DaoAI Camera Studio User Manual

Release 0.1

DaoAI

Oct 26, 2023

GETTING STARTED

This page gives an overview of DaoAI's product line. DaoAI offers a wide range of 3D cameras for different working distances and use cases. Compared to our competitors, our cameras provide:

- Accurate, high-resolution captures under a variety of lighting conditions
- Fast capture speed and on-cam processing
- Powerful filters and post-processing tools to improve the quality of generated point clouds
- Stable capture performance under temperatures ranging from 0-40 degrees Celsius
- IP65 rated dust tight and water-resistant case
- Easy integration with most major robotics brands

Our DaoAI Camera Studio application is the supporting graphical user interface software that allows users to directly interact with our DaoAI 3D Camera systems.

CHAPTER

ONE

QUICK START GUIDE

This a quick start guide for the fully user manual, we recommend that you follow the user guide for the details setting and studying.

BP SMALL

Unbox

In the DaoAI Box you will find:

- DaoAI BP SMALL camera
- 24V power supply
- Power extension cables (6m)
- Ethernet (CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

Install

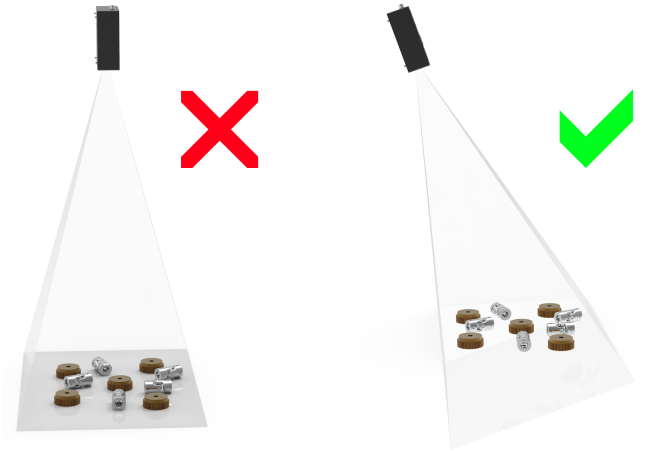
Requirements PCs:

- Windows 10 or Ubuntu 20.04
- Compatible GPU
- 16 GB RAM or more

Follow the steps outlined in *Software Installation*

Download DaoAI Camera Studio

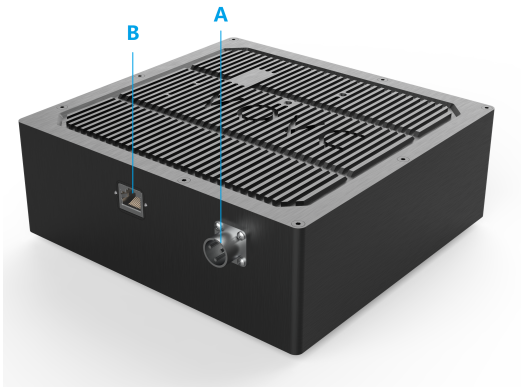
Mount



3D cameras use active lighting to detect your target objects. To minimize direct reflections from the background and reduce potential artifacts, it is recommended that you mount your camera at a slight angle to get the best results. You can test different positions in DaoAI Studio. See available *Mechanical interface*.

For more information see *Working Distance and Camera Positioning* and *Mechanical Installation*.

Connect



A. Power Connector 24V, 10A DC

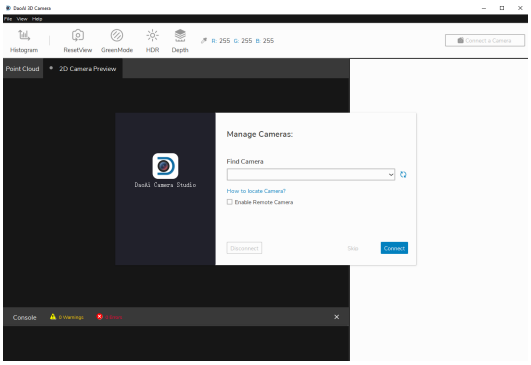
B. Ethernet Connector CAT 6 or higher

Ensure that all connections are screwed in tightly.

Network Configuration to connect to the camera.

For more information see *Connectivity and Power Supply*.

Lanuch



After installing the software, Launch “DaoAI 3D EN.exe” English version or “DaoAI 3D CN.exe” Chinese version, you will be greeted with the DaoAI Camera Studio startup window.

See the *Connecting & Disconnecting Camera* tutorials. Check out the connect camera Configuration page to lean how to connect your camera.

For more inforation see *User Guide*.

BP MEDIUM

Unbox

In the DaoAI Box you will find:

- DaoAI BP MEDIUM camera
- 24V power supply
- Power extension cables (6m)
- Ethernet (CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

Install

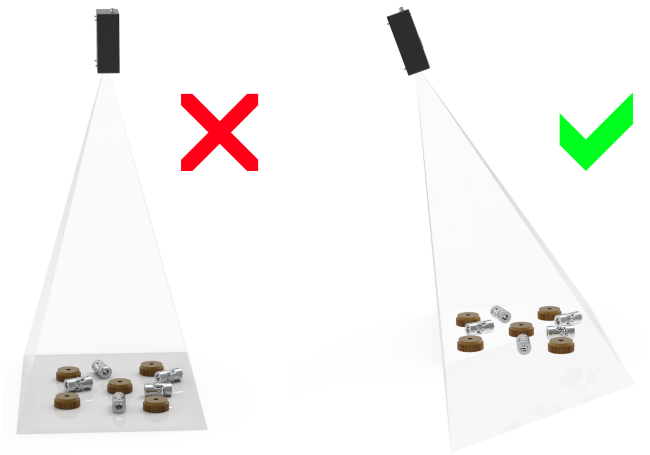
Requirements PCs:

- Windows 10 or Ubuntu 20.04
- Compatible GPU
- 16 GB RAM or more

Follow the steps outlined in *Software Installation*

Download DaoAI Camera Studio

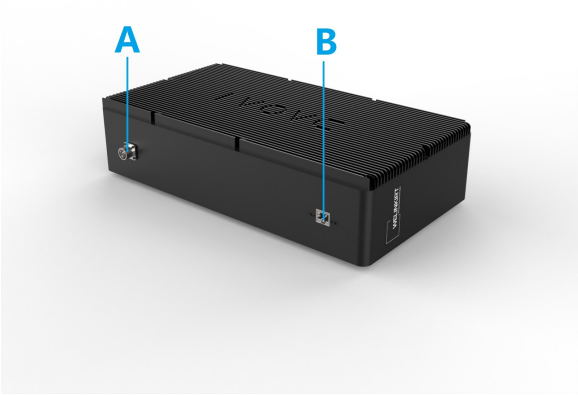
Mount



3D cameras use active lighting to detect your target objects. To minimize direct reflections from the background and reduce potential artifacts, it is recommended that you mount your camera at a slight angle to get the best results. You can test different positions in DaoAI Studio. See available *Mechanical interface*.

For more information see *Working Distance and Camera Positioning* and *Mechanical Installation*.

Connect



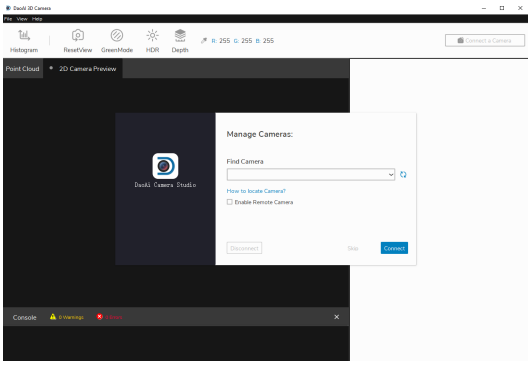
A. Power Connector 24V, 10A DC
B. Ethernet Connector CAT 6 or higher

Ensure that all connections are screwed in tightly.

Network Configuration to connect to the camera.

For more information see *Connectivity and Power Supply*.

Lanuch



After installing the software, Launch “DaoAI 3D EN.exe” English version or “DaoAI 3D CN.exe” Chinese version, you will be greeted with the DaoAI Camera Studio startup window.

See the *Connecting & Disconnecting Camera* tutorials. Check out the connect camera Configuration page to lean how to connect your camera.

For more inforation see *User Guide*.

BP LARGE

Unbox

In the DaoAI Box you will find:

- DaoAI BP LARGE camera
- 24V power supply
- Power extension cables (6m)
- Ethernet (CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

Install

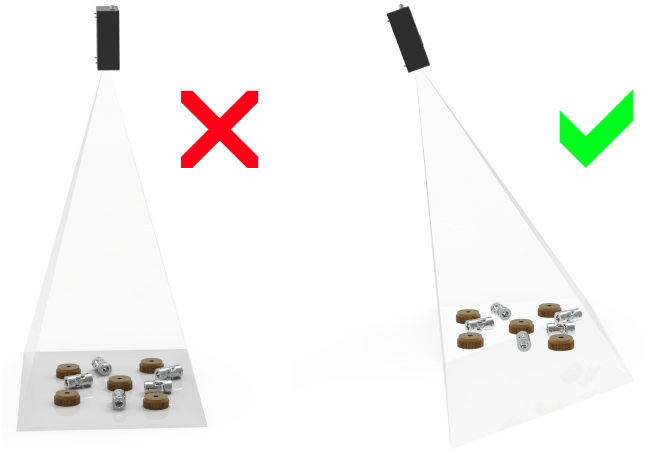
Requirements PCs:

- Windows 10 or Ubuntu 20.04
- Compatible GPU
- 16 GB RAM or more

Follow the steps outlined in *Software Installation*

Download DaoAI Camera Studio

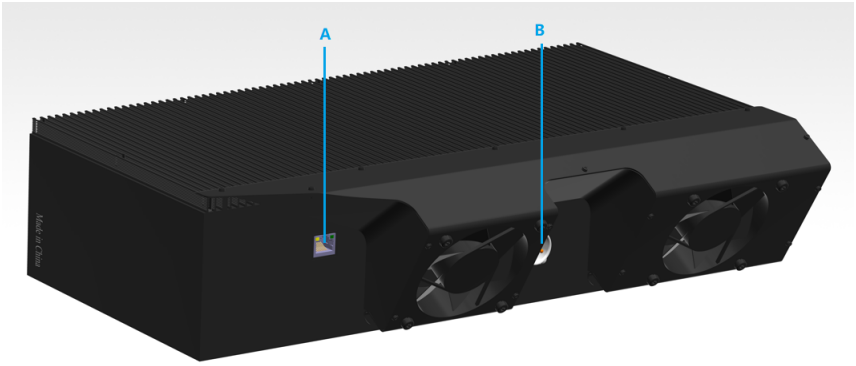
Mount



3D cameras use active lighting to detect your target objects. To minimize direct reflections from the background and reduce potential artifacts, it is recommended that you mount your camera at a slight angle to get the best results. You can test different positions in DaoAI Studio. See available *Mechanical interface*.

For more information see *Working Distance and Camera Positioning* and *Mechanical Installation*.

Connect



A. Ethernet Connector CAT 6 or higher

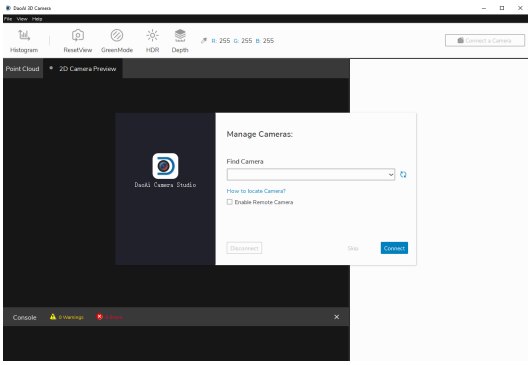
B. Power Connector 24V, 10A DC

Ensure that all connections are screwed in tightly.

Network Configuration to connect to the camera.

For more information see *Connectivity and Power Supply*.

Lanuch



After installing the software, Launch “DaoAI 3D EN.exe” English version or “DaoAI 3D CN.exe” Chinese version, you will be greeted with the DaoAI Camera Studio startup window.

See the *Connecting & Disconnecting Camera* tutorials. Check out the connect camera Configuration page to lean how to connect your camera.

For more inforation see *User Guide*.

BP AMR

Unbox

In the DaoAI Box you will find:

- DaoAI BP AMR camera
- 24V power supply
- Power extension cables (6m)
- Ethernet (CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

Install

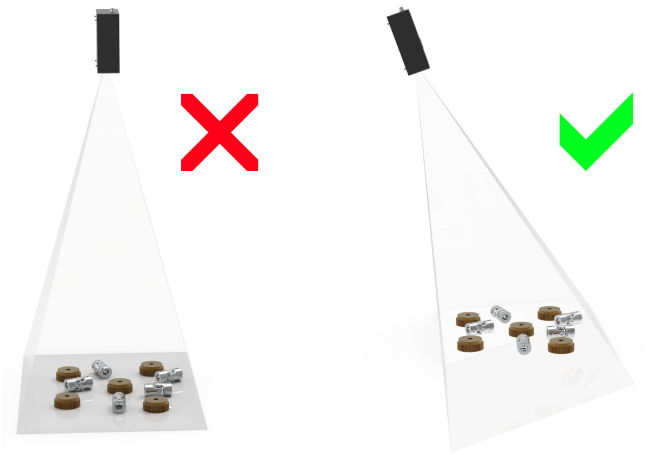
Requirements PCs:

- Windows 10 or Ubuntu 20.04
- Compatible GPU
- 16 GB RAM or more

Follow the steps outlined in *Software Installation*

Download DaoAI Camera Studio

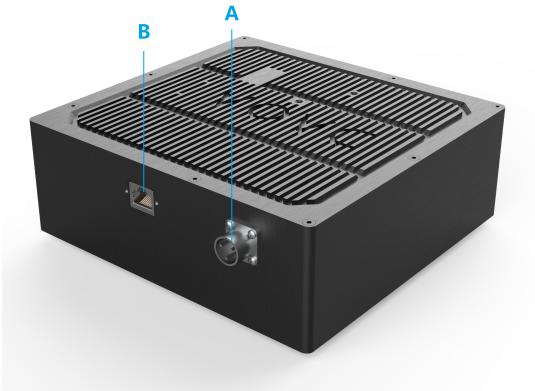
Mount



3D cameras use active lighting to detect your target objects. To minimize direct reflections from the background and reduce potential artifacts, it is recommended that you mount your camera at a slight angle to get the best results. You can test different positions in DaoAI Studio. See available *Mechanical interface*.

For more information see *Working Distance and Camera Positioning* and *Mechanical Installation*.

Connect



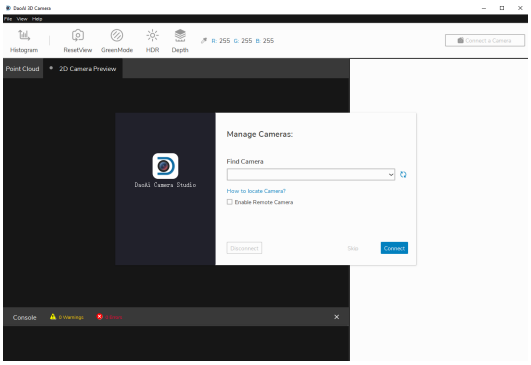
A. Power Connector 24V, 10A DC
B. Ethernet Connector CAT 6 or higher

Ensure that all connections are screwed in tightly.

Network Configuration to connect to the camera.

For more information see *Connectivity and Power Supply*.

Lanuch



After installing the software, Launch “DaoAI 3D EN.exe” English version or “DaoAI 3D CN.exe” Chinese version, you will be greeted with the DaoAI Camera Studio startup window.

See the *Connecting & Disconnecting Camera* tutorials. Check out the connect camera Configuration page to lean how to connect your camera.

For more inforation see *User Guide*.

BP AMR-GPU

Unbox

In the DaoAI Box you will find:

- DaoAI BP AMR-GPU camera
- 24V power supply
- Power extension cables (6m)
- Ethernet (CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

Install

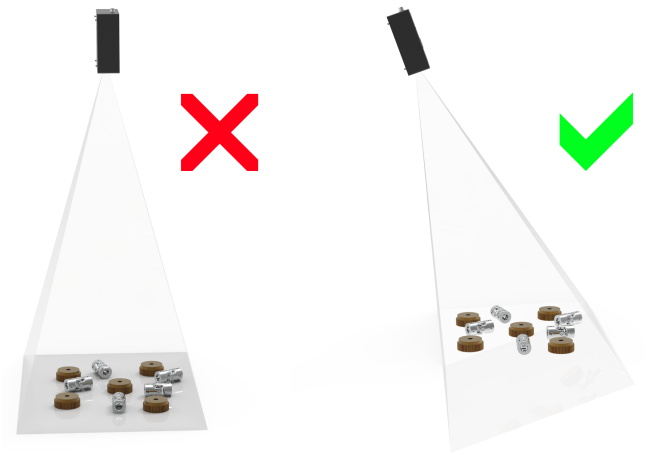
Requirements PCs:

- Windows 10 or Ubuntu 20.04
- Compatible GPU
- 16 GB RAM or more

Follow the steps outlined in *Software Installation*

Download DaoAI Camera Studio

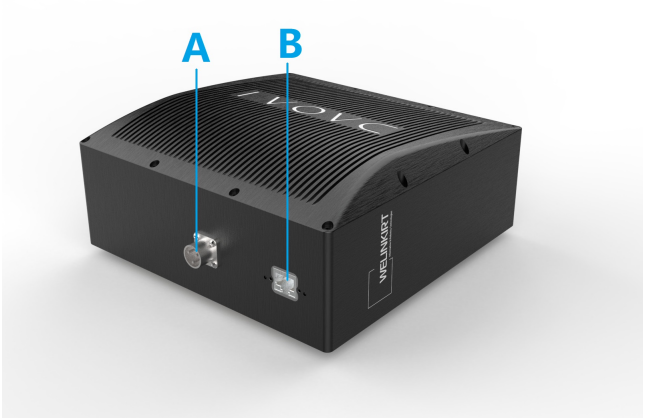
Mount



3D cameras use active lighting to detect your target objects. To minimize direct reflections from the background and reduce potential artifacts, it is recommended that you mount your camera at a slight angle to get the best results. You can test different positions in DaoAI Studio. See available *Mechanical interface*.

For more information see *Working Distance and Camera Positioning* and *Mechanical Installation*.

Connect



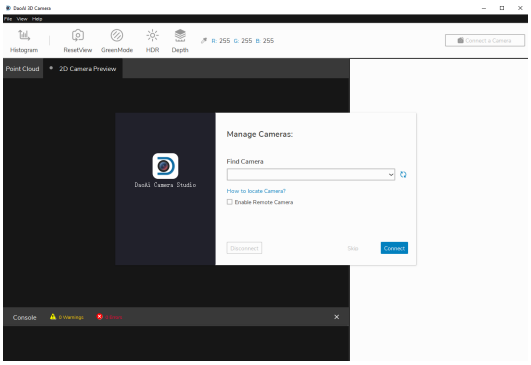
A. Power Connector 24V, 10A DC
B. Ethernet Connector CAT 6 or higher

Ensure that all connections are screwed in tightly.

Network Configuration to connect to the camera.

For more information see *Connectivity and Power Supply*.

Lanuch



After installing the software, Launch “DaoAI 3D EN.exe” English version or “DaoAI 3D CN.exe” Chinese version, you will be greeted with the DaoAI Camera Studio startup window.

See the *Connecting & Disconnecting Camera* tutorials. Check out the connect camera Configuration page to lean how to connect your camera.

For more inforation see *User Guide*.

BP LASER

Unbox

In the DaoAI Box you will find:

- DaoAI BP LASER camera
- 24V power supply
- Power extension cables (6m)
- Ethernet (CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

Install

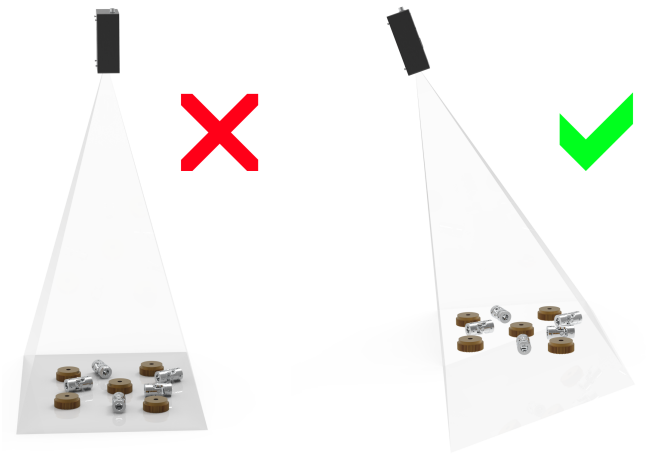
Requirements PCs:

- Windows 10 or Ubuntu 20.04
- Compatible GPU
- 16 GB RAM or more

Follow the steps outlined in *Software Installation*

Download DaoAI Camera Studio

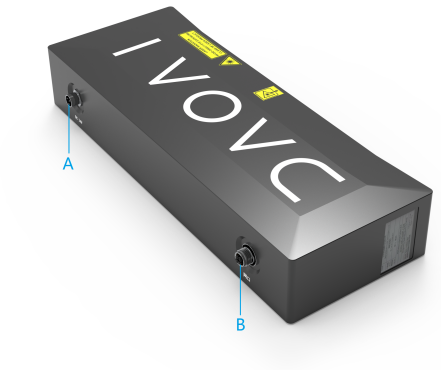
Mount



3D cameras use active lighting to detect your target objects. To minimize direct reflections from the background and reduce potential artifacts, it is recommended that you mount your camera at a slight angle to get the best results. You can test different positions in DaoAI Studio. See available *Mechanical interface*.

For more information see *Working Distance and Camera Positioning* and *Mechanical Installation*.

Connect



A. Power Connector 24V, 10A DC

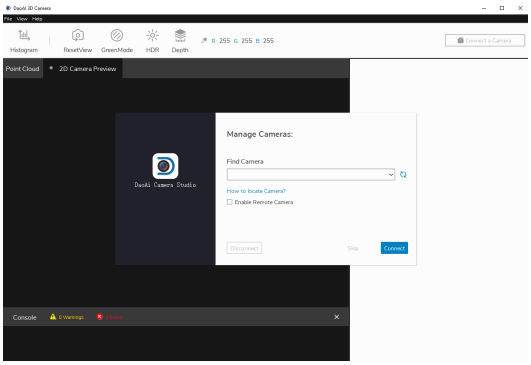
B. Ethernet Connector CAT 6 or higher

Ensure that all connections are screwed in tightly.

Network Configuration to connect to the camera.

For more information see *Connectivity and Power Supply*.

Lanuch



After installing the software, Launch “DaoAI 3D EN.exe” English version or “DaoAI 3D CN.exe” Chinese version, you will be greeted with the DaoAI Camera Studio startup window.

See the *Connecting & Disconnecting Camera* tutorials. Check out the connect camera Configuration page to lean how to connect your camera.

For more inforation see *User Guide*.

Warning: Do NOT deliberately look into or stare into the beam – this can cause serious injury to your eye.

CHAPTER

TWO

USER GUIDE

This guide will help you get started with DaoAI Camera Studio. The detailed datasheet for the DaoAI Camera Studio is available at the following links:

2.1 Unboxing

BP SMALL

In the DaoAI Camera Studio Box you will find:

- DaoAI BP Small Camera
- Power supply
- Power extension cables (6m)
- Ethernet(CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

BP MEDIUM

In the DaoAI Camera Studio Box you will find:

- DaoAI BP Medium Camera
- Power supply
- Power extension cables (6m)
- Ethernet(CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

BP LARGE

In the DaoAI Camera Studio Box you will find:

- DaoAI BP Large Camera
- Power supply
- Power extension cables (6m)

- Ethernet(CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

BP AMR

In the DaoAI Camera Studio Box you will find:

- DaoAI BP AMR Camera
- Power supply
- Power extension cables (6m)
- Ethernet(CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

BP AMR-GPU

In the DaoAI Camera Studio Box you will find:

- DaoAI BP AMR-GPU Camera
- Power supply
- Power extension cables (6m)
- Ethernet(CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

BP LASER

In the DaoAI Camera Studio Box you will find:

- DaoAI BP LASER Camera
- Power supply
- Power extension cables (6m)
- Ethernet(CAT 6) cables (10m)

Optional/as ordered:

- Calibration Board
- On-Arm mount

Warning: Do NOT deliberately look into or stare into the beam – this can cause serious injury to your eye.

2.2 System Requirements

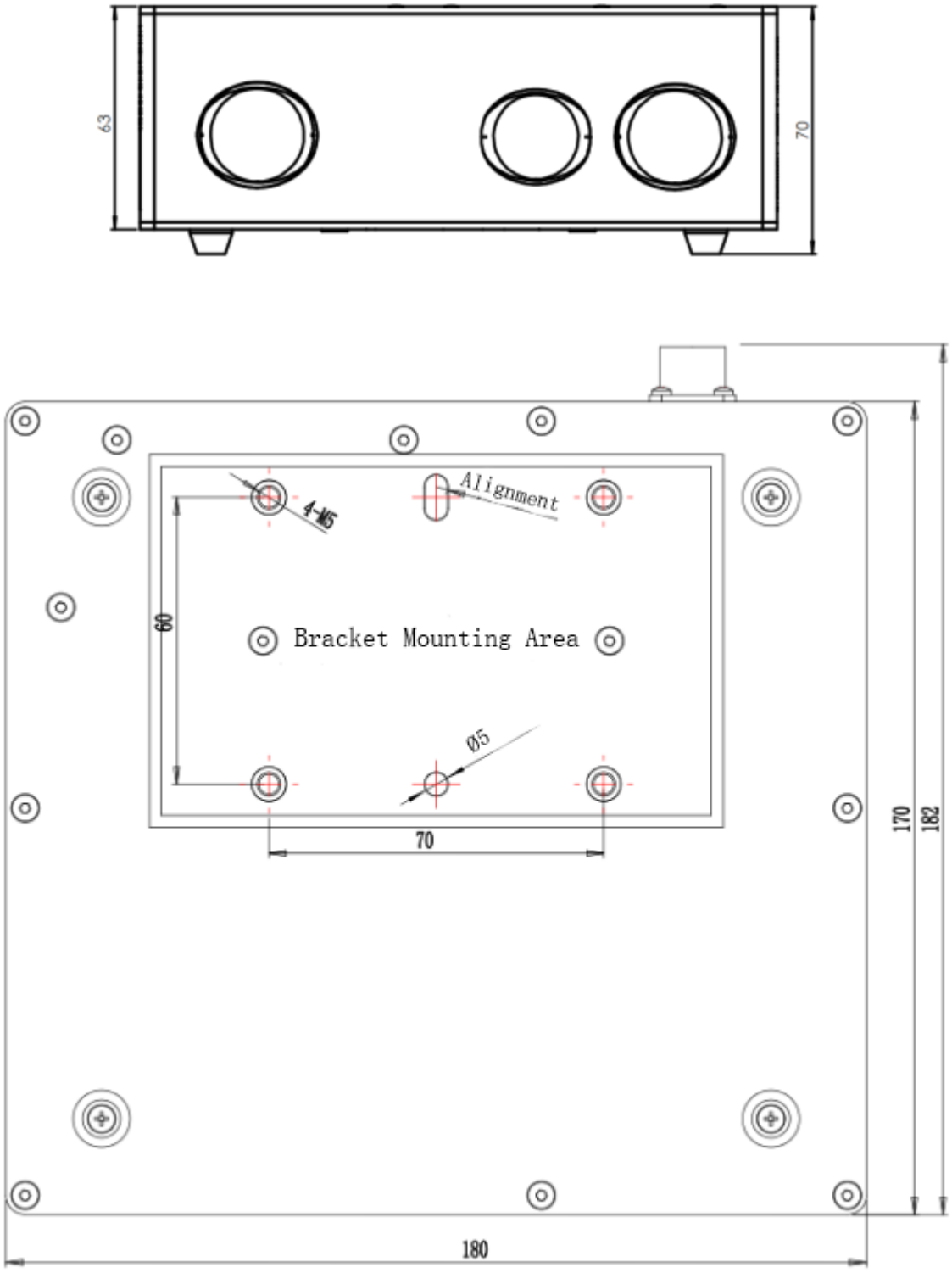
OS	Windows 10 or Linux Ubuntu 20.04
Dedicated GPU	A dedicated GPU gives the best performance with DaoAI Camera Studio. It is also the best choice if the GPU will be used for more than DaoAI's computations. A medium to high-end NVIDIA GPU with at least 3 GB of memory is required for optimal performance. This is the preferred solution for DaoAI Camera Studio. **Recommendations:** • NVIDIA GeForce GTX 1050Ti or better • NVIDIA GeForce MX150 or better
USB	SuperSpeed USB3 port
Ethernet	1 Gbps adapter connected via PCI Express or Thunderbolt 3

2.3 Mechanical Installation

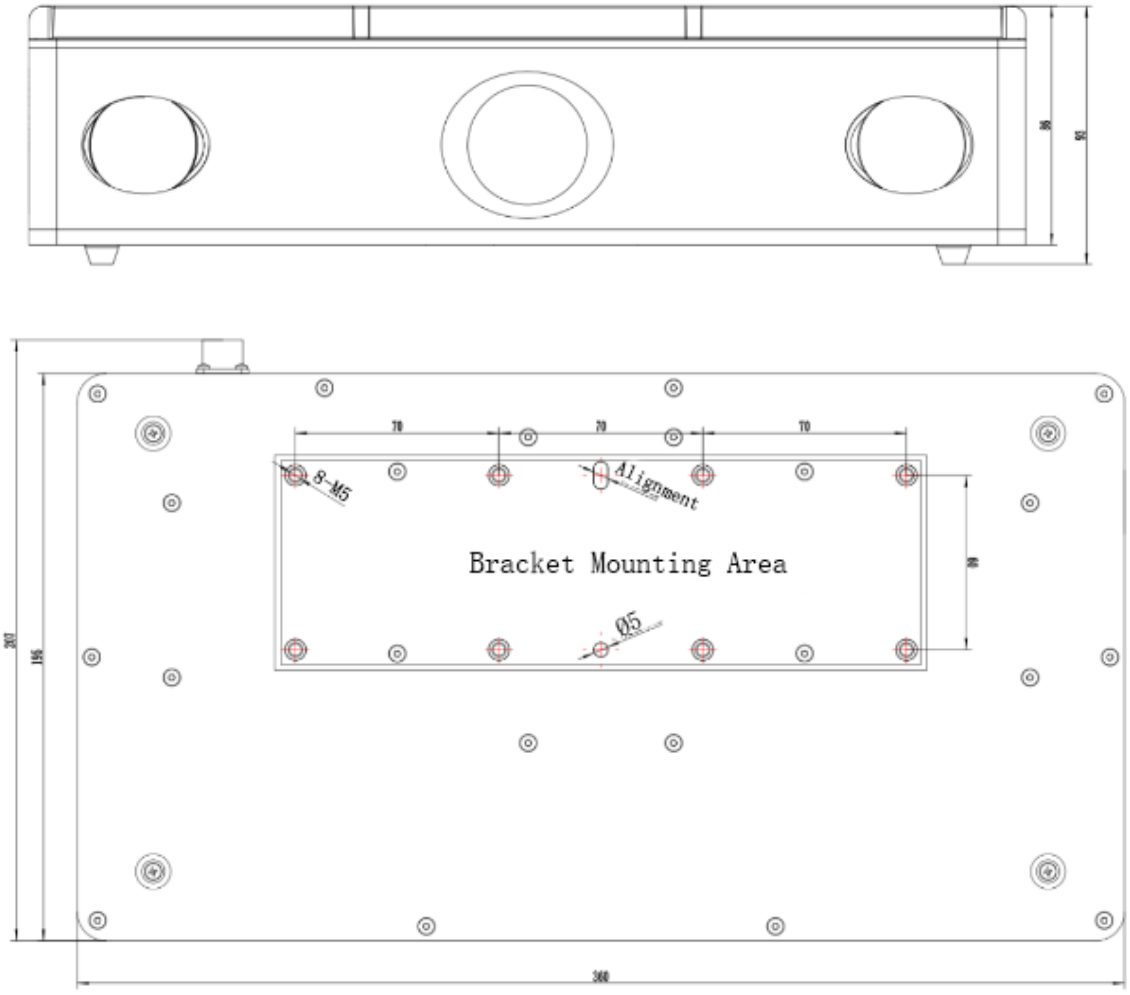
2.3.1 Mechanical interface

Dimensions

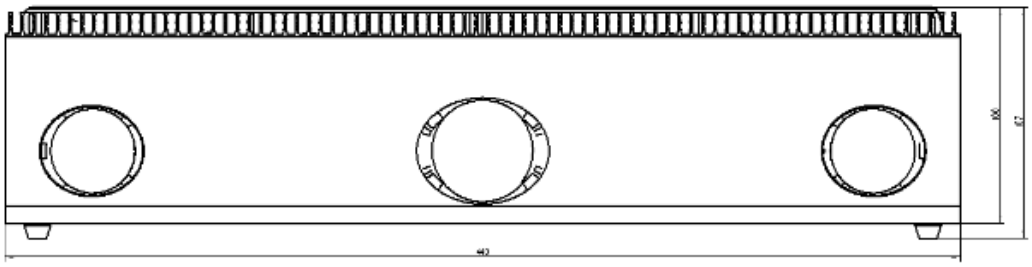
BP SMALL

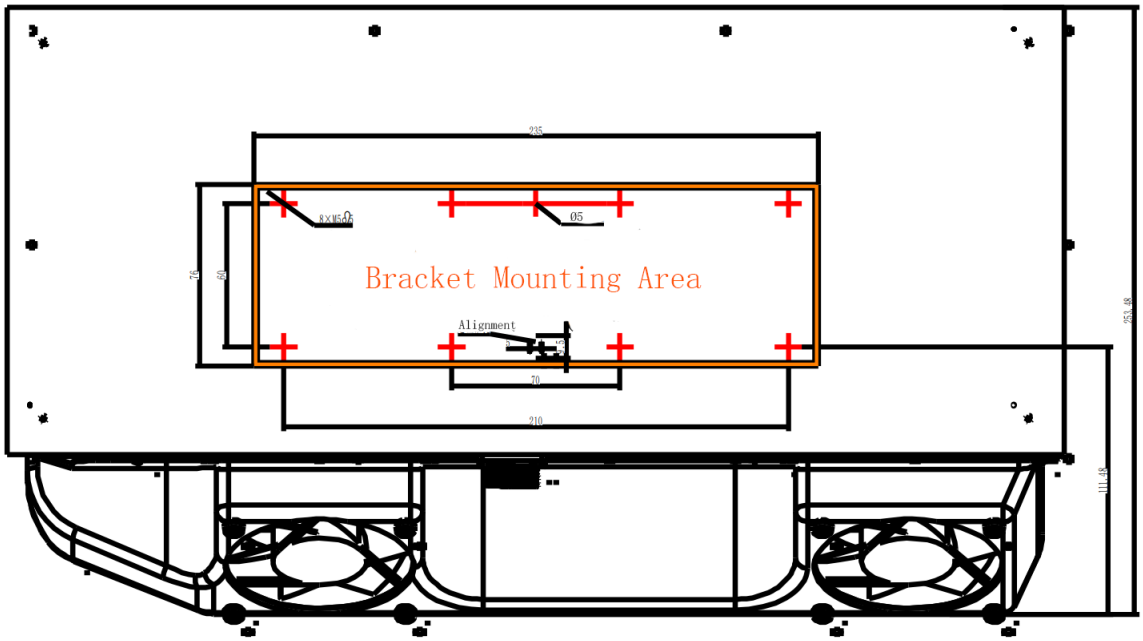

63
70
4-M5
Alignment
60
Bracket Mounting Area
Ø5
70
170
182
180

BP MEDIUM

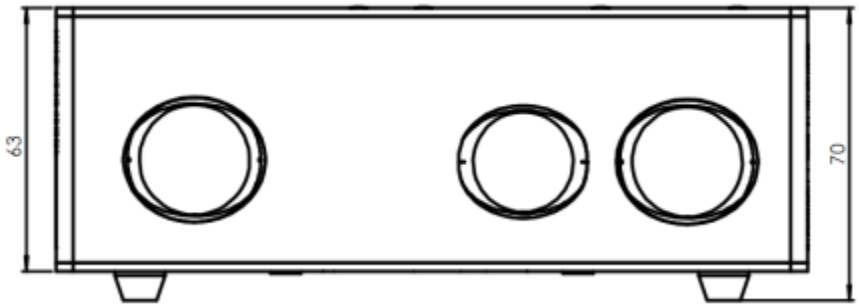


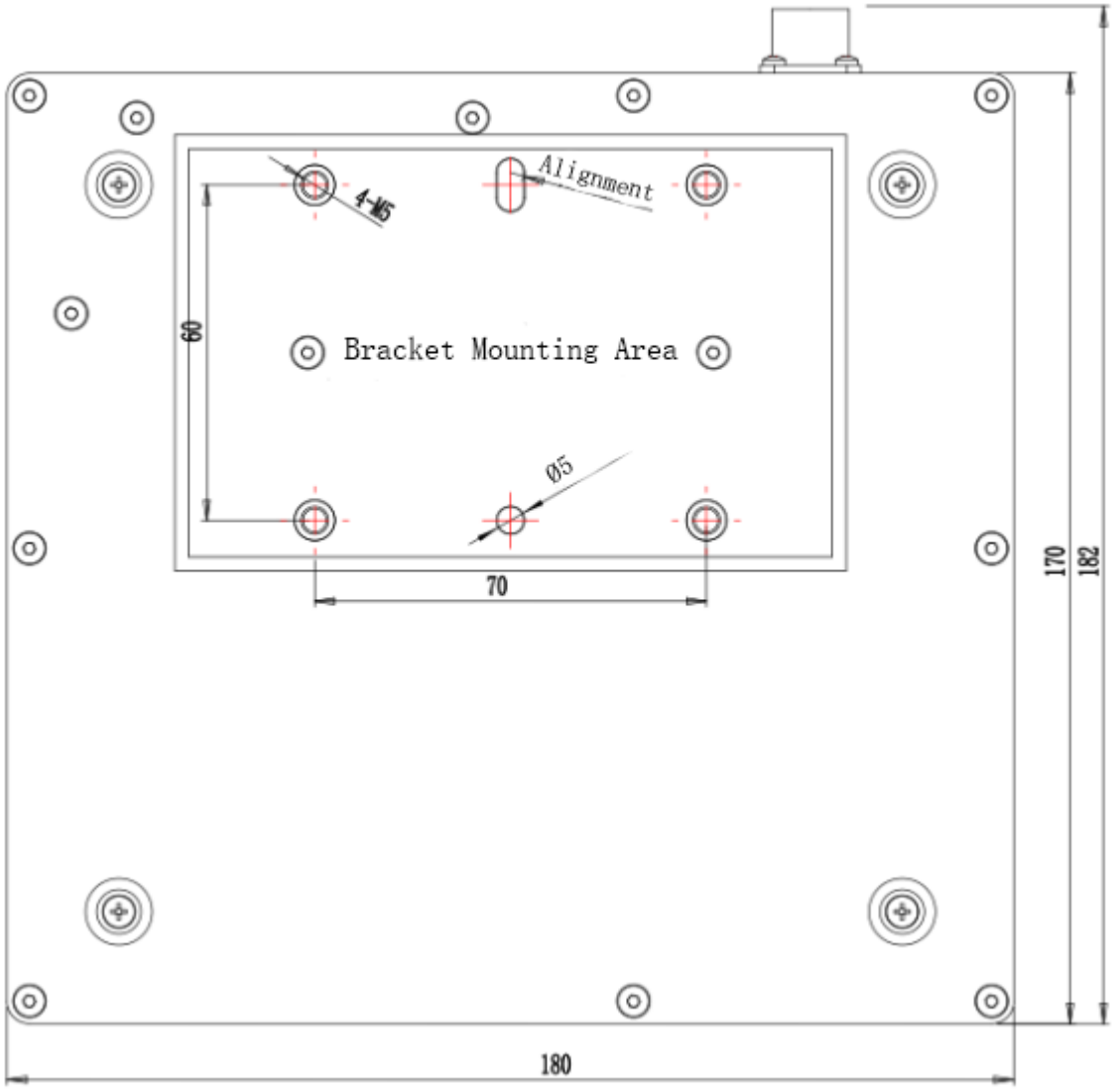
BP LARGE



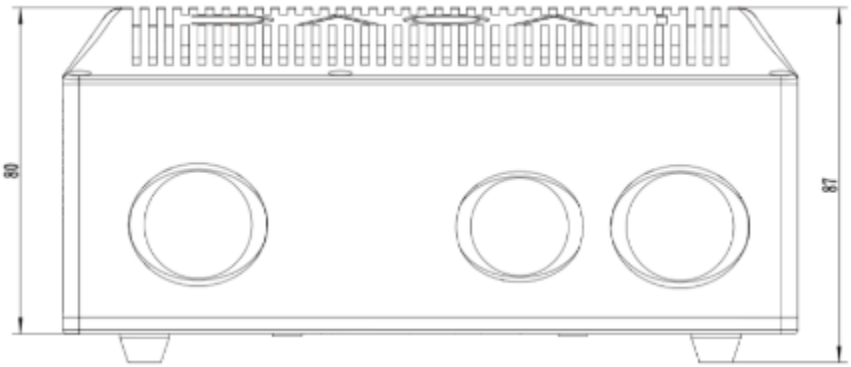


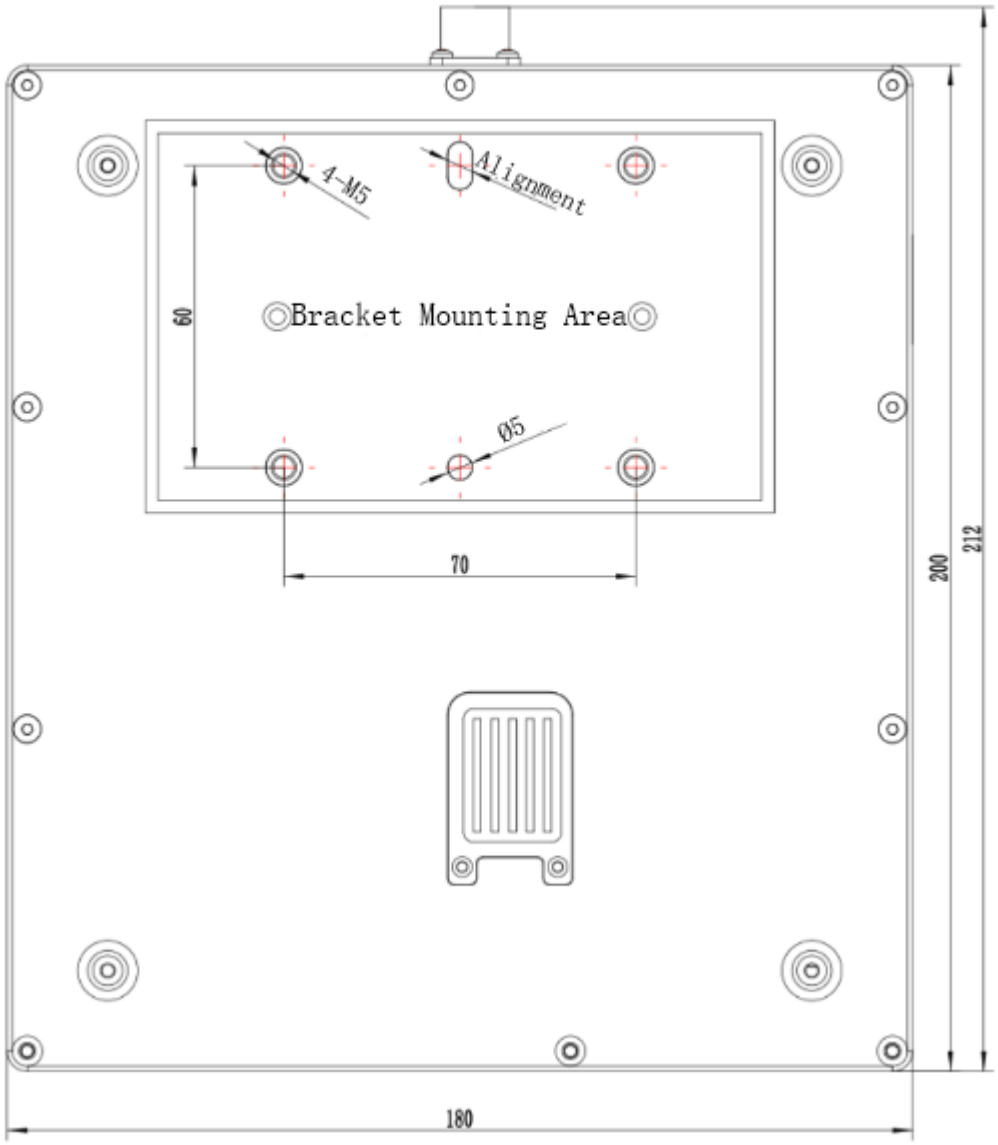
BP AMR



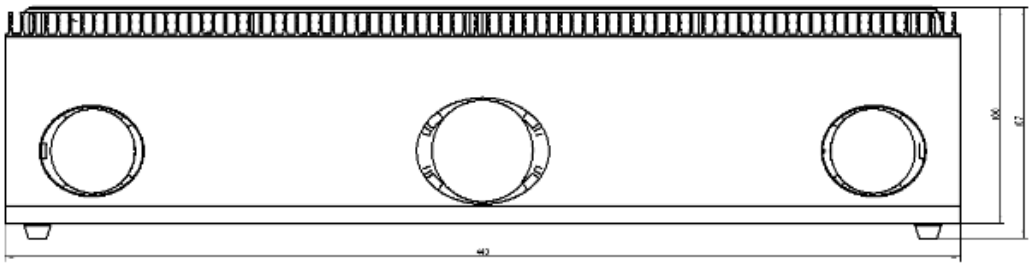

Alignment
4-M5
Bracket Mounting Area
60
Ø5
70
170
182
180

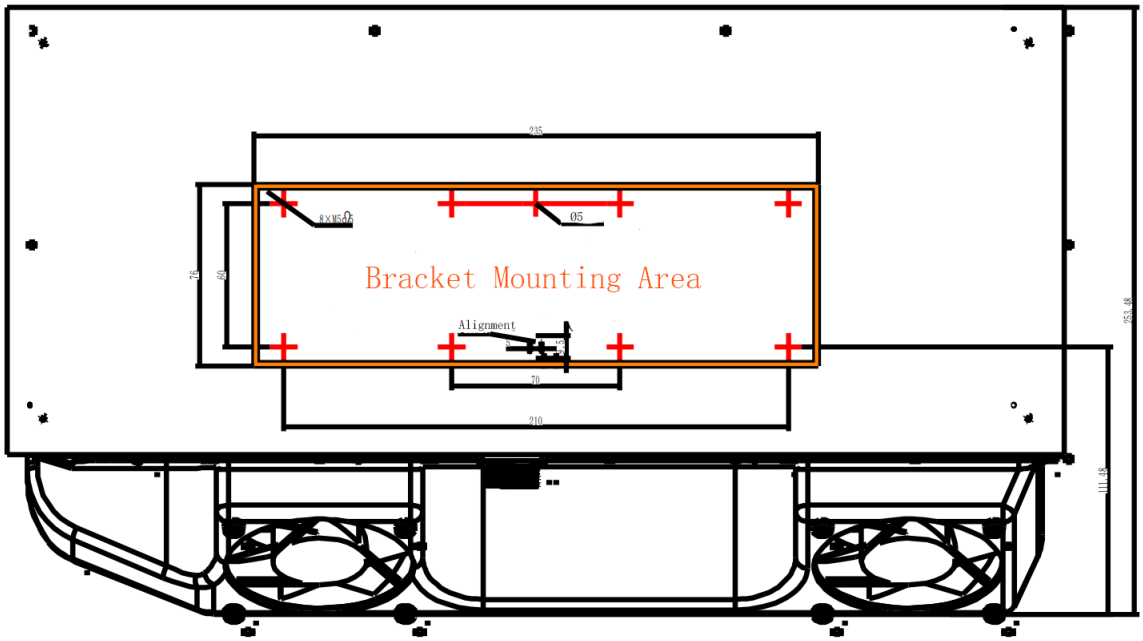
BP AMR-GPU


80
87



BP LASER

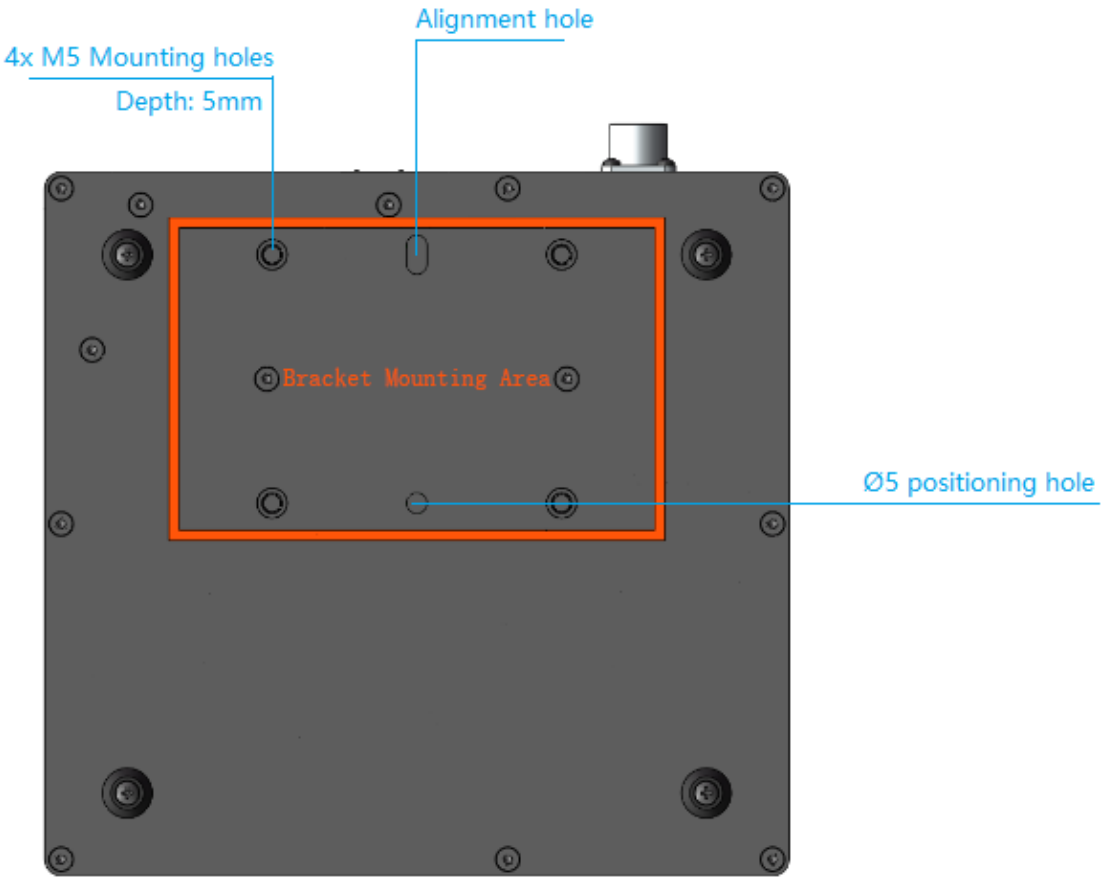




Mounting Specifications

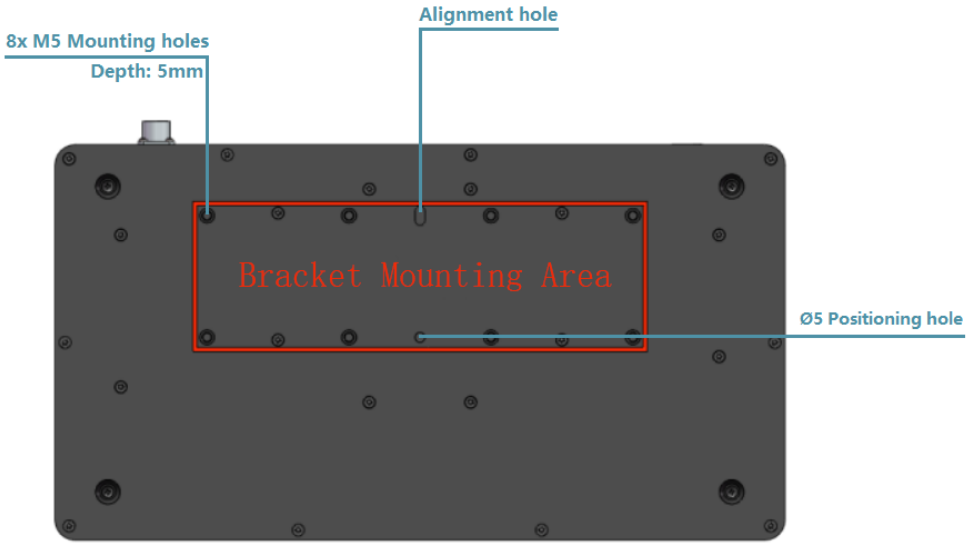
BP SMALL

The DaoAI BP SMALL camera has four M5 mounting holes, one Ø5 positioning hole, and one obround alignment hole. To ensure not to damage the threads, we recommend not exceeding the specified maximum torque value when fastening the screws.



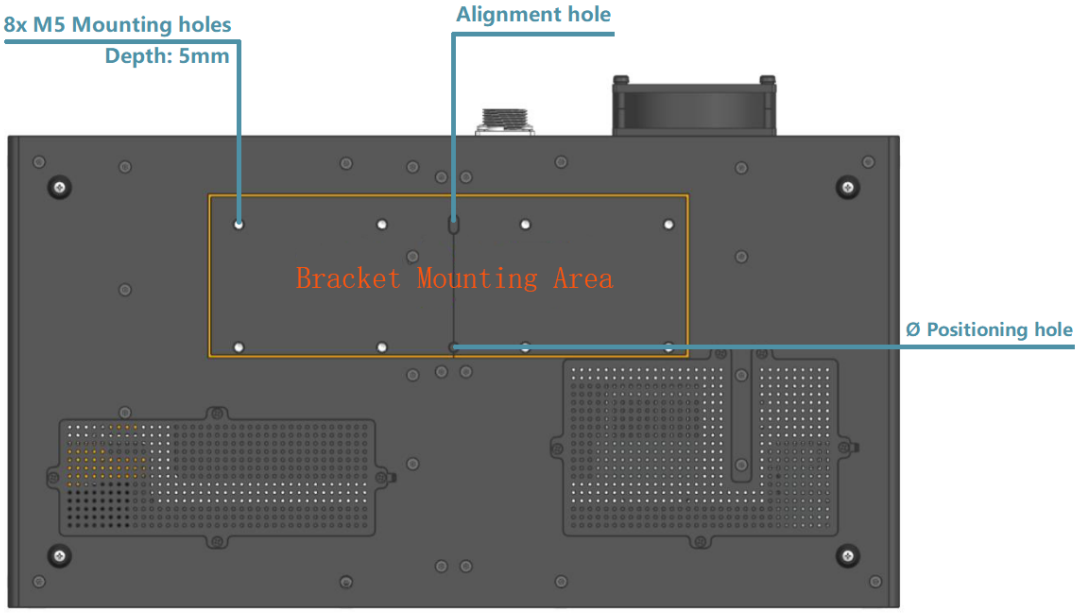
BP MEDIUM

The DaoAI BP MEDIUM camera has eight M5 mounting holes, one Ø5 positioning hole, and one obround alignment hole. To ensure not to damage the threads, we recommend not exceeding the specified maximum torque value when fastening the screws.



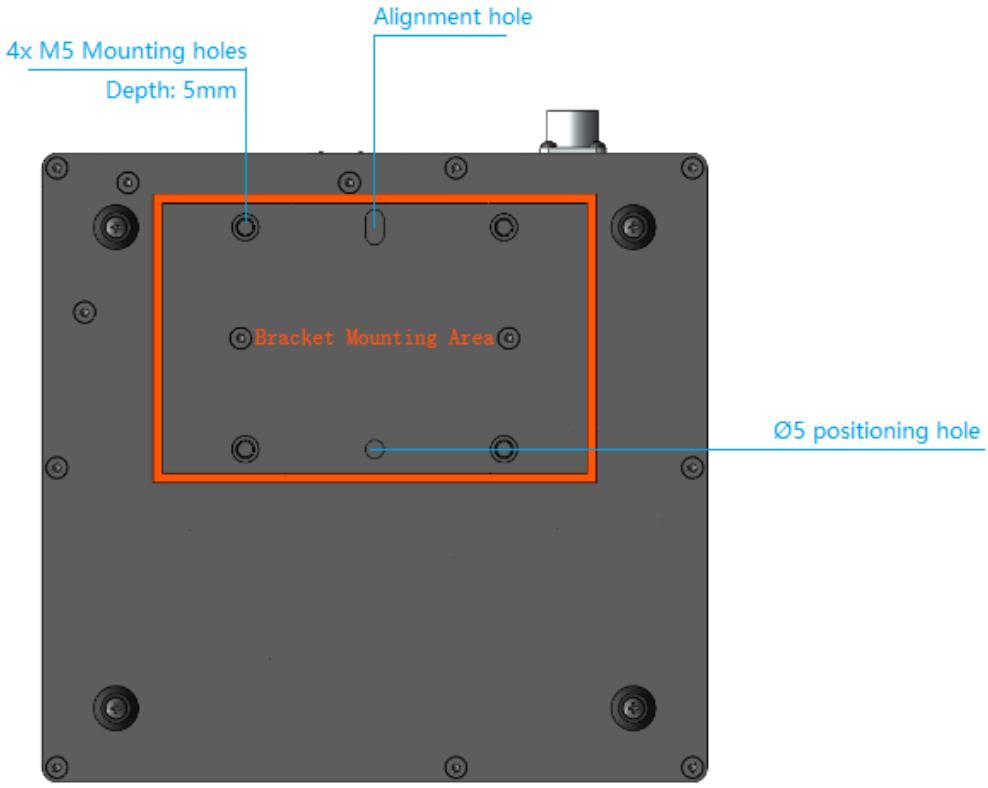
BP LARGE

The DaoAI BP LARGE camera has eight M5 mounting holes, one Ø5 positioning hole, and one obround alignment hole. To ensure not to damage the threads, we recommend not exceeding the specified maximum torque value when fastening the screws.



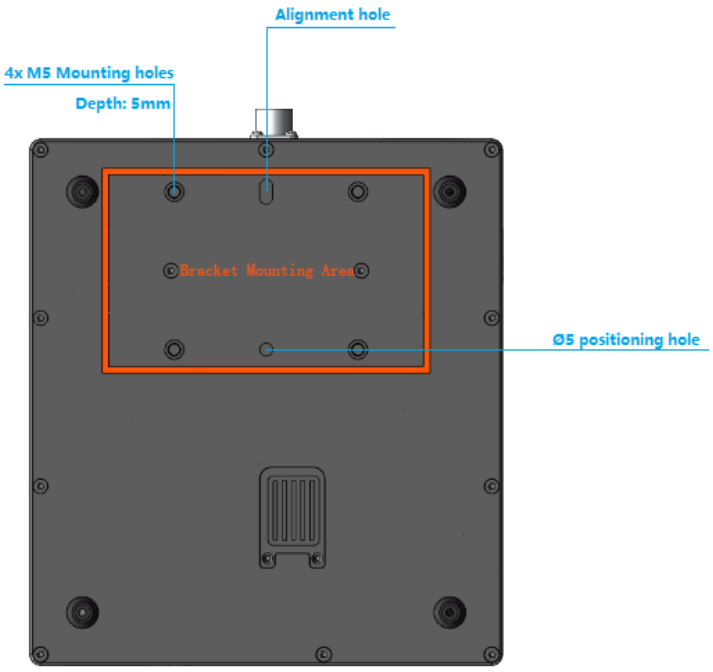
BP AMR

The DaoAI BP AMR camera has four M5 mounting holes, one Ø5 positioning hole, and one obround alignment hole. To ensure not to damage the threads, we recommend not exceeding the specified maximum torque value when fastening the screws.



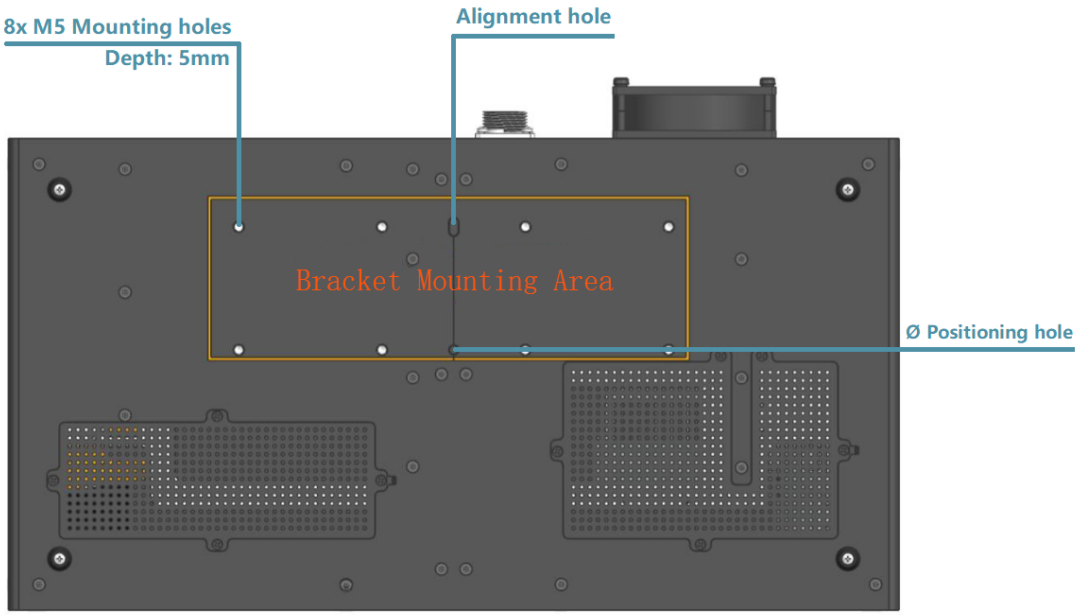
BP AMR-GPU

The DaoAI BP AMR-GPU camera has four M5 mounting holes, one Ø5 positioning hole, and one obround alignment hole. To ensure not to damage the threads, we recommend not exceeding the specified maximum torque value when fastening the screws.



BP LASER

The DaoAI BP LASER camera has eight M5 mounting holes, one Ø5 positioning hole, and one obround alignment hole. To ensure not to damage the threads, we recommend not exceeding the specified maximum torque value when fastening the screws.

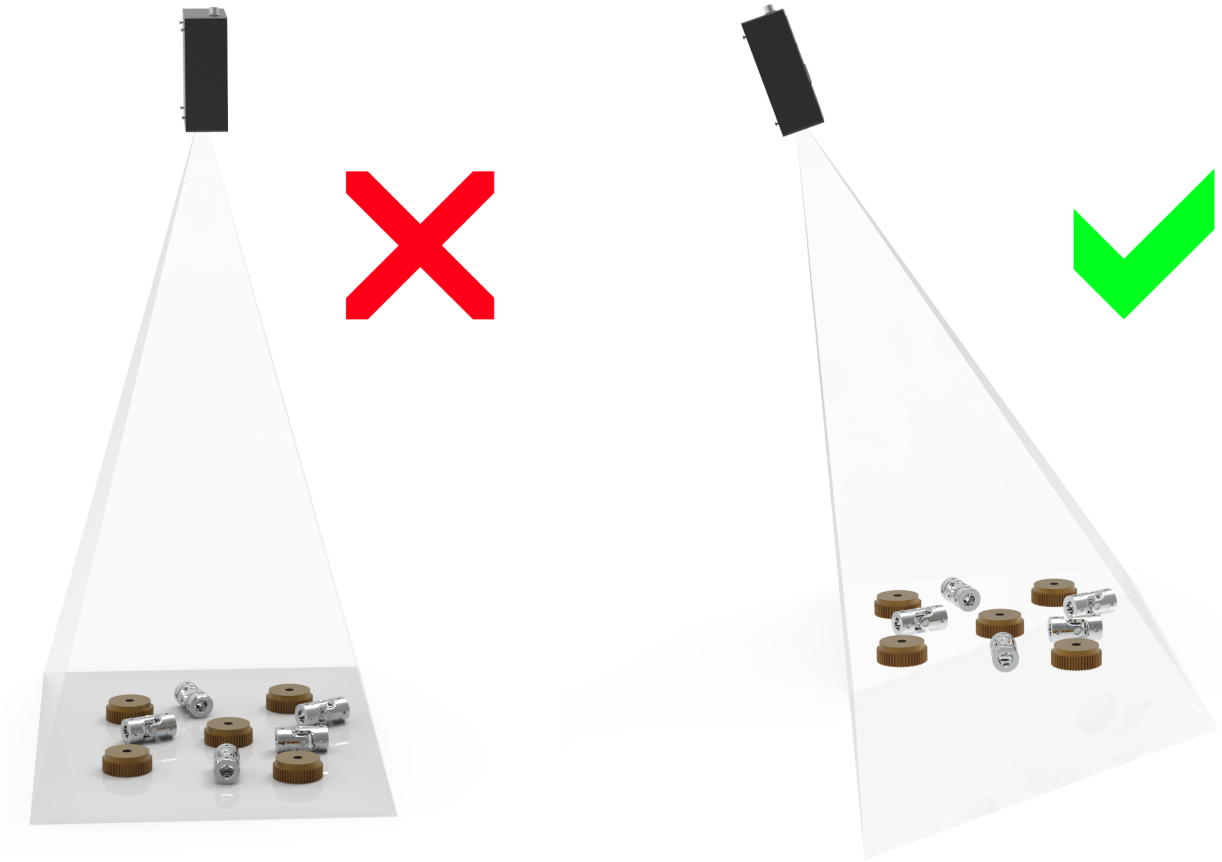


2.3.2 Positioning Correctly

The camera and the projector have an angle with respect to the center axis. This should be considered if it is desired to have the camera perpendicular to the scene.

If possible, mount the camera at a slight tilt angle to avoid reflections and interference from the background. This also frees up space above the scene for easier access for tools and robots. Check out available DaoAI mounts.

Ambient light might reduce performance. Consider blocking direct light affecting the scene.

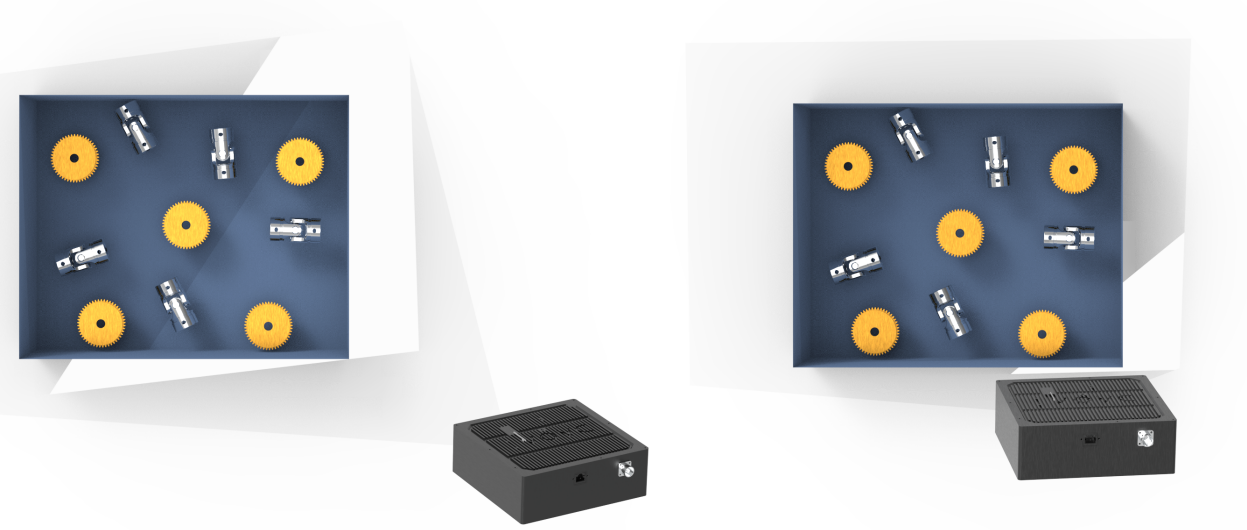


Note: Camera tilting is more important if the scene contains specular surfaces.

Please checkout *Working Distance and Field-of-View* Working Distance and Camera Positioning for more information on how to correctly position your camera.

In bin-picking applications

For bin-picking applications, place the DaoAI BP camera projector above the back edge or above the rear corner of the bin (see images below). Pan and tilt it so that the 2D camera is looking at the center of the bin. The projector rays should not fall on the inner surfaces of the two walls closest to the projector; they should almost be parallel to those two walls. Mounting the camera this way minimizes inter reflections from the bin walls.



Cooling clearance

DaoAI BP LARGE cameras use active and passive cooling, other BP cameras use passive cooling, they all allow some space around the device for airflow, and do not block the air opening on its front and rear sides. See the datasheets for the operating temperature range for your camera.

Signal protection

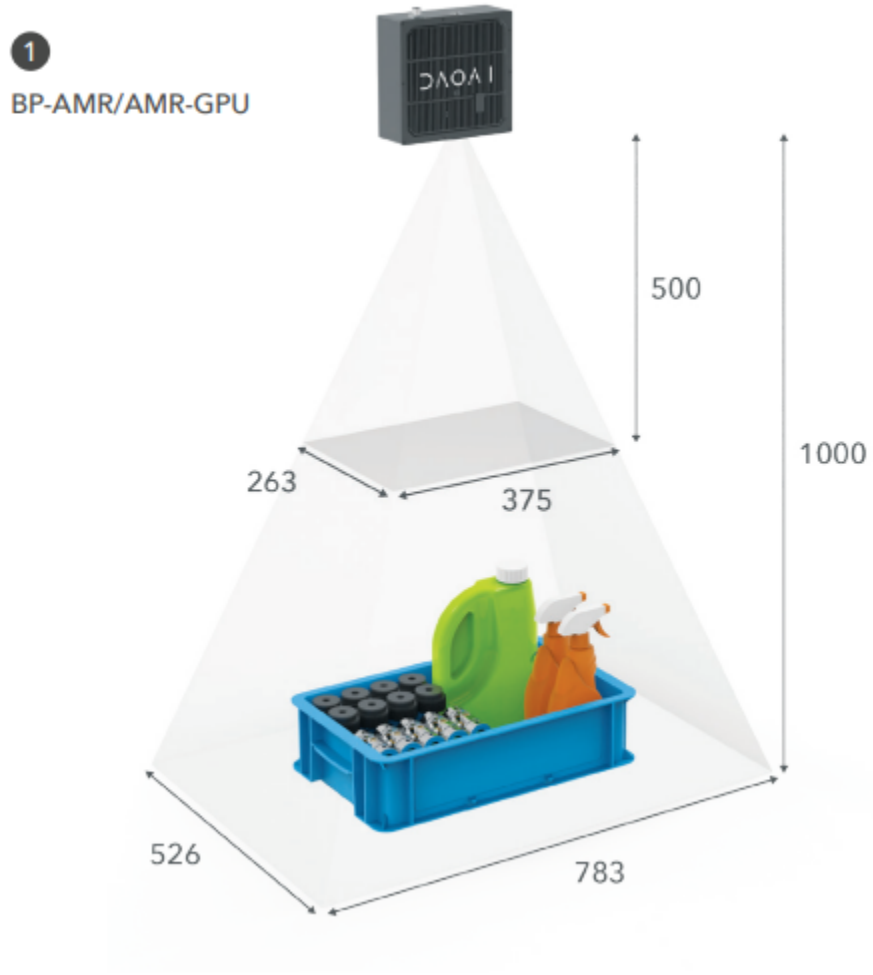
Do not install DaoAI BP cameras and cables next to high voltage devices that can generate high levels of electromagnetic disturbance. Do not route camera cabling through the same trunks/conduits with AC power cables and cables emitting high levels of disturbance.

Continue reading about *Connectivity and Power Supply*.

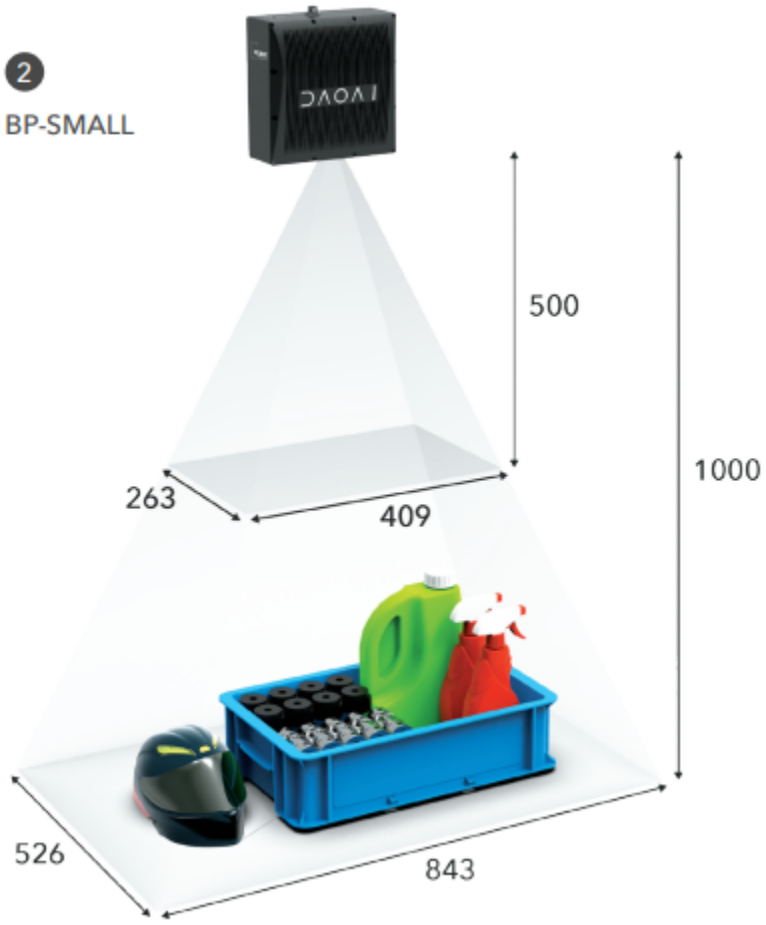
2.3.3 Working Distance and Field-of-View

This series has five bin-picking camera models for various workspace sizes and applications such as bin picking, palletizing, and part loading.

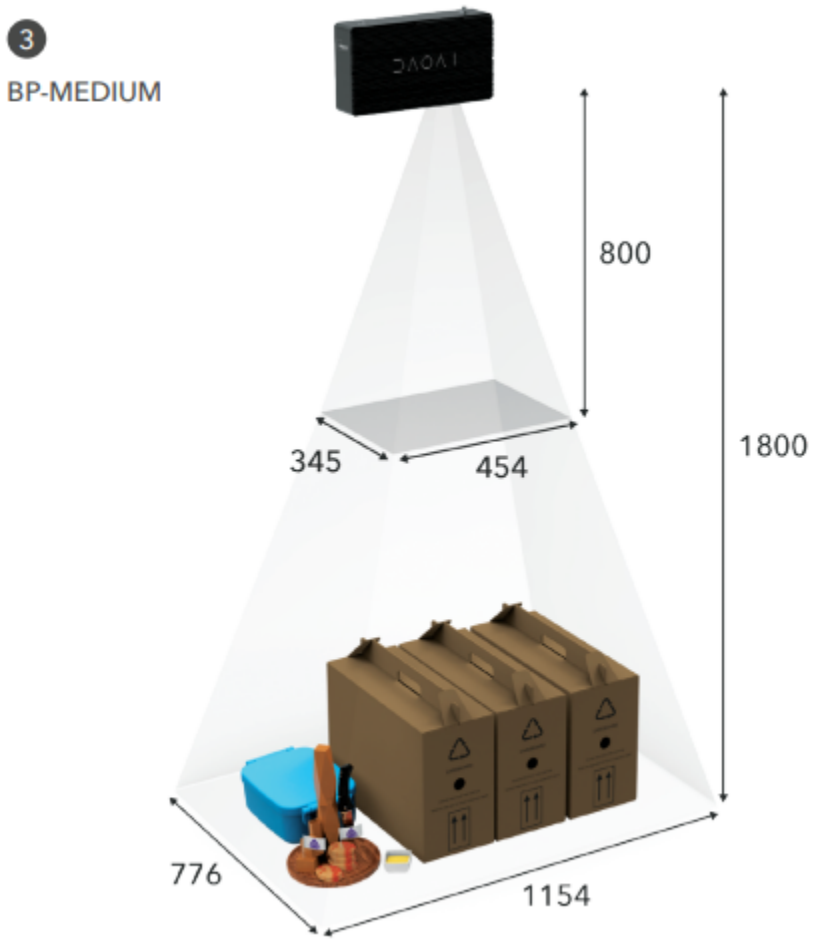
BP ARM/AMR-GPU



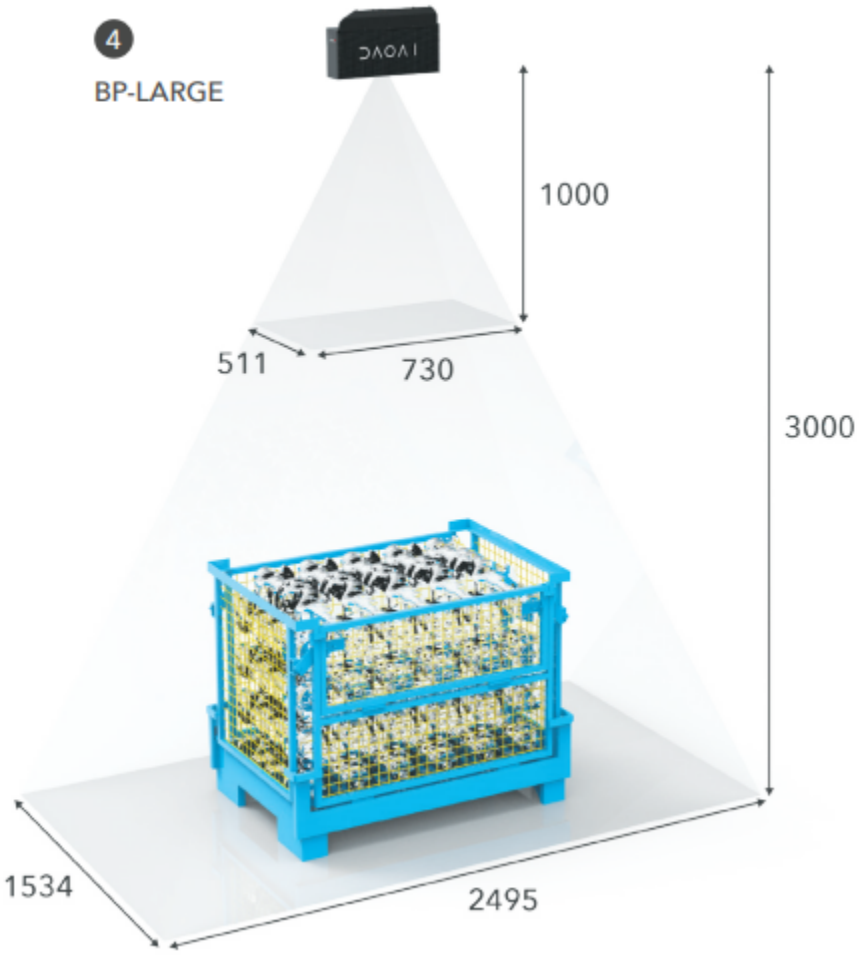
BP SMALL



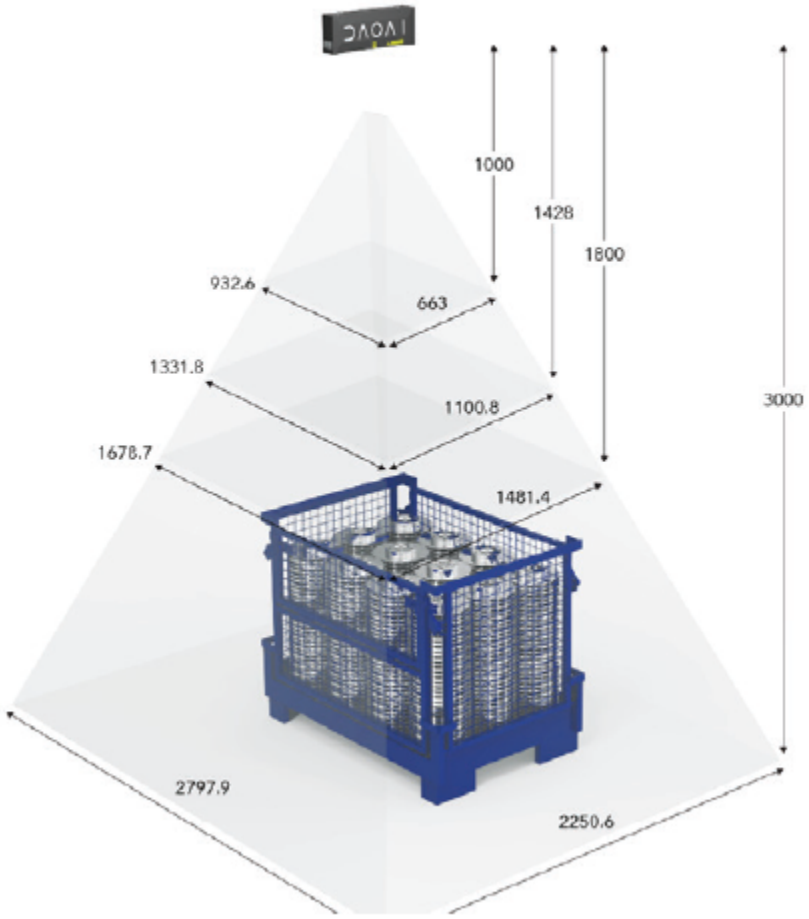
BP MEDIUM



BP LARGE



BP LASER



2.4 Connectivity and Power Supply

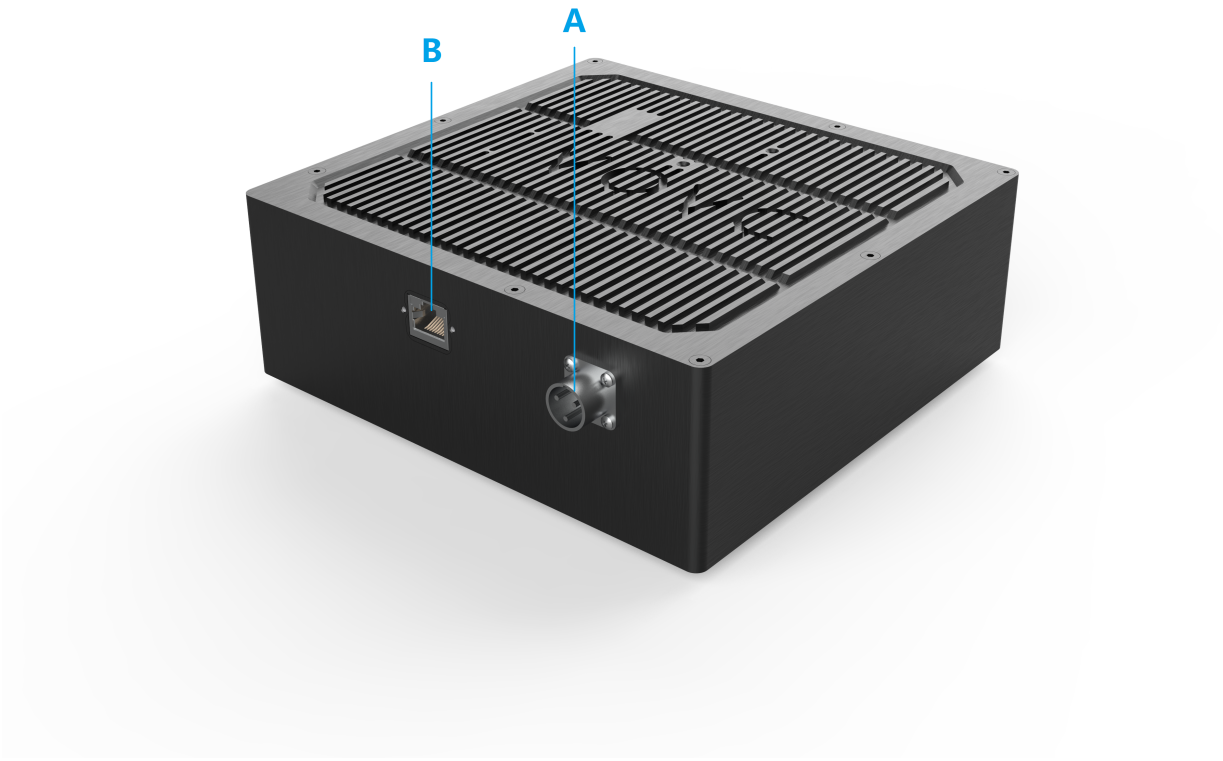
2.4.1 Hardware Setup

This page will provide instructions on how to connect your DaoAI camera to be used with DaoAI Camera Studio.

To set up the camera, plug in the power cable and connect it to your machine that has Camera Studio installed via USB (Or Ethernet for remote controlled cameras)

2.4.2 Connectors

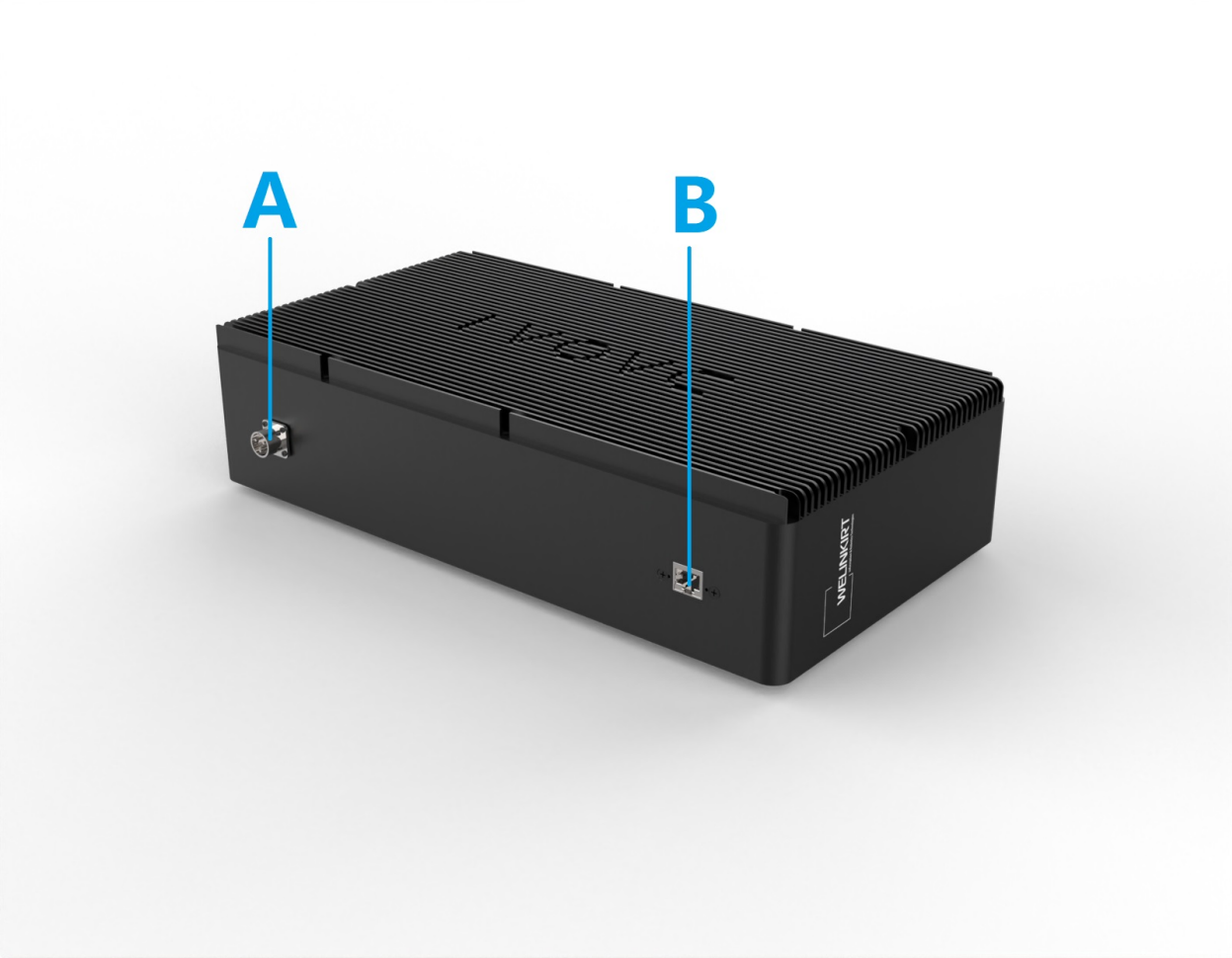
BP SMALL



A. Power Connector 24V, 10A DC

B. Ethernet Connector CAT 6 or higher

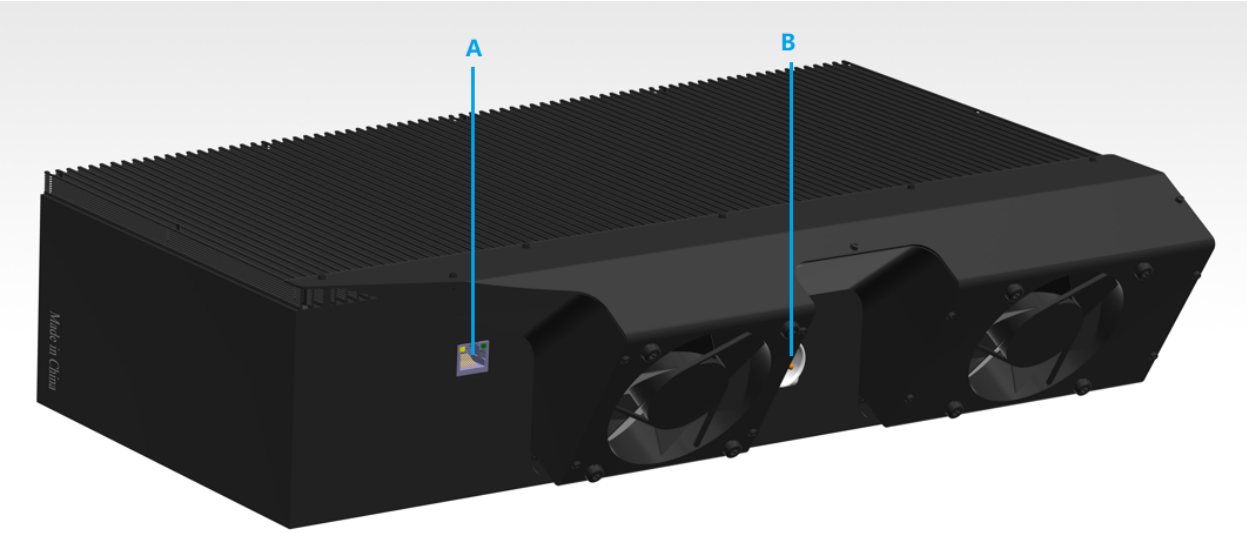
BP MEDIUM



A. Power Connector 24V, 10A DC

B. Ethernet Connector CAT 6 or higher

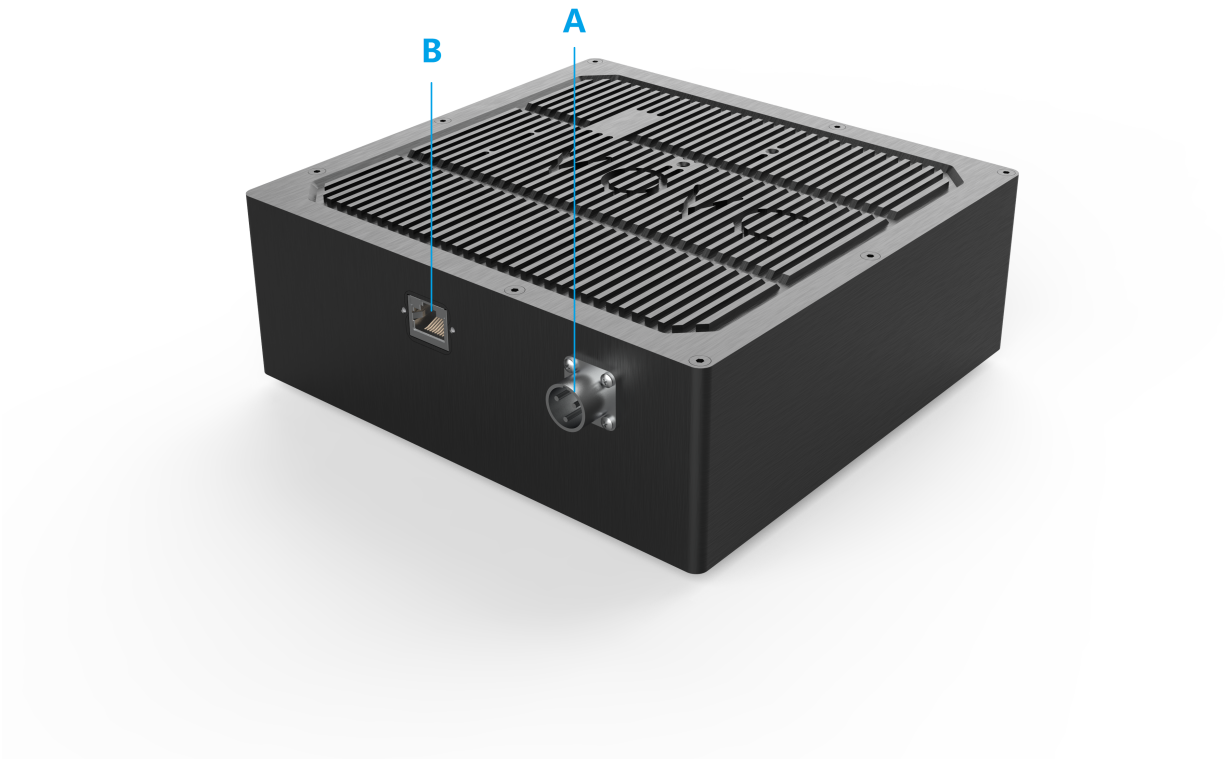
BP LARGE



A. Ethernet Connector CAT 6 or higher

B. Power Connector 24V, 15A DC

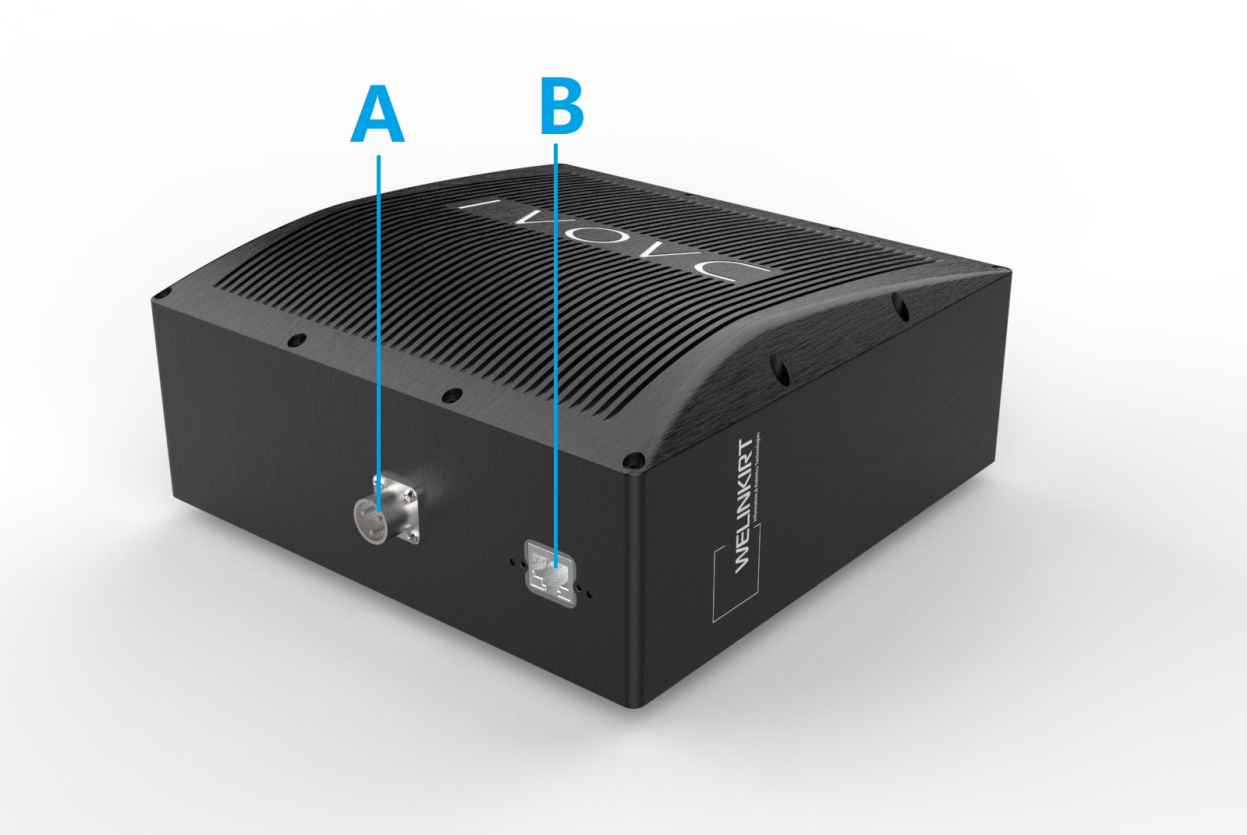
BP AMR



A. Power Connector 24V, 10A DC

B. Ethernet Connector CAT 6 or higher

BP AMR-GPU



A. Power Connector 24V, 10A DC

B. Ethernet Connector CAT 6 or higher

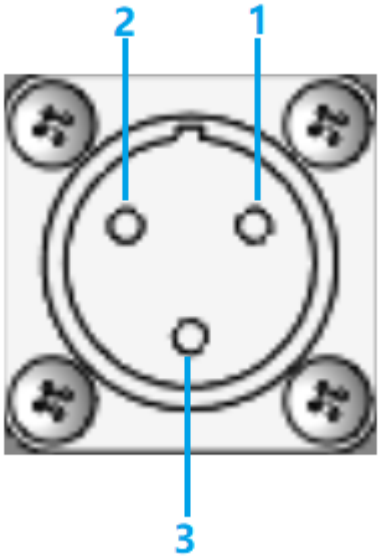
BP LASER

A. Power Connector 24V, 15A DC

B. Ethernet Connector CAT 6 or higher

Power supply interface

BP SMALL

Pinout	Pin	Purpose
	1	DC24V
	2	GND
	3	Reserved, do not connect

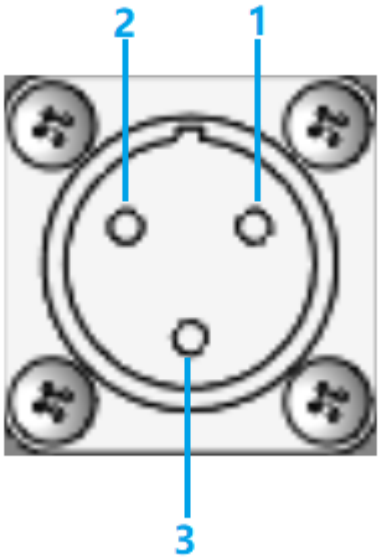
BP MEDIUM

Pinout	Pin	Purpose
	1	DC24V
	2	GND
	3	Reserved, do not connect
	4	Reserved, do not connect
	5	Reserved, do not connect
	6	Reserved, do not connect

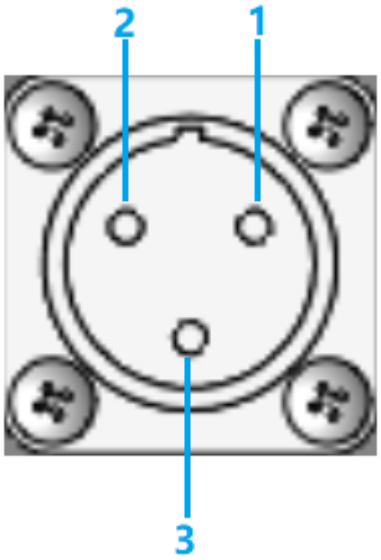
BP LARGE

Pinout	Pin	Purpose
	1	DC24V
	2	GND
	3	Reserved, do not connect
	4	Reserved, do not connect
	5	Reserved, do not connect
	6	Reserved, do not connect

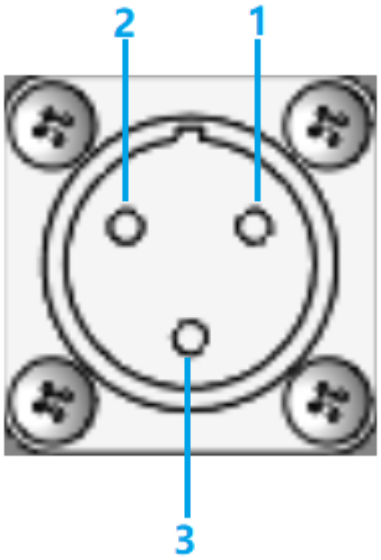
BP AMR

Pinout	Pin	Purpose
	1	DC24V
	2	GND
	3	Reserved, do not connect

BP AMR-GPU

Pinout	Pin	Purpose
	1	DC24V
	2	GND
	3	Reserved, do not connect

BP LASER

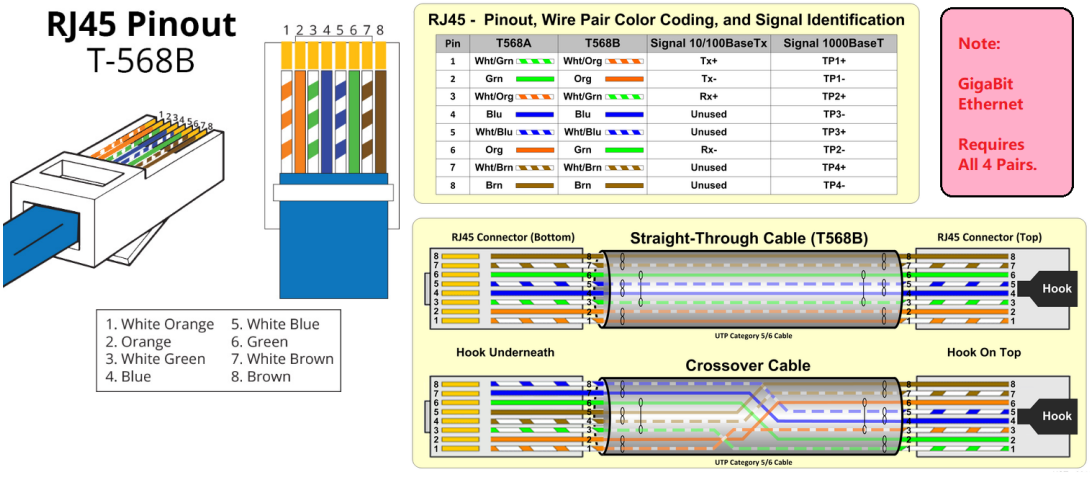
Pinout	Pin	Purpose
	1	DC24V
	2	GND
	3	Reserved, do not connect

Data cable

BP SMALL

BP Small use a ethernet cable for data transmission.

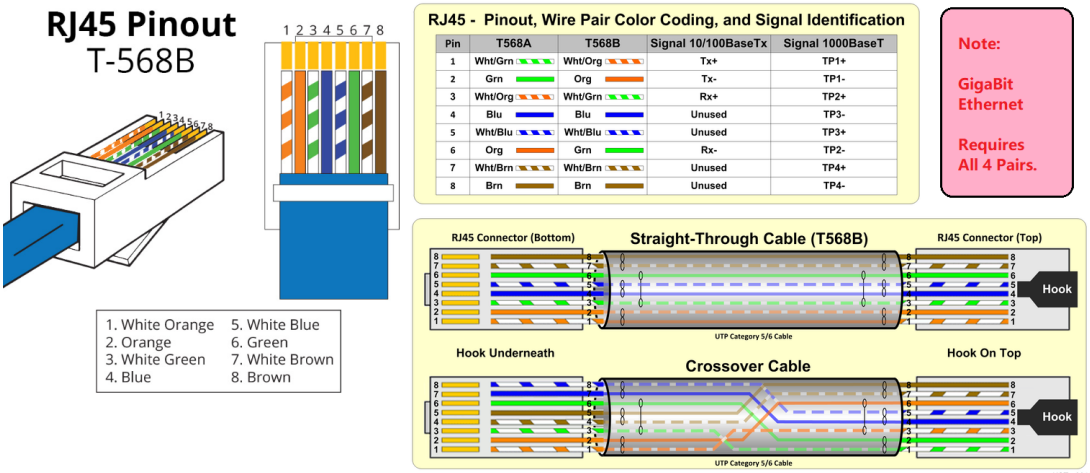
The table below provides the ethernet cable pinout.



BP MEDIUM

BP Medium use a ethernet cable for data transmission.

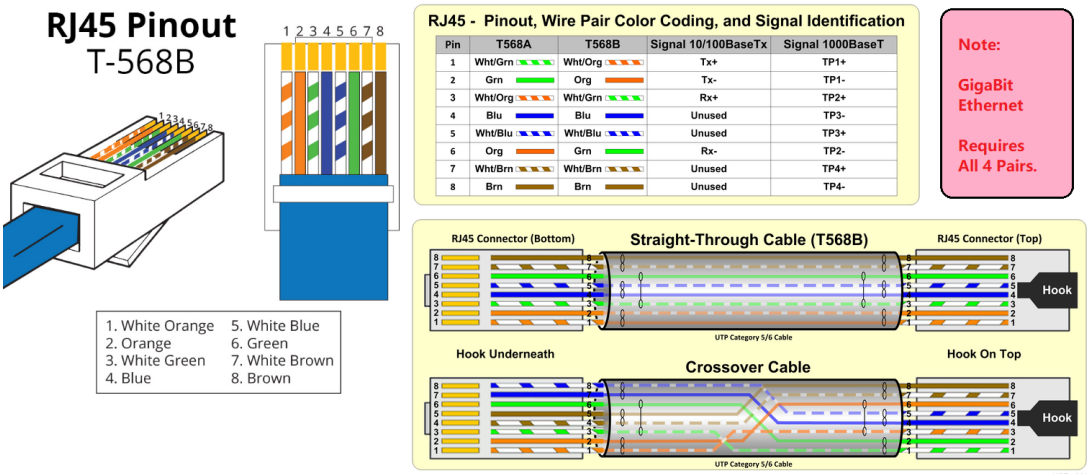
The table below provides the ethernet cable pinout.



BP LARGE

BP Large use a ethernet cable for data transmission.

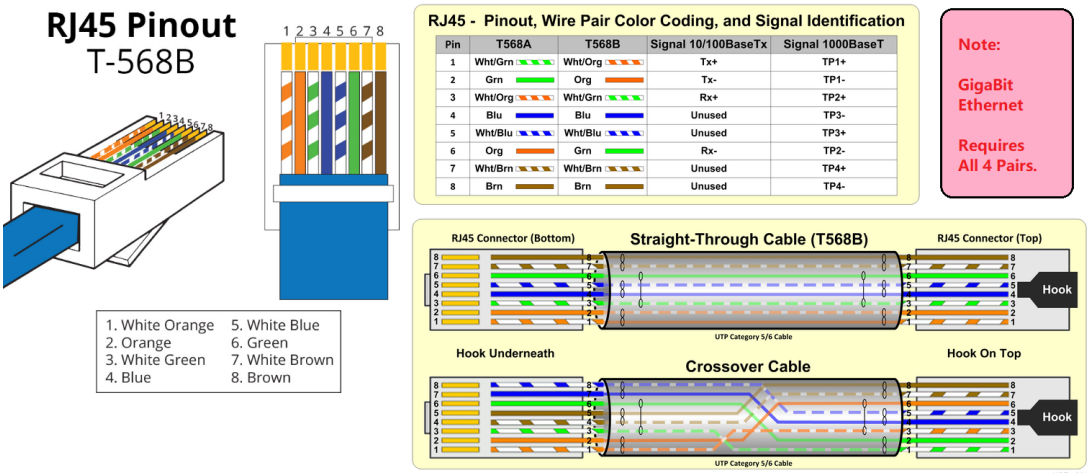
The table below provides the ethernet cable pinout.



BP AMR

BP AMR use a ethernet cable for data transmission.

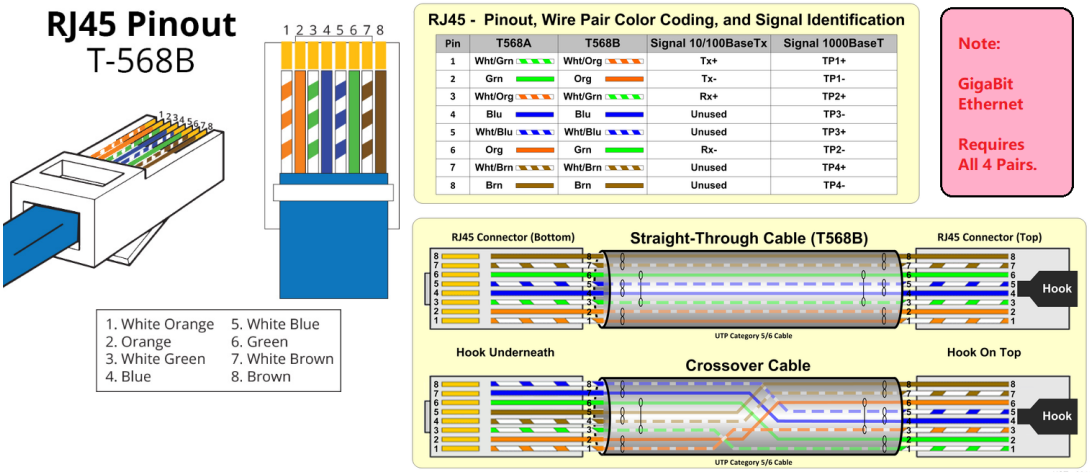
The table below provides the ethernet cable pinout.



BP AMR-GPU

BP AMR-GPU use a ethernet cable for data transmission.

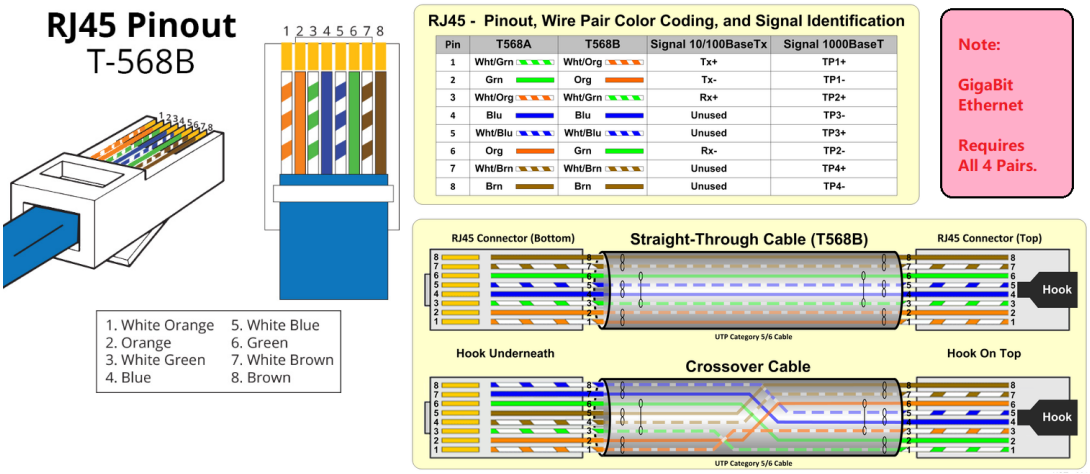
The table below provides the ethernet cable pinout.



BP LASER

BP Laser use a ethernet cable for data transmission.

The table below provides the ethernet cable pinout.



2.4.3 Connecting to the computer

BP SMALL

1. Plug the power supply first into the “24V “
2. Plug the ethernet cable into the camera and connect it to your computer
3. Plug the power supply into a power outlet.

Note: On disconnect, follow the procedure in reverse, disconnect mains power first. Ensure that all connections are screwed in tightly. Check *System Requirements* for performance considerations

---

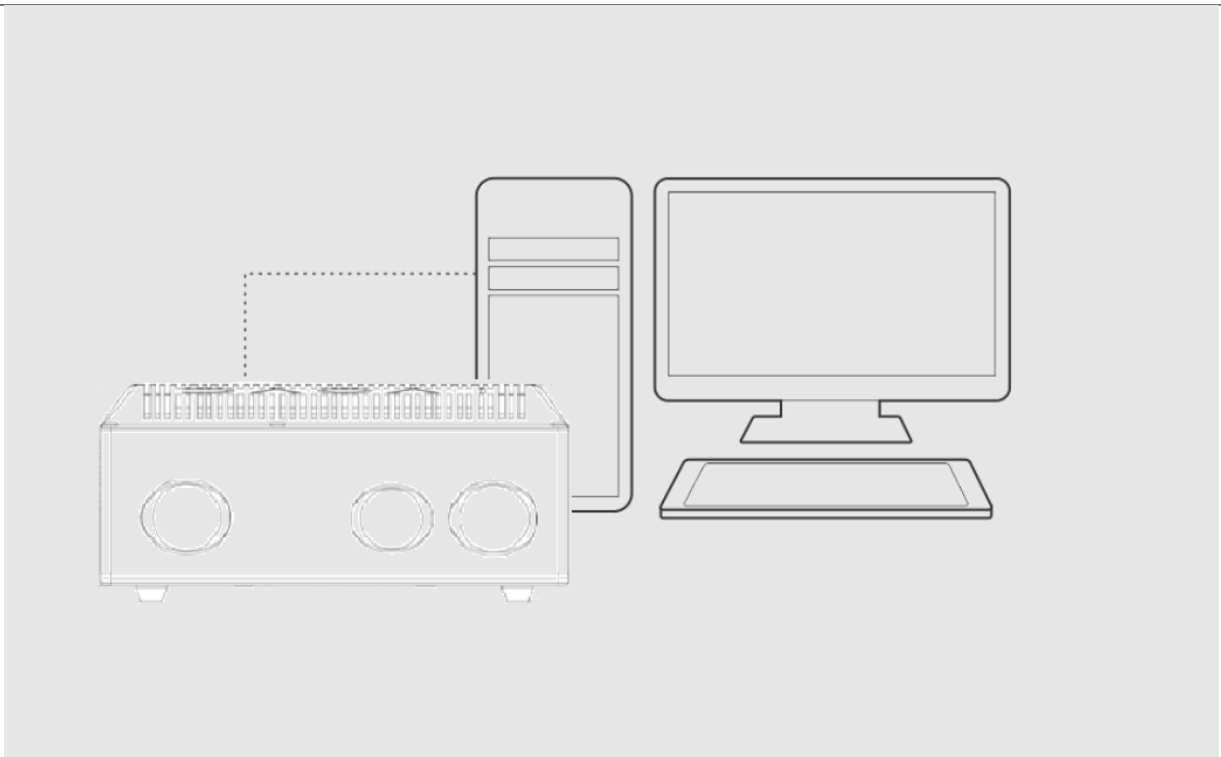
Use the AC/DC adapter supplied with the unit to ensure compliance with emission and immunity standards.

The DaoAI BP Small camera uses Ethernet communication and needs 1 Gbps for performance.

Network Topology

The DaoAI BP Small camera supports the following network topologies:

Direct connection



Continue to *Software Installation* where you will also find Network Configuration.

BP MEDIUM

1. Plug the power supply first into the “24V”
2. Plug the ethernet cable into the camera and connect it to your computer
3. Plug the power supply into a power outlet.

Note: On disconnect, follow the procedure in reverse, disconnect mains power first. Ensure that all connections are screwed in tightly. Check *System Requirements* for performance considerations

---

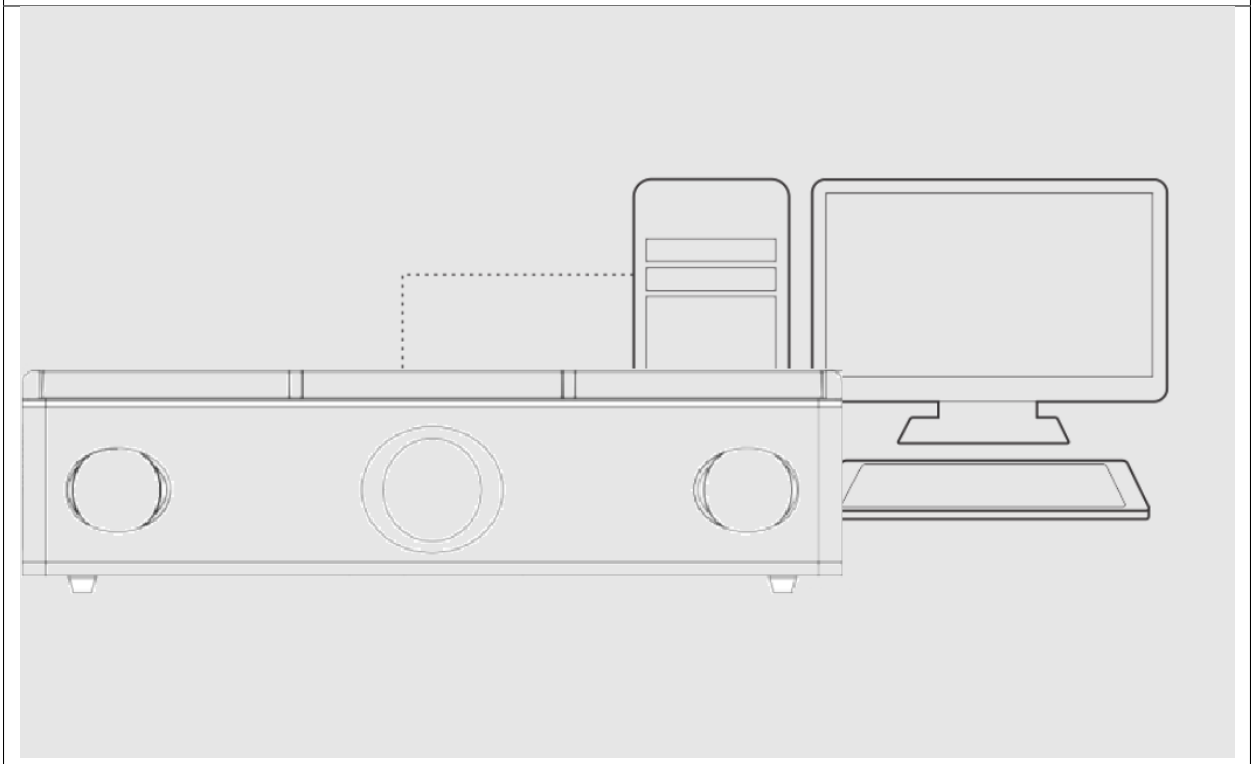
Use the AC/DC adapter supplied with the unit to ensure compliance with emission and immunity standards.

The DaoAI BP Medium camera uses Ethernet communication and needs 1 Gbps for performance.

Network Topology

The DaoAI BP Medium camera supports the following network topologies:

Direct connection



Continue to *Software Installation* where you will also find Network Configuration.

BP LARGE

1. Plug the power supply first into the “24V”
2. Plug the ethernet cable into the camera and connect it to your computer
3. Plug the power supply into a power outlet.

Note: On disconnect, follow the procedure in reverse, disconnect mains power first. Ensure that all connections are screwed in tightly. Check *System Requirements* for performance considerations

---

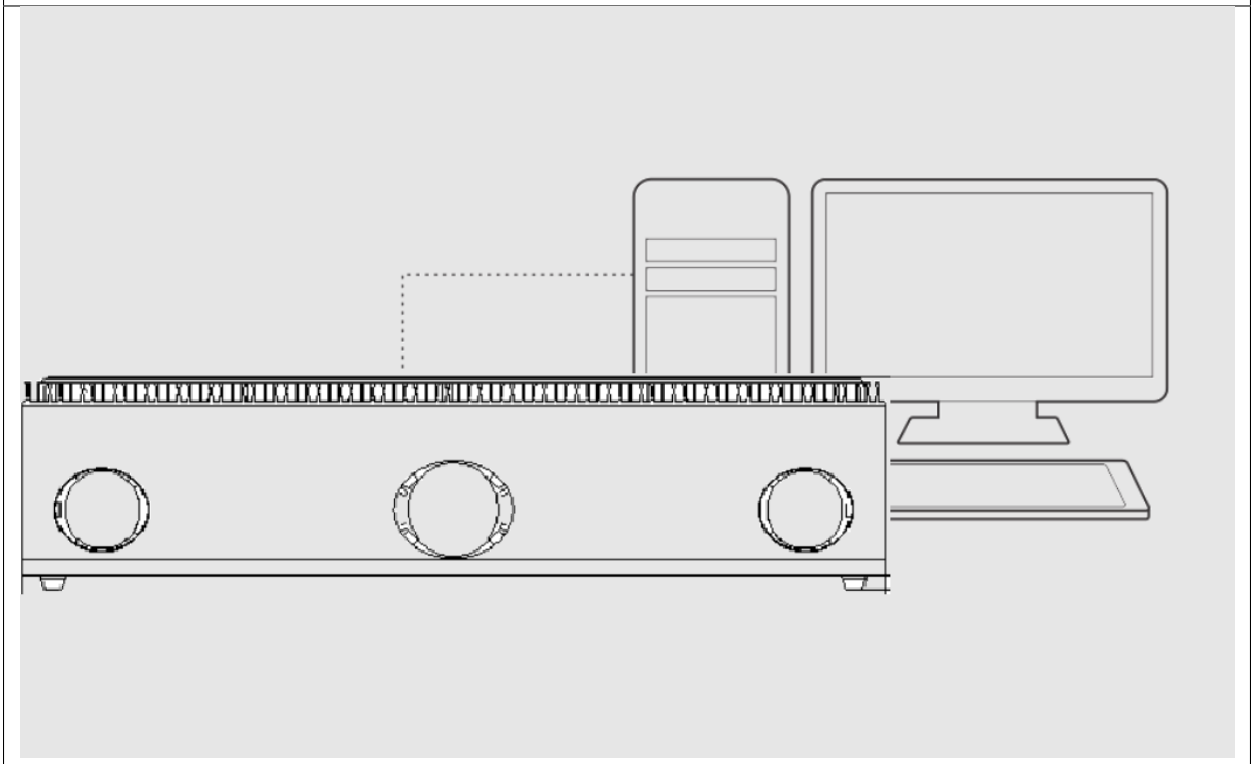
Use the AC/DC adapter supplied with the unit to ensure compliance with emission and immunity standards.

The DaoAI BP LARGE camera uses Ethernet communication and needs 1 Gbps for performance.

Network Topology

The DaoAI BP large camera supports the following network topologies:

Direct connection



Continue to *Software Installation* where you will also find Network Configuration.

BP AMR

1. Plug the power supply first into the “24V”
2. Plug the ethernet cable into the camera and connect it to your computer
3. Plug the power supply into a power outlet.

Note: On disconnect, follow the procedure in reverse, disconnect mains power first. Ensure that all connections are screwed in tightly. Check *System Requirements* for performance considerations

---

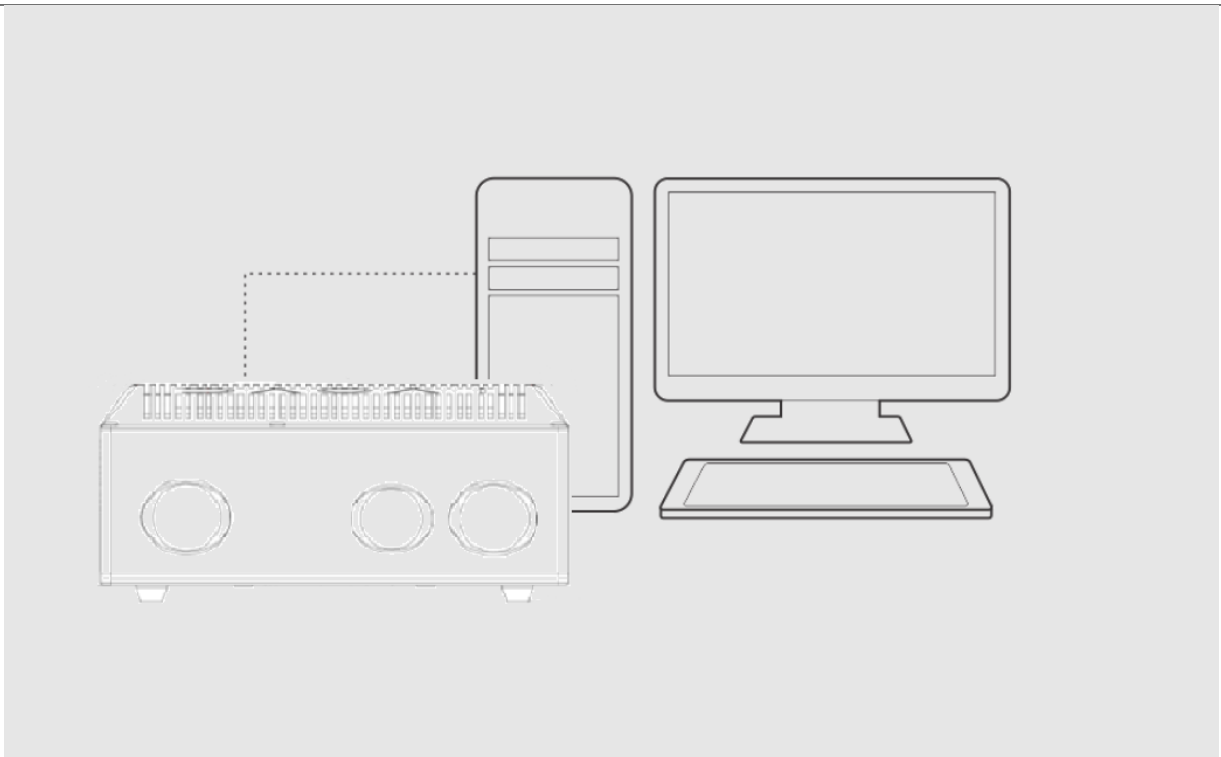
Use the AC/DC adapter supplied with the unit to ensure compliance with emission and immunity standards.

The DaoAI BP AMR camera uses Ethernet communication and needs 1 Gbps for performance.

Network Topology

The DaoAI BP AMR camera supports the following network topologies:

Direct connection



Continue to *Software Installation* where you will also find Network Configuration.

BP AMR-GPU

1. Plug the power supply first into the “24V”
2. Plug the ethernet cable into the camera and connect it to your computer
3. Plug the power supply into a power outlet.

Note: On disconnect, follow the procedure in reverse, disconnect mains power first. Ensure that all connections are screwed in tightly. Check *System Requirements* for performance considerations

---

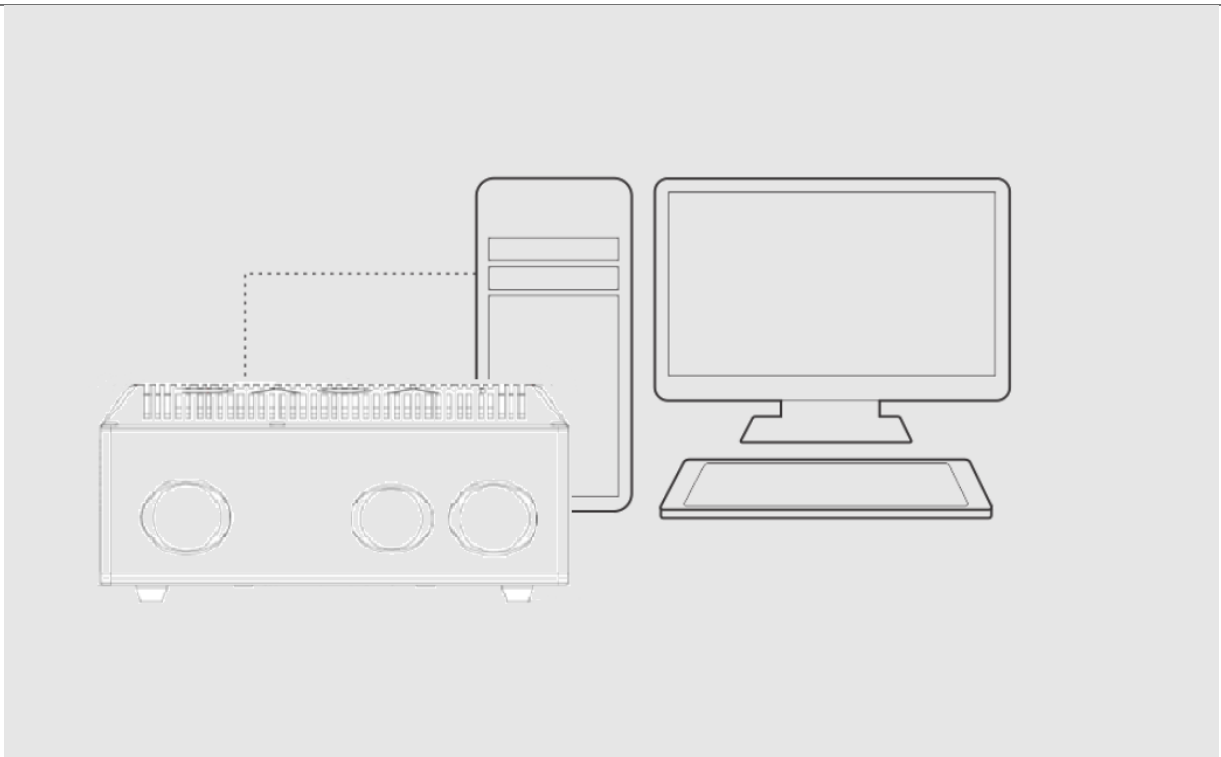
Use the AC/DC adapter supplied with the unit to ensure compliance with emission and immunity standards.

The DaoAI BP AMR camera uses Ethernet communication and needs 1 Gbps for performance.

Network Topology

The DaoAI BP AMR-GPU camera supports the following network topologies:

Direct connection



Continue to *Software Installation* where you will also find Network Configuration.

BP LASER

1. Plug the power supply first into the “24V”
2. Plug the ethernet cable into the camera and connect it to your computer
3. Plug the power supply into a power outlet.

Note: On disconnect, follow the procedure in reverse, disconnect mains power first. Ensure that all connections are screwed in tightly. Check *System Requirements* for performance considerations

---

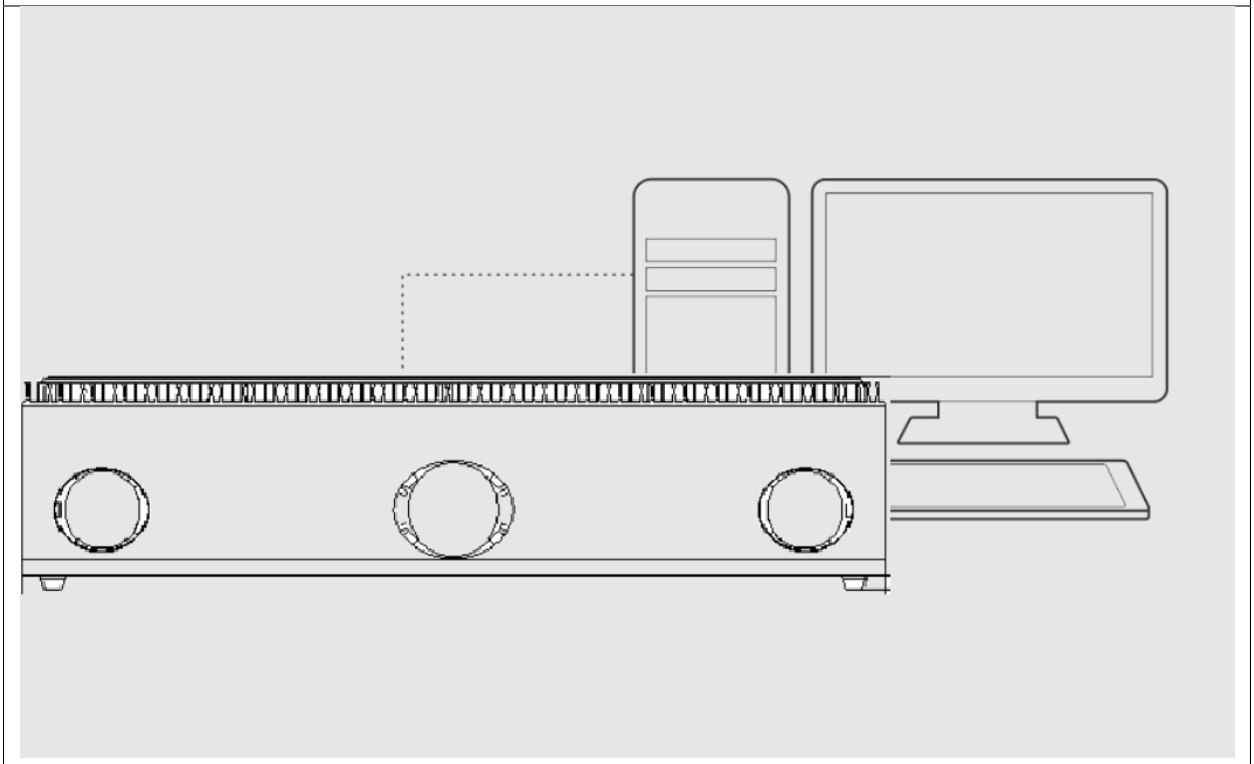
Use the AC/DC adapter supplied with the unit to ensure compliance with emission and immunity standards.

The DaoAI BP LASER camera uses Ethernet communication and needs 1 Gbps for performance.

Network Topology

The DaoAI BP laser camera supports the following network topologies:

Direct connection



Continue to *Software Installation* where you will also find Network Configuration.

2.5 Service and Maintenance

All of the DaoAI Cameras contain no user-serviceable parts inside. The product warranty will be void if opened.

Please follow the instructions below to ensure that your DaoAI camera is well maintained:

- Check screw connections and connectors at regular intervals.
- The cameras use passive cooling, please allow some space around the device for optimal airflow.
- To remove dust or other accumulated particles, it is recommended to use a small vacuum cleaner or a small canister of compressed air. This applies to both the glasses and in between the ribs of the heat sink.
- Regularly clean optical glass parts of the device. Please refer to *Cleaning Instructions* for more details.
- Check and update the calibration on a yearly basis by performing Infield Correction (and Hand-Eye Calibration) when necessary.

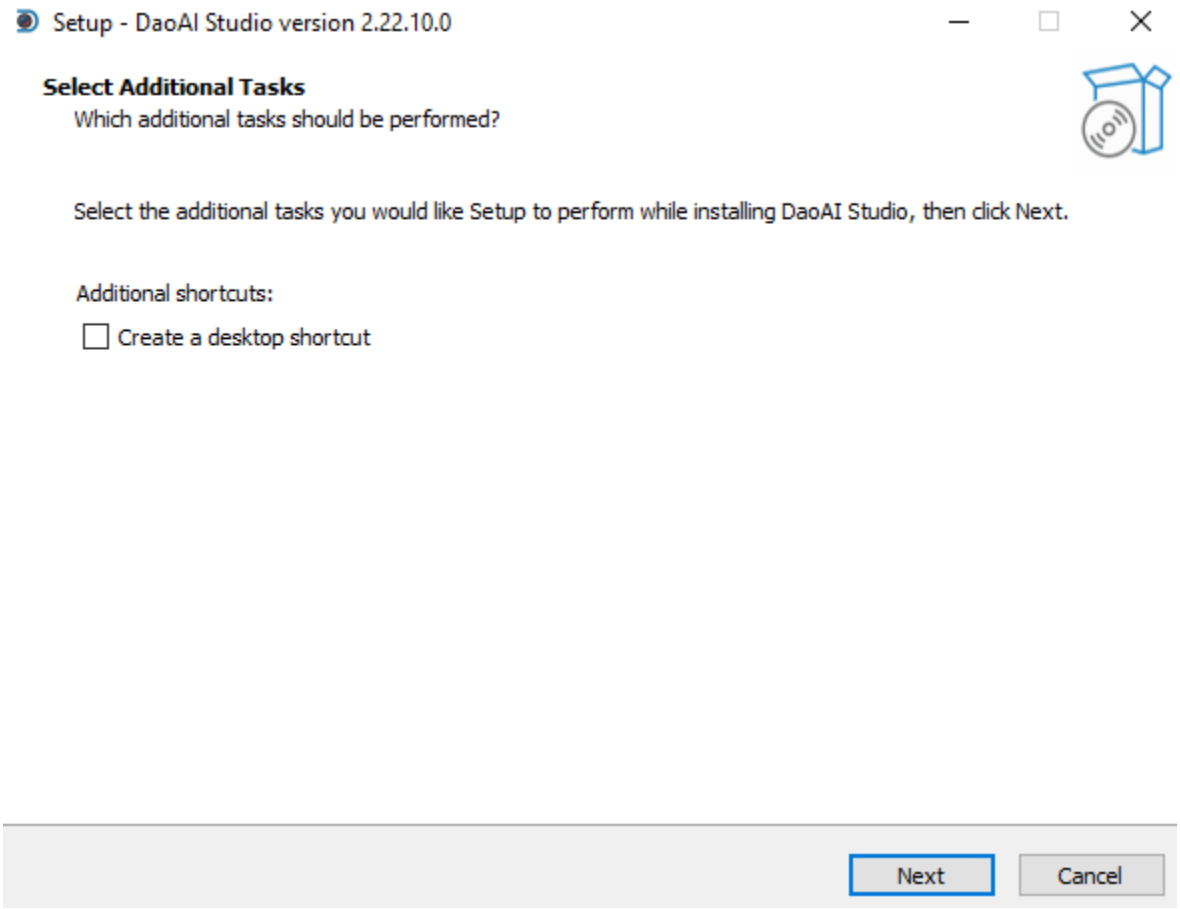
2.6 Software Installation

Before Installing the DaoAI Camera Studio Software:

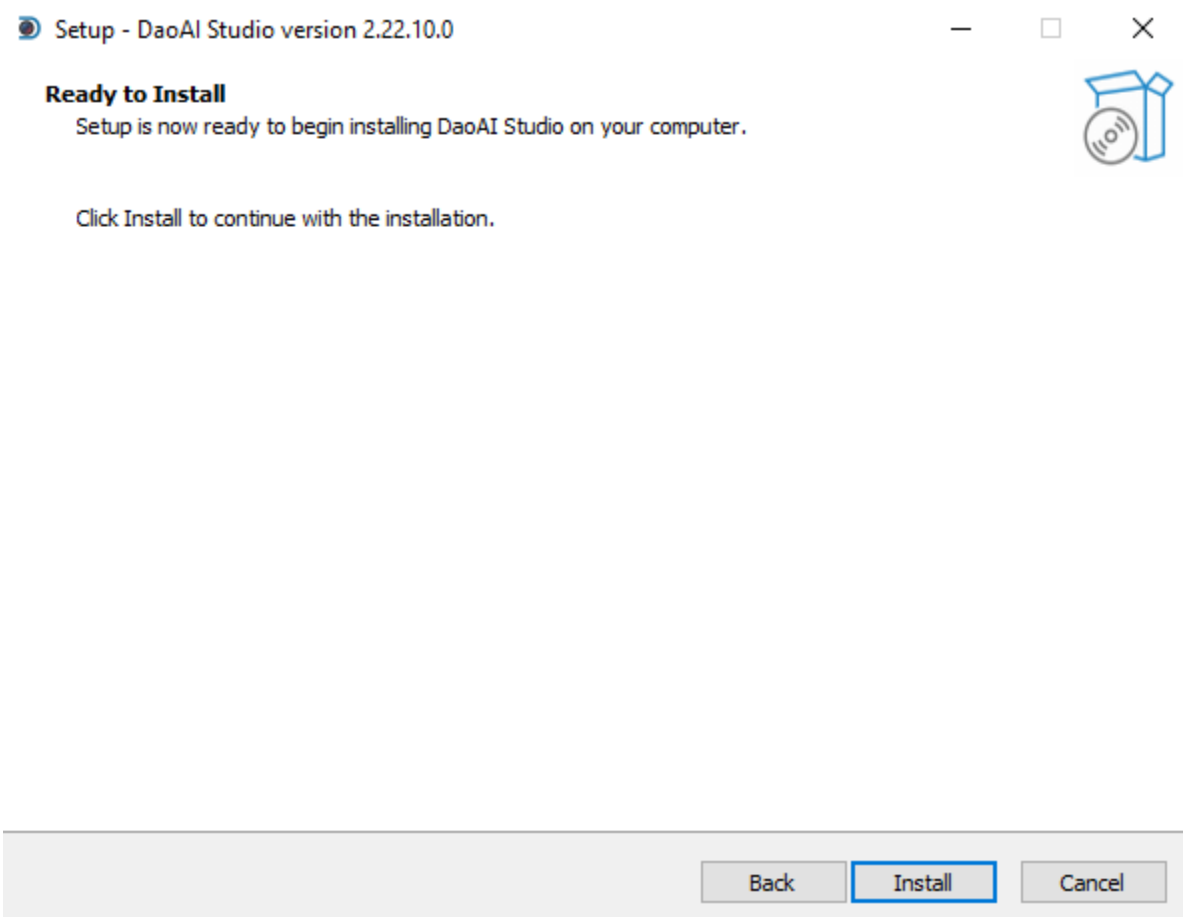
- Check the GPU Requirements (GTX 1050 Ti)
- Check GPU driver is up to date

Installation Steps:

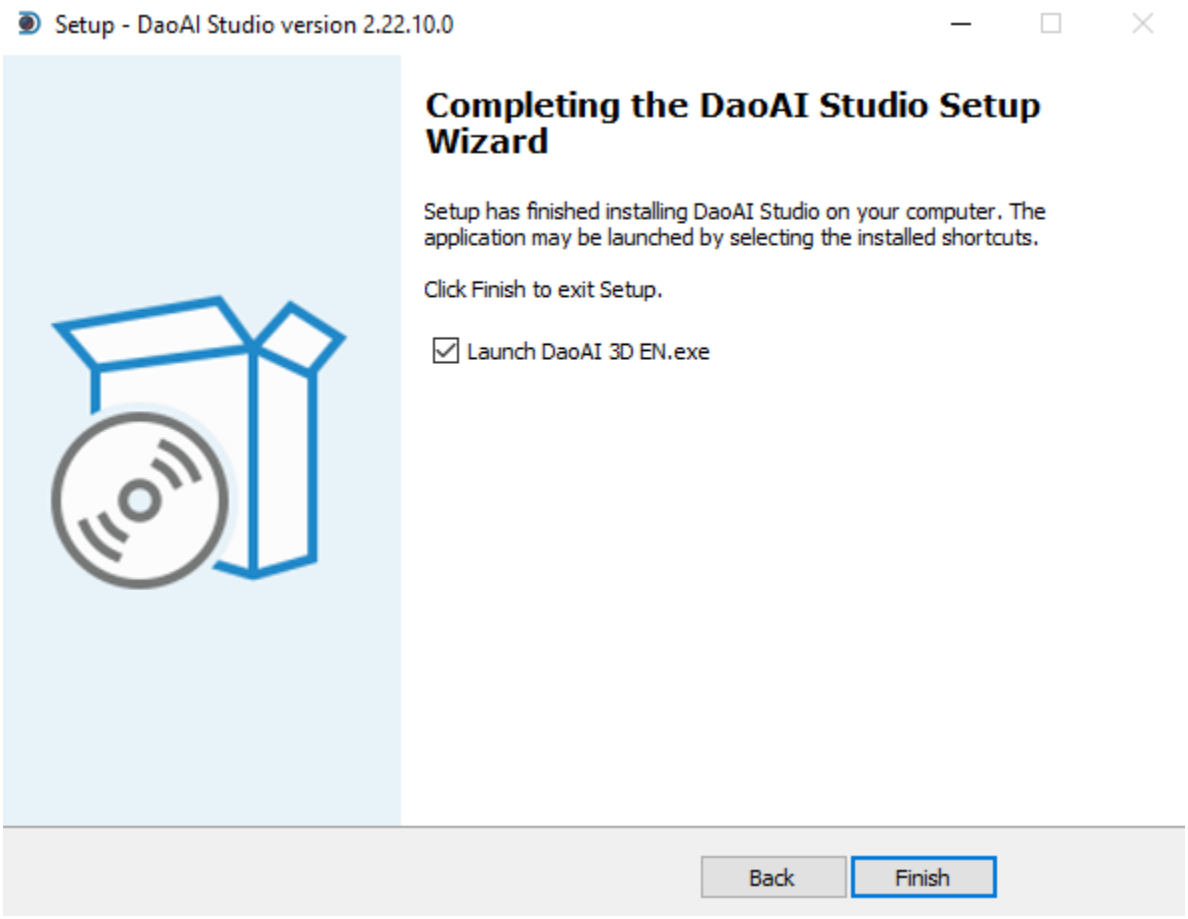
1. Download the latest DaoAI Camera Studio Software Installer.
2. Run the full installer that starts with DaoAI_Studio. (e.g. DaoAI_Studio_2.22.10.0_103_full.exe)
3. Follow the steps. You can customize the settings as you see fit.



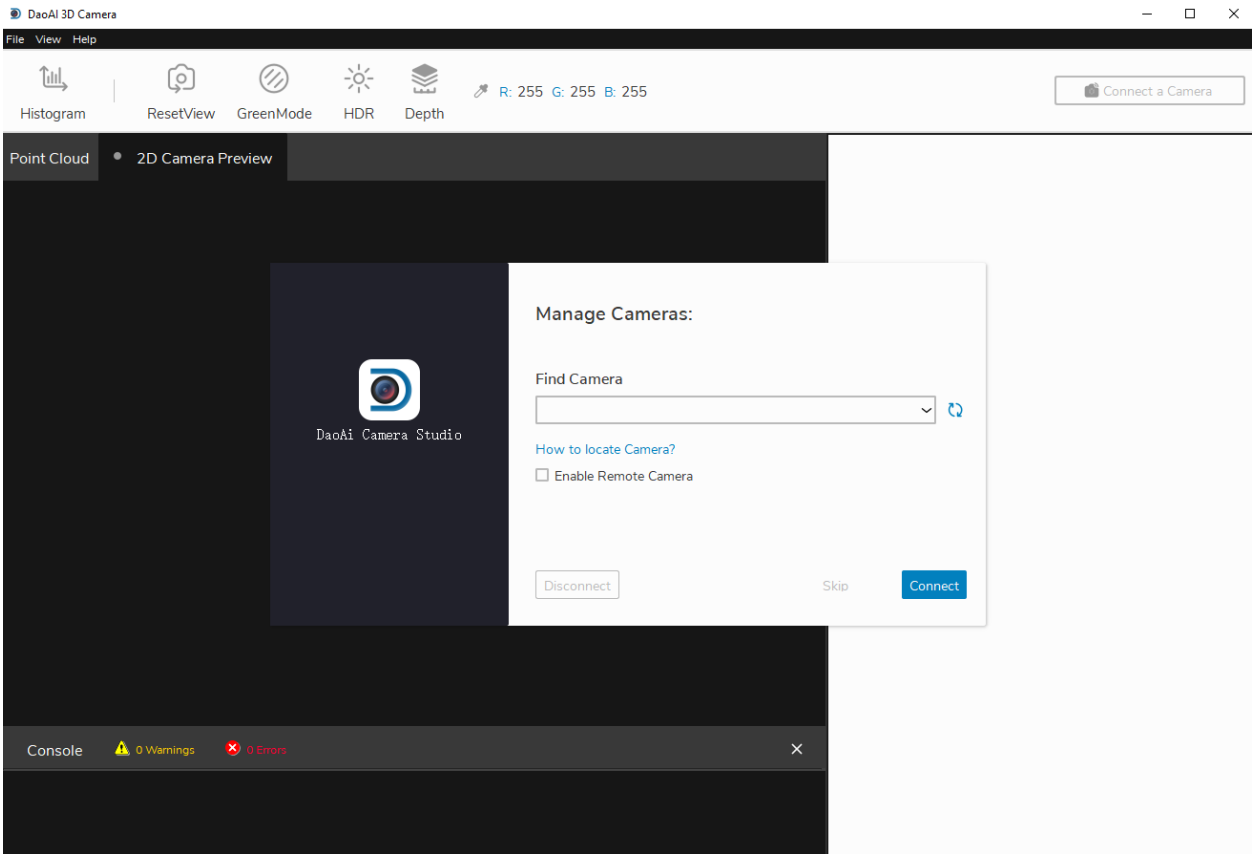
4. Click “Install”, the the installation will begin. Note that the installation could take up to a few minutes to complete.



5. Click “Finish”, and the software installation process is completed.



6. If you have the "Launch DaoAI 3D EN.exe" checkbox selected (default), you will be greeted with the DaoAI Camera Studio startup window.



2.6.1 Network Configuration

Default Configuration

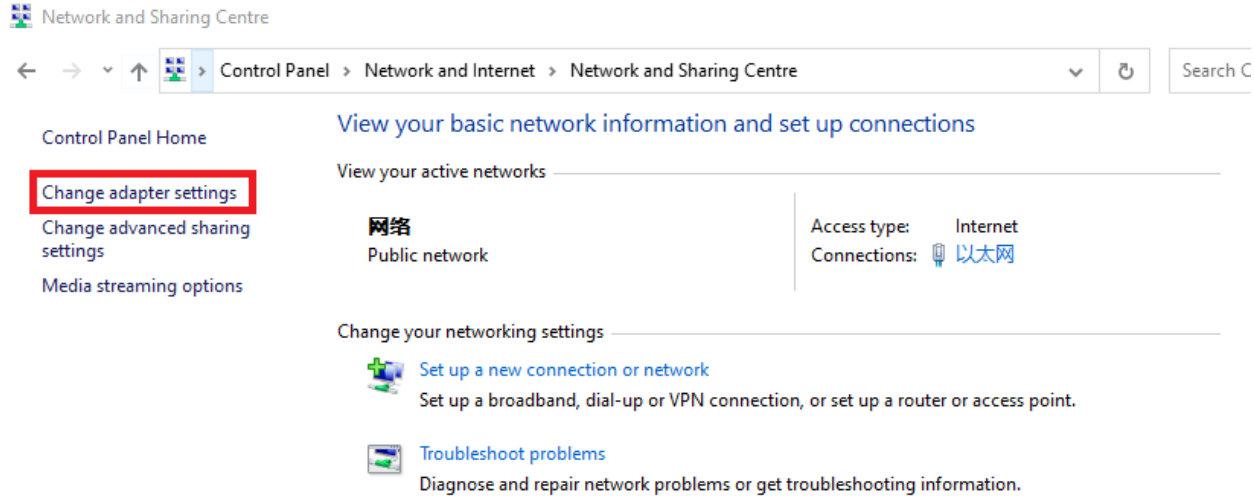
	IP	Subnet mask
BP-AMR	192.168.1.12	255.255.255.0
Other BP Series	192.168.1.2	255.255.255.0

Any IP address in the subnet range is valid (e.g. For BP series camera: 192.168.1.0 - 192.168.1.255).
If you have not configured the camera IP address before, the camera's IP address will be the default IP address as in the above table.

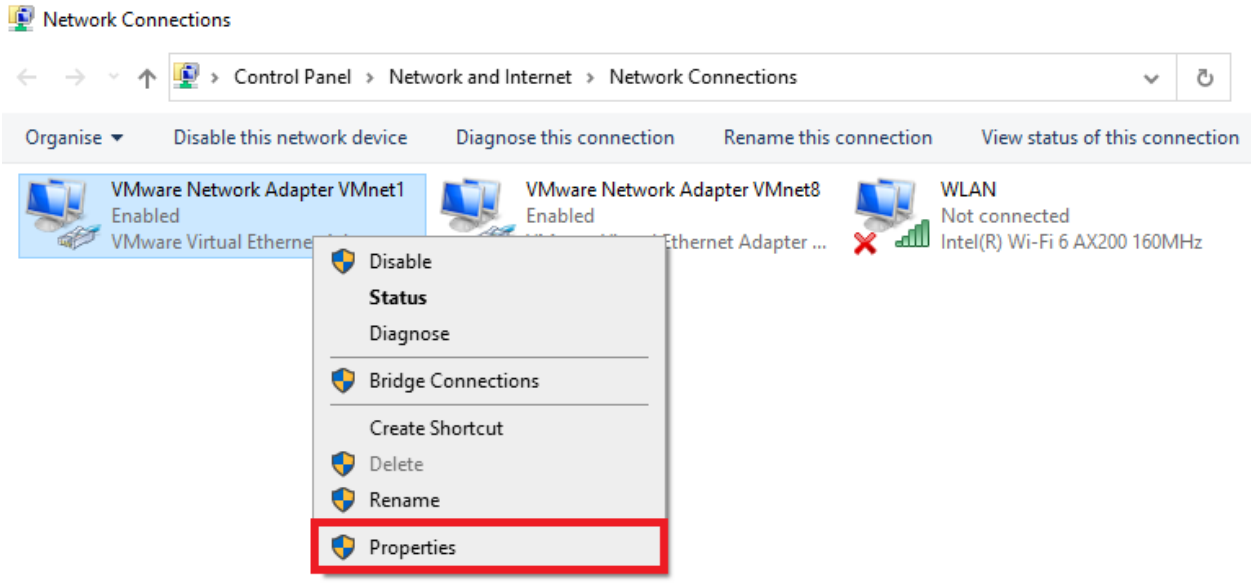
Static IP Network Configuration - PC

Navigate to:

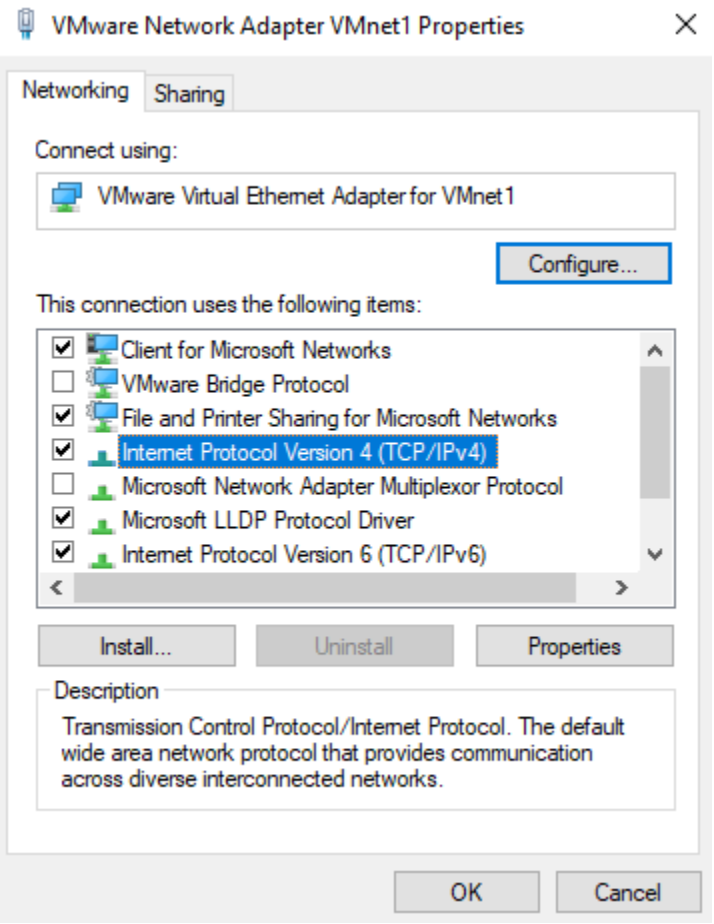
Control Panel → Network and Internet → Network and Sharing Center → Change Adapter Settings.



Right click → Properties.



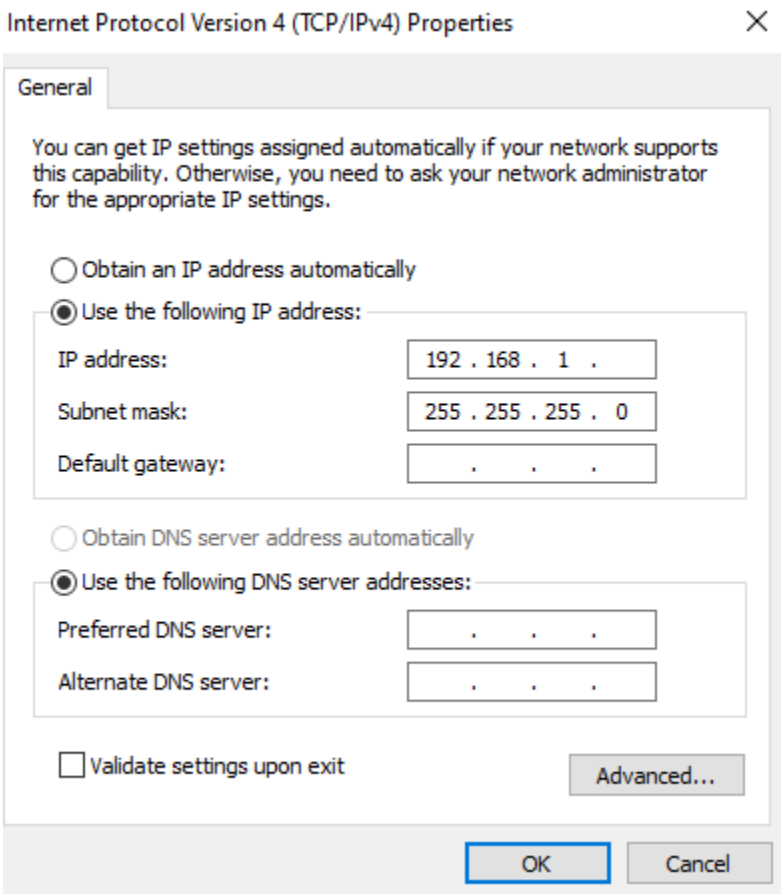
Double click Internet Protocol Version 4 (TCP/IPv4).



Select **Use the following IP address** → set the IP address to 192.168.1.x.

If this is the first time you are setting up the camera, the IP address should be any number other than 2 (e.g. enter 192.168.1.9).

Enter Subnet mask: 255.255.255.0



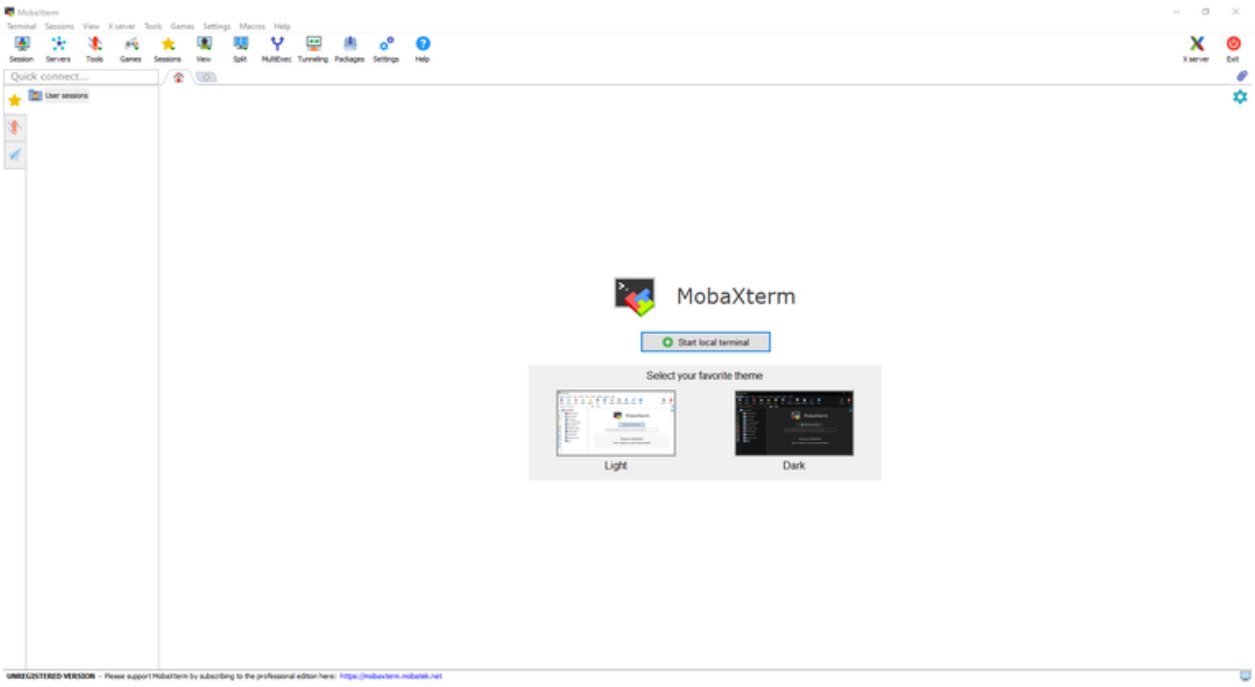
Click **OK** to finish configuration.

Note: Please make sure the IP addresses of all connected cameras are different from each other, otherwise you may run into problem when trying to connect *multiple cameras*.

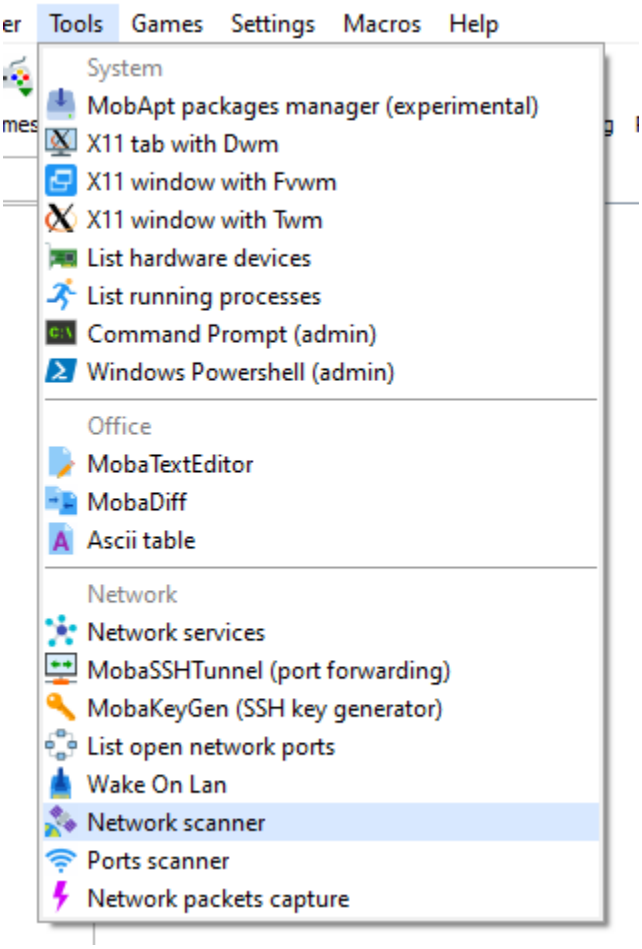
Check Cameras' IP Addresses

If you don't know a camera's IP address, you can use MobaXterm to check.

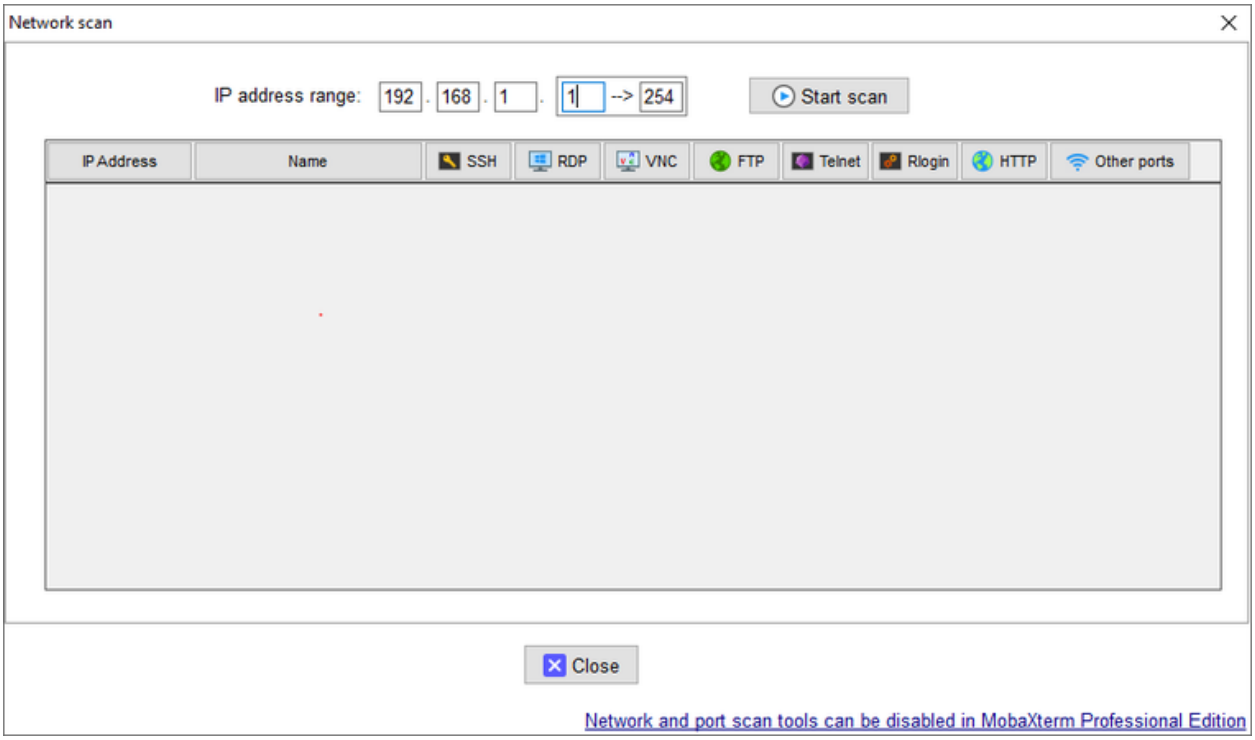
Open MobaXterm.



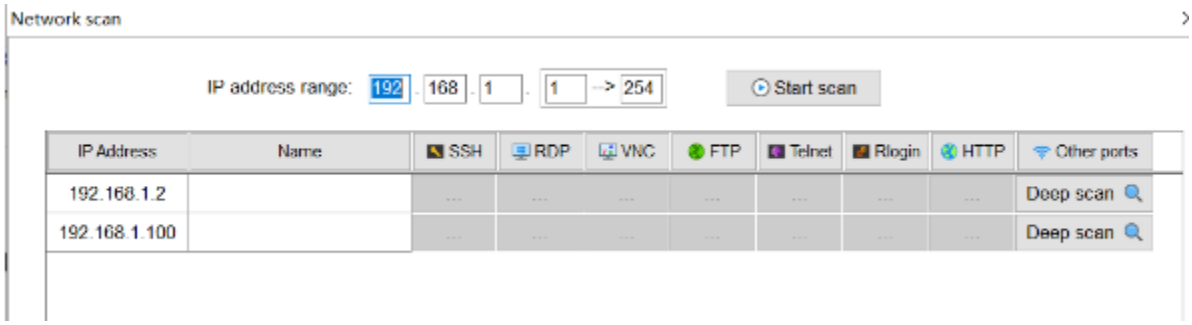
From **Tools** dropdown list, select **Network Scanner**.



Input **192.168.1.1 –> 254** in **IP Address Range**, then click **Start Scan**.



After scanning, it will list all discoverable IP addresses in the local network.



CHAPTER

THREE

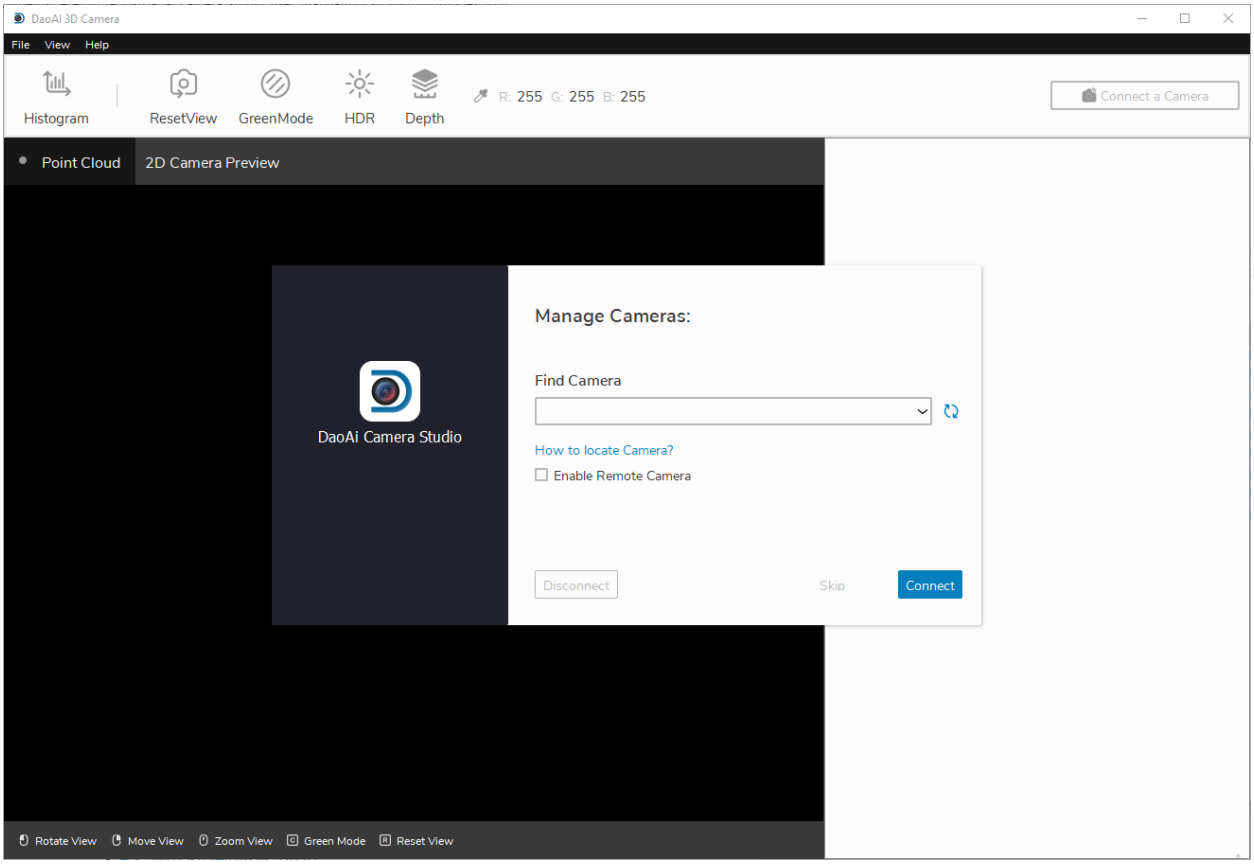
CAMERA STUDIO GUIDE

3.1 Connecting & Disconnecting Camera

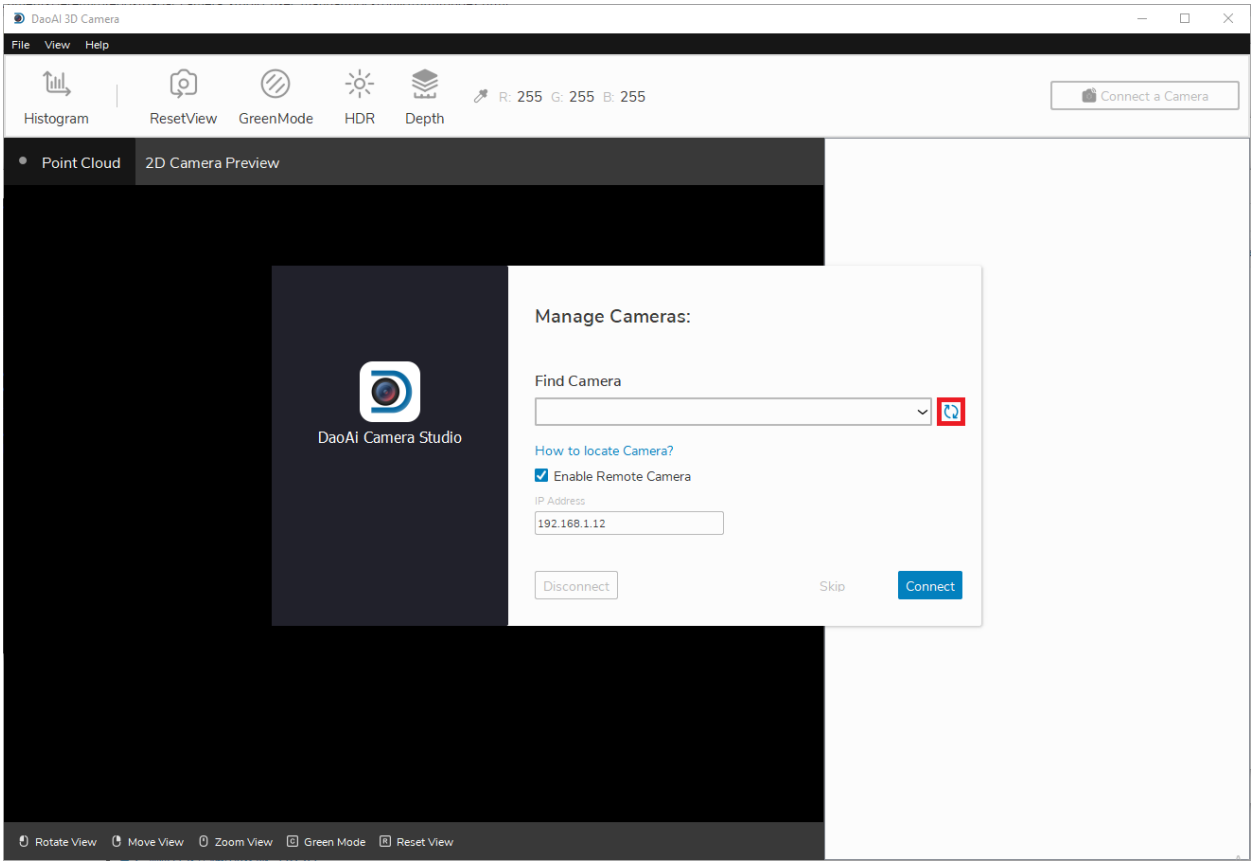
The first step of using the DaoAI Camera Studio is to connect your cameras.

3.1.1 Connecting

When starting up the Camera Studio, at first you will see the **Manage Cameras** window.



The refresh button updates the list of connected cameras.



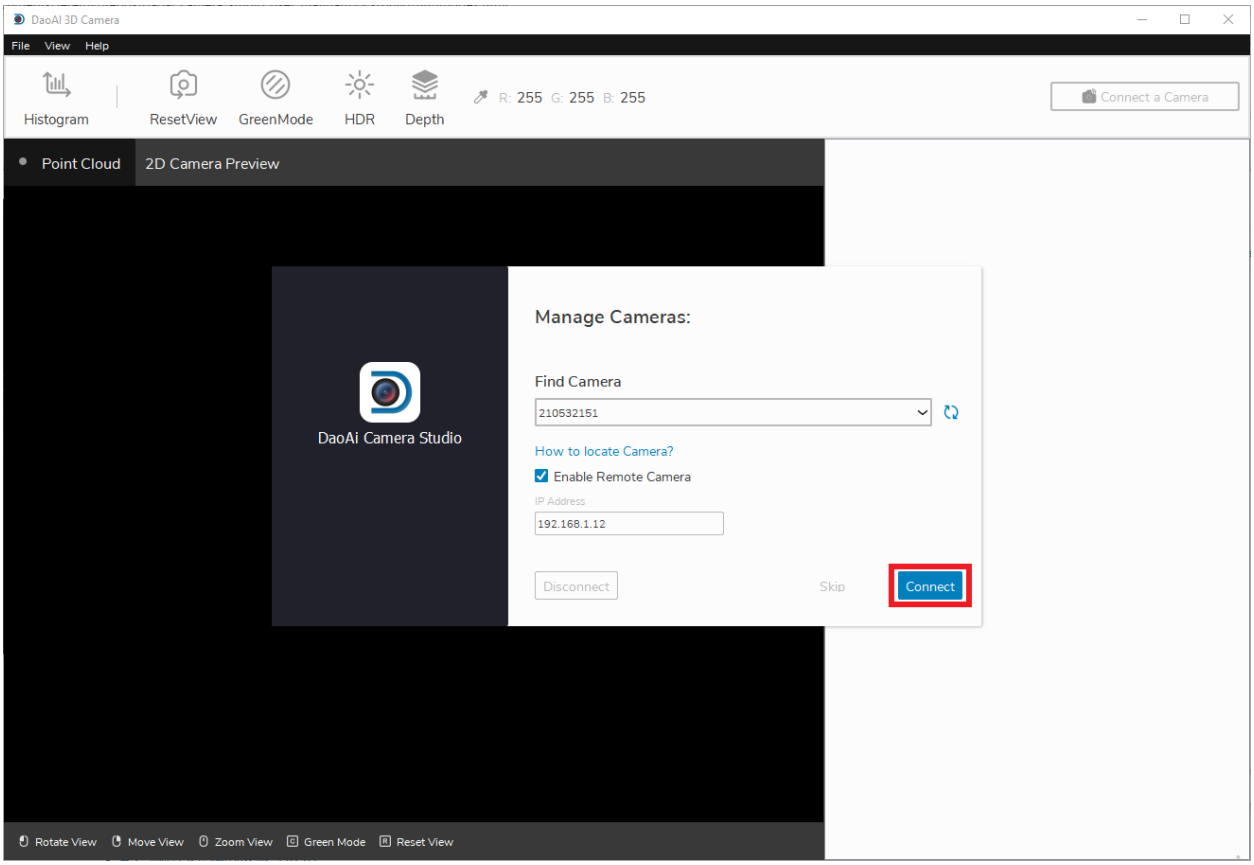
Note: If you are using a remote controlled camera, you will have check the *Enable Remote Cameras* checkbox and specify the camera's IP address first before clicking refresh.

Default IP for Remote Camera

- **192.168.1.2**: BP-L camera, BP-M camera, BP-S camera, BP-AMR-GPU camera, and IN cameras
- **192.168.1.12**: BP-AMR camera

If you still can't find your camera, check out

Once you detect the camera you wish to connect to, click the **Connect** button. If there are multiple cameras connected, you can choose which camera to connect to from the dropdown list.

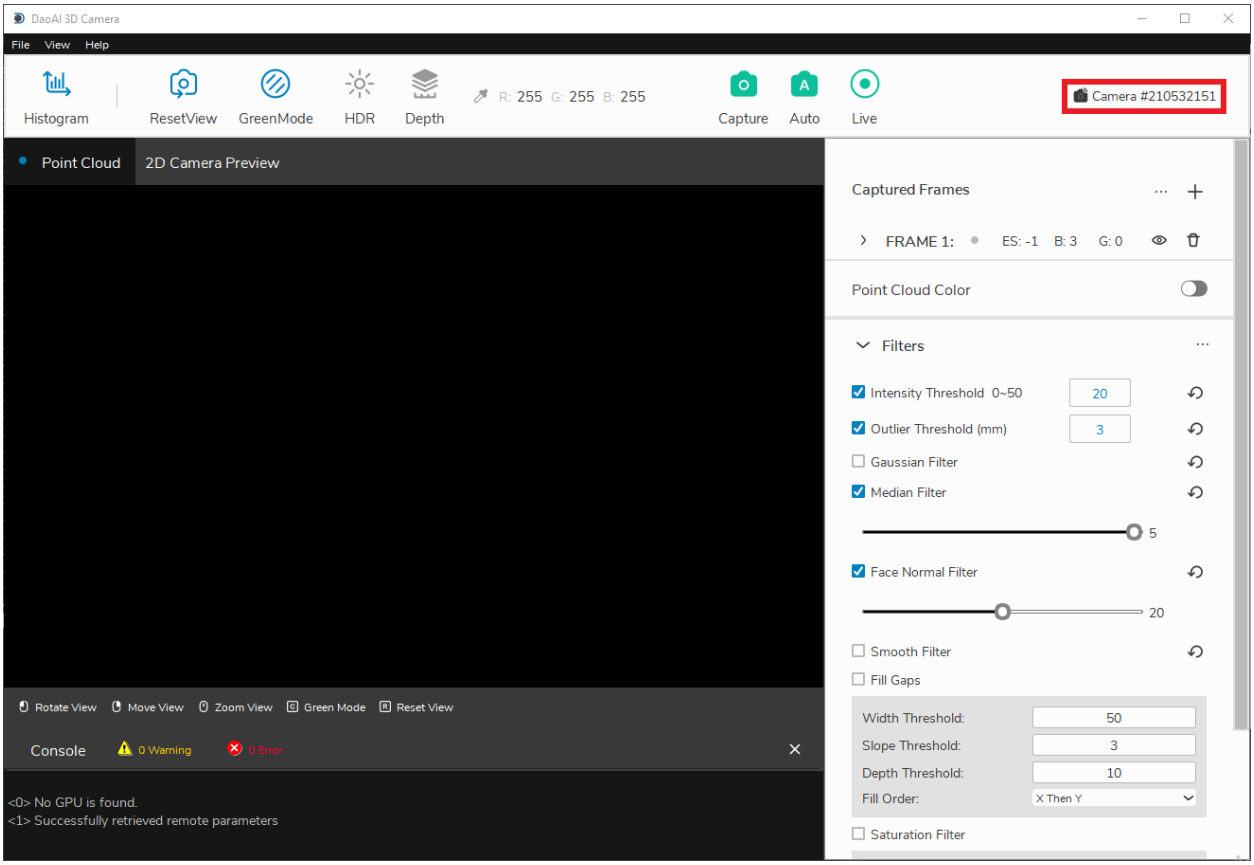


After connecting, you should see the main window.

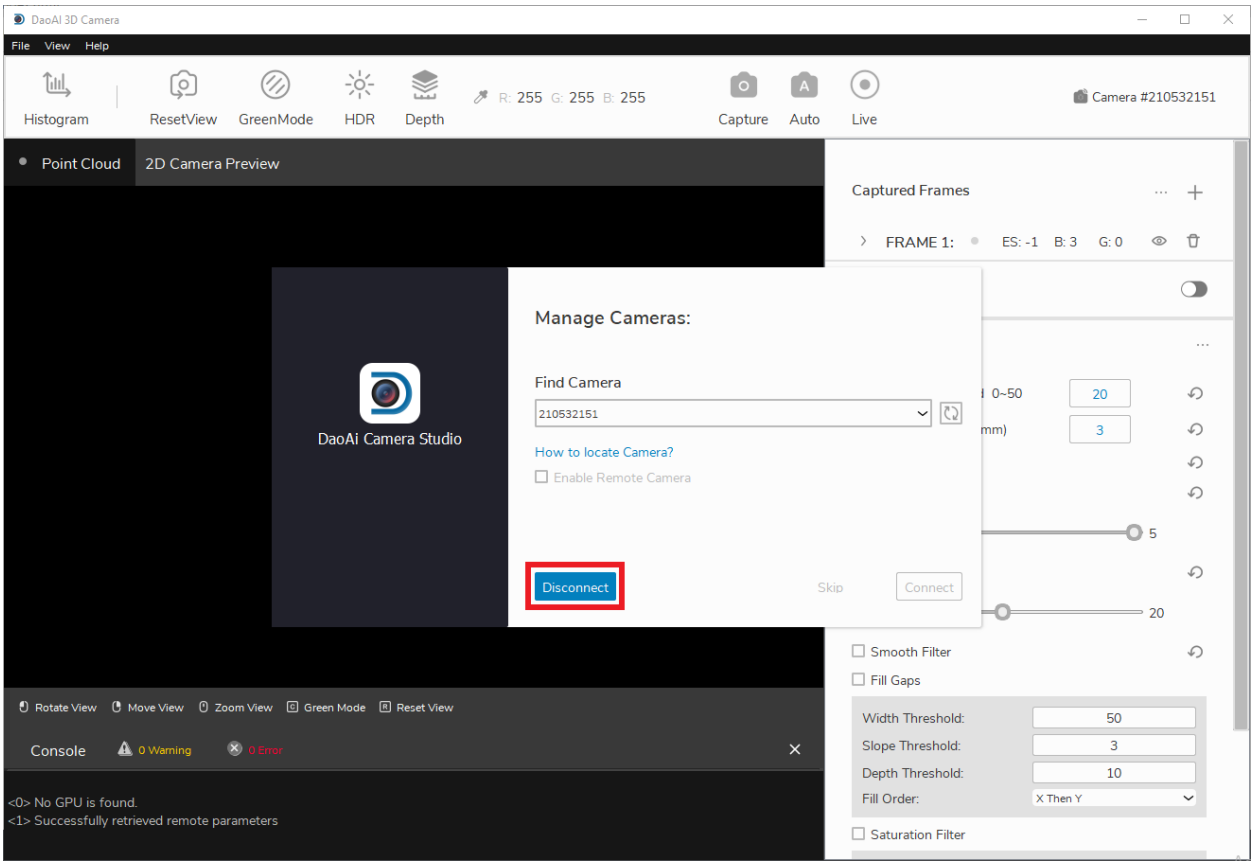
Note: Normally, if multiple cameras are physically connected, they will all appear in the camera selection list. However, DaoAI Camera Studio only supports establishing a connection with a single camera at a time. To capture with multiple cameras using DaoAI Camera Studio, please start another instance of DaoAI Camera Studio and connect the other camera.

3.1.2 Disconnecting

To disconnect your camera, first click the camera ID in the top right area of the main window.



From there, you should see the **Manage Cameras** window, where you can click disconnect.



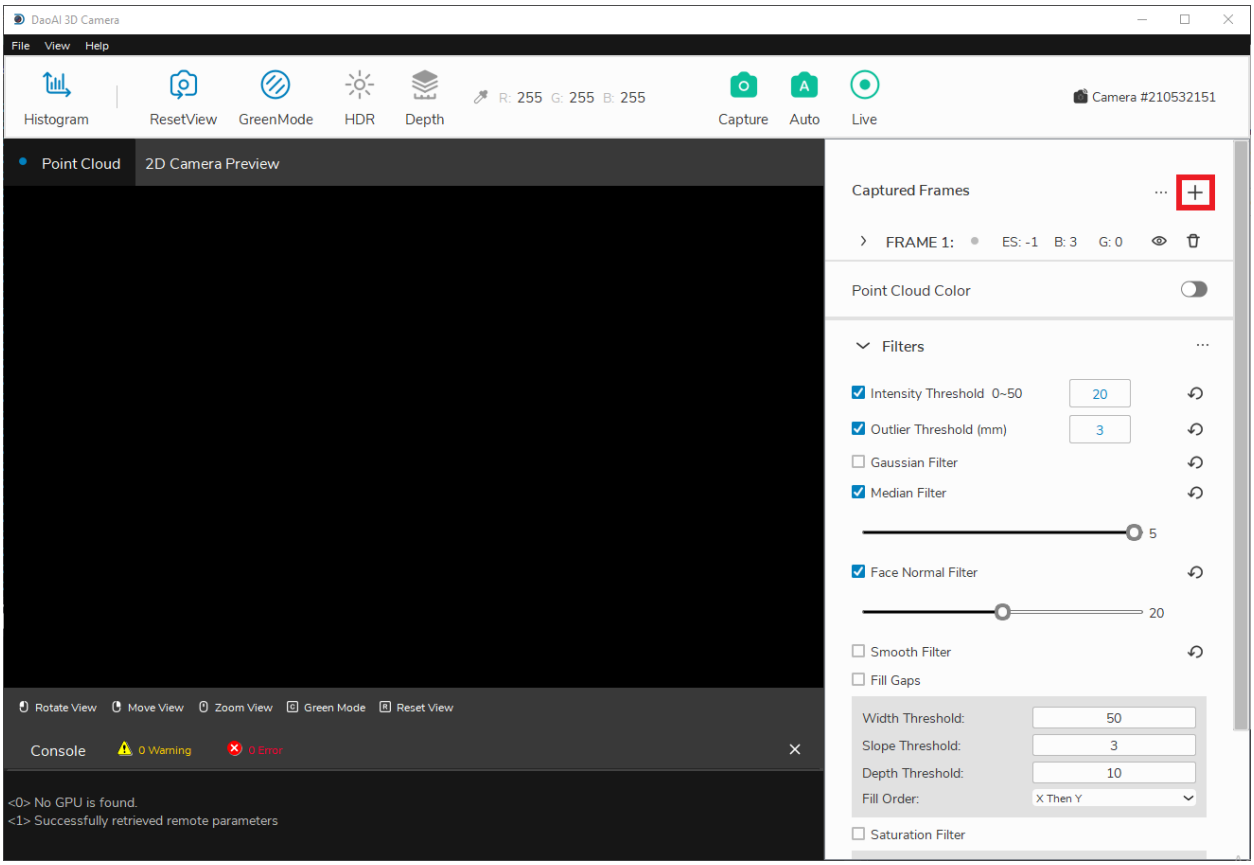
3.2 Adding & Editing Frames

This page will provide instructions on how to add frames.

Adding additional frames

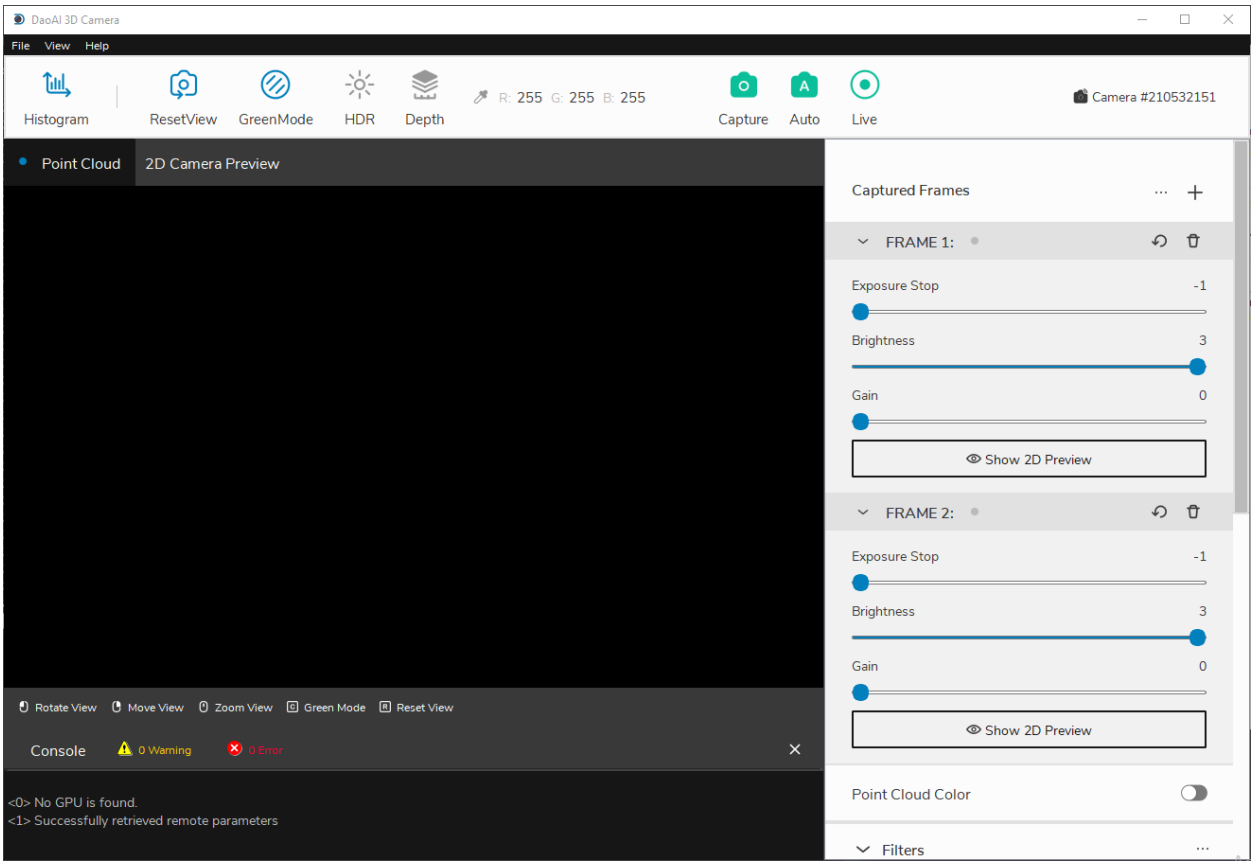
Frames are used to control the capture settings, allowing you to alter properties like exposure stop, brightness, and gain. By adding additional frames, you can capture multiple sets of images for each capture.

In the right panel of the main window, you should see the existing frames. If you haven't made any changes, there should be just one frame. To add another frame, click the plus **+** button.



Editing Frame Parameters

You can edit the settings of the existing frames by clicking their drop downs and adjusting the sliders.

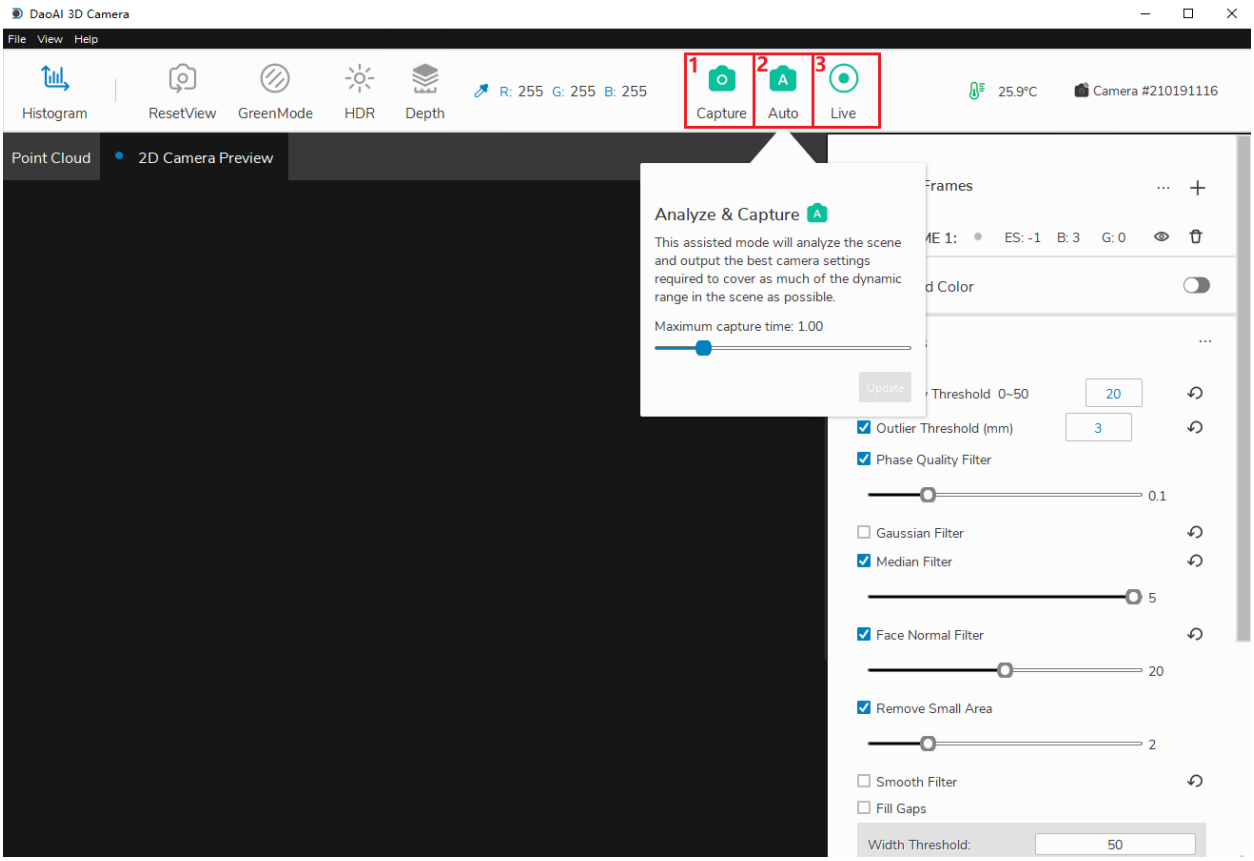


After you have all the frames you need, you can start doing captures using these different frame settings.

3.3 Capturing Images

This page will provide instructions on how perform captures in Camera Studio.

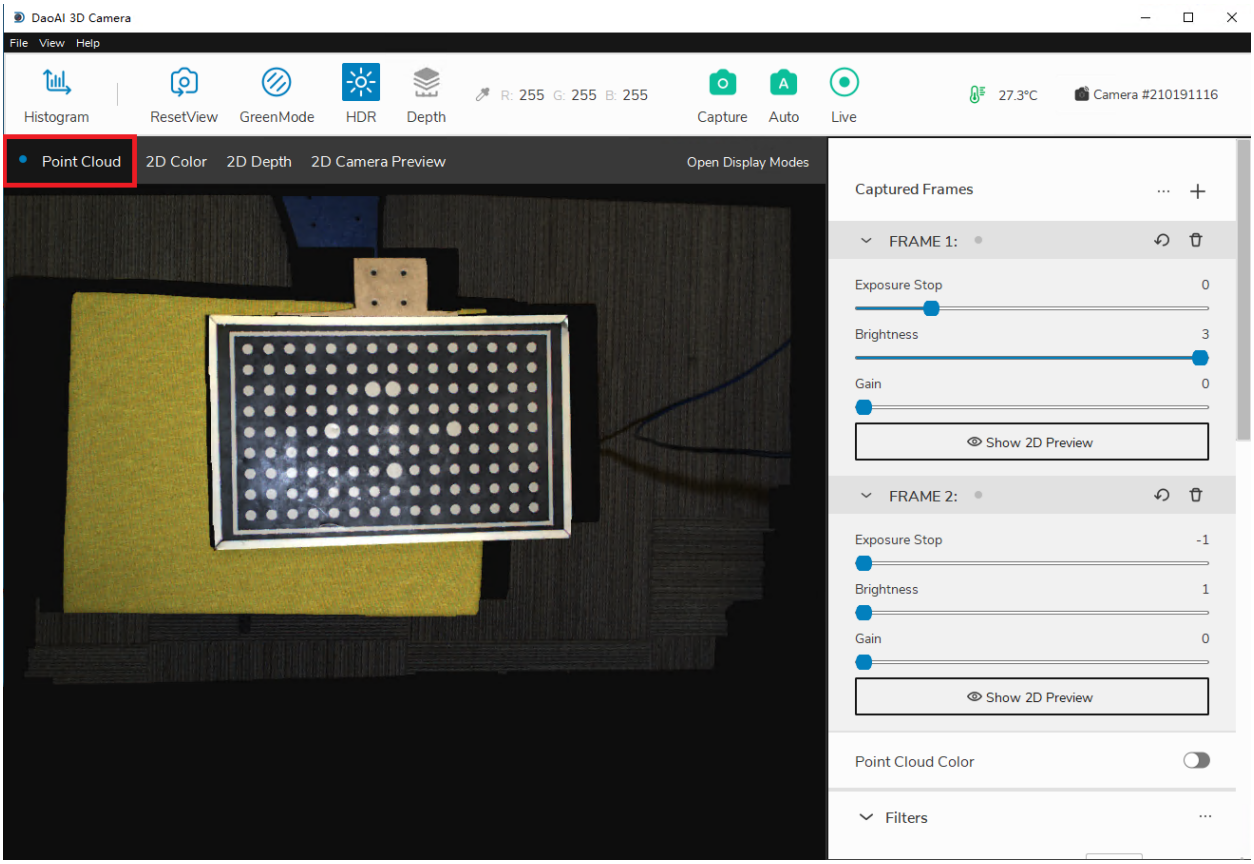
There are three capture modes to choose from in Camera Studio:



1. **Capture** performs a single capture using the current frames.
2. **Auto** Analyze the scene and automatically generate frame settings based off “Maximum capture time”, then perform a single capture.
3. **Live** performs continuous captures using the current frames.

3.4 Saving Point Clouds

This page will provide simple instructions for saving point clouds in DaoAI Camera Studio.



1. Capture an image (see *Capturing Images*).
2. Select the "Point Cloud" tab to view the captured point clouds (under the "Histogram" button).
3. Select "Files" (the first button in the upper-left corner).
4. Select "Save 3D data" from the File dropdown menu to save the point clouds.

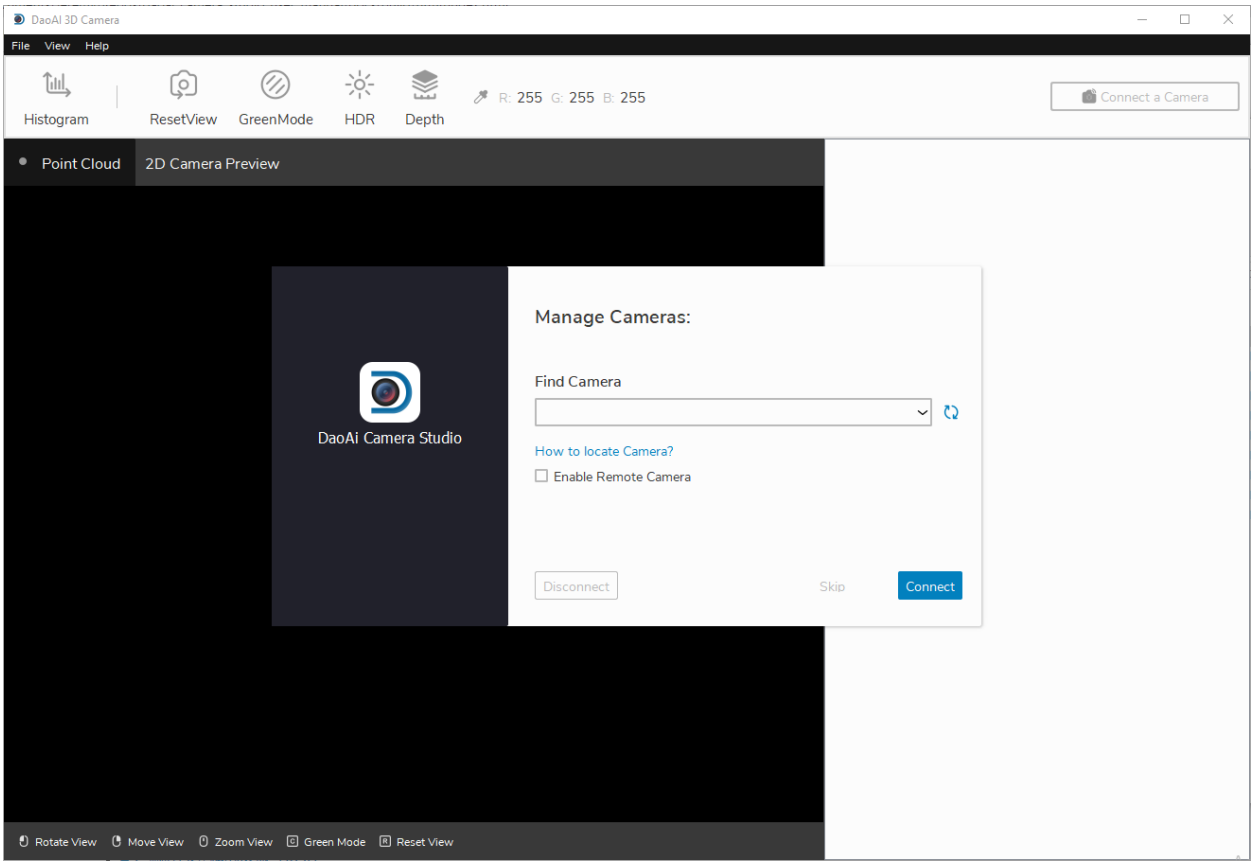
3.5 Control Panel

3.5.1 Connecting & Disconnecting Camera

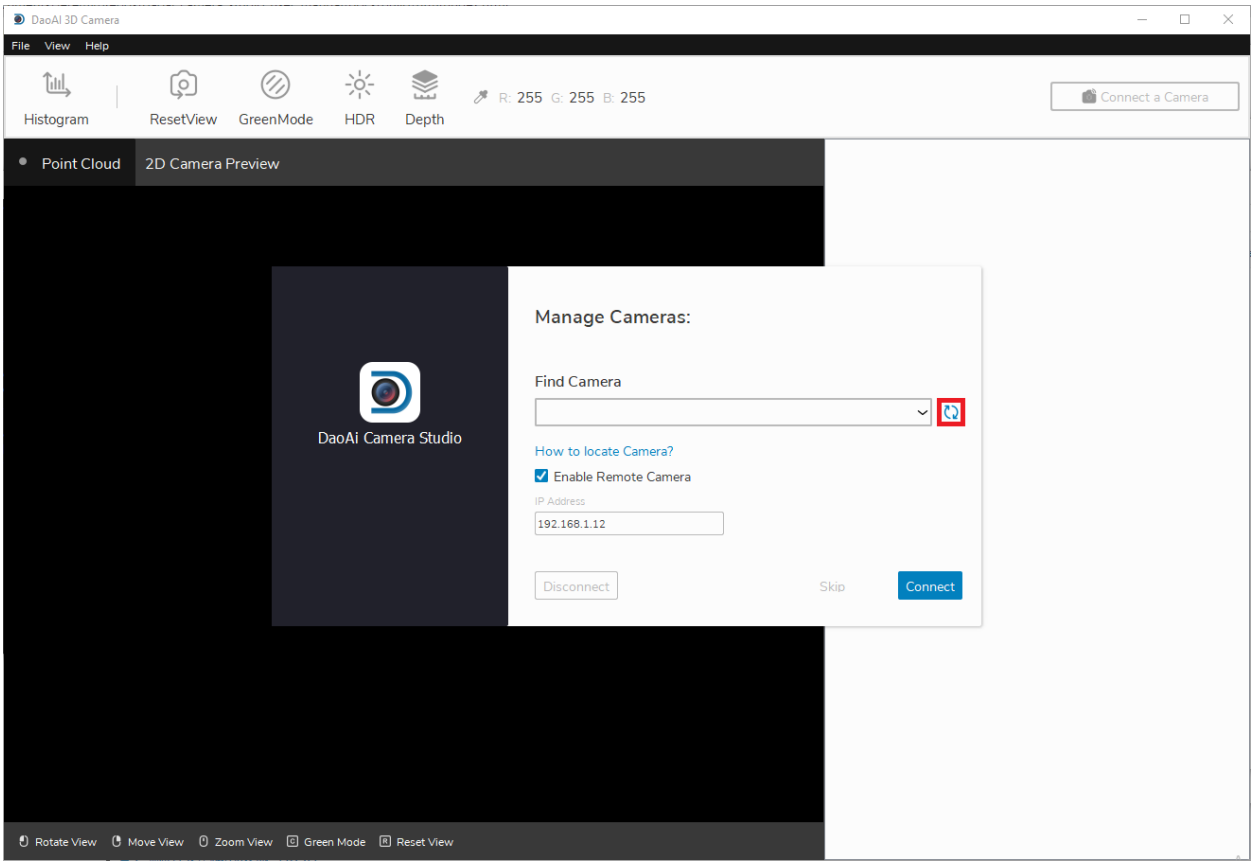
The first step of using the DaoAI Camera Studio is to connect your cameras.

Connecting

When starting up the Camera Studio, at first you will see the **Manage Cameras** window.



The refresh button updates the list of connected cameras.



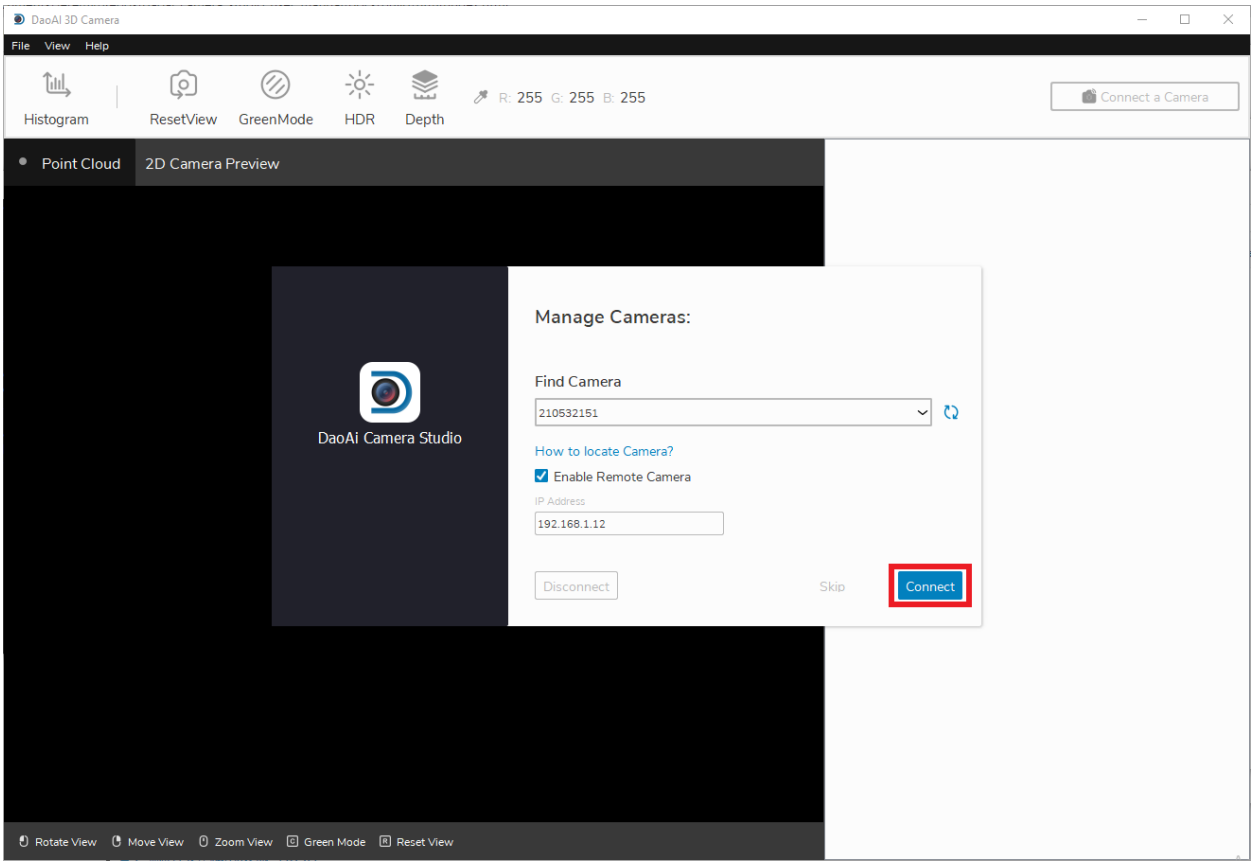
Note: If you are using a remote controlled camera, you will have check the *Enable Remote Cameras* checkbox and specify the camera's IP address first before clicking refresh.

Default IP for Remote Camera

- **192.168.1.2**: BP-L camera, BP-M camera, BP-S camera, BP-AMR-GPU camera, and IN cameras
- **192.168.1.12**: BP-AMR camera

If you still can't find your camera, check out

Once you detect the camera you wish to connect to, click the **Connect** button. If there are multiple cameras connected, you can choose which camera to connect to from the dropdown list.

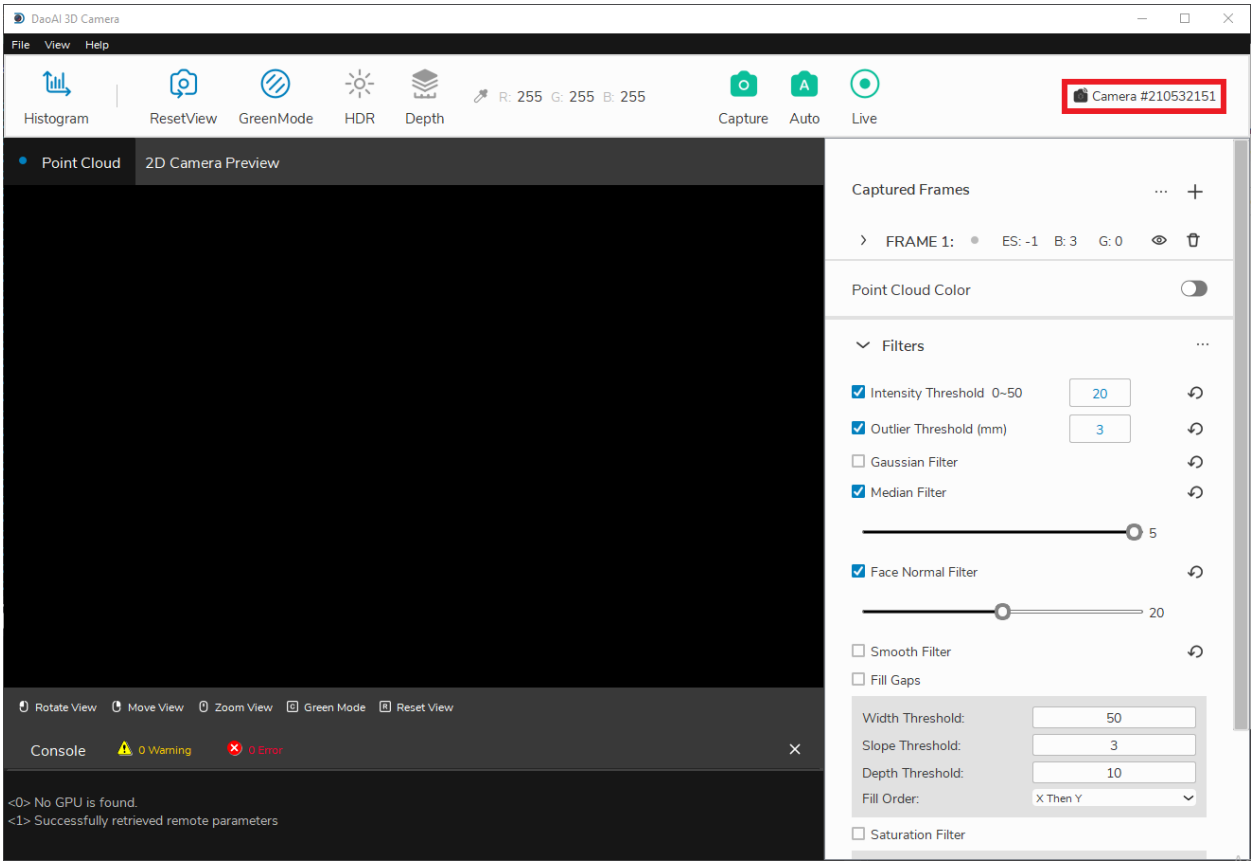


After connecting, you should see the main window.

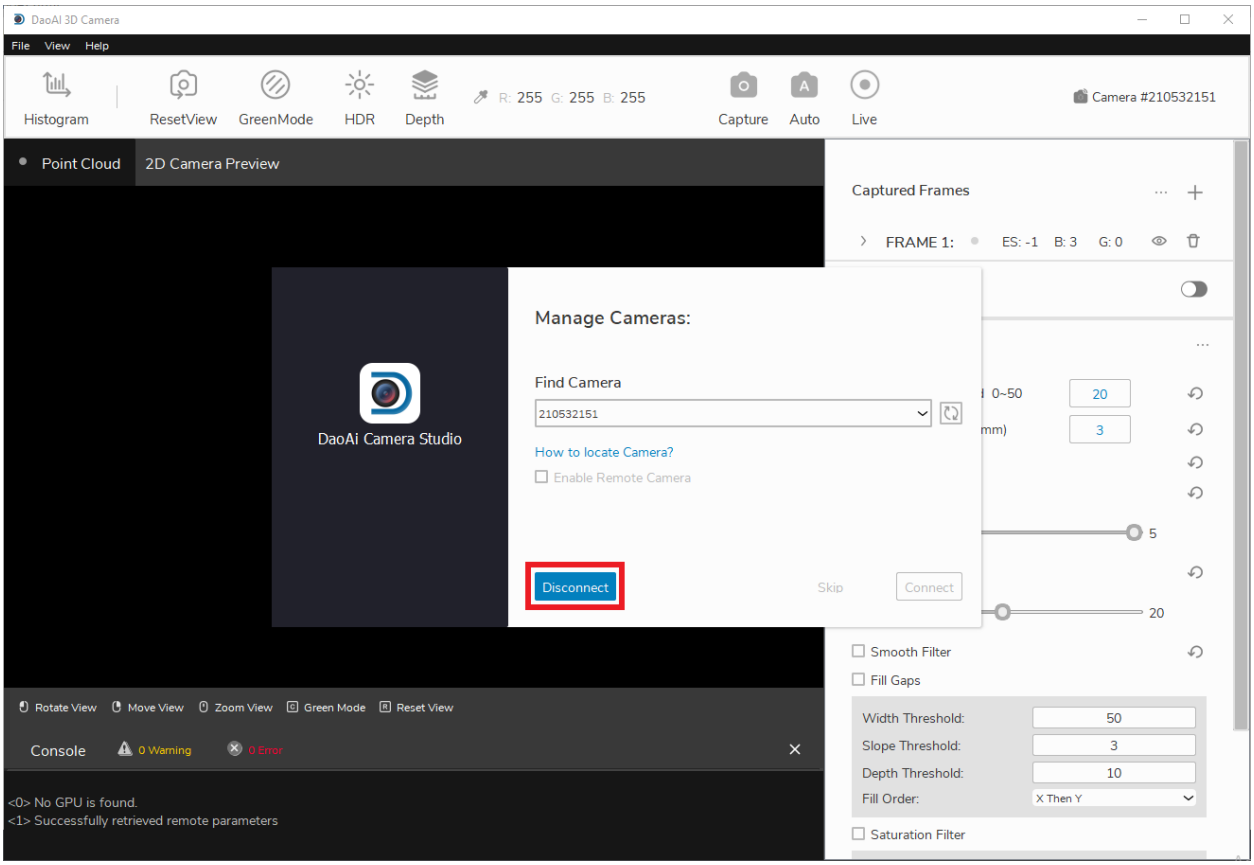
Note: Normally, if multiple cameras are physically connected, they will all appear in the camera selection list. However, DaoAI Camera Studio only supports establishing a connection with a single camera at a time. To capture with multiple cameras using DaoAI Camera Studio, please start another instance of DaoAI Camera Studio and connect the other camera.

Disconnecting

To disconnect your camera, first click the camera ID in the top right area of the main window.



From there, you should see the **Manage Cameras** window, where you can click disconnect.



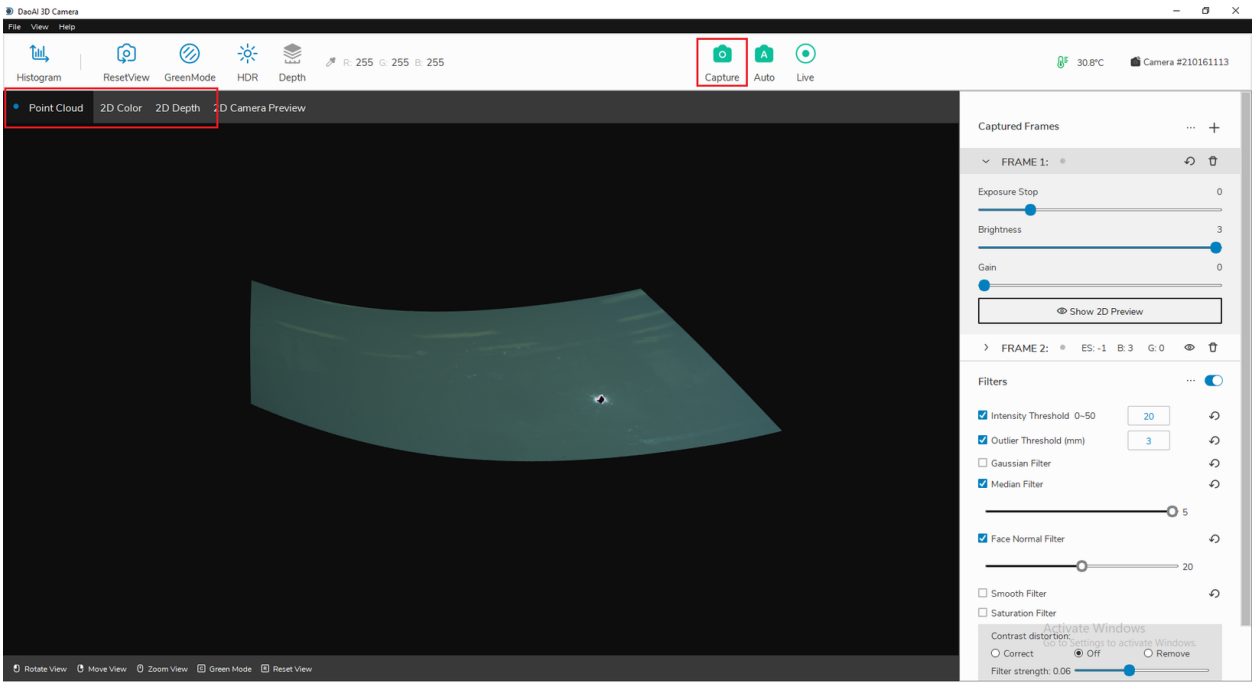
3.5.2 Captures

Performing captures is one of the main features of our Camera Studio software. Camera Studio has 3 capture modes to choose from:

- Single Capture
- Auto Capture
- Live Capture

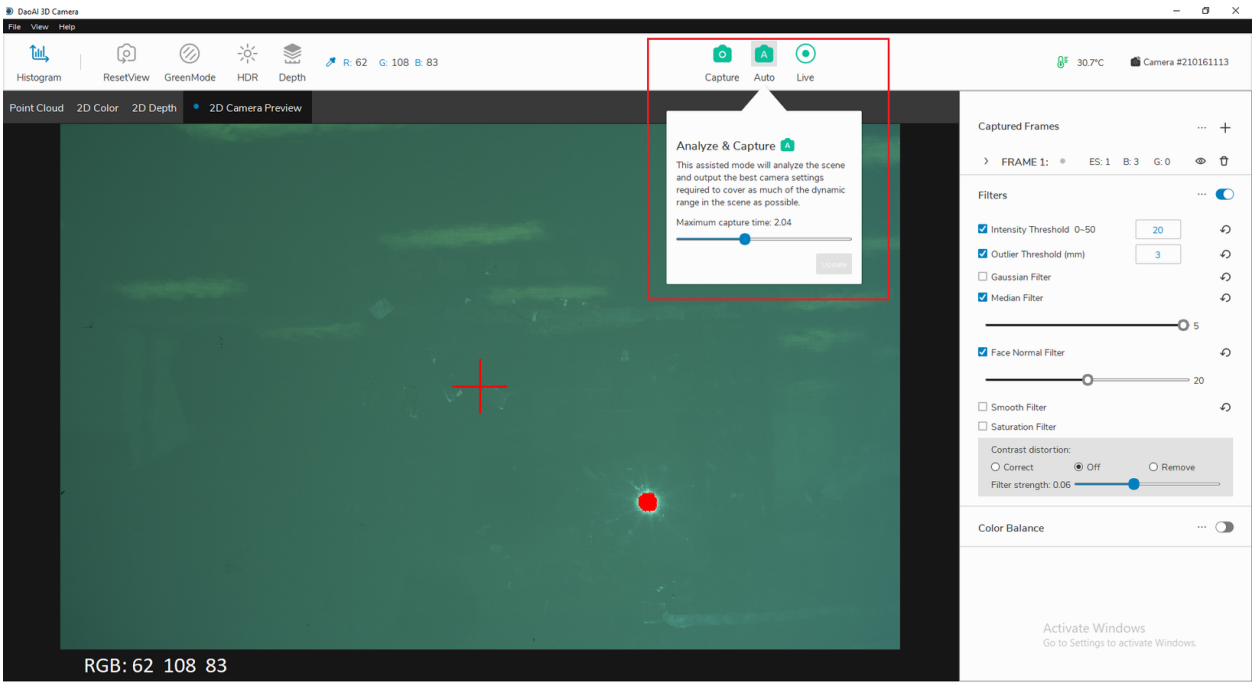
Single Capture

When you click “Capture” on the top bar, the camera will capture images using all the frames and their corresponding settings. For all capture modes, you can view the depth map (Depth), color map (Color), and the point cloud map (Point Cloud) by switching between these tabs.



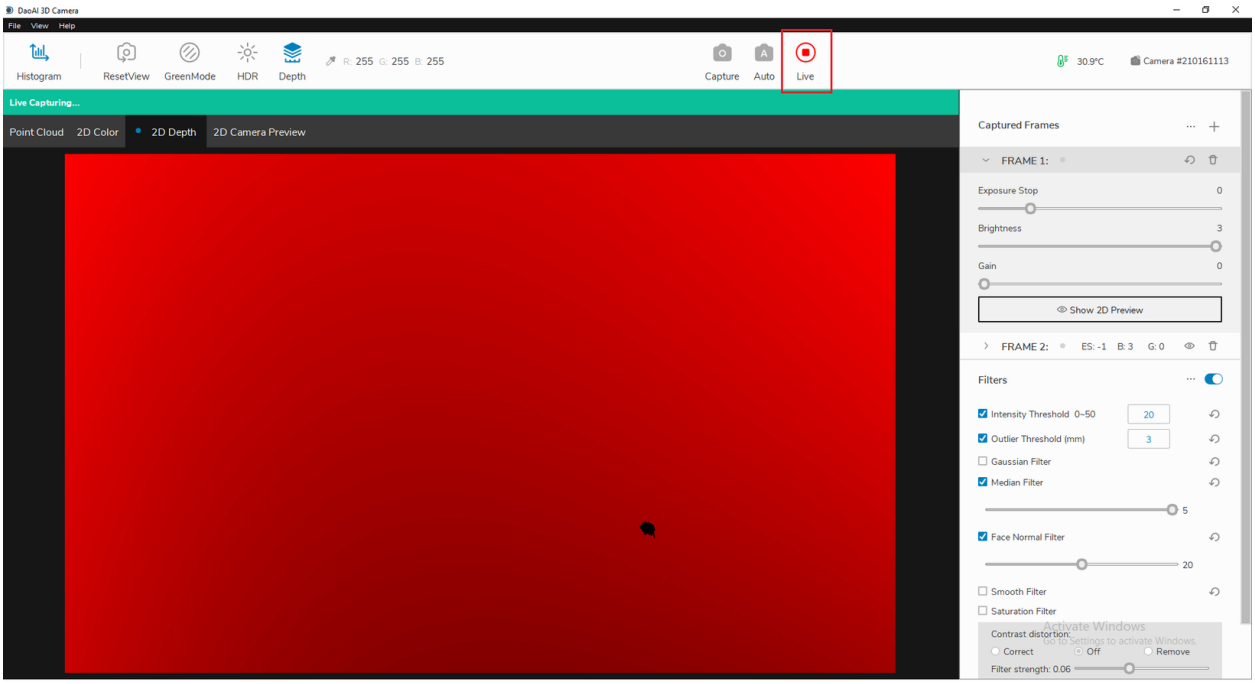
Auto Capture

When you click “Auto”, an analysis of the image environment will be conducted and frames will be automatically generated in order to maximize the dynamic range of the final image. Then, a single capture will follow using the new frame settings. In this mode, you can also specific the Maximum capture time. This setting will change the maximum cumulative exposure time for the frames it can generate based off of the initial analysis.



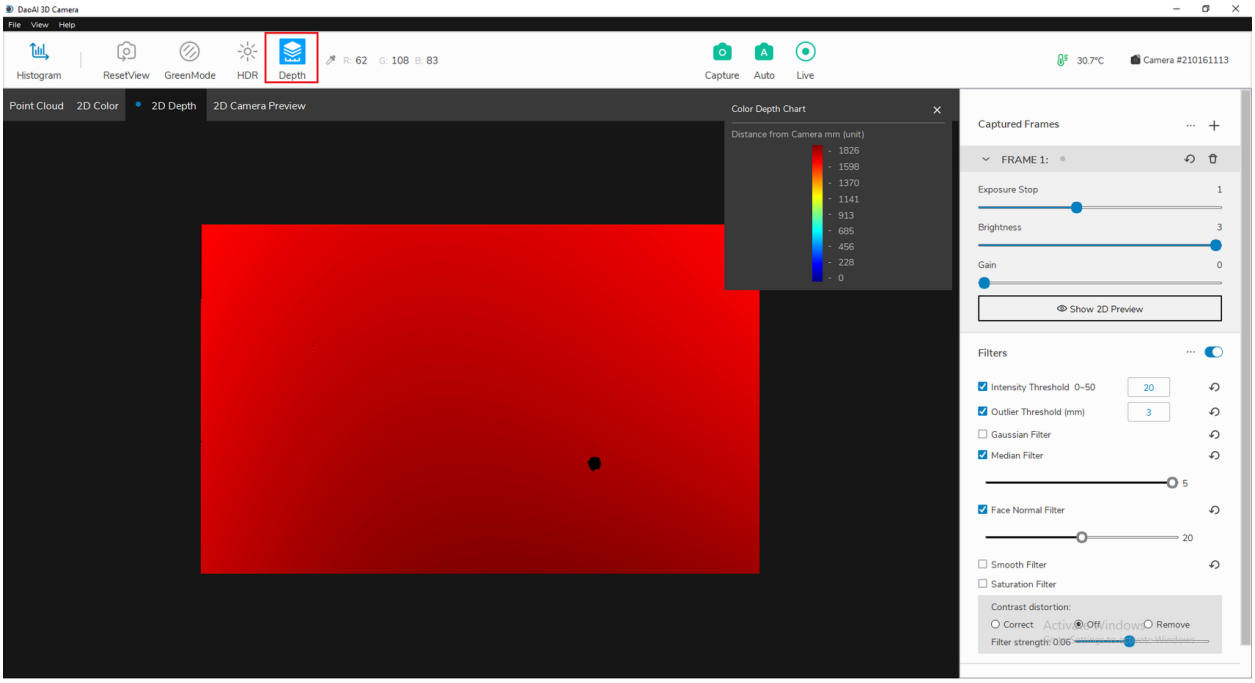
Live Capture

When you click “Live”, Camera Studio will continuously perform single captures. To stop performing captures, simply click the button again (while Live mode is running).



3.5.3 Depth

The 2D depth display tab encodes the depth values of each valid pixel as its color.
Blue represents short distance, red represents long distance from the camera, in unit millimeter.
To analyze the specific values, you can toggle the "Depth" button and the depth chart legend will pop up.



3.5.4 Frames

Frames are used to adjust exposure stop, brightness, and gain levels that are applied when doing captures. Camera Studio allows you to add multiple frames with different settings to capture multiple image groups and fine tune the quality of your resulting images.

Adding Frames

When an image does not meet the expected requirements, you can achieve more precise exposure levels by adding more frames and with different exposure file settings. For example, using two frames set at 2 different exposure values, -1 and 0, to make up for the situation where a single frame -1 is too dark or a single frame 0 is too bright.

Click the "+" button on the right frame settings menu in order to add more frames. Similarly, you can click on the trash can icon to delete a corresponding frame. The minimum number of frames is 1.

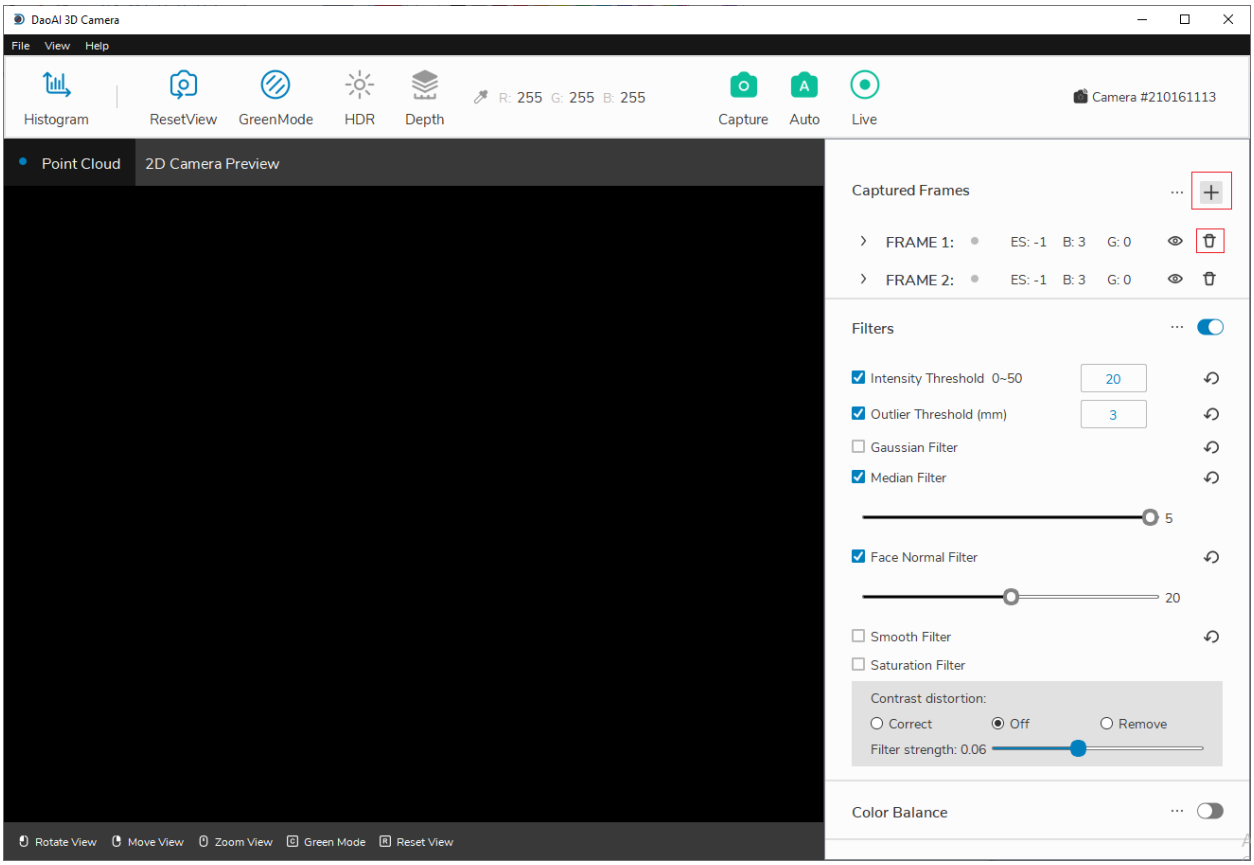


Fig. 1: Add and remove buttons

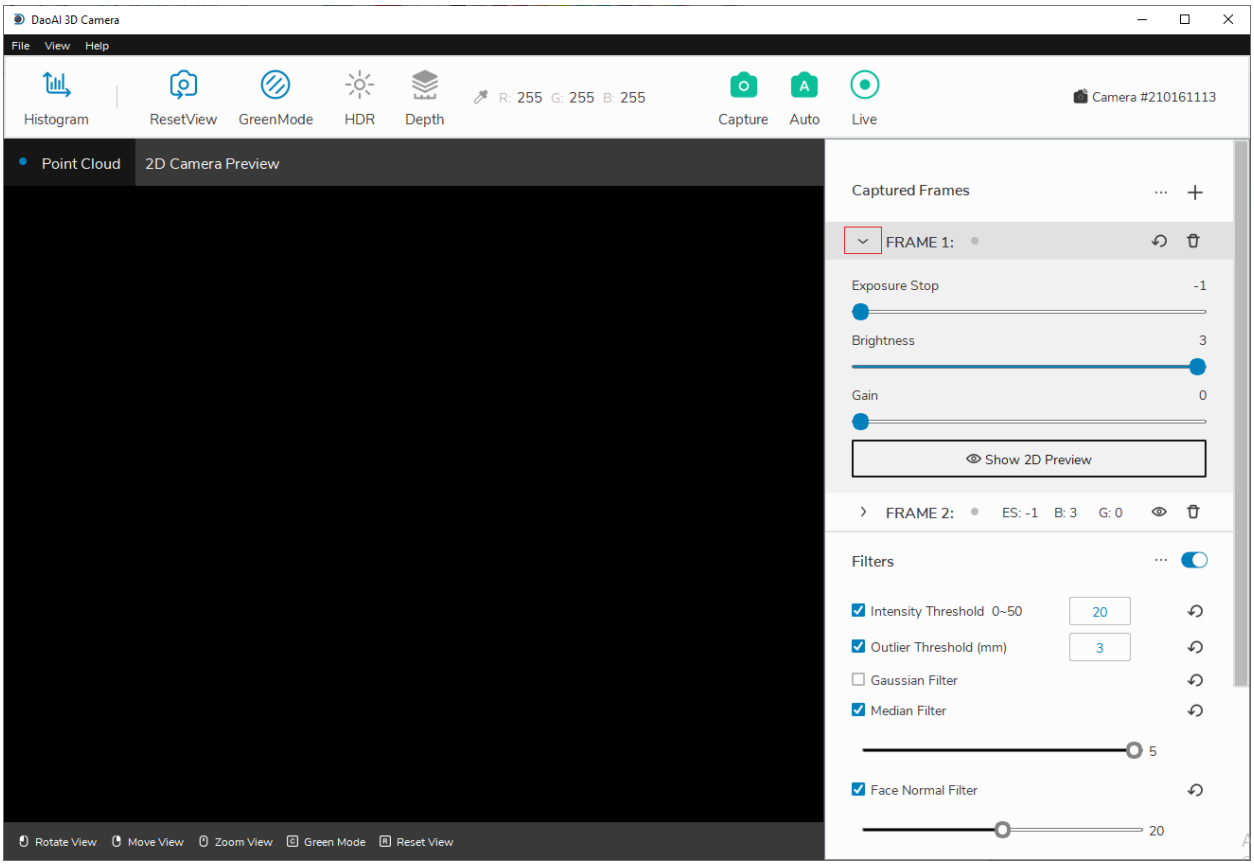
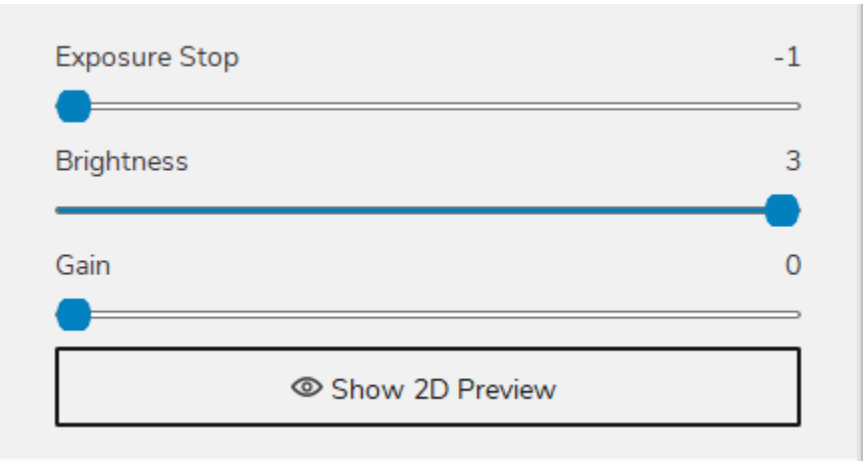


Fig. 2: Dropdown menu reveals parameters

Frame Parameters

All three parameters serves the purpose of adjusting final image brightness and each increase of one in any field will double the final image brightness.

When attempting to increase image brightness, you should prioritize “Brightness”, then “Exposure Stop”, and increase “Gain” last.



Exposure Stop

Responsible for adjusting the exposure time. The level of exposure stops are -1, 0, 1, 2, 3, 4. Since increasing “Exposure Stop” increases capturing time, you should only increase this field when “Brightness” has reached maximum value. The most commonly used exposure levels are -1, 0, and 1.

Brightness

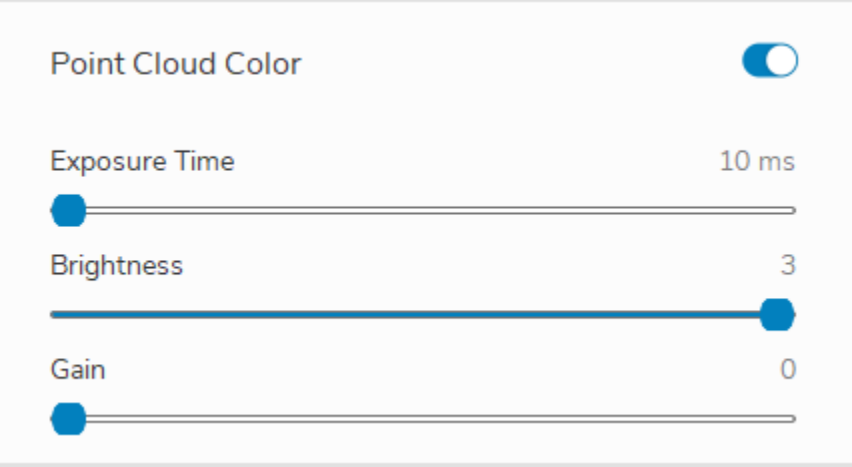
Responsible for adjusting the brightness of the projecter. The possible values are 1, 2, 3. Changing this field does not increase capturing time and should prioritize this field when adjusting image brightness.

Gain

Responsible for increasing the ISO value or sensitivity of the camera. The span ranges from 0, 1, 2, 3, 4 where 0 is the base point 0dB and 4 corresponds to the highest value 24dB (each slider increment causes 6dB change). When the gain is increased or decreased by 6dB, the gain effect is twice as much as that of the previous one. The conversion formula is: 20 x log (gain multiple) = + /-gain dB value. Increasing “Gain” will reduce signal-noise ratio, therefore we should increase this field lastly.

Point Cloud Color

Independently capture the color for Point Cloud.



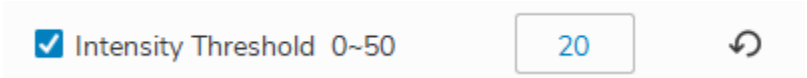
When enabled, a settings section will expand as the image above and when capture, will collect an additional image using the setting for the Point Cloud's color. The additional image will be captured by projecting a white screen.

The settings are similar to "Frame Parameters" except the "Exposure Time", which is in miliseconds, allowing more precisive controls.

3.5.5 Filters

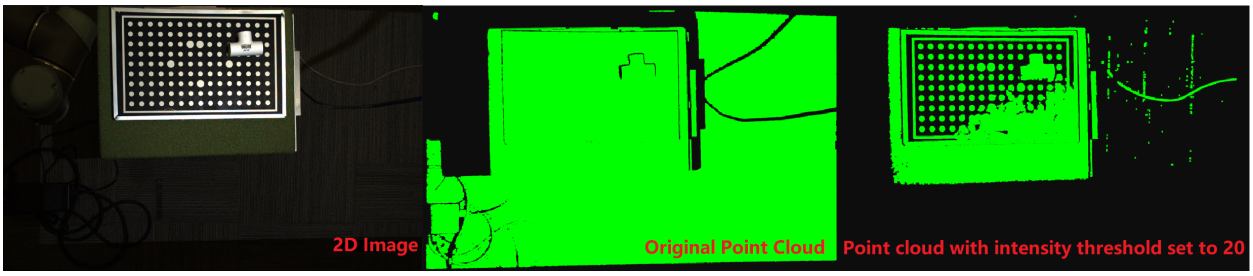
This section describes the filters used for pre and post processing images in order to improve the point cloud quality.

Intensity Threshold



Our intensity metric is based on the average pixel intensity value from the four fringe projections from capture. The threshold helps filter out outliers that are caused by low-quality pixels within dark areas in an image. Generally, we see that for objects with low reflectivity, the algorithm is not able to conduct 3D reconstruction and thus computes incorrect points there. The value of the intensity threshold corresponds to the averaged RGB value we want to filter at.

e.g. When the intensity threshold is 20, all the pixels in the final image with values <= 20 will be filtered out.



Outlier Threshold

Filter out points which are more than a certain distance from their nearest neighboring point. For example, if set to 3mm, it is determined whether the straightline distance between two adjacent points is greater than 3mm. If it is greater than 3mm, the point is filtered out. However, if there are multiple outliers close to each other, this filter will not be able to filter out those points.

Phase Quality Filter

This filter is used to filter out low contrast areas. The track bar can be used to set the filtering strength. Higher value will filter out more areas. For example, using a phase quality filter of 0.3, some area of the floor is filtered out in the point cloud.

Gaussian Filter

This filter applies a moving average window to each pixel of the captured image. The gaussian kernel size can be set to be 3x3 , 5x5, 7x7, 9x9, 11x11, and 13x13. This applies a smoothing effect on the point cloud and also helps to remove outlier points.

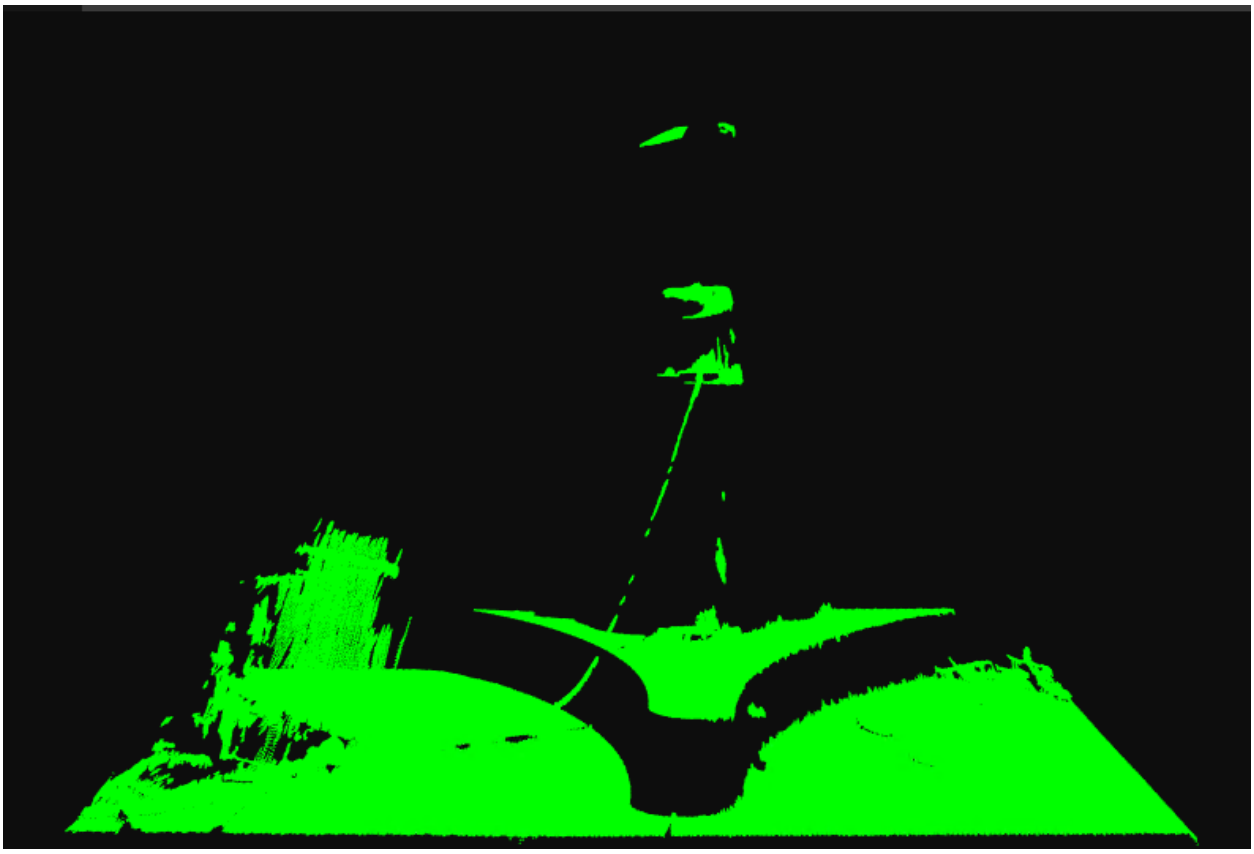


Fig. 3: Point cloud without outlier threshold set

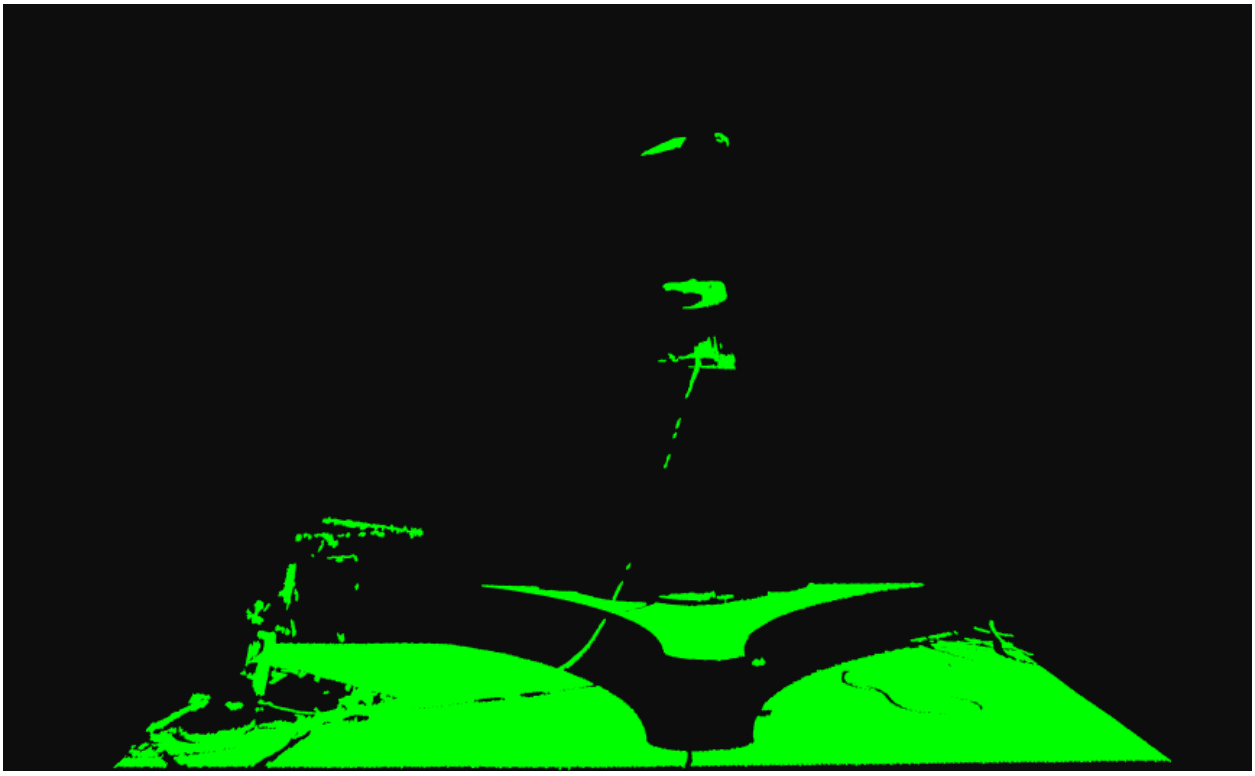


Fig. 4: Point cloud with outlier threshold set to 3mm

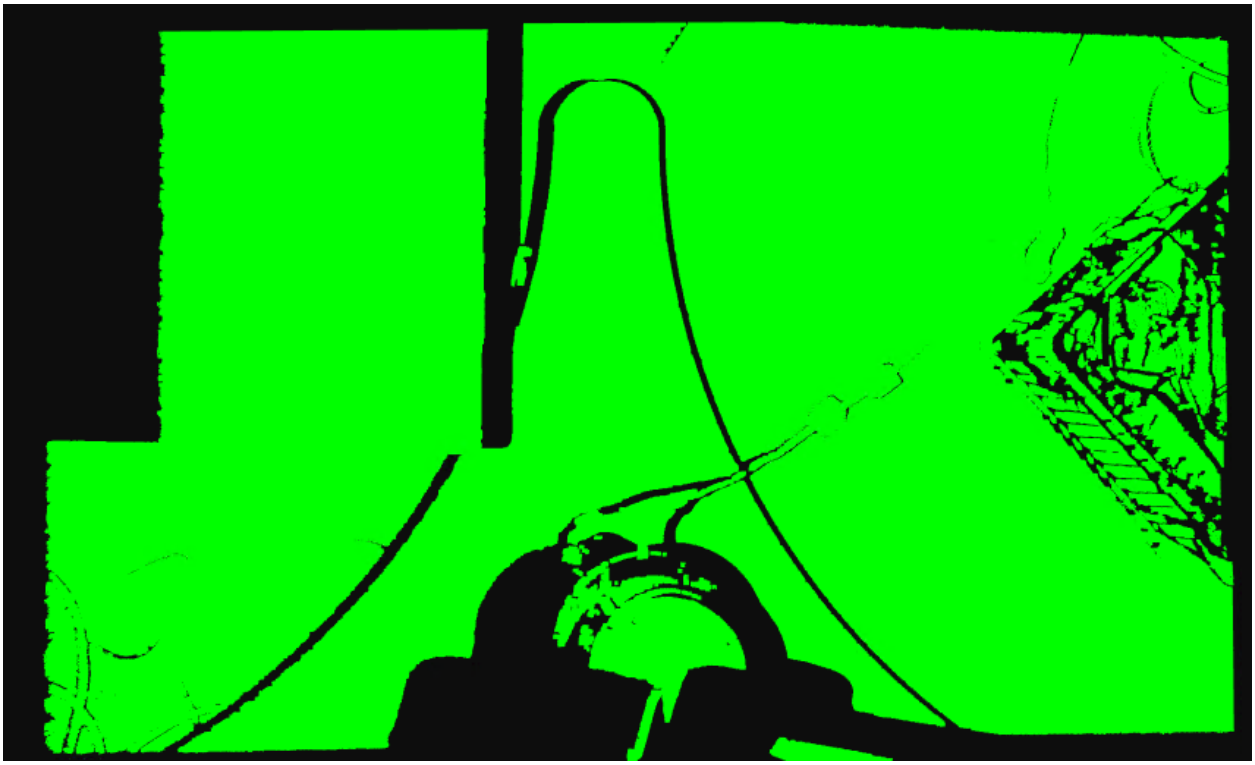


Fig. 5: Point cloud without phase quality set

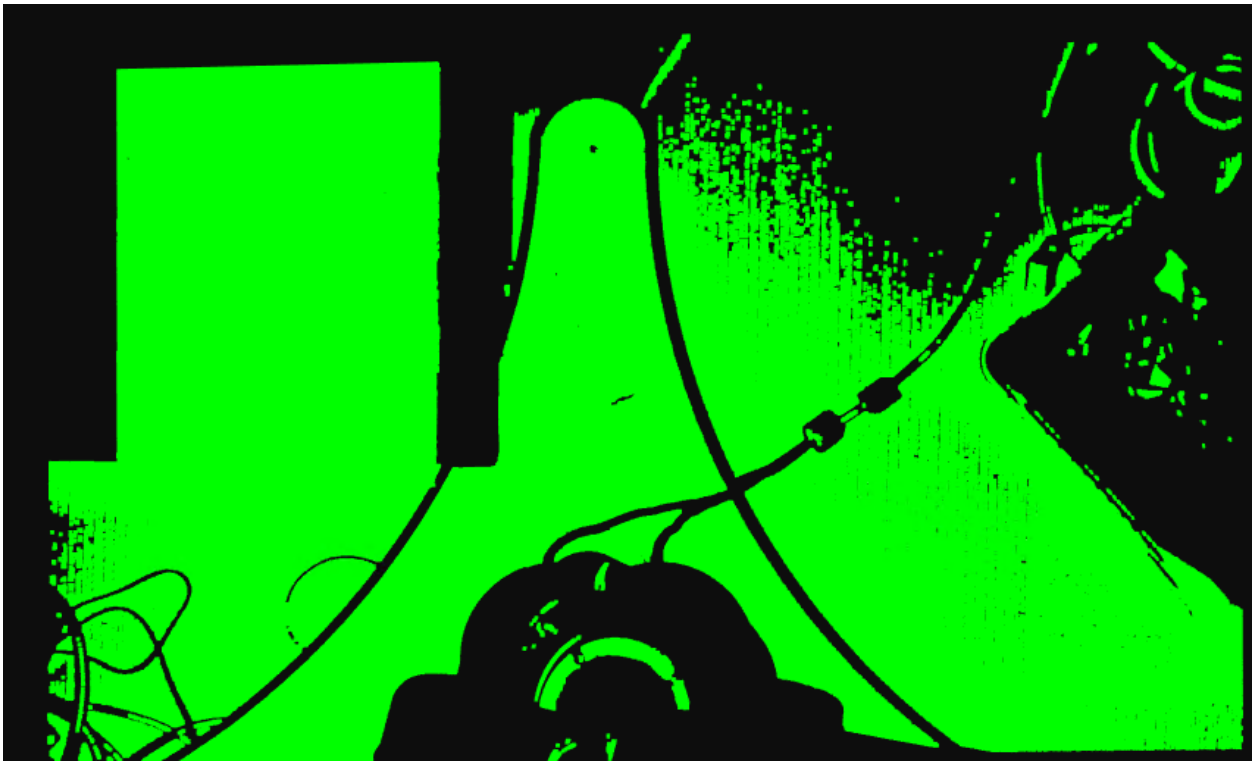


Fig. 6: Point cloud with phase_quality set to 0.3

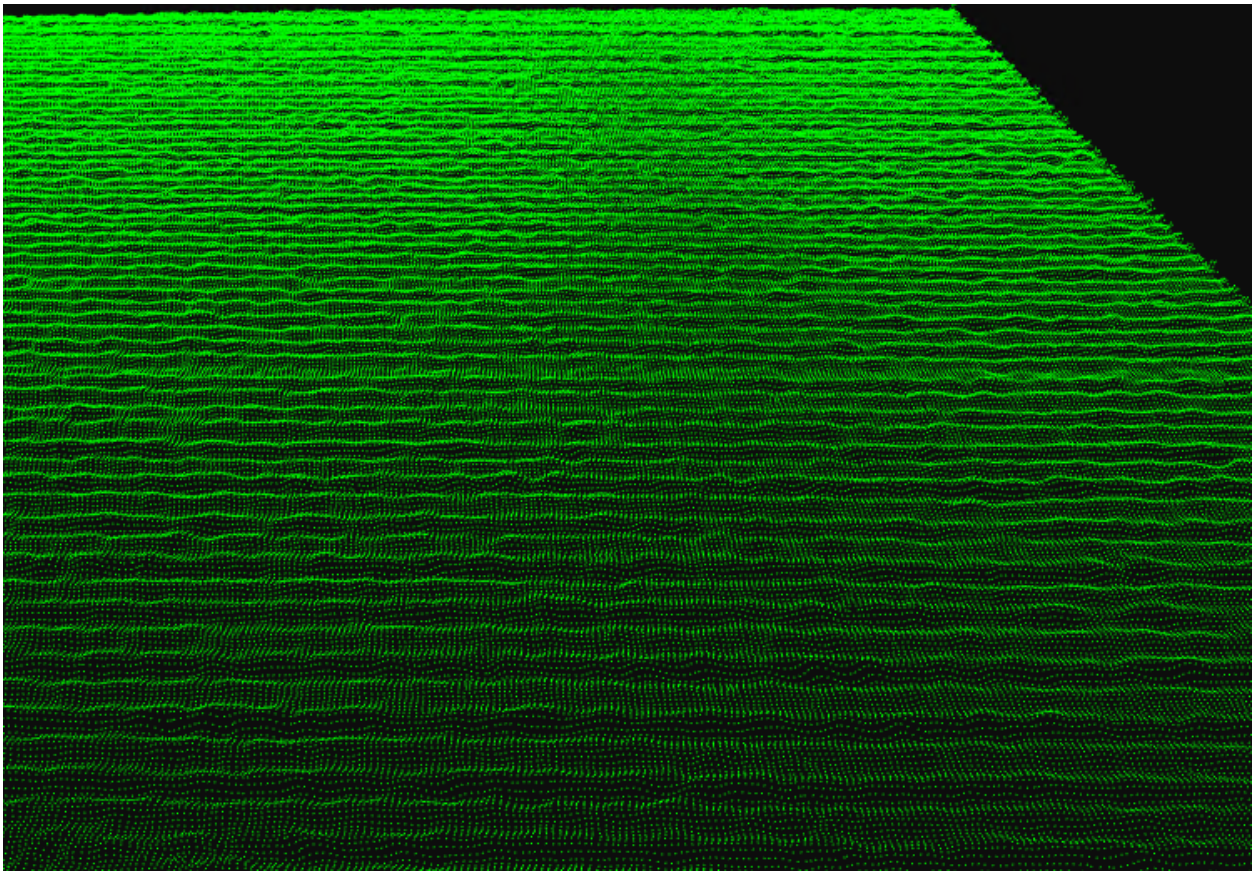


Fig. 7: Original point cloud

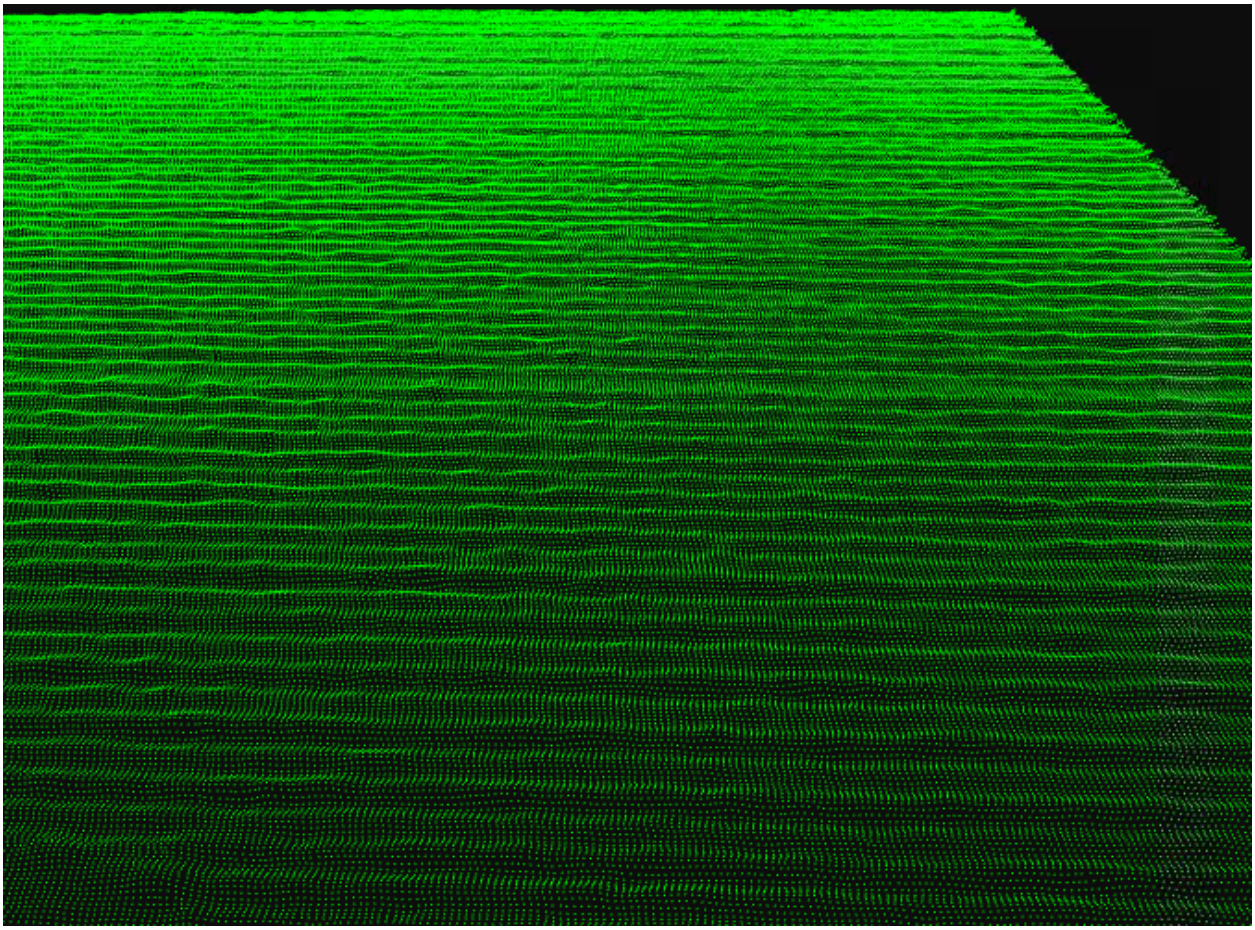


Fig. 8: Point cloud with Gaussian filter of kernel size 5x5

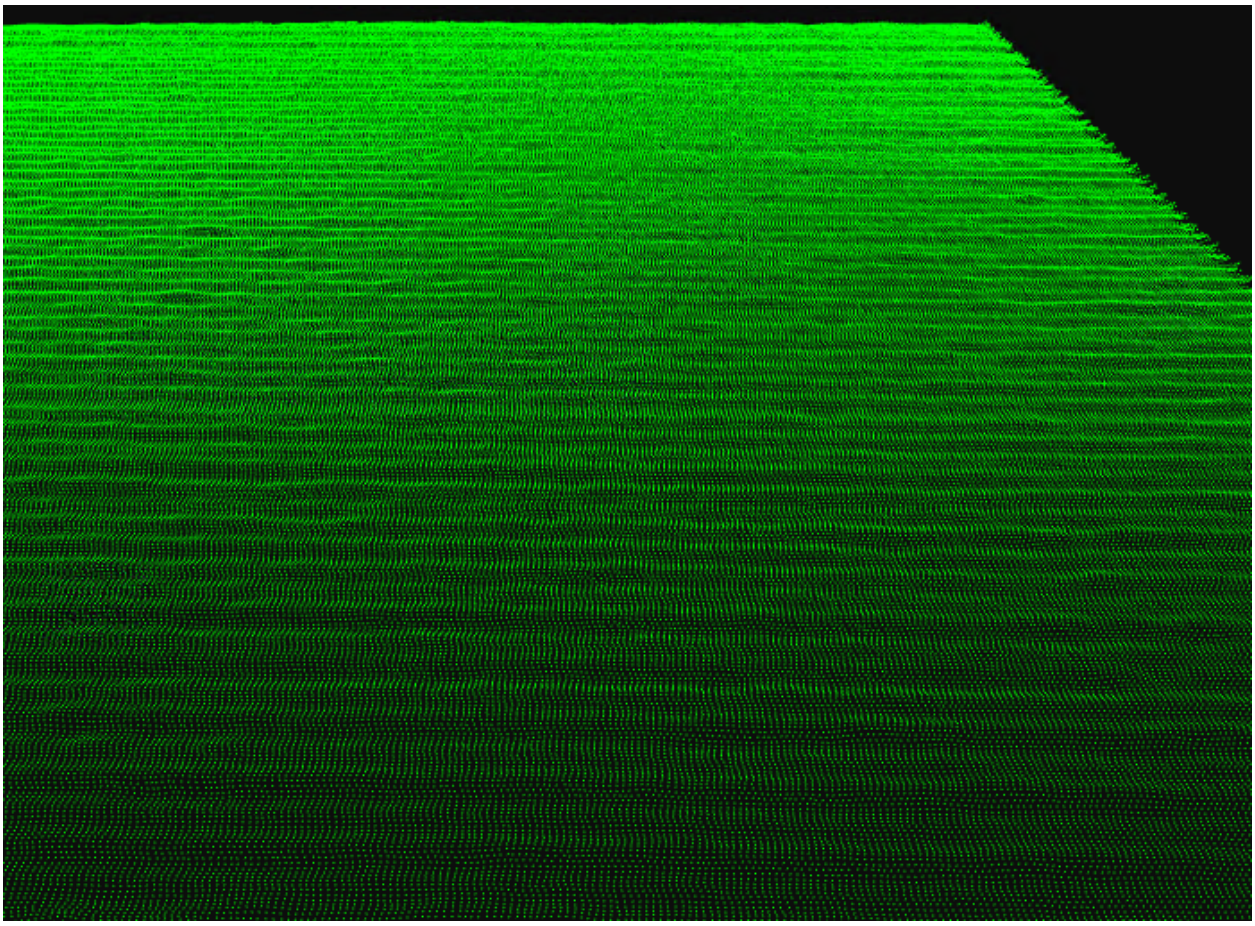
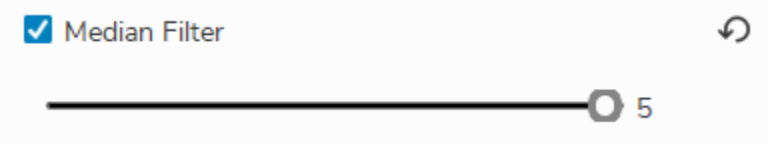


Fig. 9: Point cloud with Gaussian filter of kernel size 9x9

Median Filter



This filter finds the median value of a sliding window to update the current pixel as. The kernel size can be 3x3 or 5x5. This applies a smoothing effect on the point and helps remove outlier points.

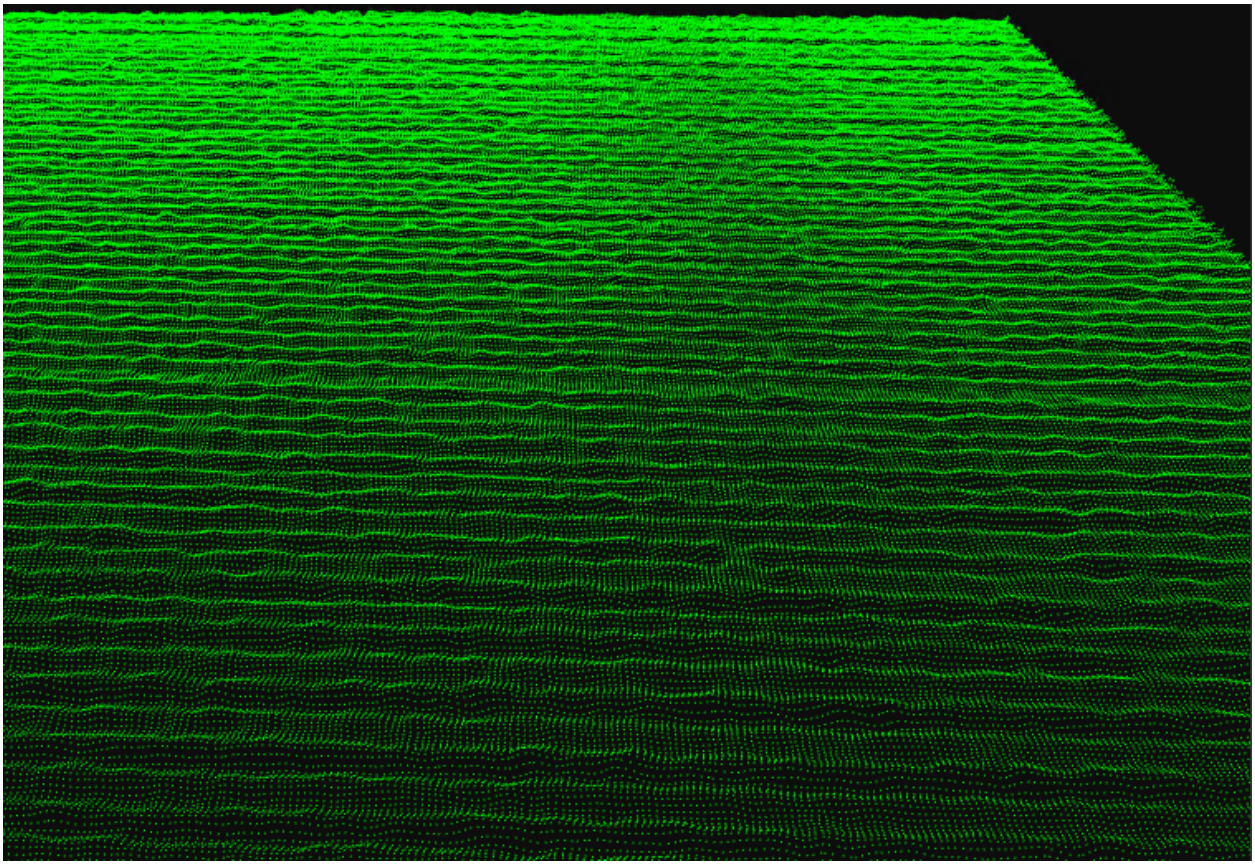


Fig. 10: Original point cloud

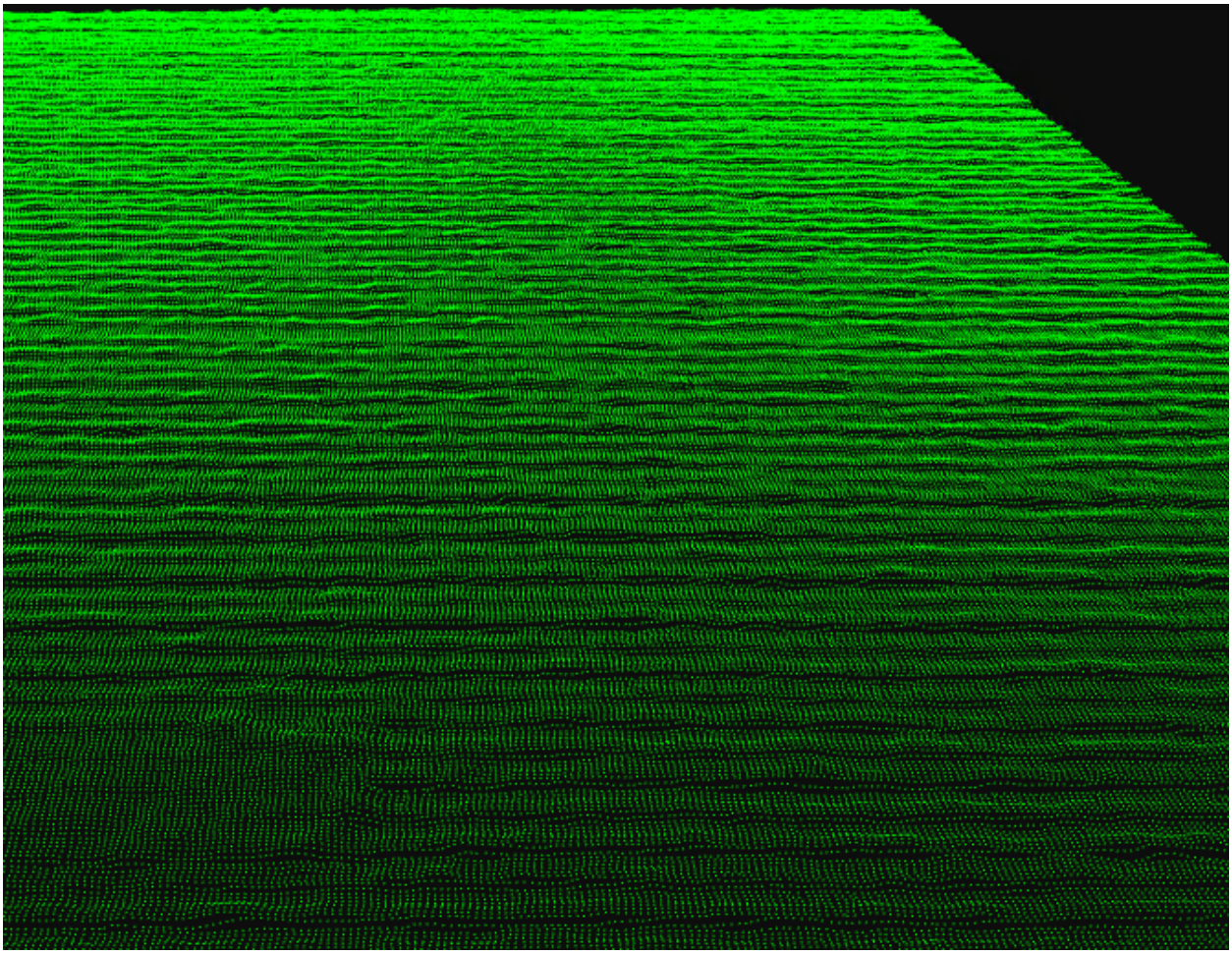
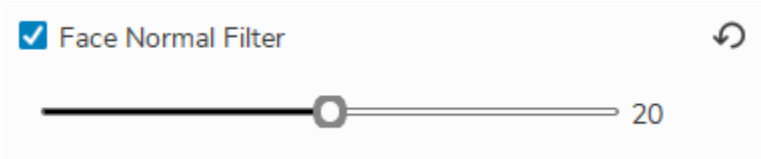


Fig. 11: Point cloud with median filter of kernel size 5x5

Face Normal Filter



This filter analyzes a point cloud polygon mesh to find the surface normal vector of any polygon. If the surface normal vector is at an angle larger than the face normal value with respect to the line of sight, then the points are filtered out. On objects with sharp corners and large areas where the surface normal is perpendicular to the line of sight, many outliers occur. An example is a box: the walls sometimes create outlier points in the point cloud due to noise, low contrast or over-saturated images.

Without this filter, the incorrect points show up on the edges of holes, sharp drop offs, vertical faces, and more.

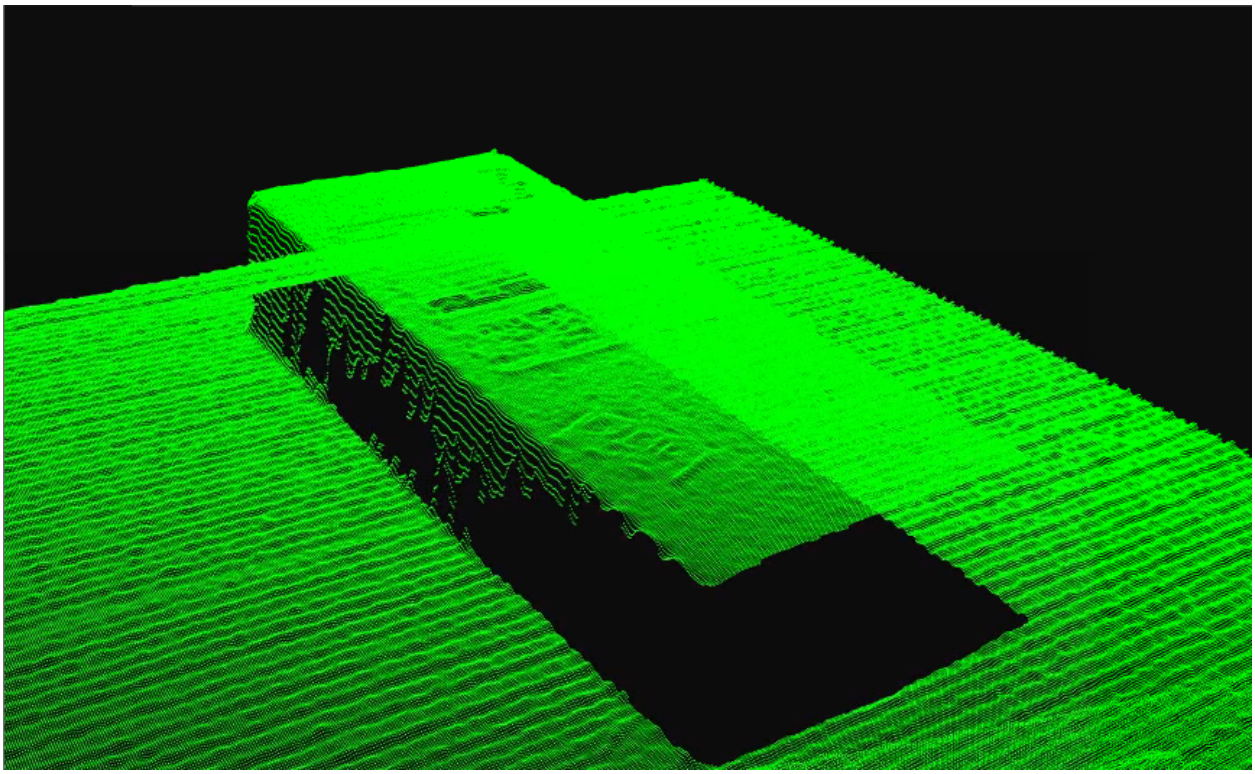


Fig. 12: Original point cloud

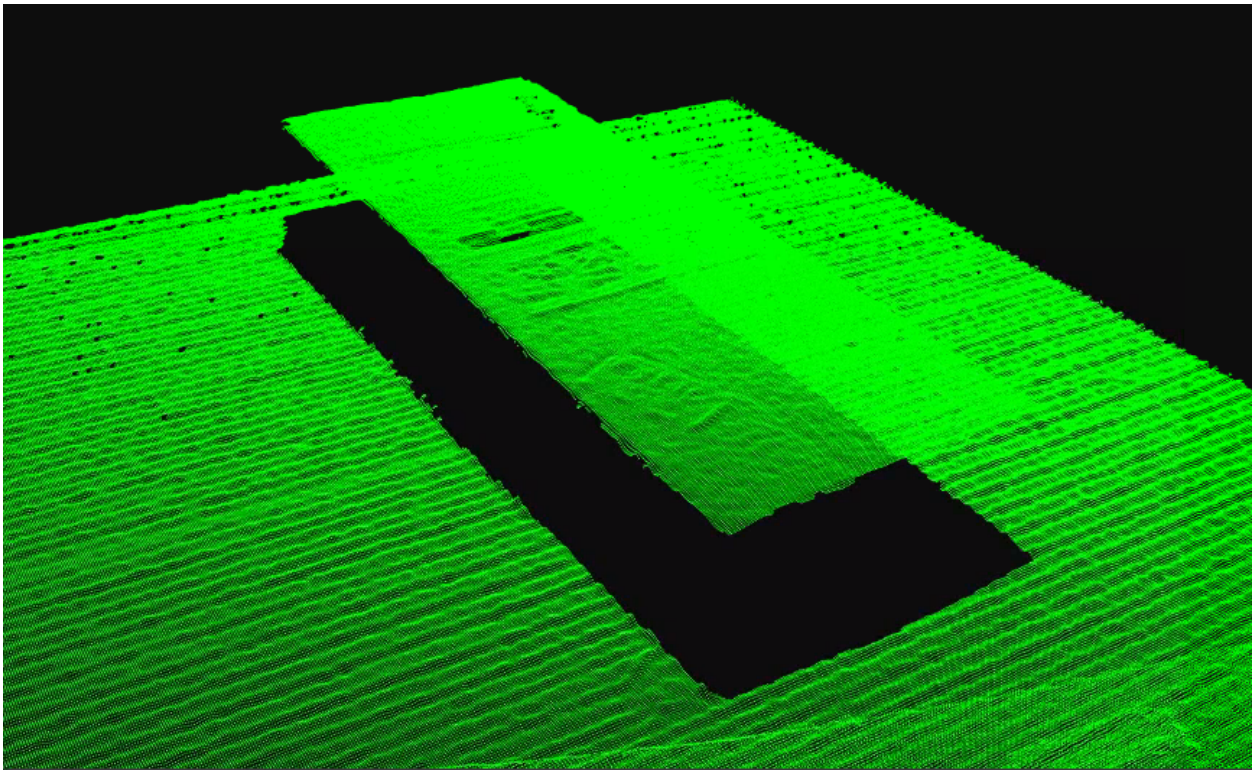


Fig. 13: Point cloud with Face Normal Filter

Remove Small Area

This filter removes small chunks of isolated point cloud. Since noise often appear in the form of small dots, they can be removed using this filter.

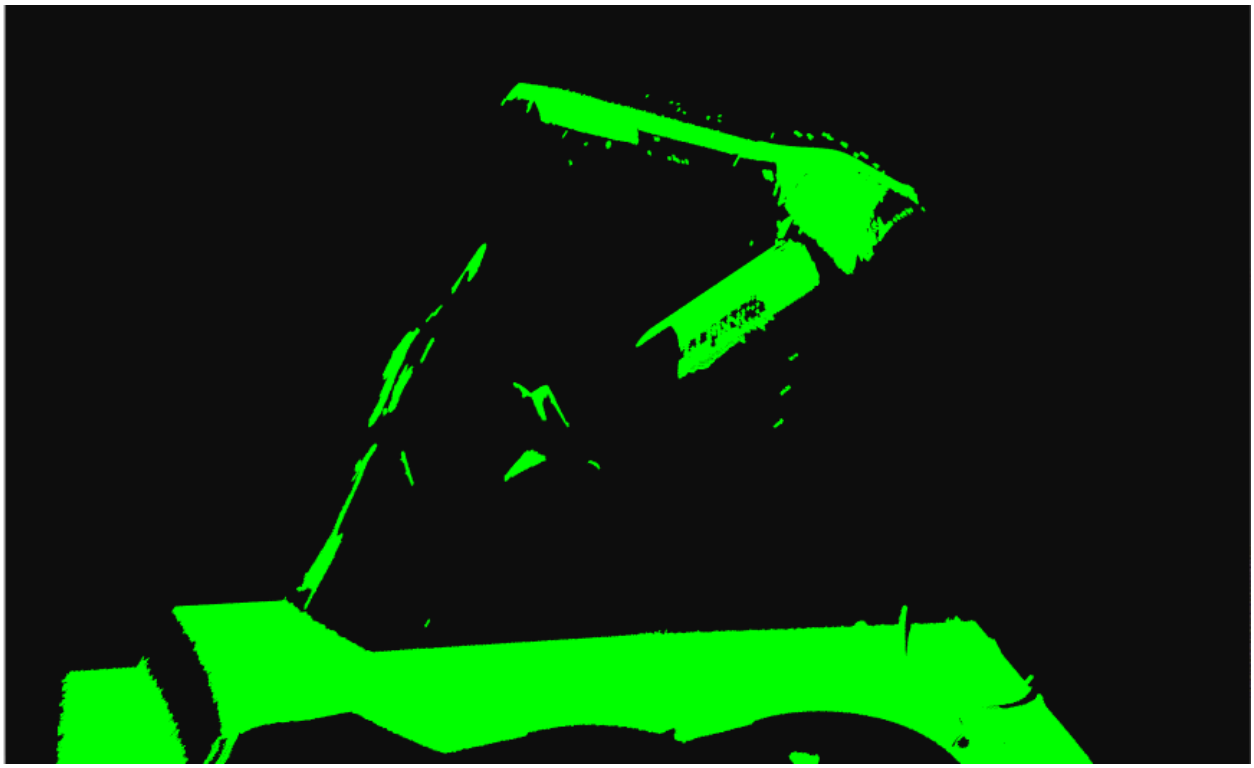


Fig. 14: Original point cloud

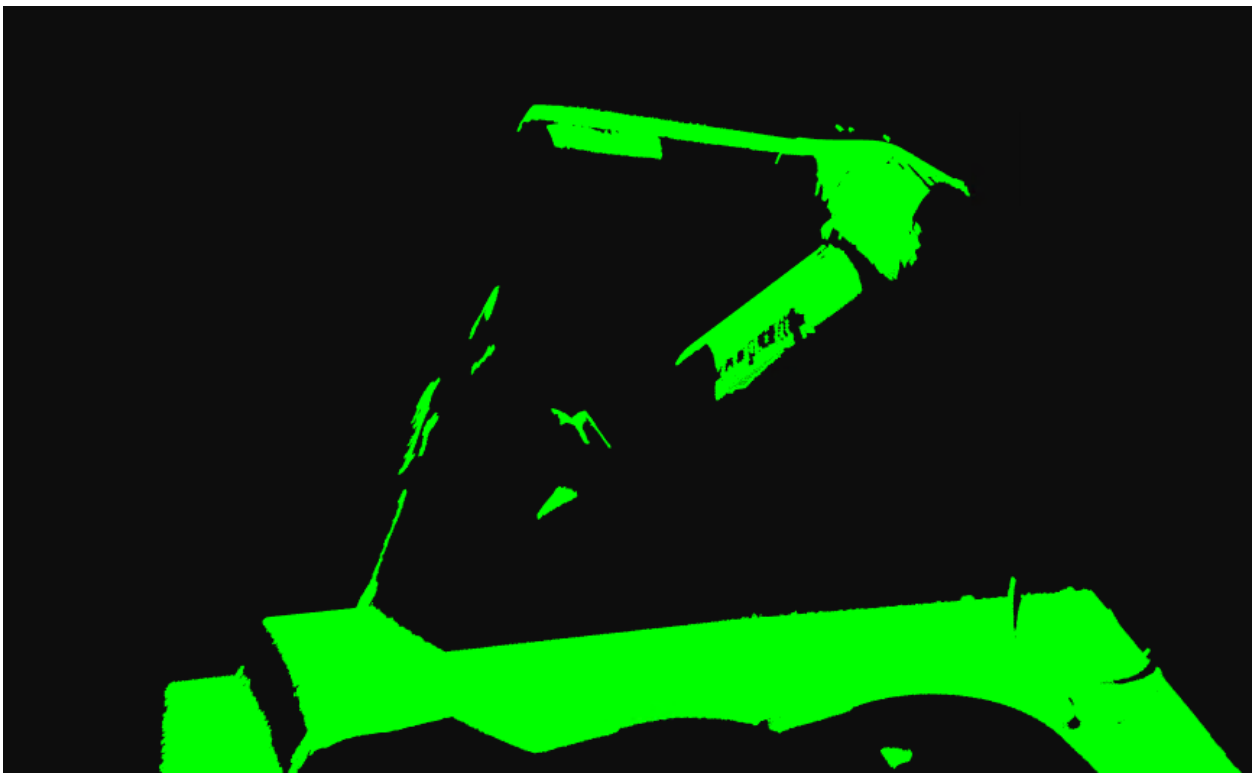


Fig. 15: Point cloud with Remove Small Area

Smooth Filter

This filter is a post-processing filter which rounds the depth value of an organized point cloud to the nearest mm. For example, if the smooth value is 0.5, each of the depth values will be rounded to the nearest 0.5mm. This filter is useful in scenarios where images with noise errors causes small oscillations on the point cloud. If you know the model is flat and you see small oscillations in the point cloud, you can use this filter to round the data to create a flat model.

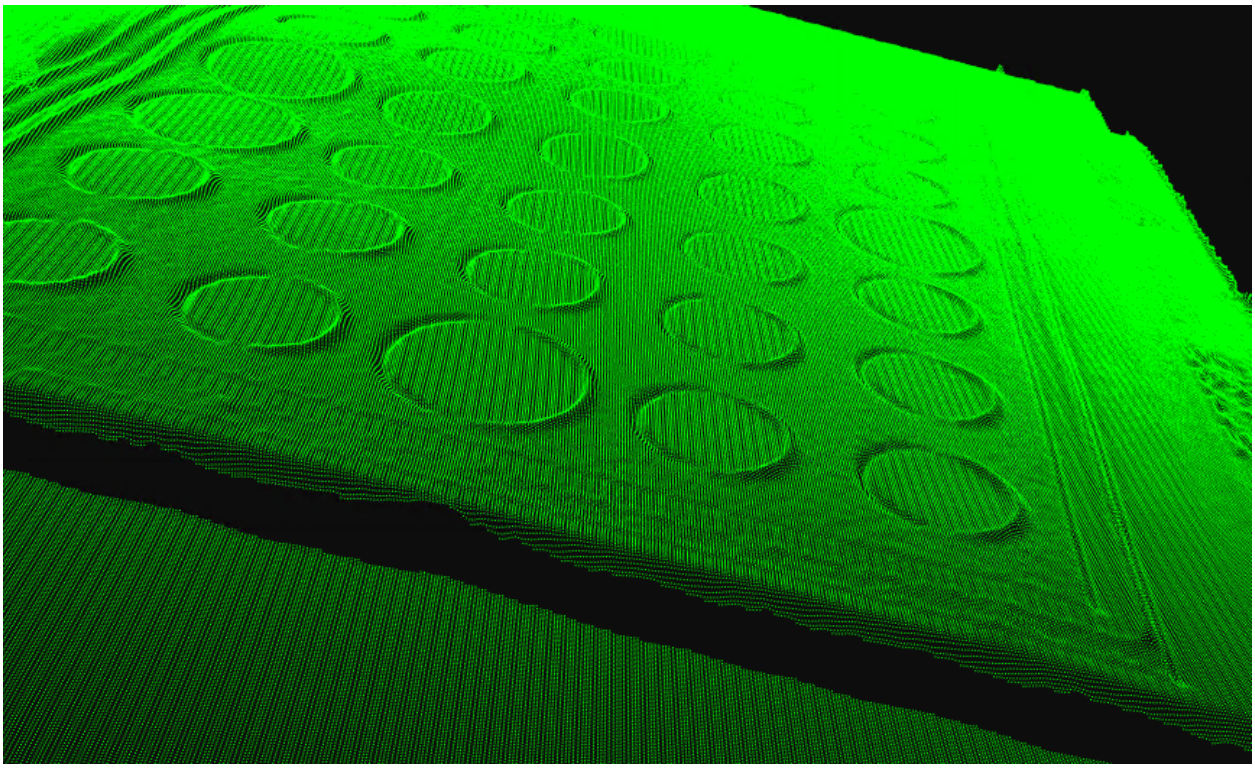


Fig. 16: Original point cloud

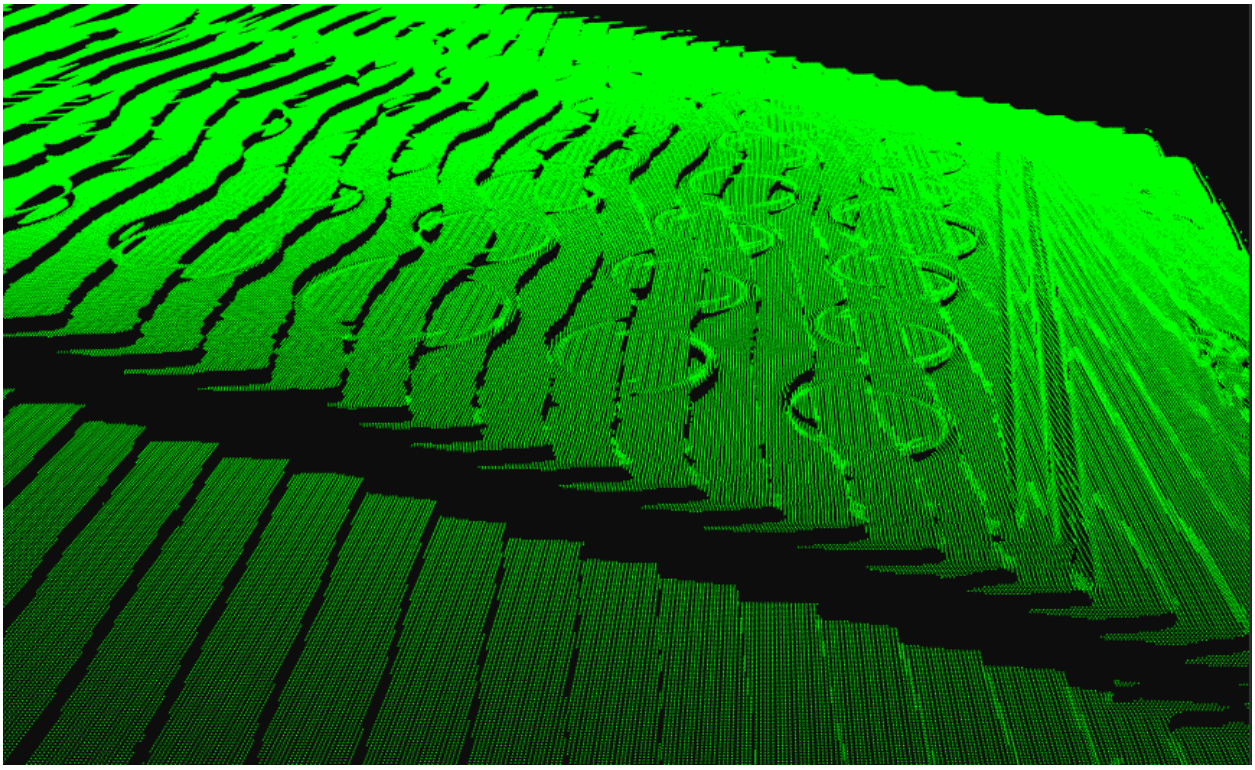
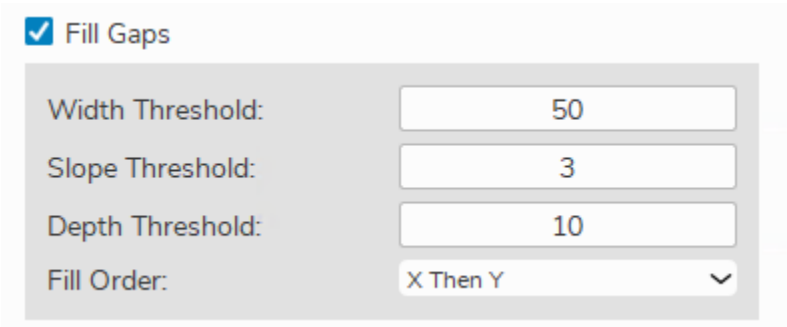


Fig. 17: Point cloud with Smooth Filter

Fill Gaps



Interpolation can be used to caltulate the coordinates of points in areas where points are missing from the point cloud. There will be scenarios where your point cloud is missing points in areas due things like reflection, poor lighting, etc. Thresholds can be set for maximum gap area width, depth, and slope, to describe the areas of the point cloud where interpolation will occur to calculate and fill in these missing points.

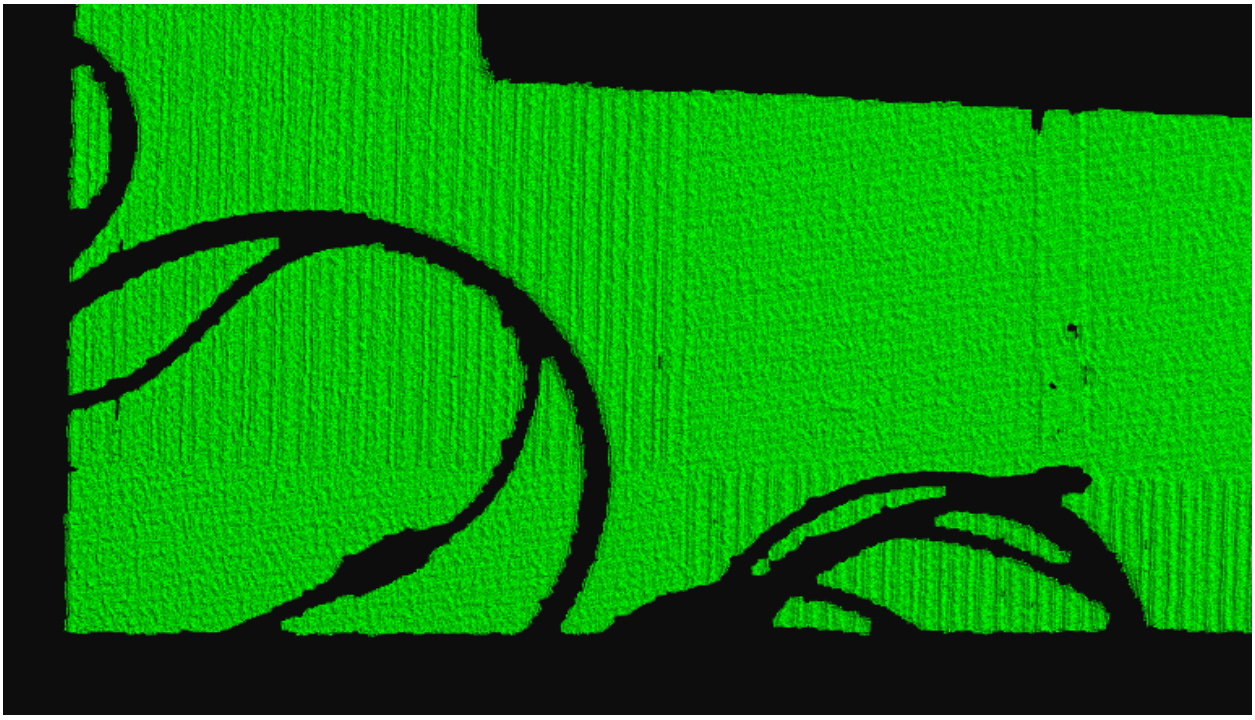


Fig. 18: Original point cloud

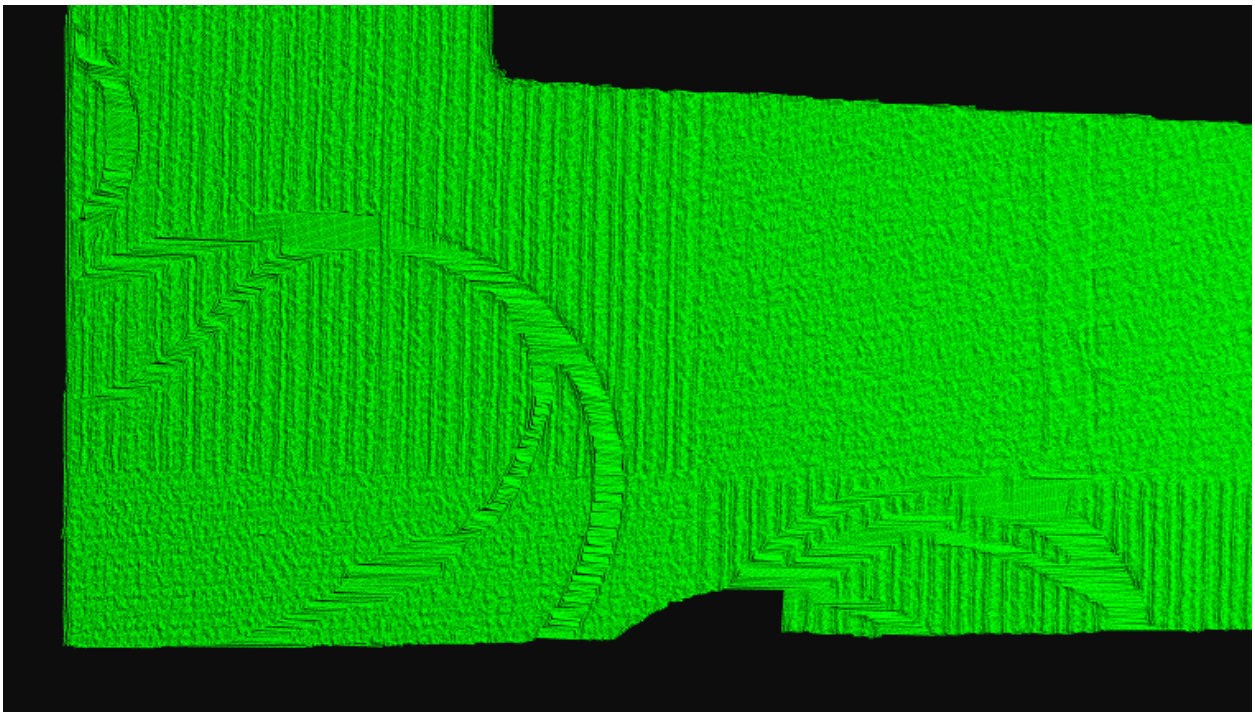


Fig. 19: Point cloud with Fill Gaps

Saturation Filter

This filter removes areas that are overexposed. When one of the three RGB channels exceeds 255, it deletes the pixel. Usually G (green) is the first overexposed channel. When using the filter, HDR mode is automatically selected and can be manually cancelled if it is not necessary.

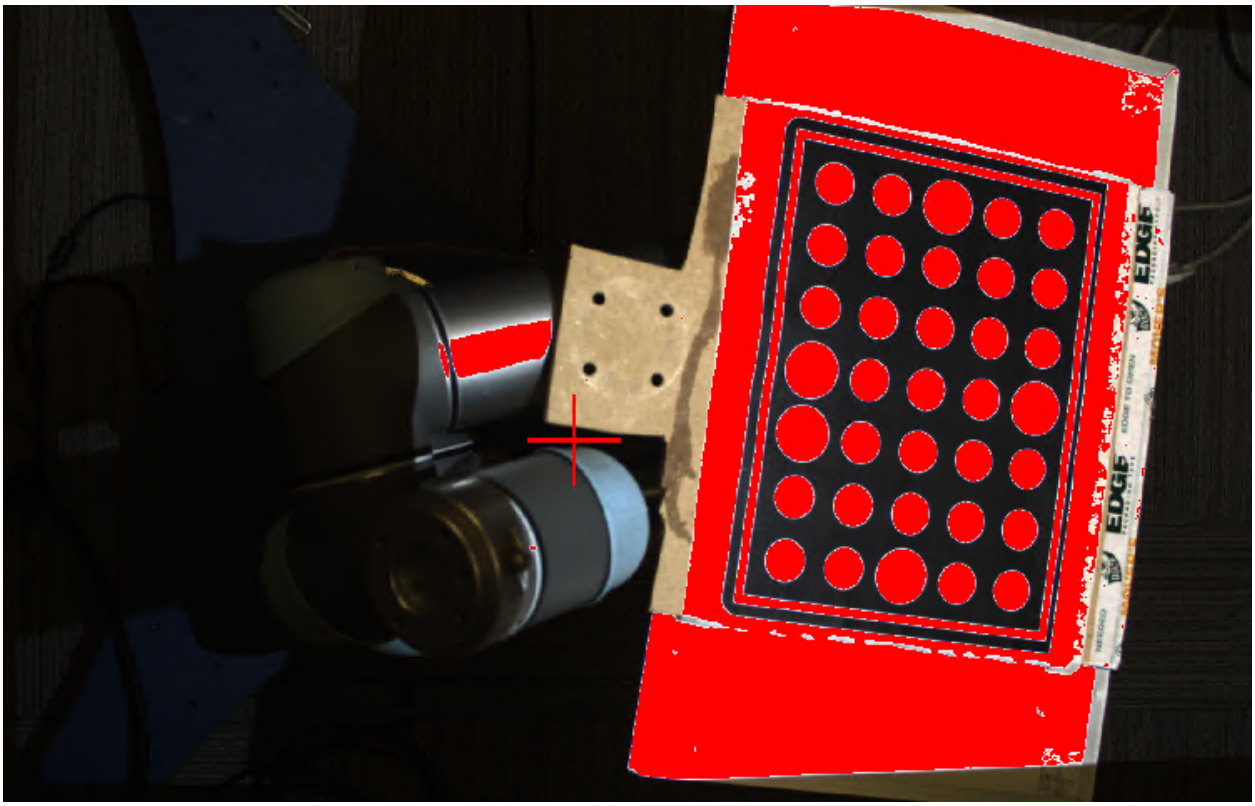


Fig. 20: Area in red marks the over-saturated area

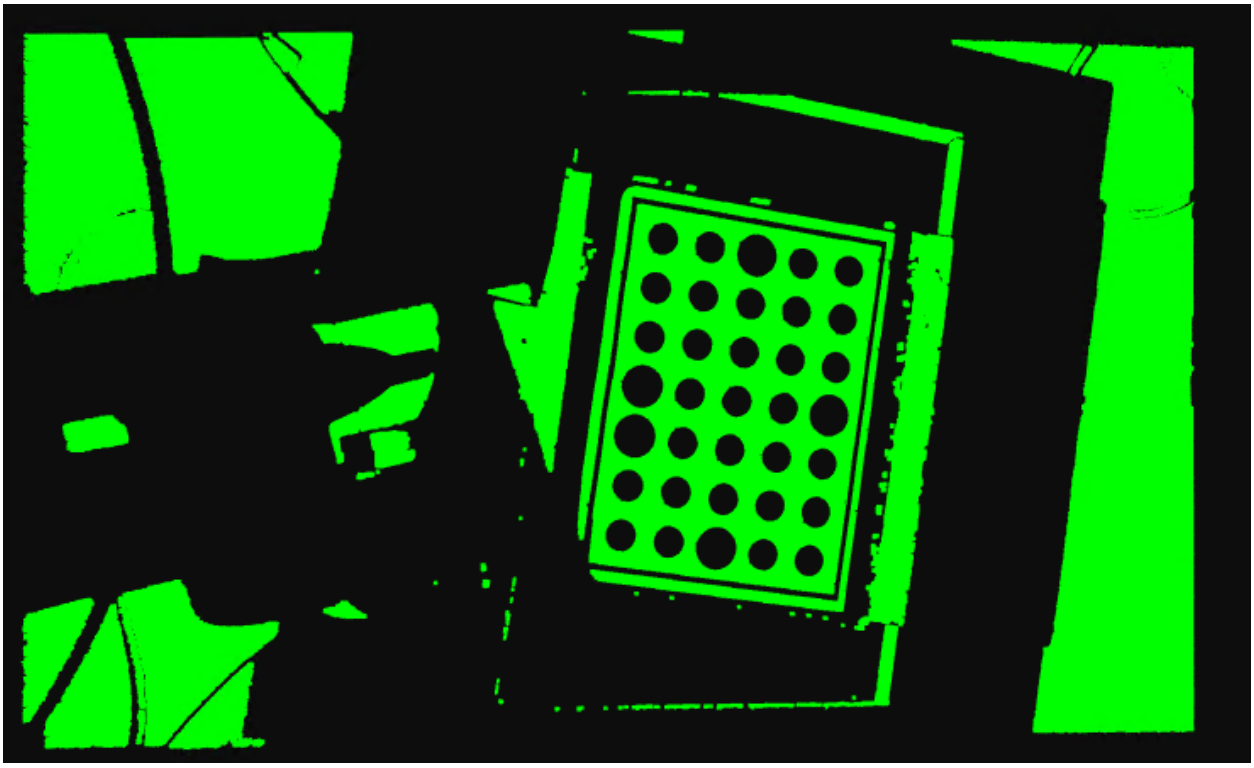
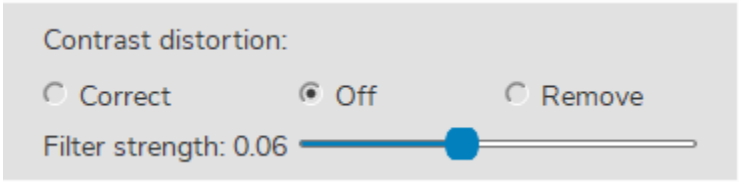


Fig. 21: Point cloud with saturation filter

Contrast Distortion Filter



Contrast distortion occurs due to imperfections in the lens and optical phenomena like diffraction and chromatic aberration. It appears when there is an abrupt contrast change from a highly absorptive to a reflective surface (e.g. in a black to white transition on a checkerboard), which leads to measurement errors in the 3D point cloud.

- If “Remove“ is selected, regions of high contrast distortion will be removed from the 3D point cloud.
- If “Correct“ is selected, measurement errors caused by contrast distortion will be compensated based on a “Strength“ value user set on the GUI. The higher “Strength“ value user use, the more measurement error will be compensated. Note that, it’s possible to over-compensate the measurement error which looks like “opposite“ contrast distortion.

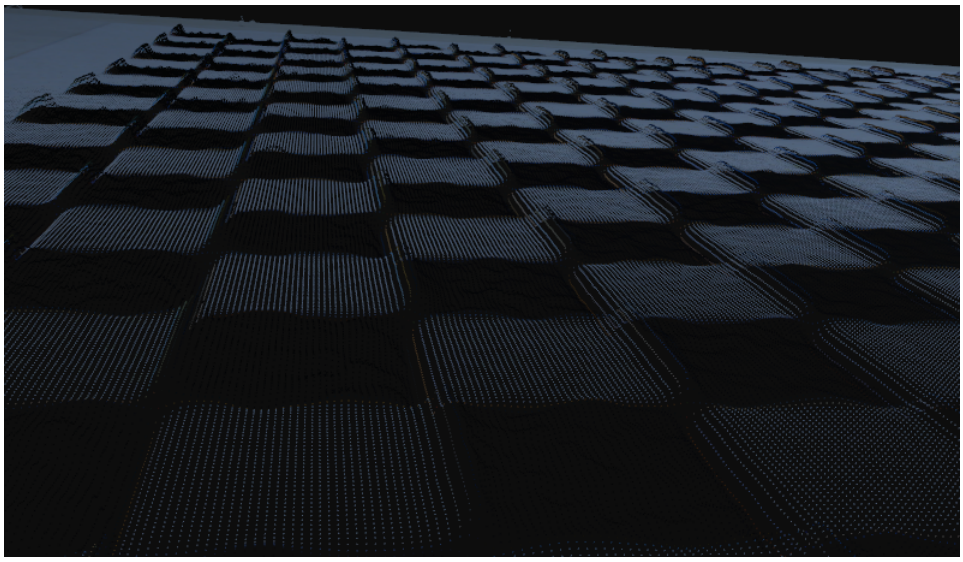


Fig. 22: Point cloud color image (contrast distortion visible)

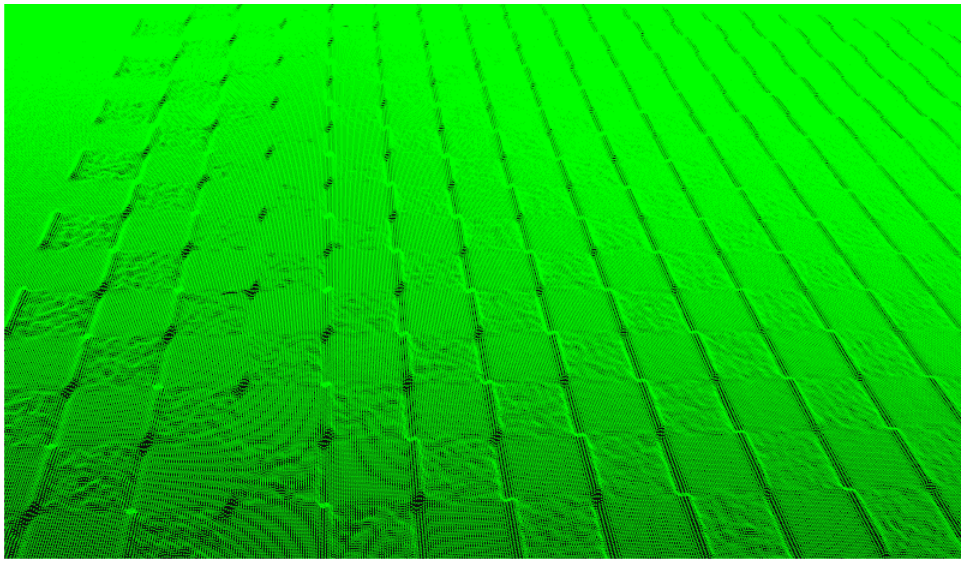


Fig. 23: Point cloud green image (contrast distortion visible)

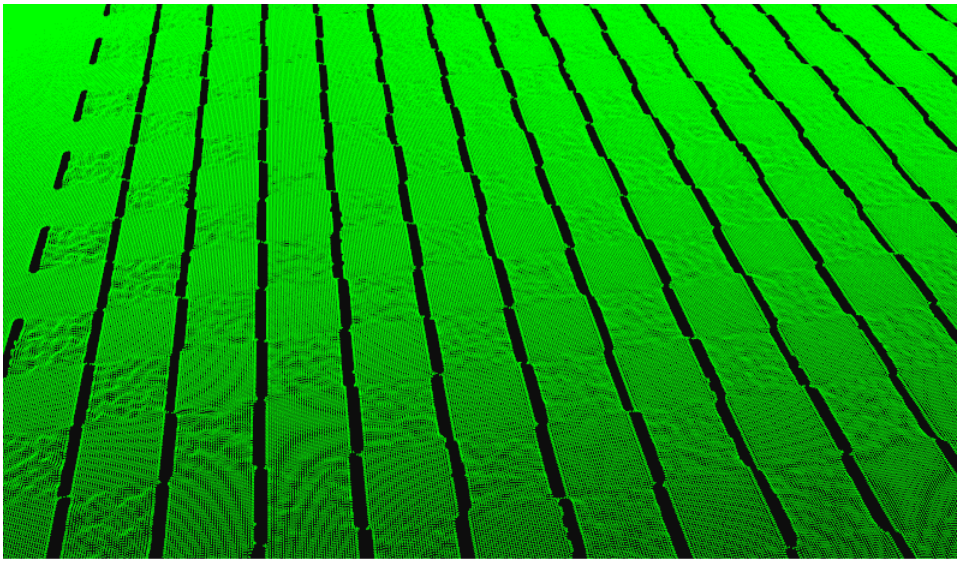


Fig. 24: Using “remove” for contrast distortion

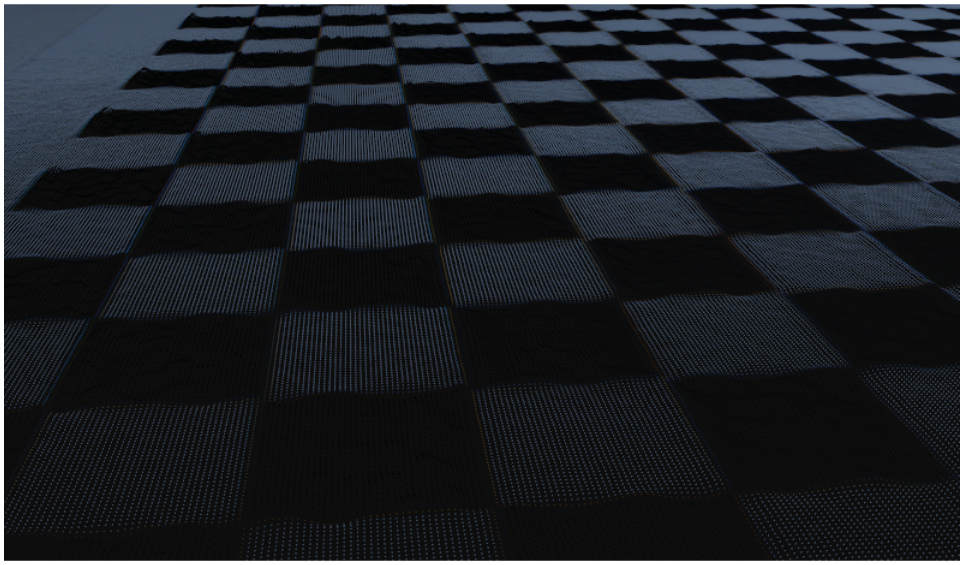


Fig. 25: Using “correct” for contrast distortion

Color Balance

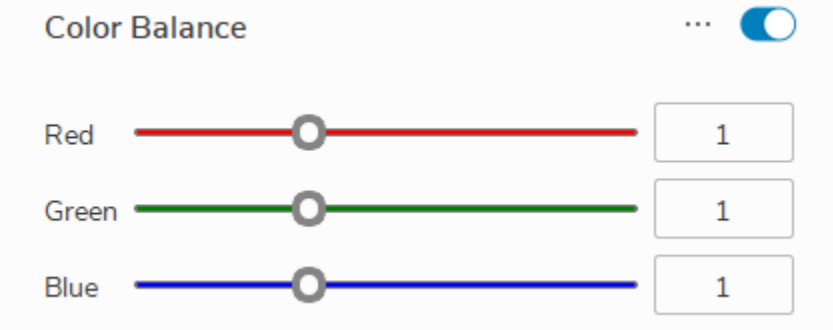


Fig. 26: Color balance controls

By increasing the corresponding value of R/G/B, you can make the image closer to that color. The default value is 1, the minimum is 0.5, and the maximum is 2.

Used when the color of the background environment is slightly inclined to a certain hue. Adjust the balance of other colors. Usually, it does not need to be set up.

Preview options are not available, a single capture is required to see the image effect.

Fig. 27: R biased color balance

Clicking the “AWB” (as the image above) to use the auto white balance tool. After clicking the “AWB”, drag to select an area of neutral color (white or gray) and click confirm. R, G, B values will be computed so that after applying color balance, the selected area can be adjusted to white.

Fig. 28: G biased color balance

Fig. 29: B biased color balance

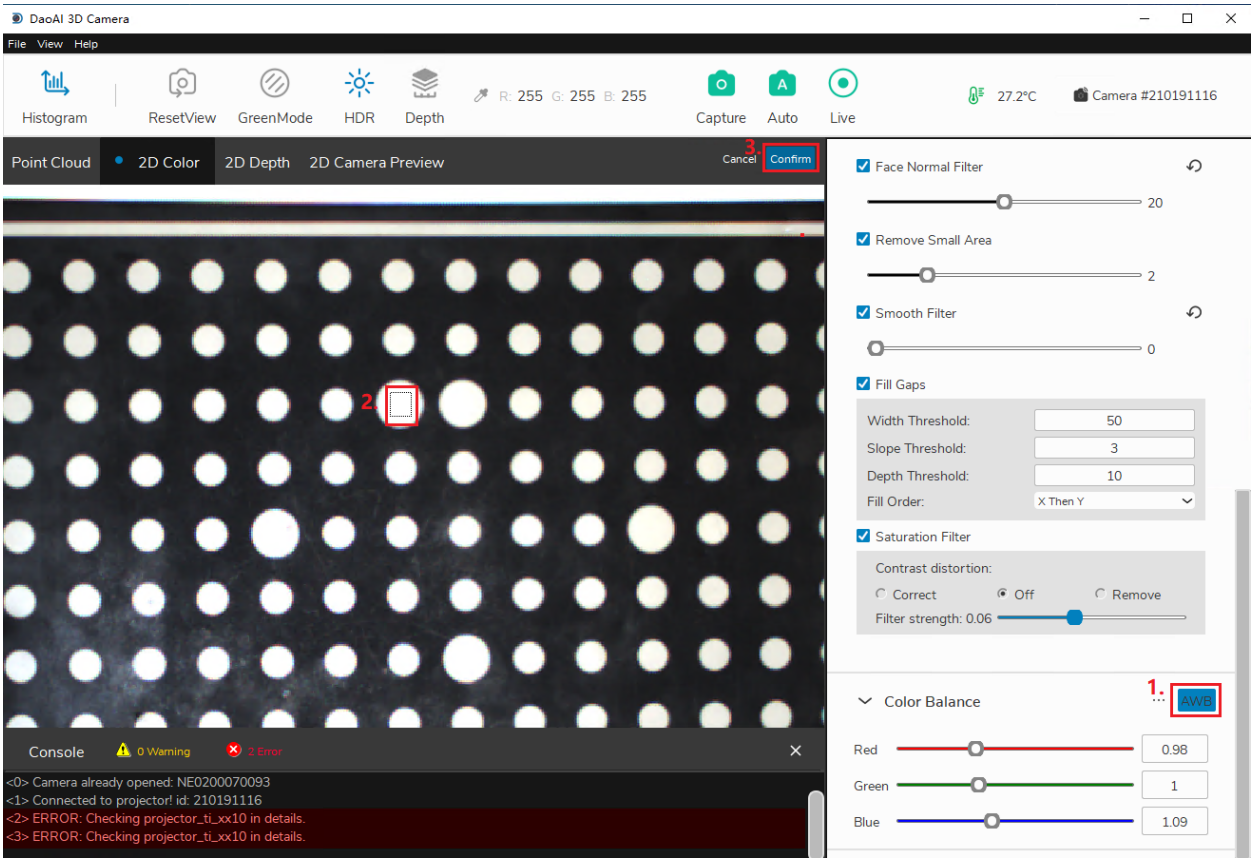


Fig. 30: drag and select an area in the display

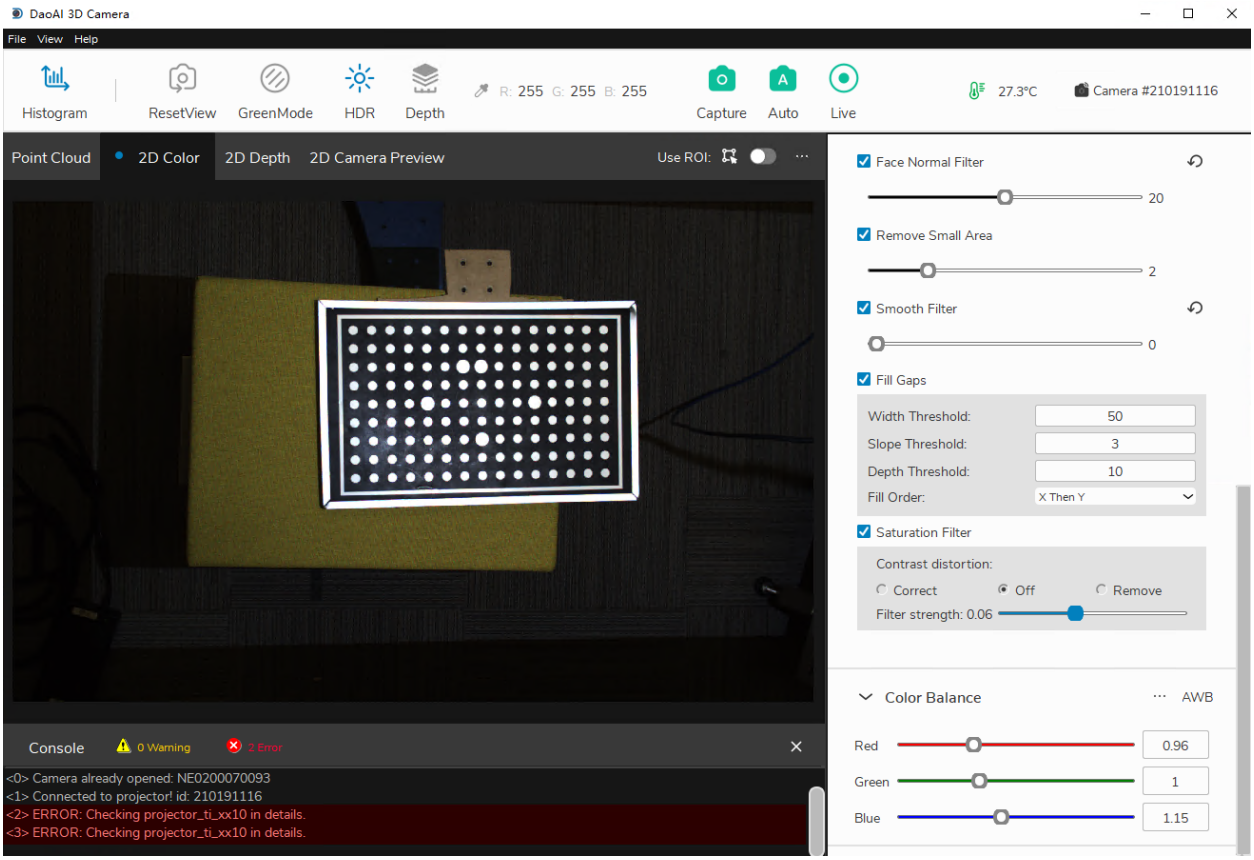


Fig. 31: image after auto white balance

3.5.6 Green Mode

The Green Mode switches between RGB point clouds and green point clouds display.

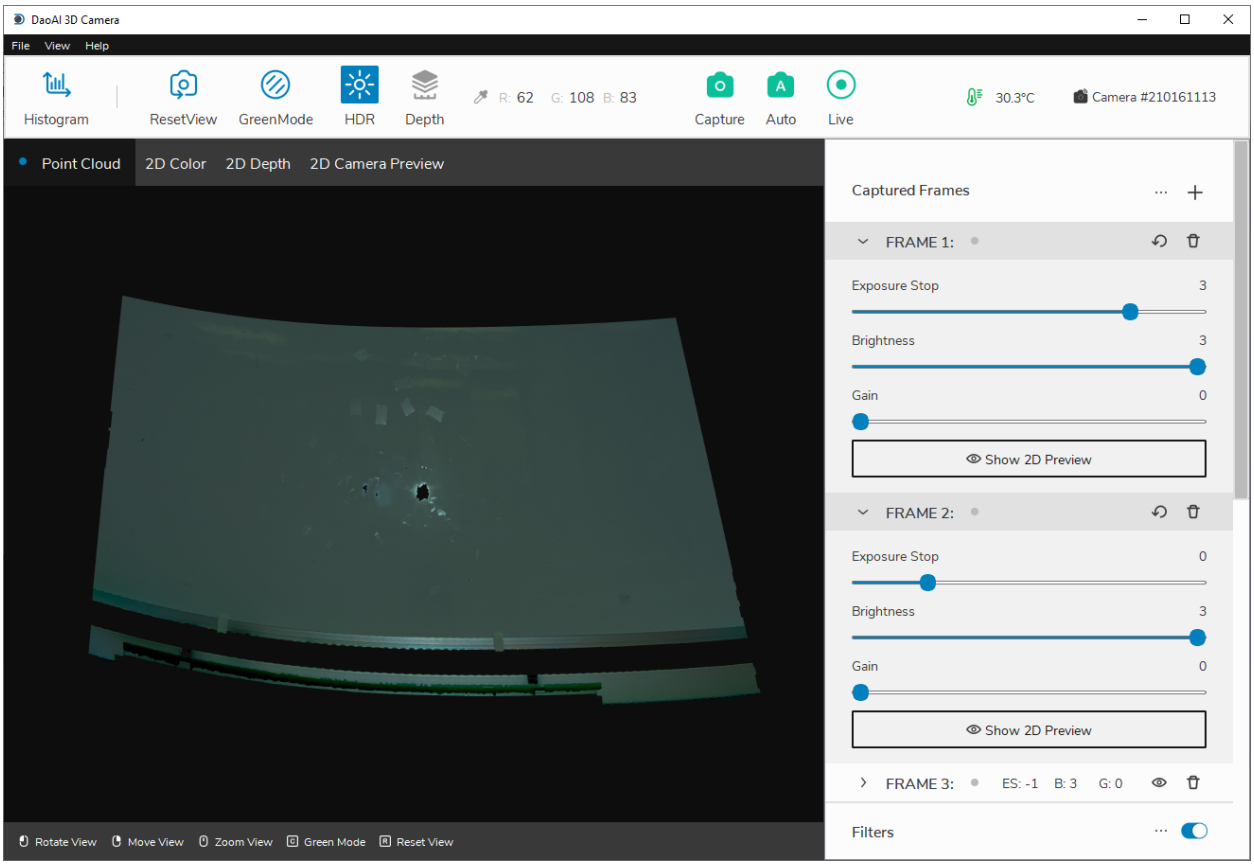


Fig. 32: RGB point clouds

3.5.7 HDR

The HDR (High Dynamic Range) mode recreates an image by increasing the difference between the darkest and the lightest parts in an image.

When HDR mode is turned on, the dynamic range of exposure is increased to larger than that of ordinary digital image technology, so that the details of the previous dark parts can be highlighted.

On the other hand, if the original image is overexposed, the HDR mode will utilize lower exposure images to recreate the final image.

Below are the same colour image with HDR off and HDR on respectively:

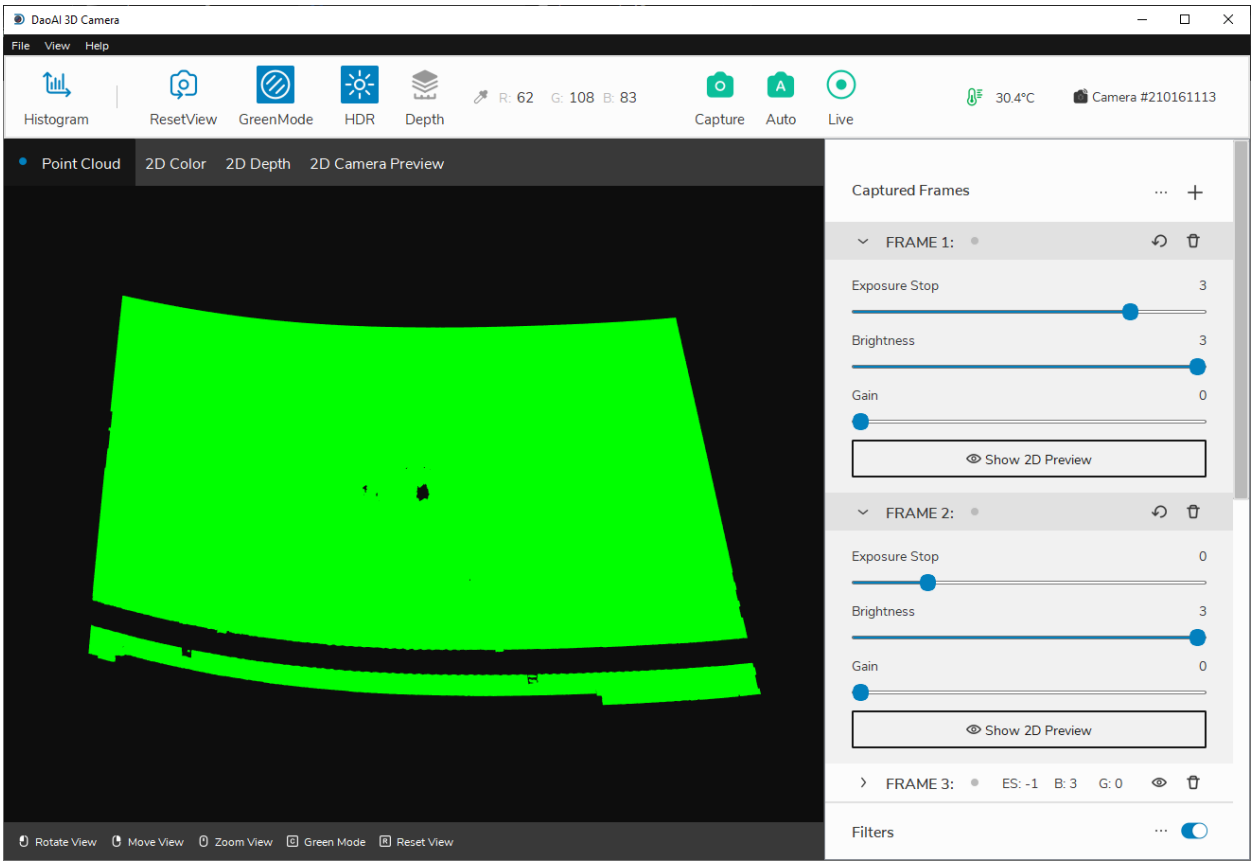


Fig. 33: Green Mode point clouds

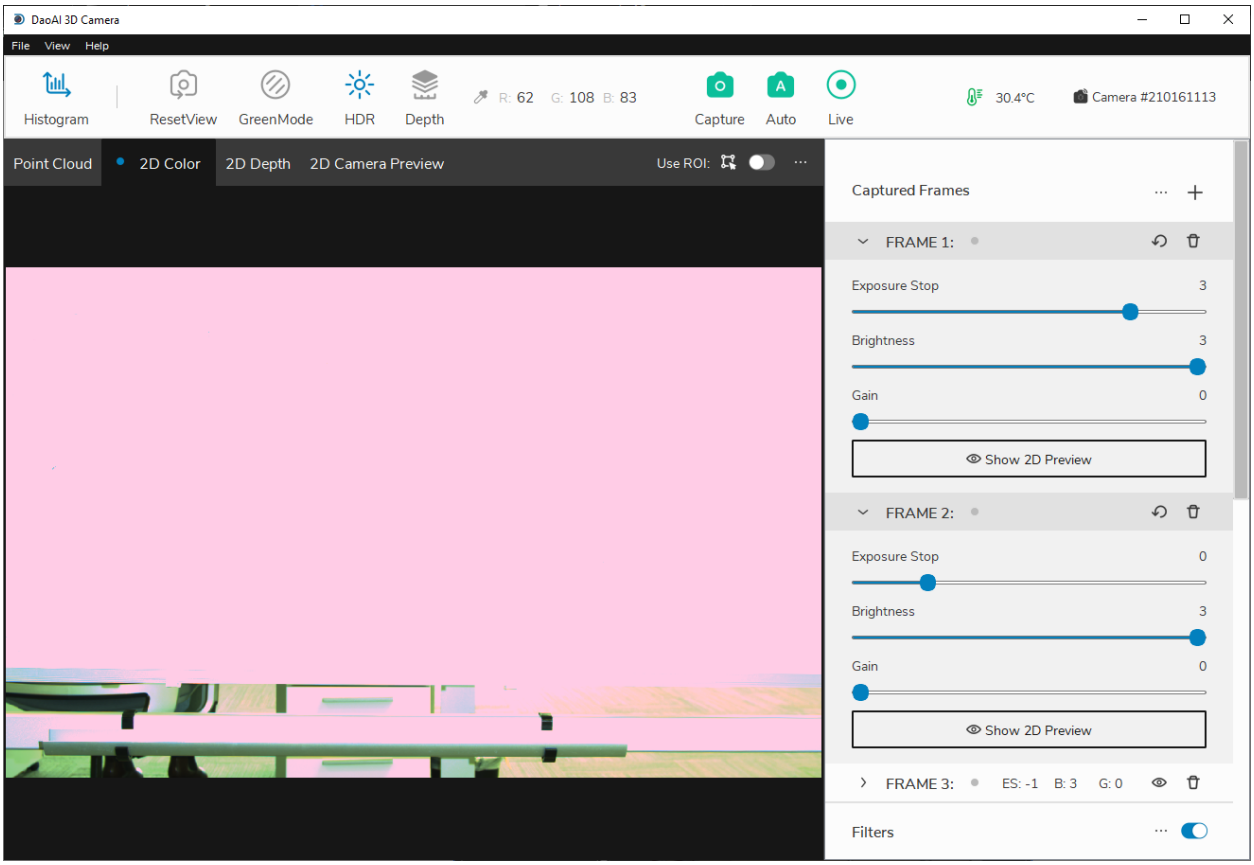


Fig. 34: Colour map with HDR off

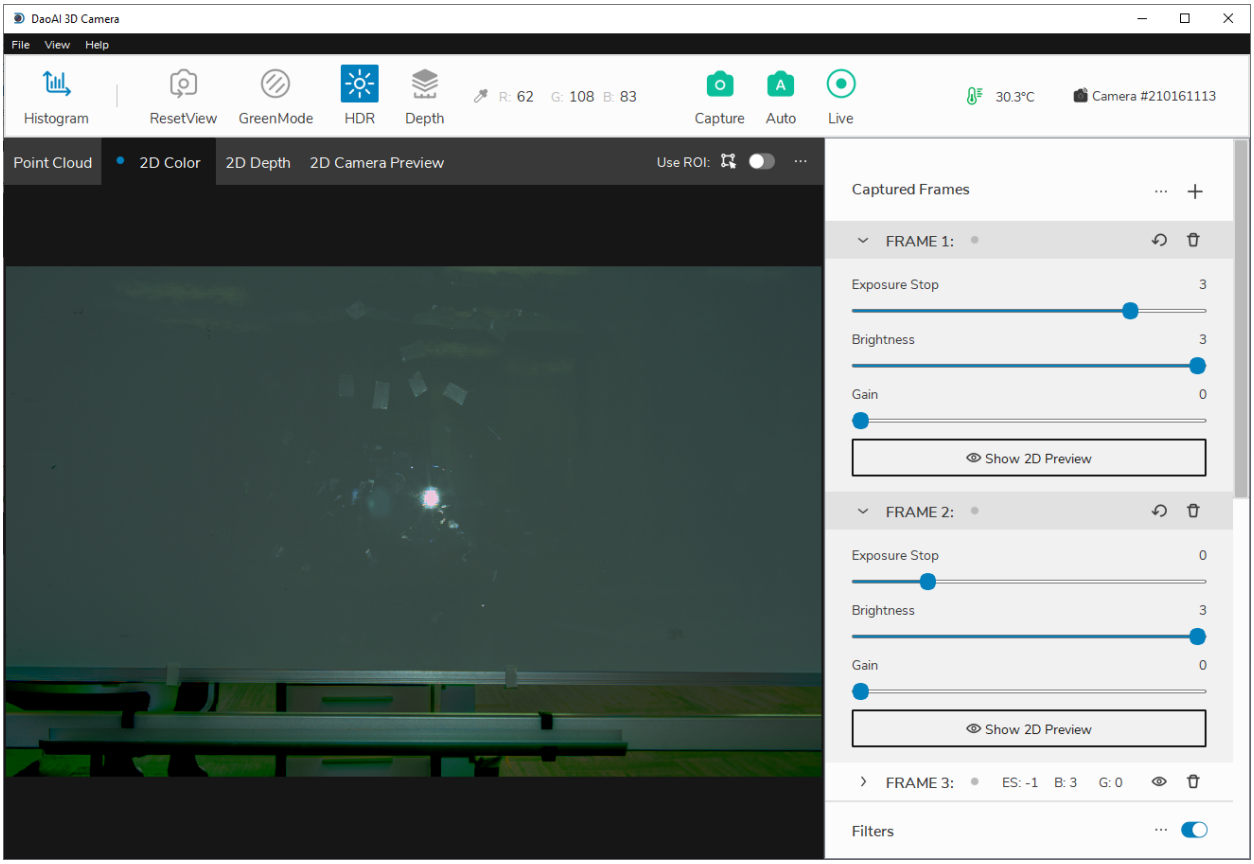


Fig. 35: Colour map with HDR on

3.5.8 Histograms

The histogram window provides a visualization of the distribution of the RGB values (0-255) of all the pixels in the image.

Goal: To have the highest column in the logarithmic histogram to around 128 to avoid 255 overexposures. This can be achieved by adding frames and adjusting exposure levels.

Click "Histogram" in the upper left corner to toggle the histogram window. The histogram plot will pop up on the screen automatically. Inside this window, there are two different tabs for switching between graphs of a linear distribution and logarithmic distribution. Overexposed pixels are marked a red color so you can easily see them. If there are a large number of pixels in 255 range, the image is overexposed. Shorter exposure time or lower brightness should be chosen to limit the overexposure.

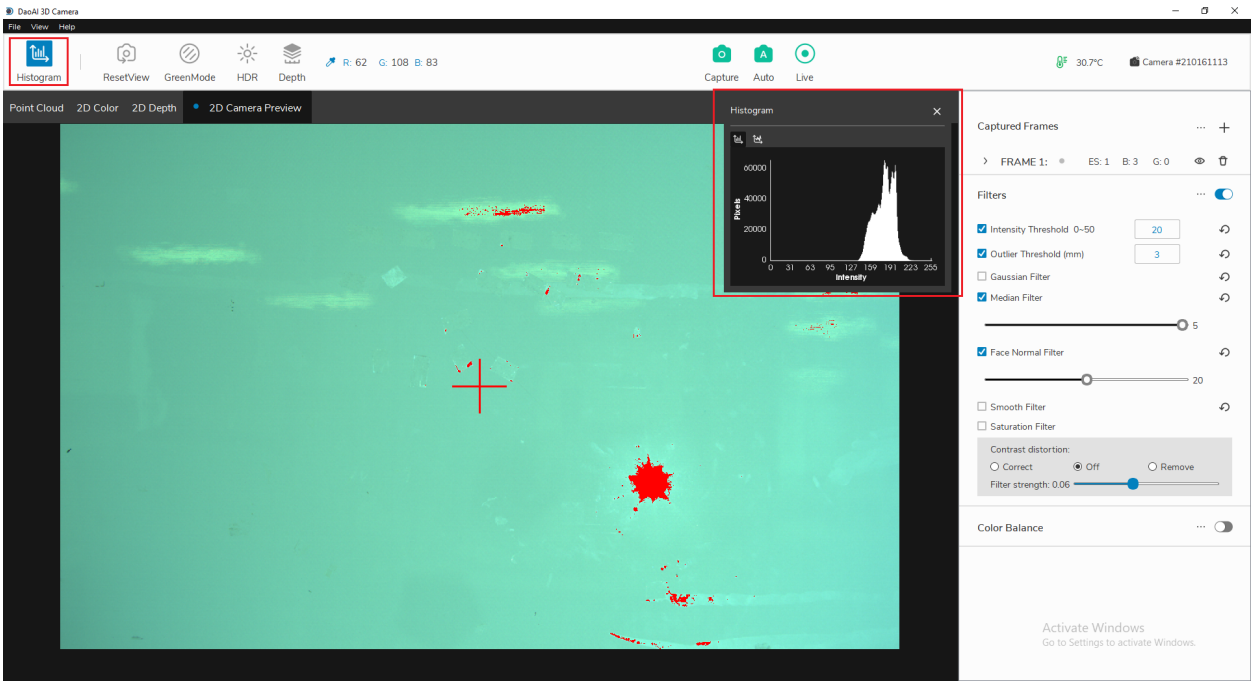


Fig. 36: Linear distribution

Logarithmic distribution can help adjusting the brightness to desired range.

For example, as the image below, we want to adjust the peak brightness from "16-32" to "128-256".

Recall that for frame parameters, an increase of one in any field will double the brightness of the final image, and the ranges in logarithmic distribution also doubles.

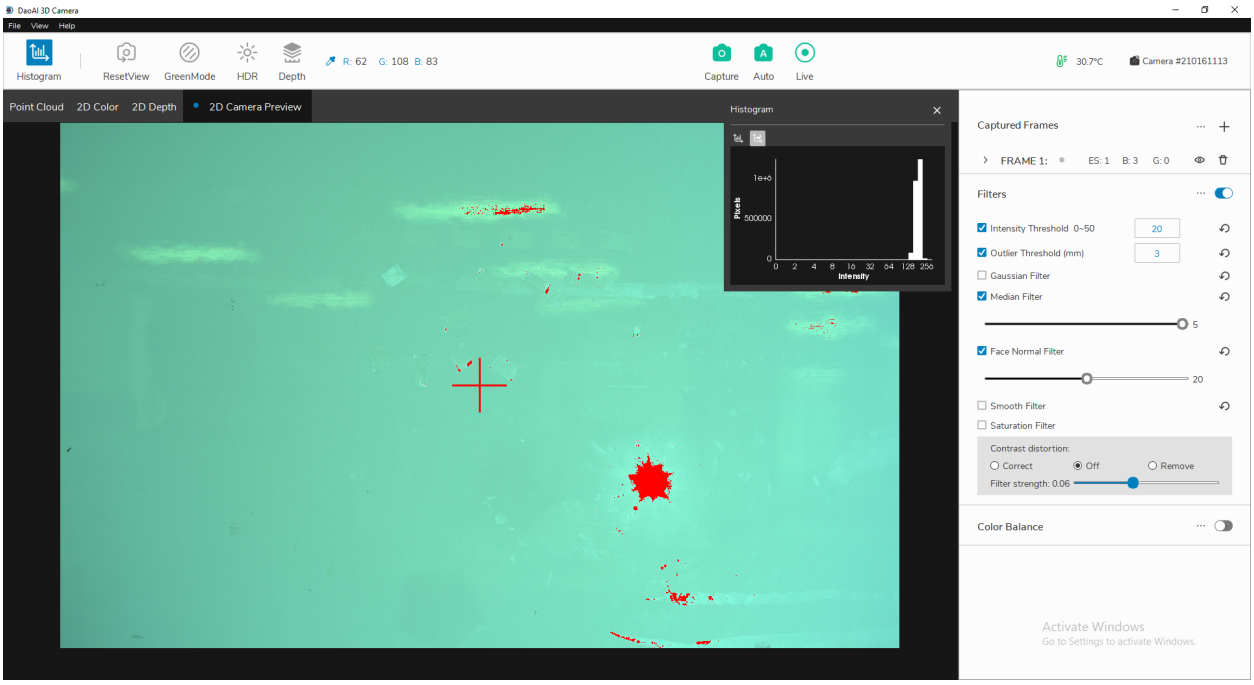
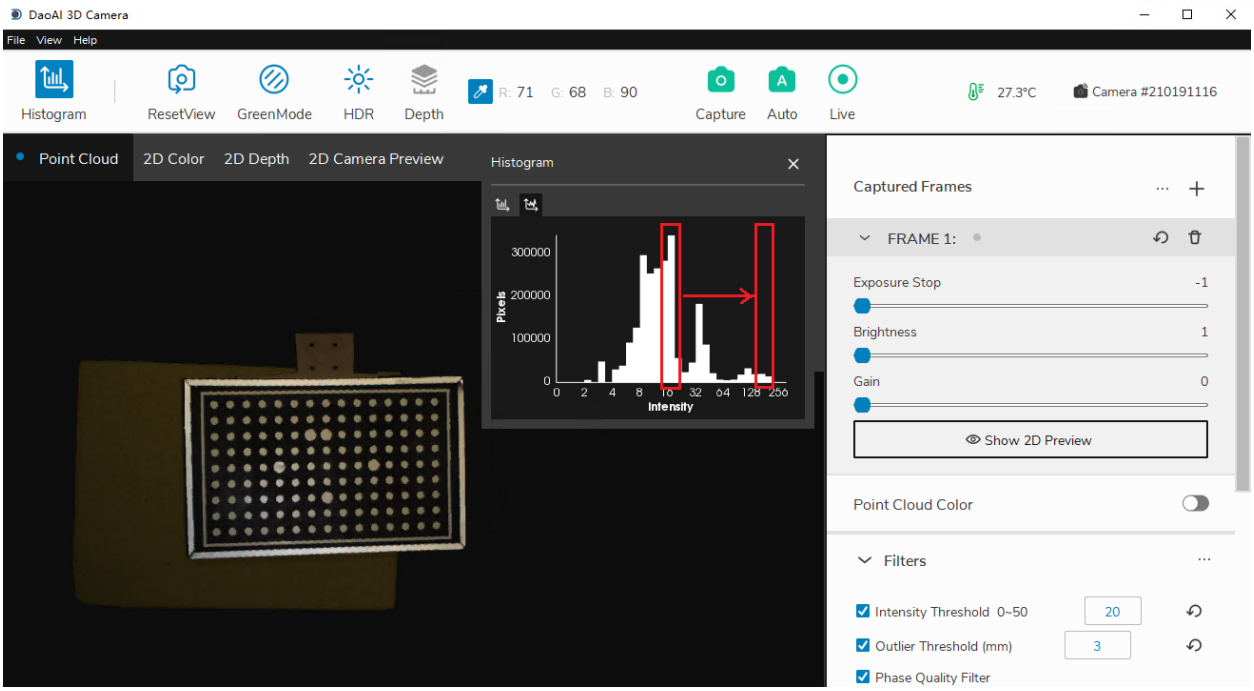
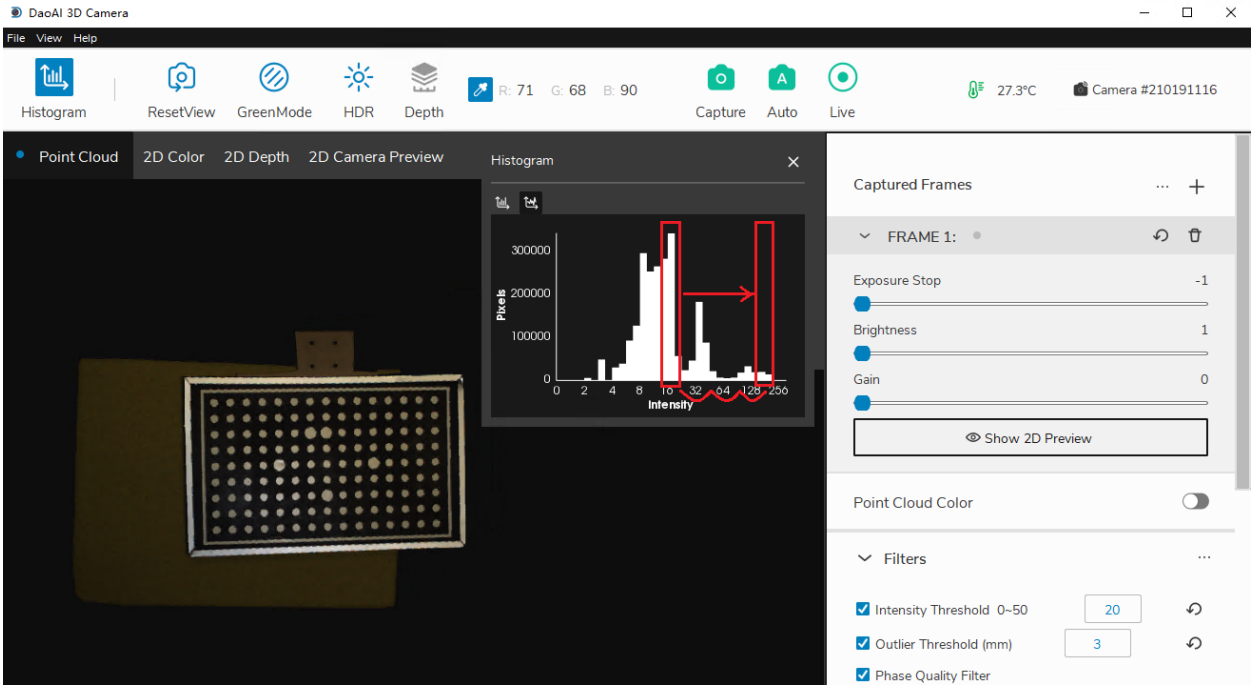


Fig. 37: Logarithmic distribution





Obeserve that there are 3 intervals from range “16-32” to “128-255”. Then adding 3 to frame parameters can move the peak brightness to our desired range.

3.5.9 Temperature Regulation

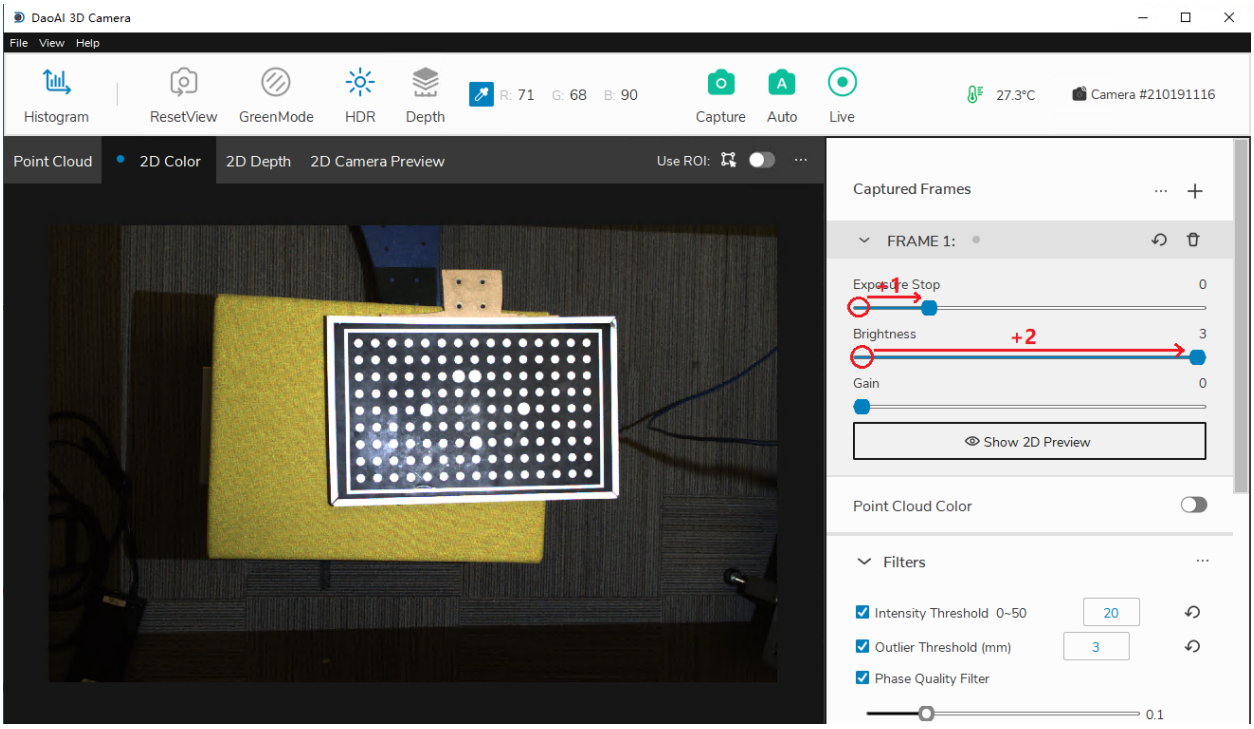
If a temperature sensor is available in your 3D system, the system will display the corresponding temperature in the top right corner of the main menu. By default, if the temperature sensor is available, a temperature control system will be enabled.

Modes There are two modes for temperature control: normal and regulating

Normal: This is the regular mode in which all software features are enabled including captures, 2D previewing, etc. While the temperature control status is normal, the temperature icon in the main window will appear green.

Regulating: By default, temperature regulation will be triggered if system temperature is above 70 degree Celsius. In this mode, the temperature of the system is abnormal and the main capture features (Auto, Live, Capture, Preview) are disabled in order for the system to regulate the temperature. When in this mode, a temperature control algorithm will be run in order to normalize the system again. While the temperature control status is regulating, the temperature icon in the main window will appear red.

While Camera Studio is regulating temperature, some functionality is disabled in order for the temperature regulating feature to work effectively. You will have to wait for the temperature to be regulated to a suitable level, or disable the temperature regulation feature before you can use these functionalities are re-enabled.



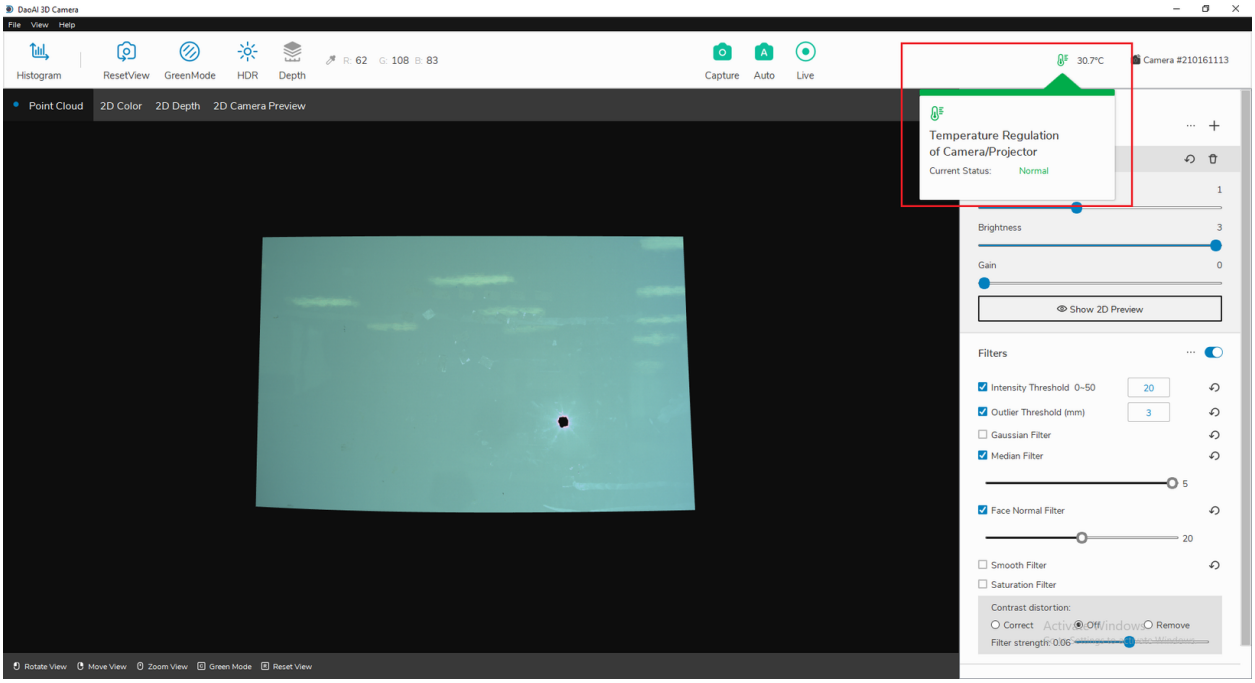


Fig. 38: Temperature icon is green

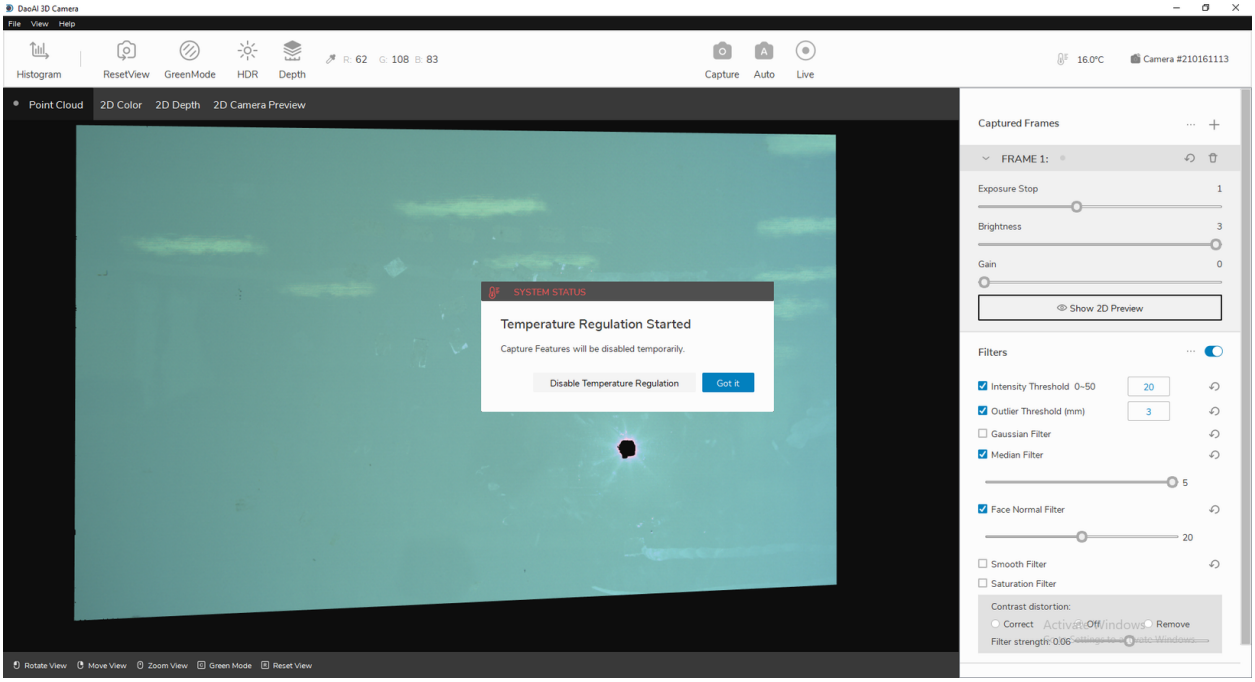


Fig. 39: Notification that temperature regulation has started

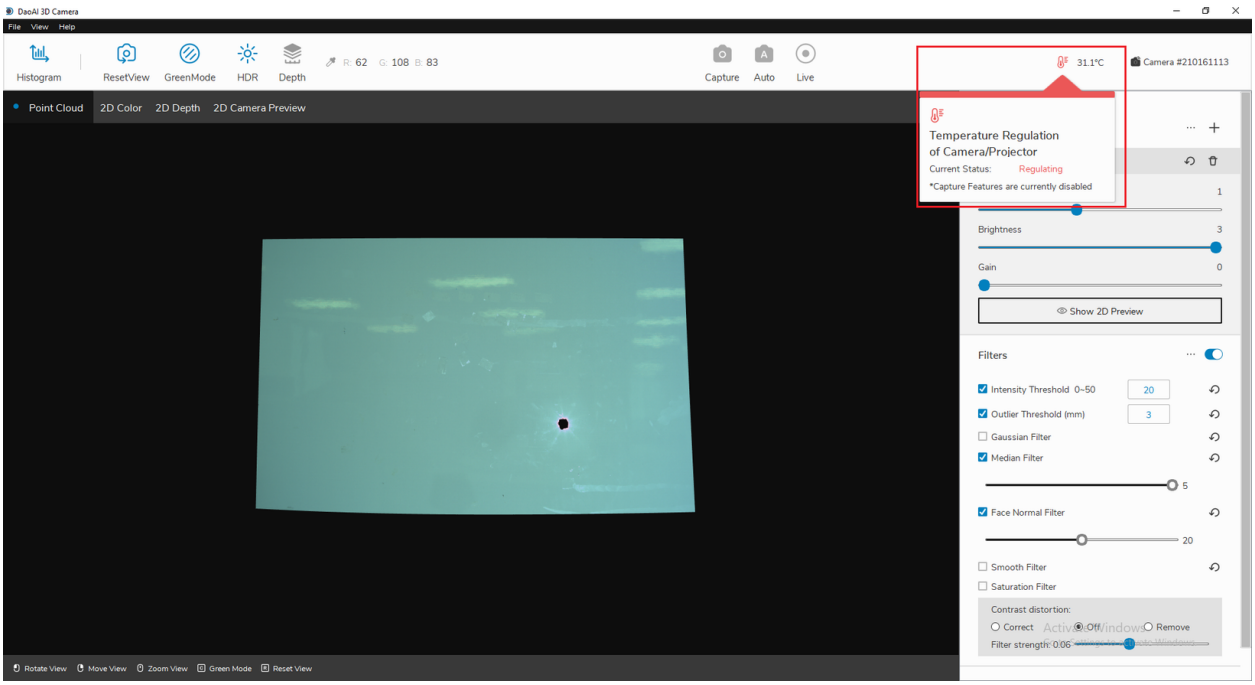
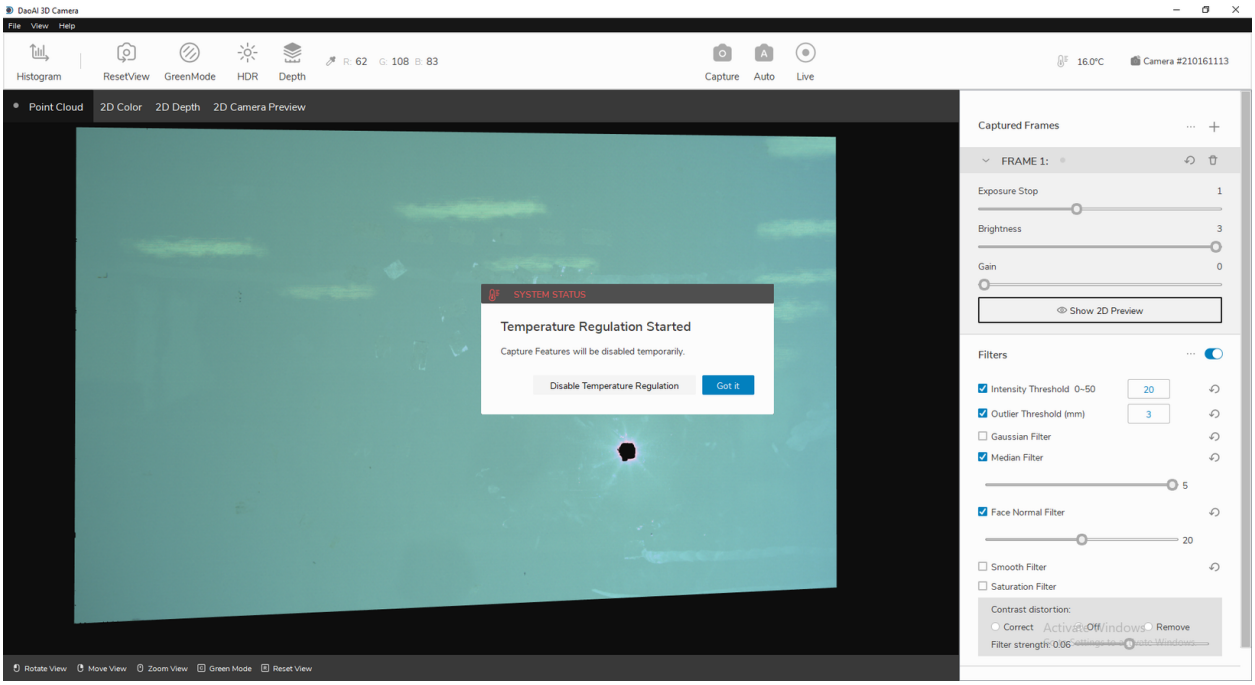


Fig. 40: Temperature icon is red

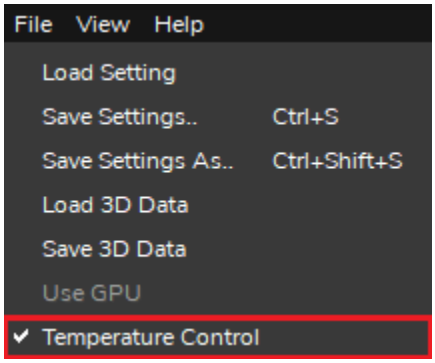
Disable Temperature Control

When camera's temperature sensor is malfunctioning, you can disable the temperature control to allow normal usage of Camera Studio.

You can click "Disable Temperature Regulation" when the warning dialgue is prompted to disable Temperature Control.



Or you can disable Temperature Control from the menu bar: Click on "files -> temperature control" and uncheck it to disable.



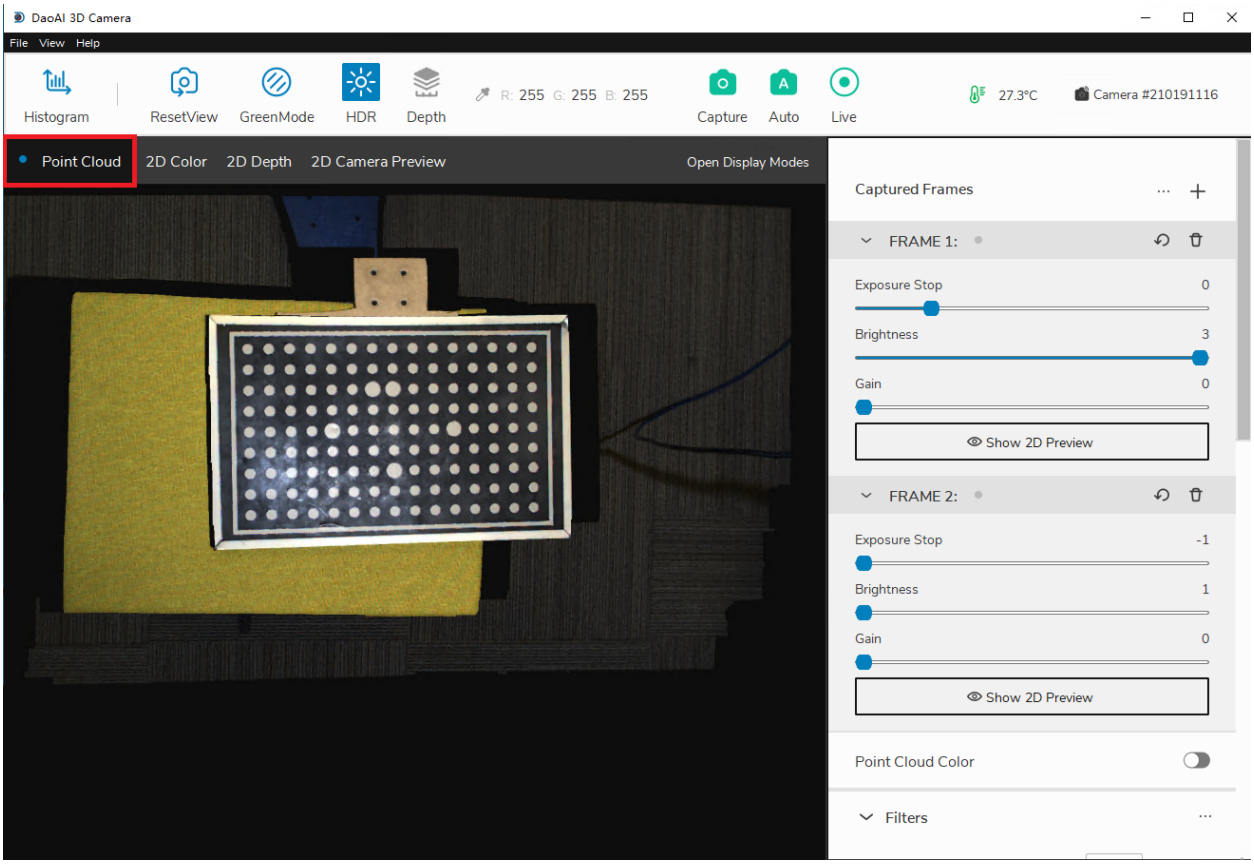
3.6 Available Views

We provide a variety of different display options to visualize the captured data:

- *Point Cloud*
- *2D Color*
- *2D Depth*
- *2D Preview*
- *Regions of Interest*

3.6.1 Point Cloud

The Point Cloud tab displays the 3D point cloud model from the captured images. The point cloud view can be rotated and moved around using the mouse.

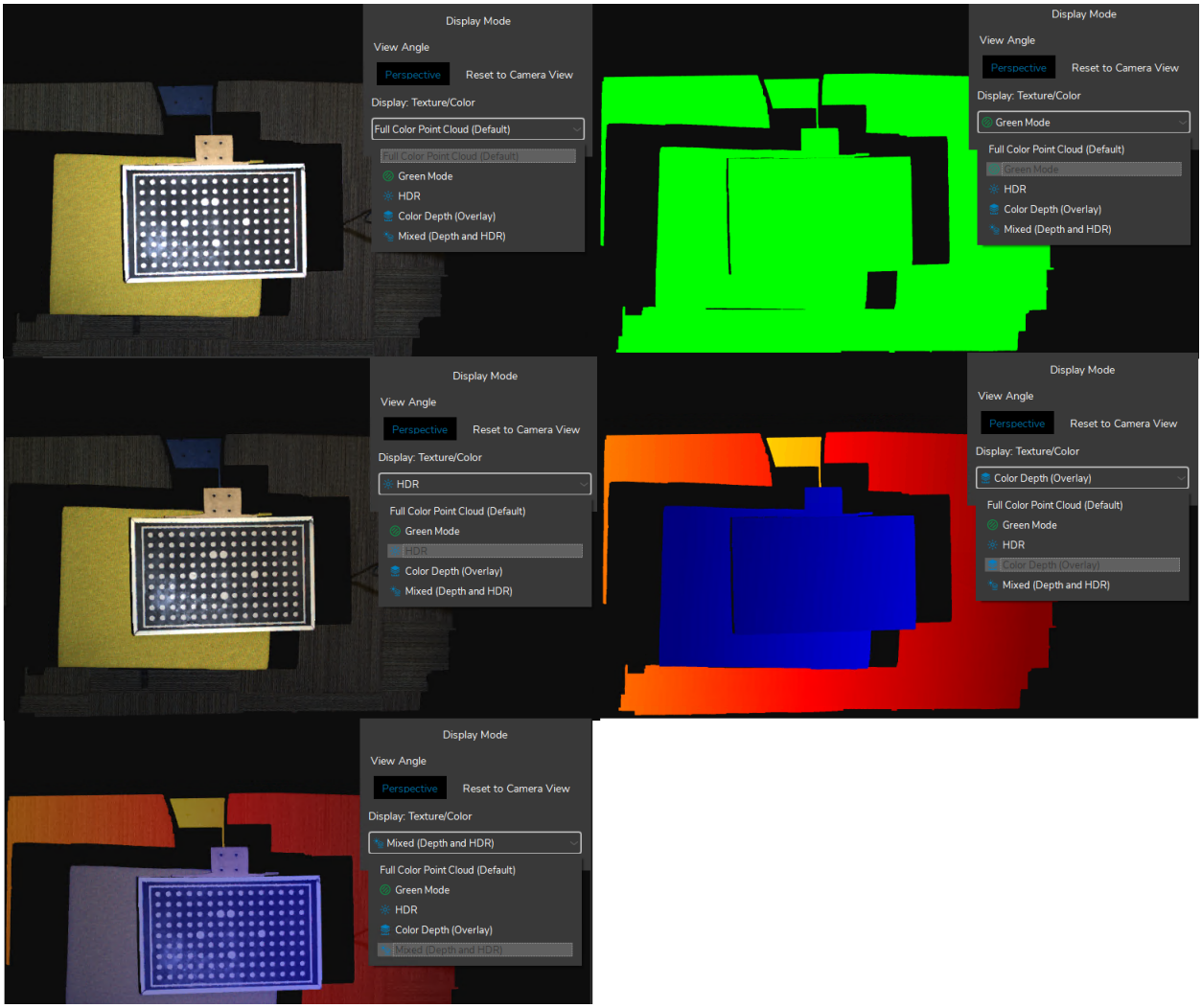


- The left mouse button rotates the point cloud.
- Right mouse button drags point cloud.

- Mouse wheel zooms in and out of point cloud.
- Keyboard button “c” toggles RGB point cloud and pure green point cloud.
- Keyboard button “r” restores the camera view to its original state.

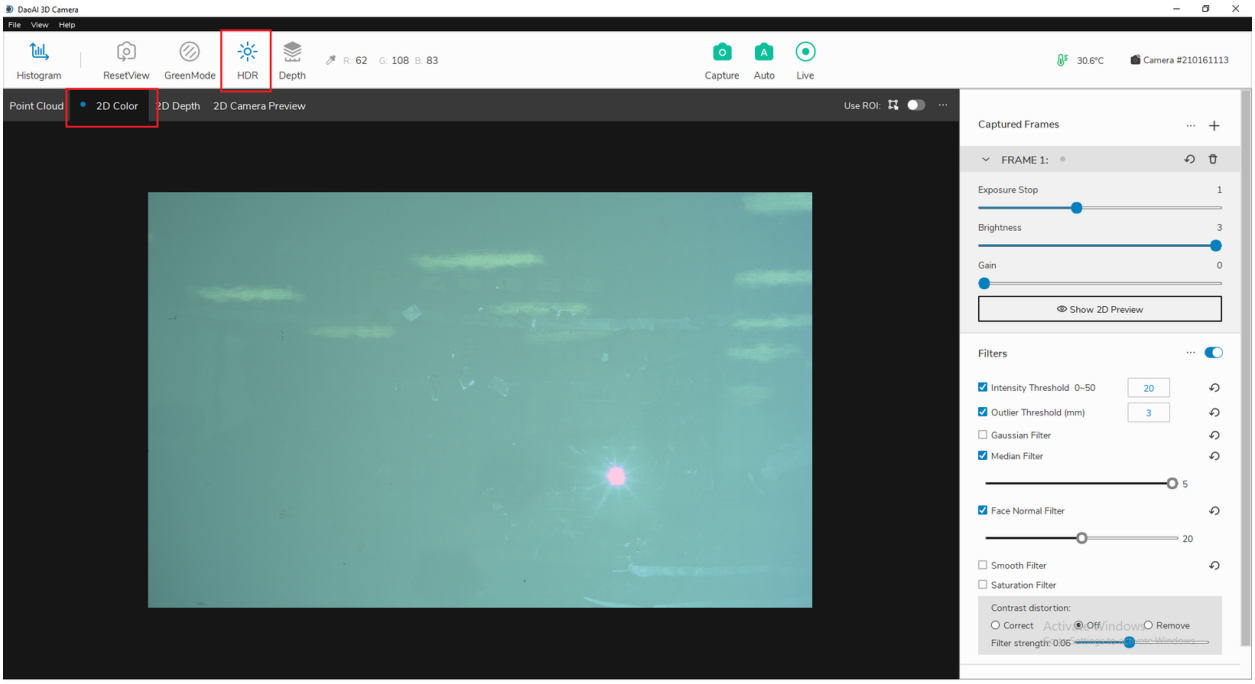
The point cloud can also be saved as a .daf, .ply, or .pcd file. This is done under “File” → “Save 3D Data”. Saving the whiteboard object into ply format can be used to detect the accuracy of 3D camera depth measurement. Similarly, a point cloud can be loaded via “File” → “Load 3D Data” - once loaded, the point cloud will be displayed in this tab.

The point cloud’s color can be switched between 5 different modes: default color, green mode, HDR, pseudocolor, and mixed color (a combination of the default and pseudocolor modes). To change the mode, click the display mode dropdown menu in the Point Cloud tab and select the color type.



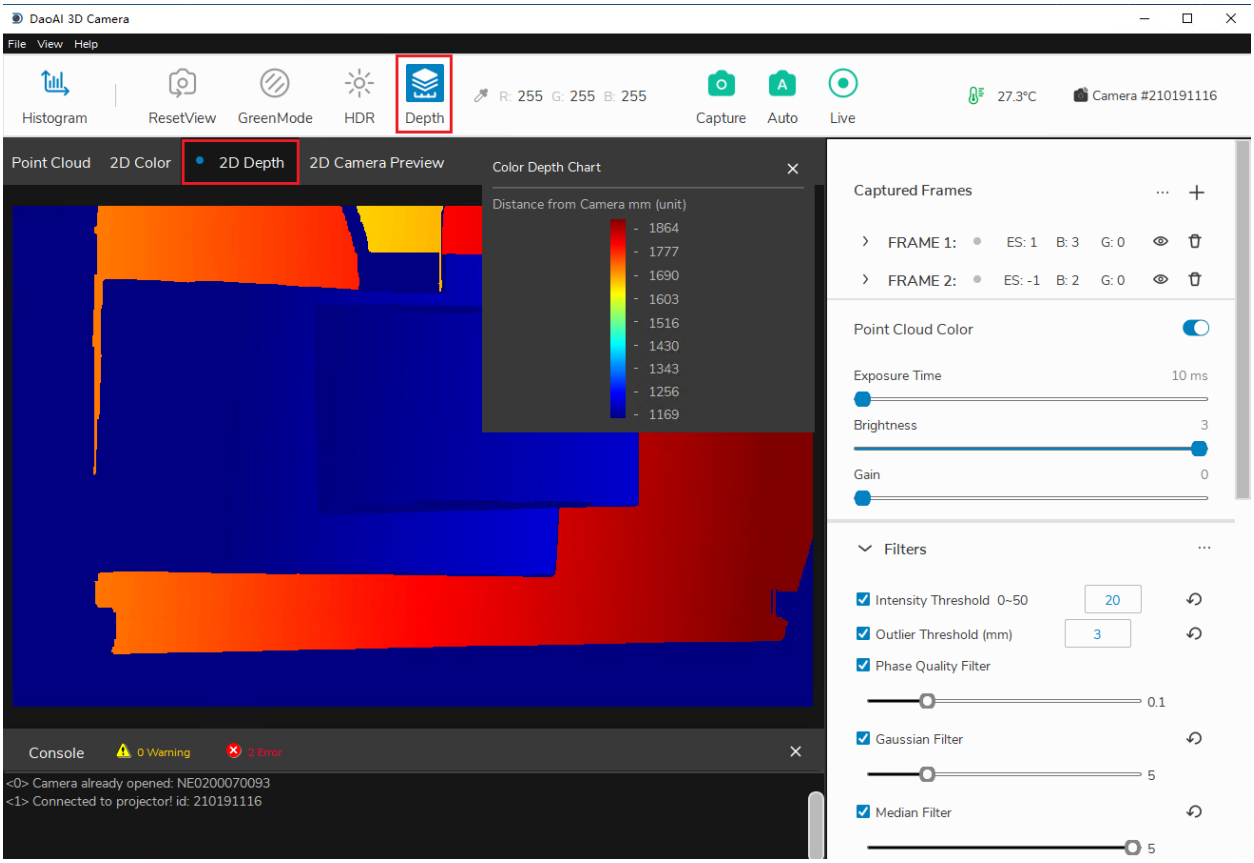
3.6.2 2D Color

The 2D Color tab shows the captured image and in this tab, the HDR mode can be selected. HDR will change the color of the image to maximize its dynamic range. HDR is useful when we have multiple frames of with settings to capture the most dynamic range.

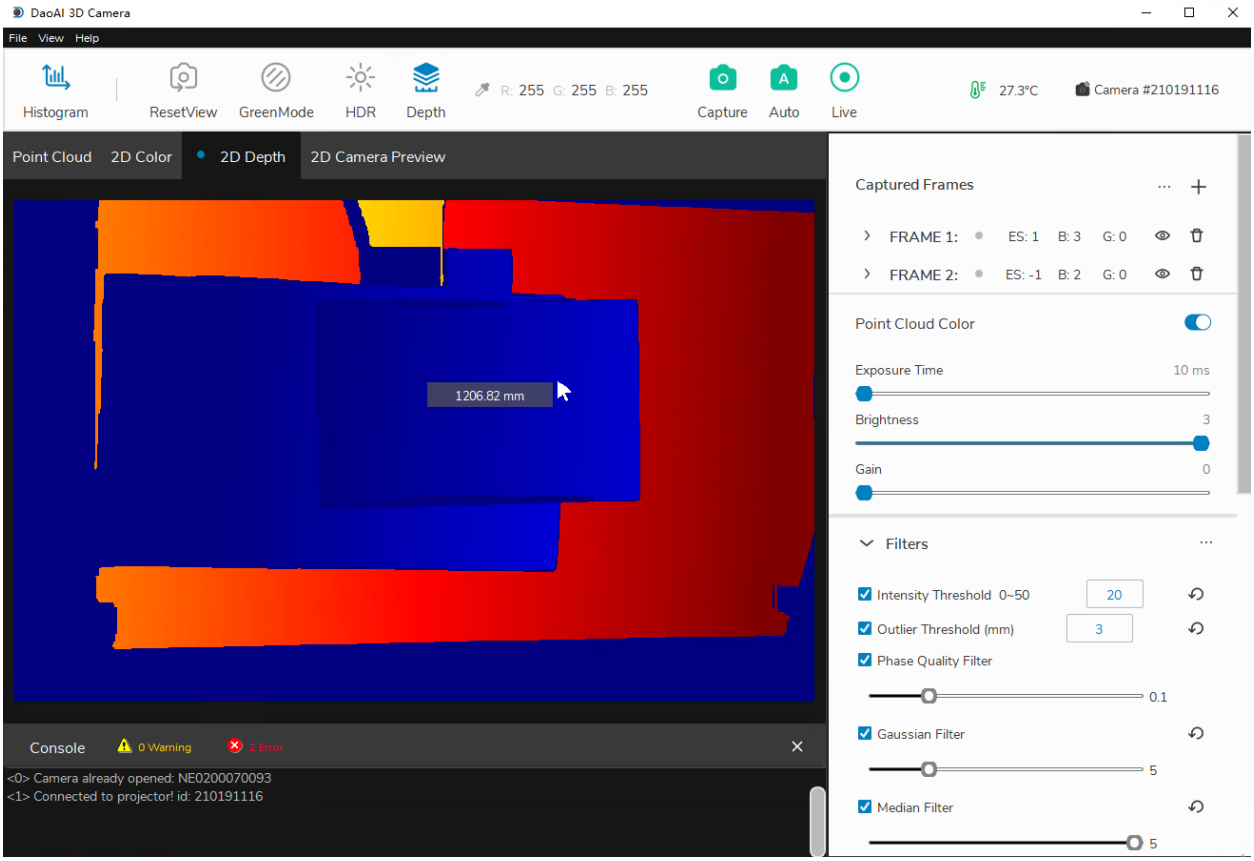


3.6.3 2D Depth

The 2D Depth display tab encodes the depth values of each valid pixel as its color. Blue represents a short distance, red represents a long distance (in millimeters) from the camera. To analyze the specific values, you can toggle the "Depth" button and the depth chart legend will pop up.

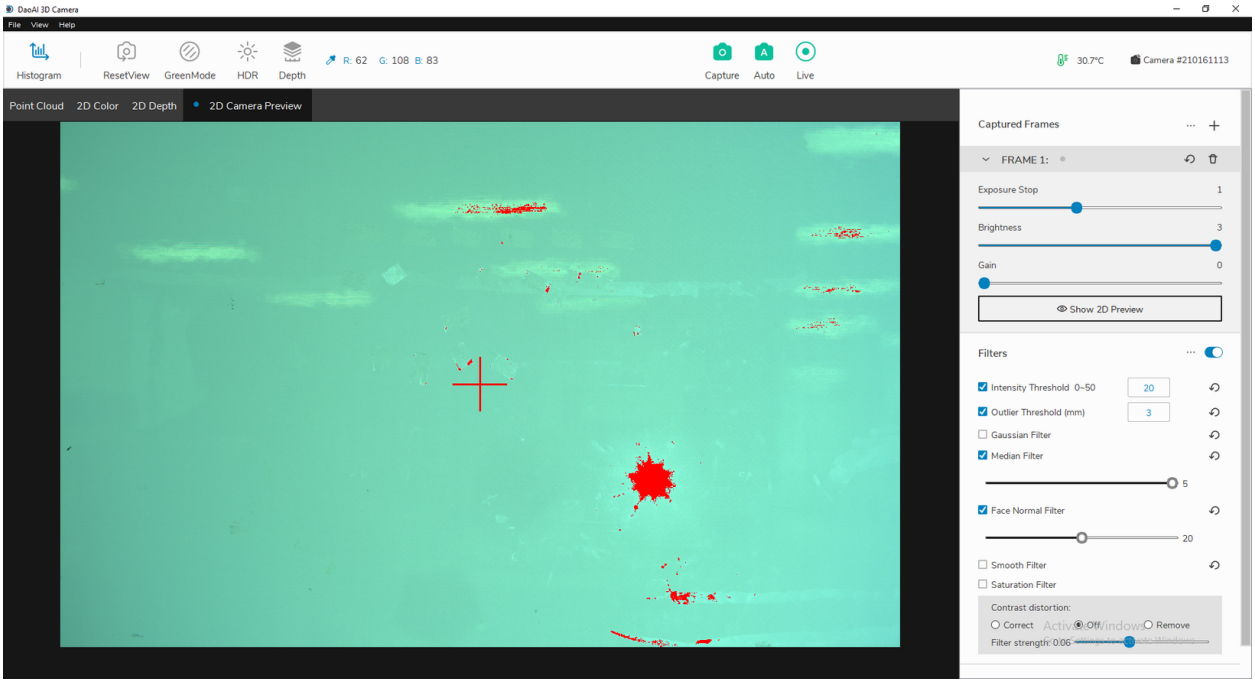


You can also hover your mouse over the image to see depth value.



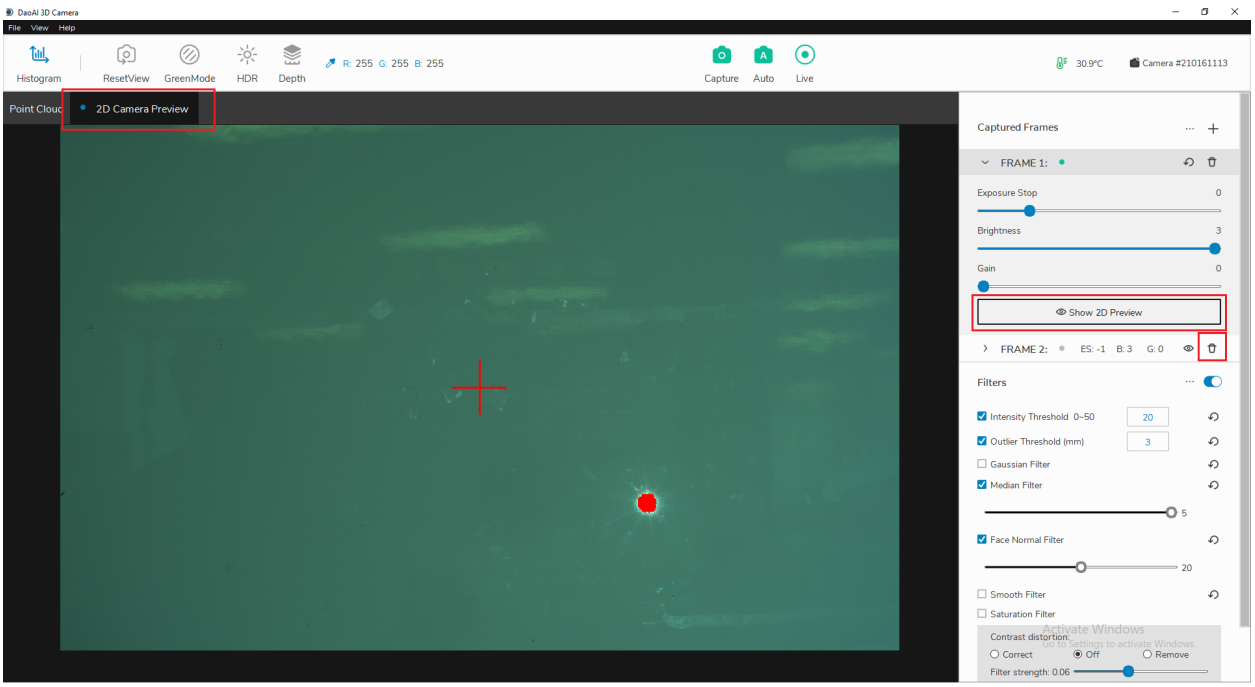
3.6.4 2D Preview

The 2D camera preview is used to confirm the position of the photographed object and the RGB value of the image in advance. (For example, the picture is a preview of a binocular camera view, you can hover the cursor over to different locations in the picture to view the local RGB values.)

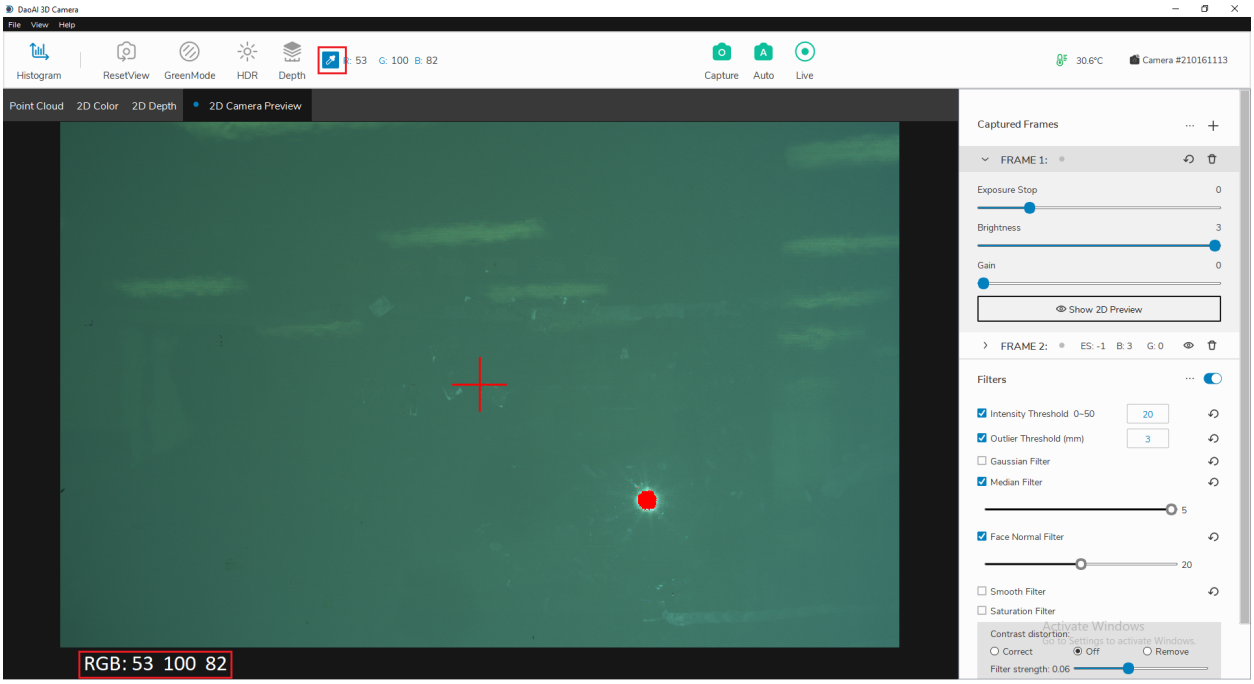


The 2D Preview feature is used to retrive a preview image of the scene before you perform a capture.

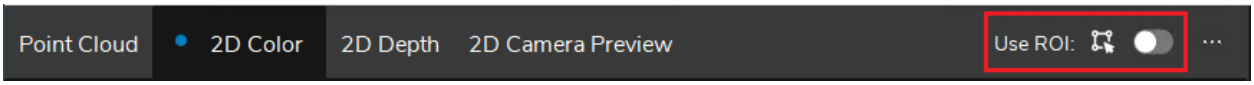
To perform a 2D Preview, click “Show 2D Preview” in the frame drop down menu corresponding to the one you want previewed. The preview will automatically appear in the “2D Camera Preview” visual tab. If the frame settings is collapsed, you can click the “eye” icon in order to preview that frame. In the “2D Camera Preview” tab, if the image has red spots, that means those pixels are overexposed under the current frame settings.



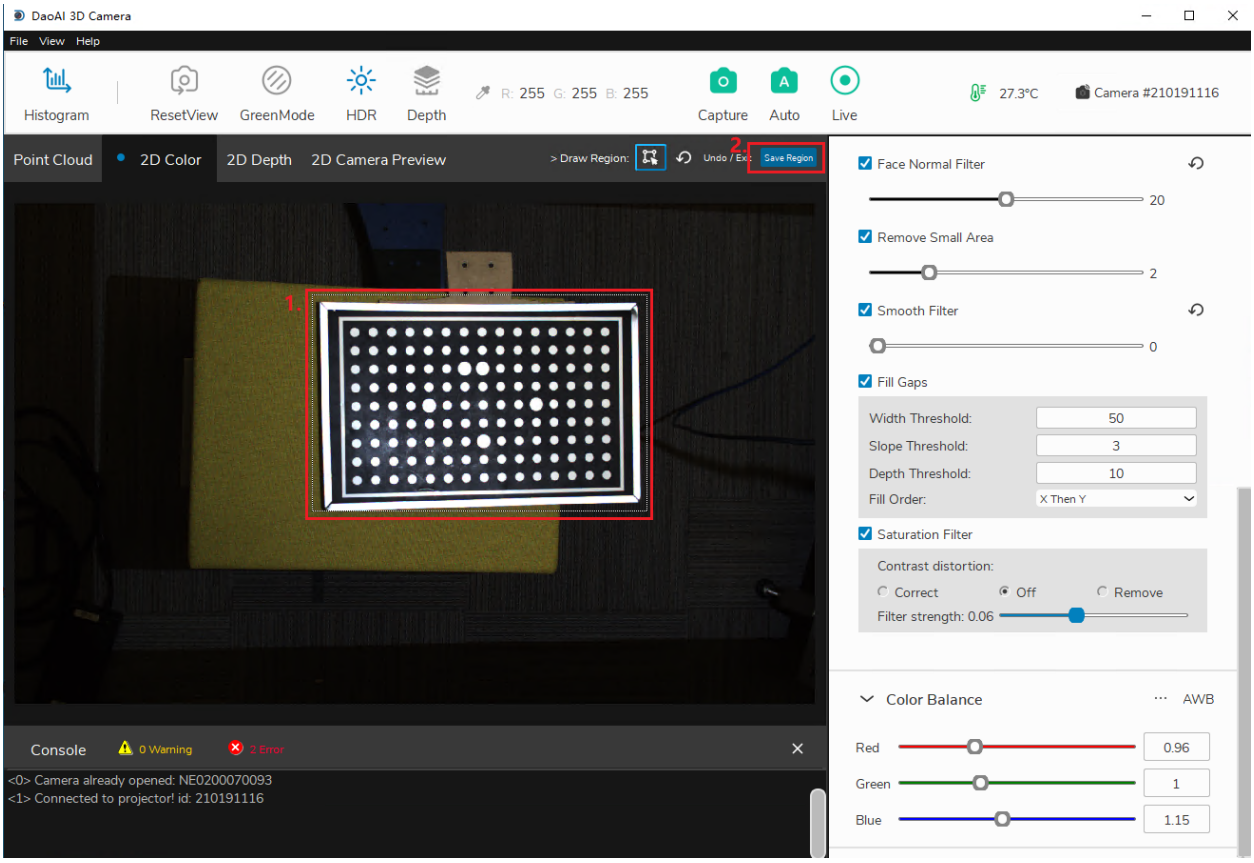
If there exists a 2D Camera Preview, you can click on the color picker icon in the top bar of the main window in order to find the RGB values of the pixel at your cursor's position in the image preview. The values will be updated on the top bar as well as displayed on the bottom left corner.



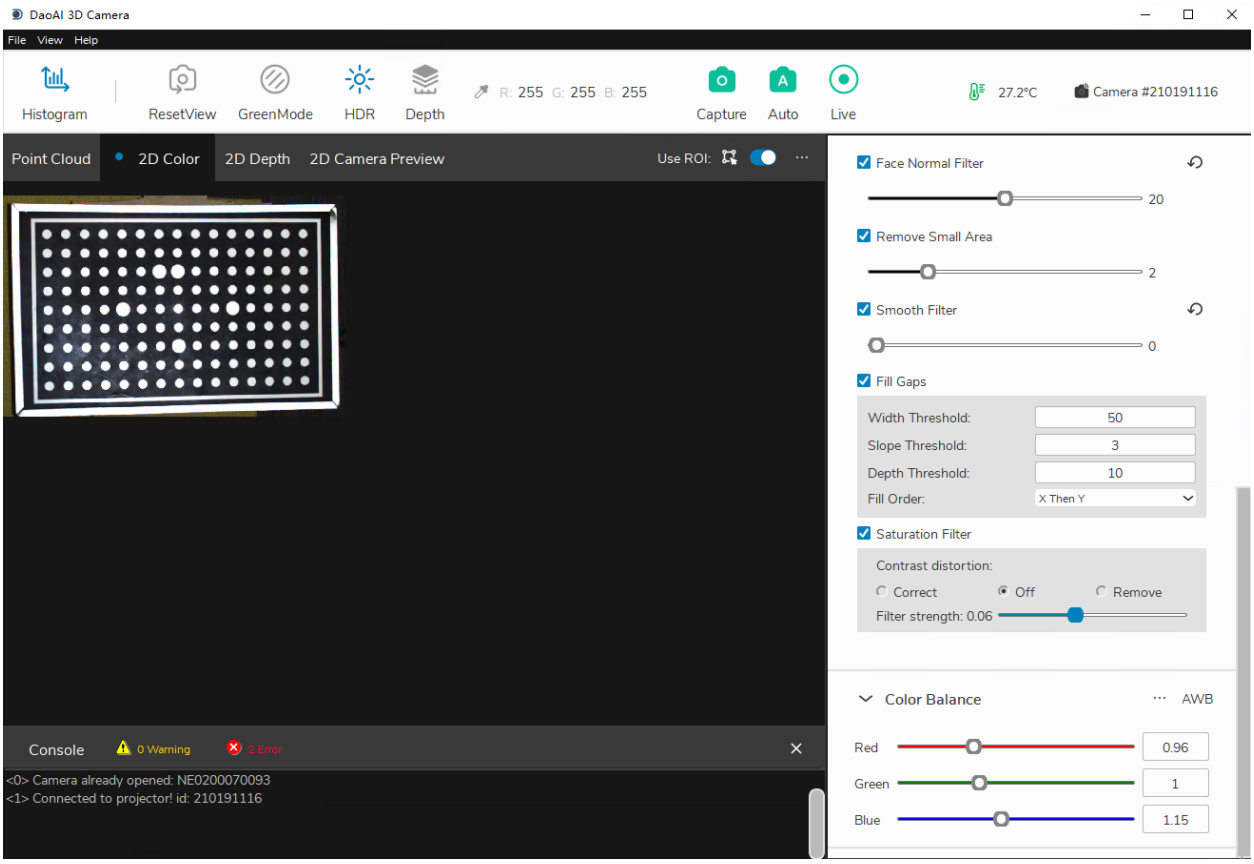
3.6.5 Regions of Interest



Regions of interest can be set within the 2D Color tab to select a cropped region of the image to be displayed. You can click the button on the top right of the tab and draw a region on the image itself.



Once you've highlighted a region, you must click "Save Region" to save it. Now whenever ROI is toggled on, all future captures will remember this saved region and will only display that region in the "Point Cloud" and "2D Color" tab.



When you decide you no longer need this ROI, you can clear the saved region by clicking the three dots beside the toggle switch and clicking “Remove ROI”.

3.7 Toolbar

Camera Studio allows you to save and load point cloud data as well as custom capture settings.

3.7.1 Save and Load

Load Point Cloud File

In the upper left corner click "File" → "Load 3D Data" to select the point cloud file (ending with .daf). Note that the path cannot contain Chinese characters. When a point cloud is displayed, the load is successful.

Save Point Cloud File

In the upper left corner, after capturing the image to generate the point cloud, click "File" → "Save 3D data" , enter the name of the new file, and click Save.

3.7.2 Save Settings (Export Settings)

Save settings file: click "File" → "Save Setting as" in the upper left corner, enter the name of the new settings file or select the overwritten settings file, and click Save.

3.7.3 Load Settings (Import Settings)

Make sure the 3D camera is connected, click "File" and "Load Setting" in the upper left corner, select the settings file (ending with .cfg). Note that the path cannot contain Chinese characters.

(The settings file saves the settings of the 3D camera when capturing images, including all the options in the main window and system settings)

3.7.4 Console

System information, results, detailed informations, warnings and errors are displayed in the Console.

Message Filtering

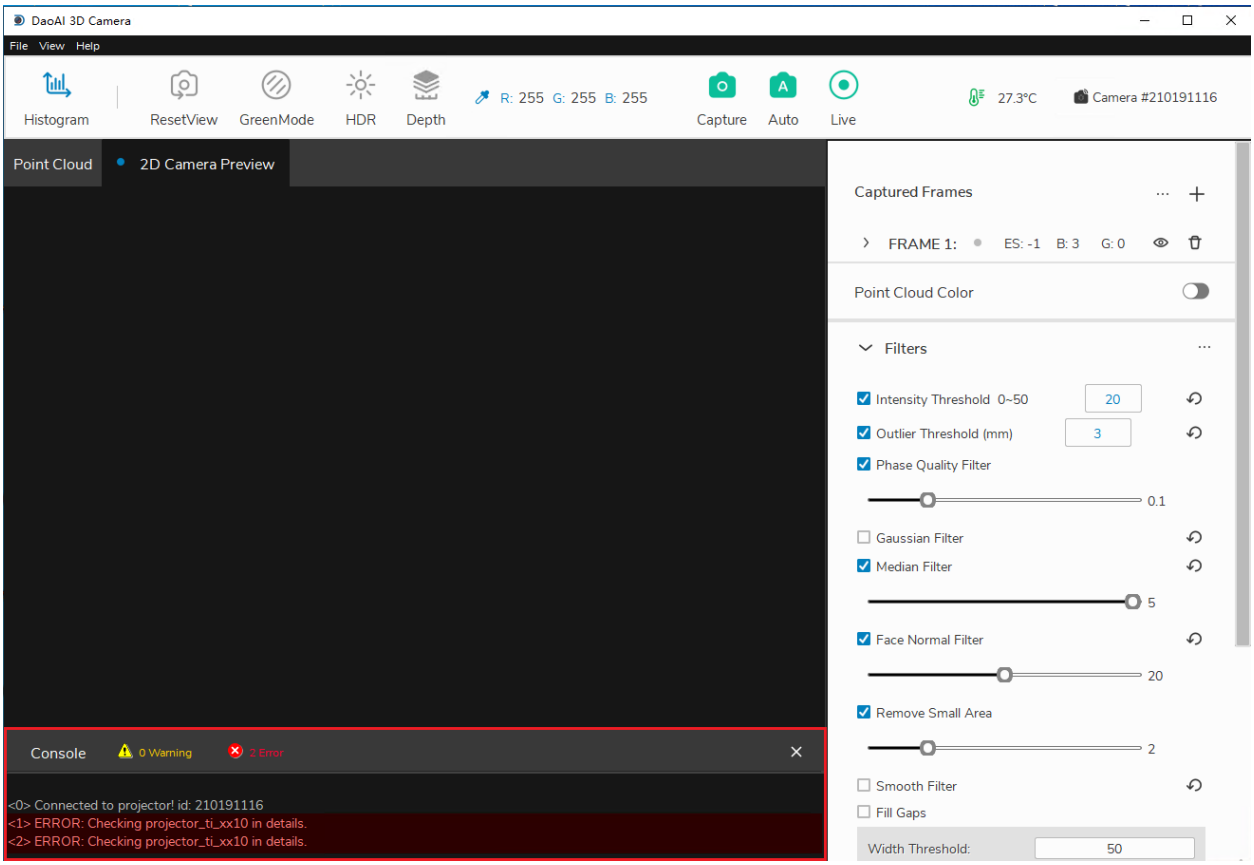
Click on the yellow warning icon or the red error icon can filter out message other than warning or error.

> **Warning:** Do not filter console when exporting log files. Log files contain messages from the console and will complete writing to disk upon program exits.

Hide and Show the Console

Click on the 'X' icon on the top-right corner of the console to hide it.

And from menu bar, click "View → Show Console" to bring it back.

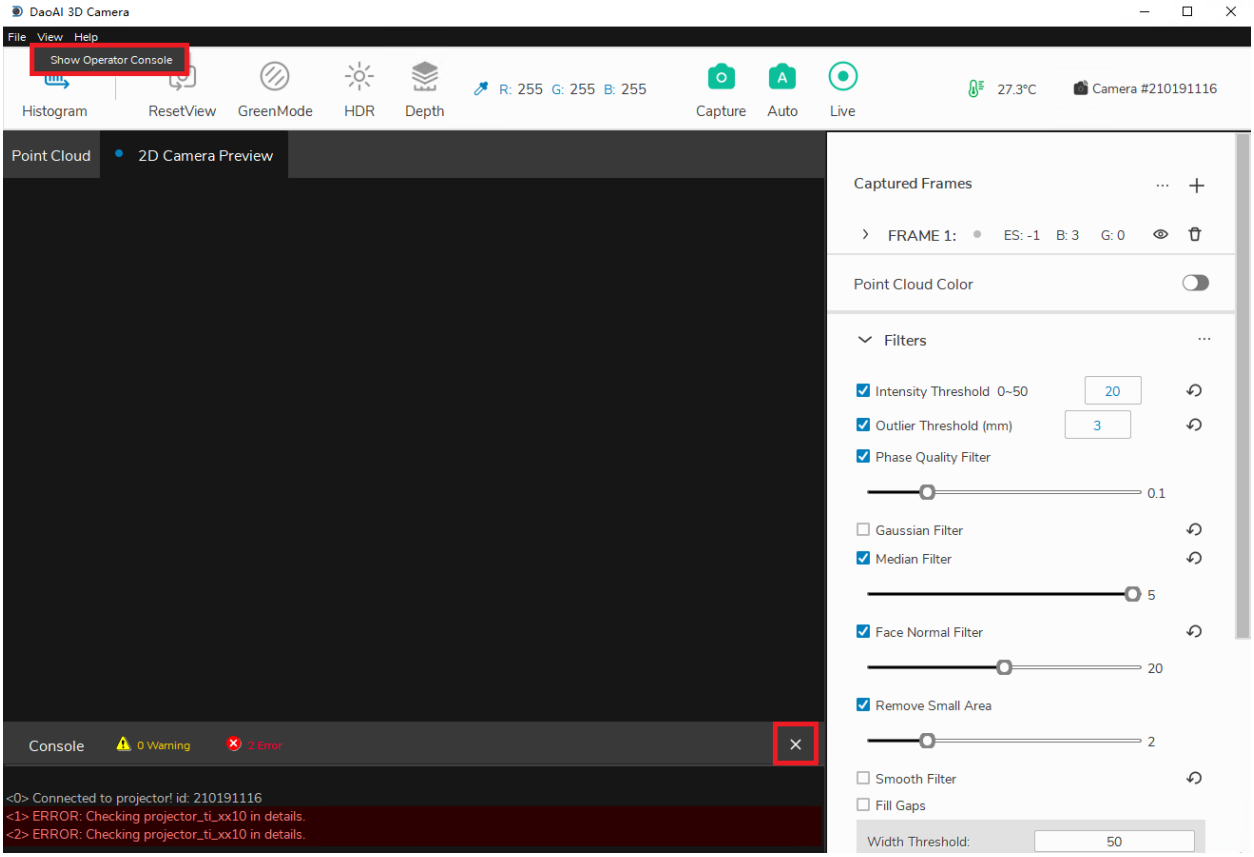


3.7.5 Dropdown Menus

3.8 Quick Reference Index

3.8.1 File

File	Function
Load Setting	Load camera settings from file.
Save Settings	Save the current camera settings.
Save Settings As	Export the current camera settings.
Load 3D Data	Load point cloud data from file.
Save 3D Data	Save point cloud data to file.
Temperature Control	Enable/Disable temperature control system.



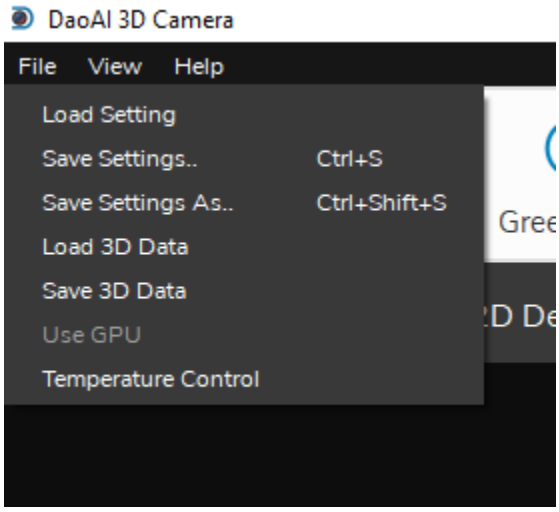


Fig. 41: File dropdown menu

3.8.2 View

View	Function
Show Operator Console	Toggle show or hide console.

3.8.3 Help

Help	Function
Search Online Help	Open the DaoAI Camera Studio User Manual in the browser.
About DaoAI Studio	Show help page, with software version and contact information.

3.8.4 Hotkeys

File	Shortcut	Function
Save Settings	Ctrl + S	Save the current camera settings.
Save Settings As	Crtl + Shift + S	Export the current camera settings.

3.8.5 Filters

Please refer to the *Filters* page for more details.

Filters	Function
Intensity Threshold	Filters out outliers caused by low quality pixels within dark areas in an image.
Outlier Threshold	Filters out points based on their distance from their nearest neighbouring point.
Phase Quality Filter	Filters out low contrast areas.
Gaussian Filter	Use a moving average window to apply a smoothing effect on the point cloud.
Median Filter	Use a median sliding window to apply a smoothing effect on the point cloud.
Face Normal Filter	Filters out points which surface normal vector is at an angle larger than face normal value.
Remove Small Area	Removes small chunk of isolated point cloud.
Smooth Filter	Rounds the depth value of an organized point cloud to the nearest mm.
Fill Gaps	Fill in missing point cloud via interpolation.
Saturation Filter	Removes overexposed areas.
Contrast Distortion Filter	Removes or corrects point cloud based on contrast.
Color Balance	Adjusts image's colour by making the image closer to one of R, G, B.

This section will provide a detailed walkthrough on using the DaoAI Camera Studio.

DaoAI Camera Studio is the graphical user interface for the DaoAI Software Development Kit. This allows the user to explore the functionality of the DaoAI Cameras, configure the camera settings in any environment conditions, and capture high defiinition 3D point clouds with high accuracy.

Note: You can download the DaoAI Camera Studio User Manual as a PDF.

CHAPTER

FOUR

DEVELOP - CREATE A NEW PROJECT

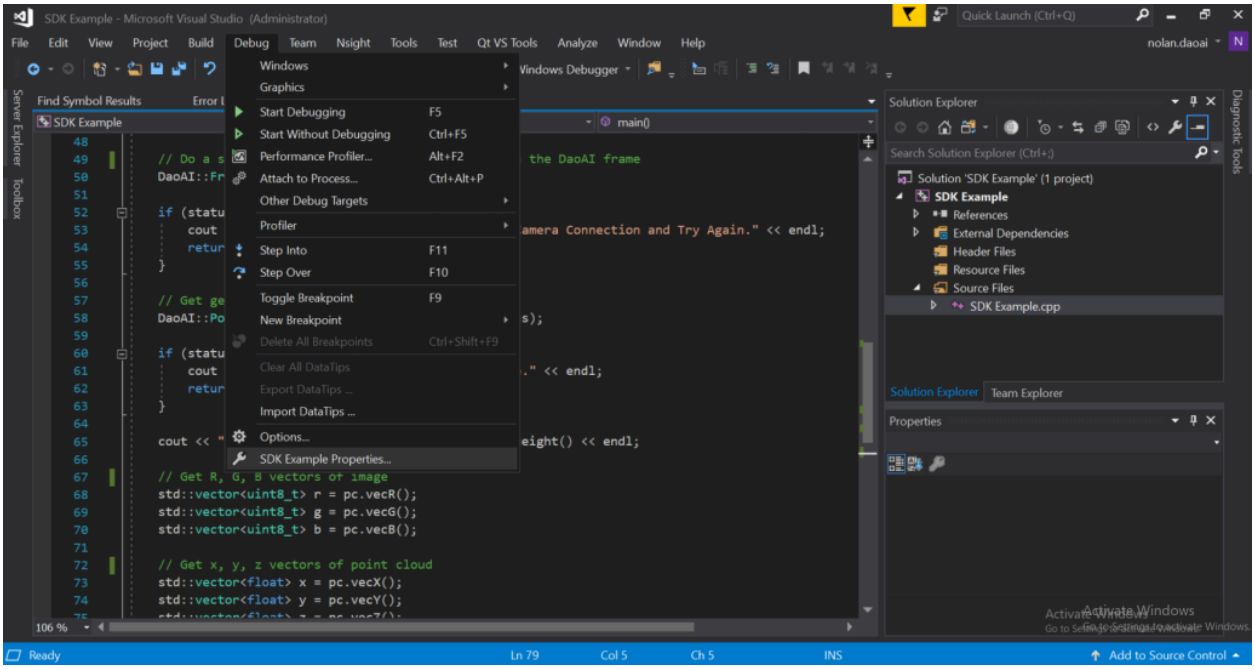
This page will walk you through how to set up a project using our SDK.

Step 1. Add the path to slc_dll.dll to the Path system variable

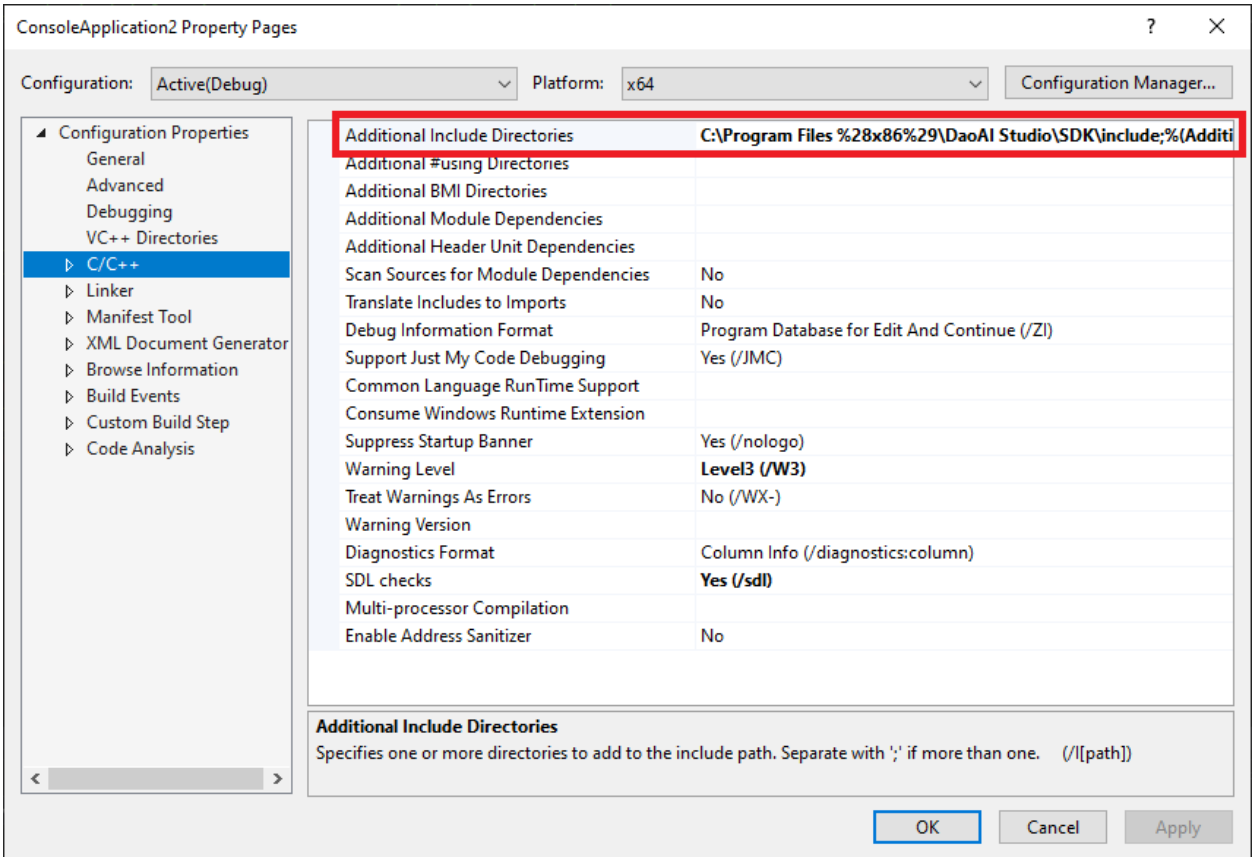
In Windows, navigate to Edit the system environment variables → Environment Variables. The new entry to the path system variable should be the <path to DaoAI Studio>\SDK\bin.

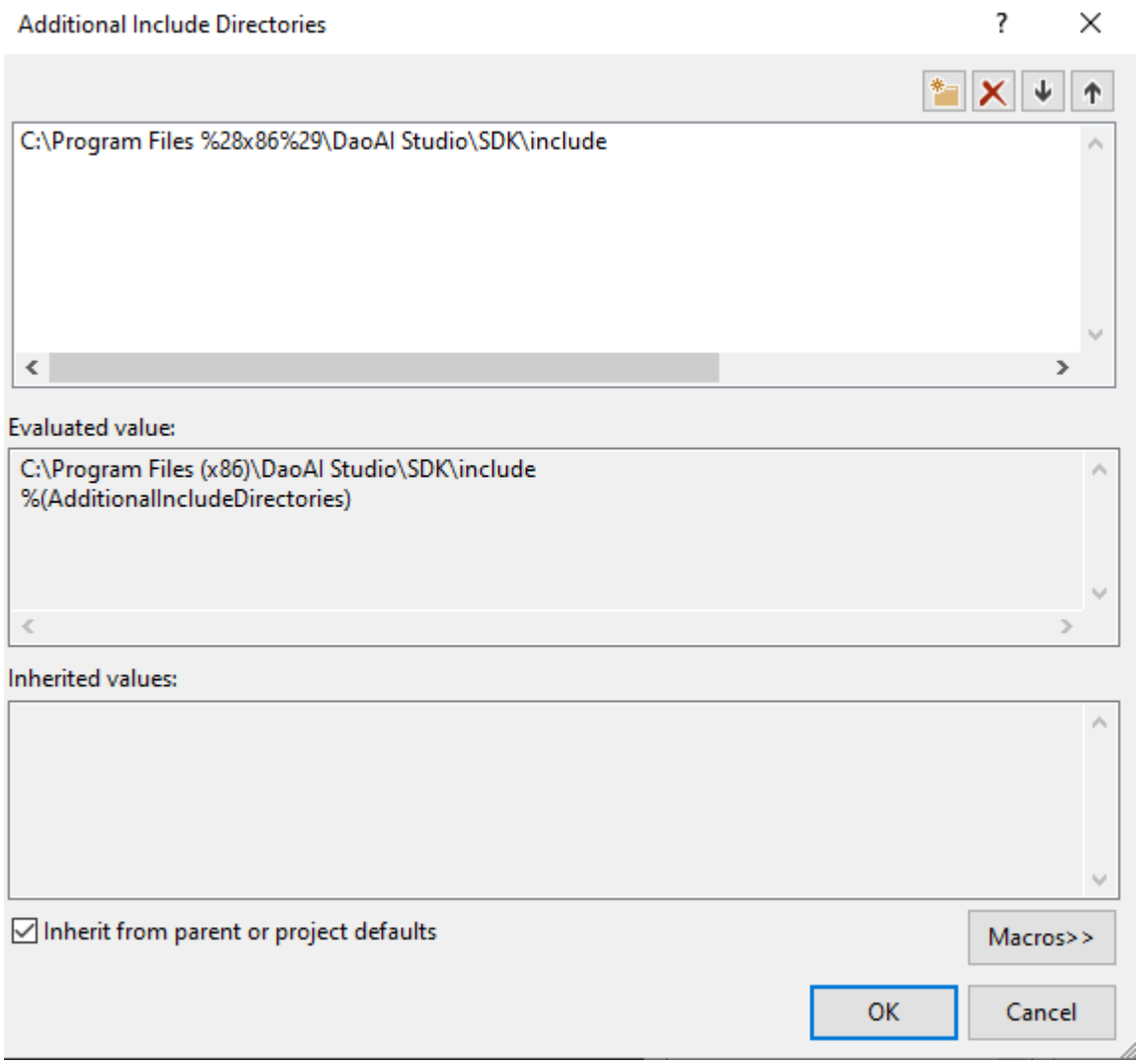
Step 2. Update include directories in Visual Studio

In Visual Studio, navigate to Debug → SDK Example Properties. You'll need to stay in the Properties menu for the remaining steps as well.



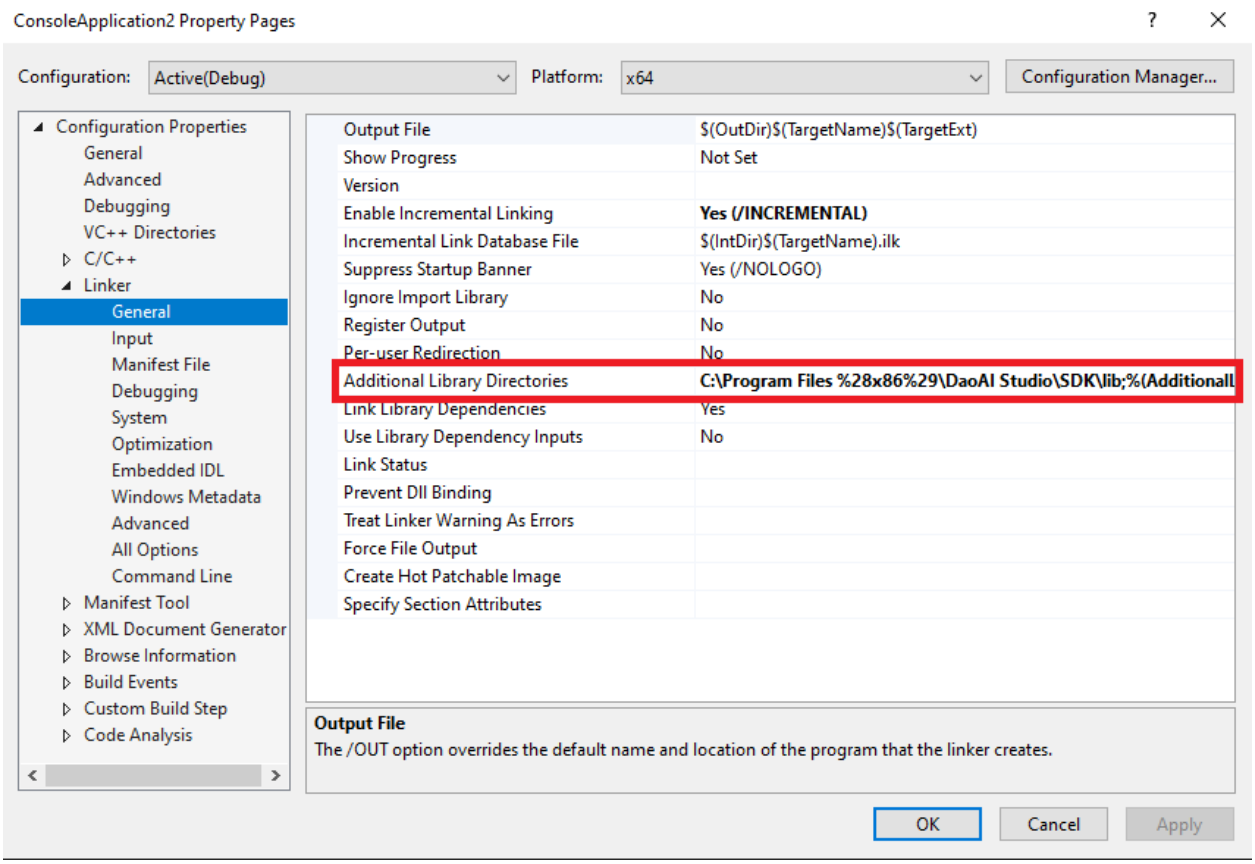
When making changes to the project properties, make sure the Configuration and Platform are set correctly (Release, x64). To update the include directories, first navigate to **Additional Include Directories** under C/C++ → General, click on the field's dropdown arrow and click Edit, and add the path to the SDK include directory, which contains the library header files. Usually is <path to SLC>/SDK/include.

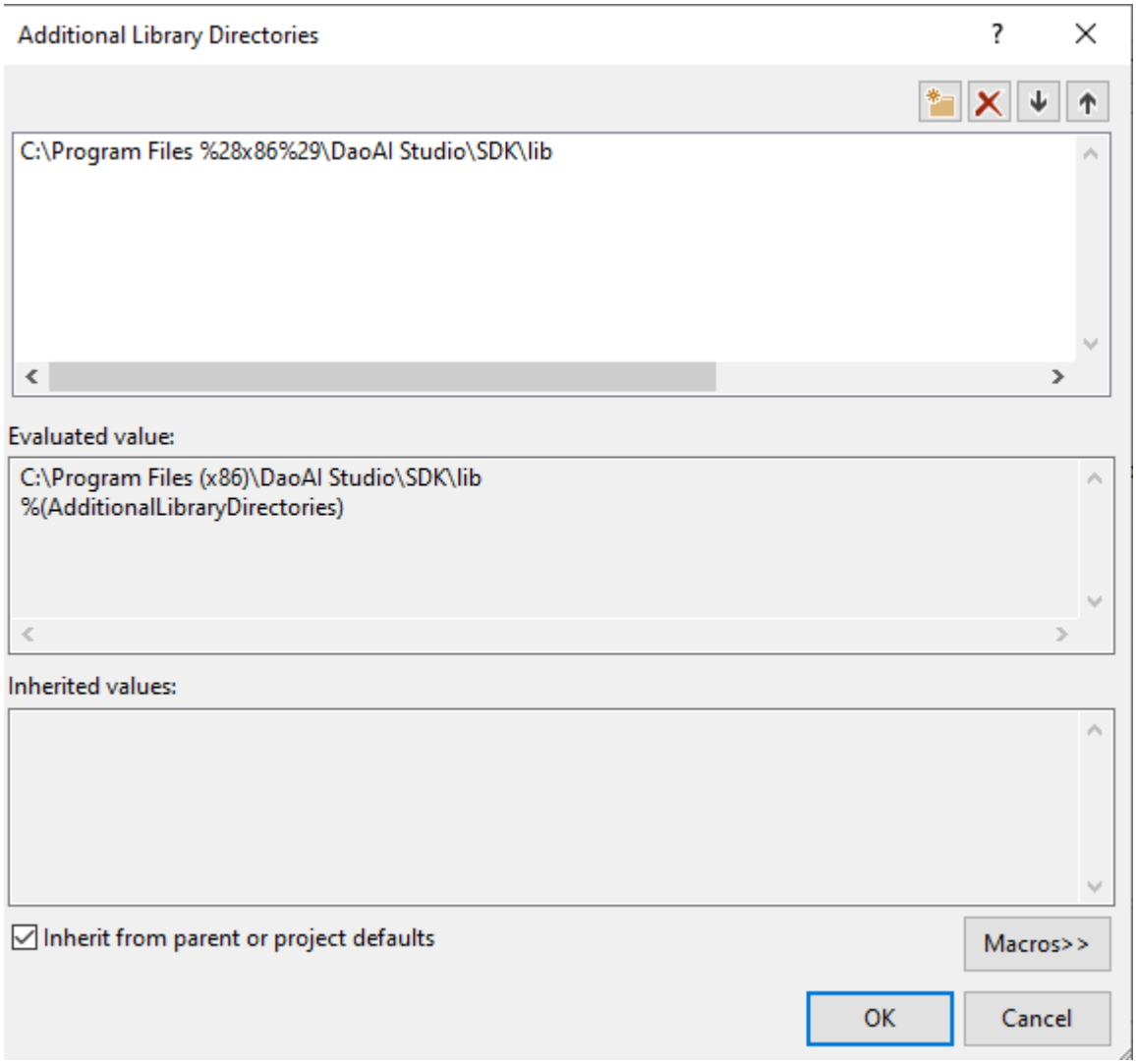

ConsoleApplication2 Property Pages
Configuration: Active(Debug)
Platform: x64
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
C/C++
Linker
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
Additional Include Directories C:\Program Files %28x86%29\DaoAI Studio\SDK\include;%(Additi
Additional #using Directories
Additional BMI Directories
Additional Module Dependencies
Additional Header Unit Dependencies
Scan Sources for Module Dependencies No
Translate Includes to Imports No
Debug Information Format Program Database for Edit And Continue (/Zi)
Support Just My Code Debugging Yes (/JMC)
Common Language RunTime Support
Consume Windows Runtime Extension
Suppress Startup Banner Yes (/nologo)
Warning Level Level3 (/W3)
Treat Warnings As Errors No (/WX-)
Warning Version
Diagnostics Format Column Info (/diagnostics:column)
SDL checks Yes (/sdl)
Multi-processor Compilation
Enable Address Sanitizer No
Additional Include Directories
Specifies one or more directories to add to the include path. Separate with ';' if more than one. (/I[path])
OK
Cancel
Apply



Step 3. Update linker settings in Visual Studio

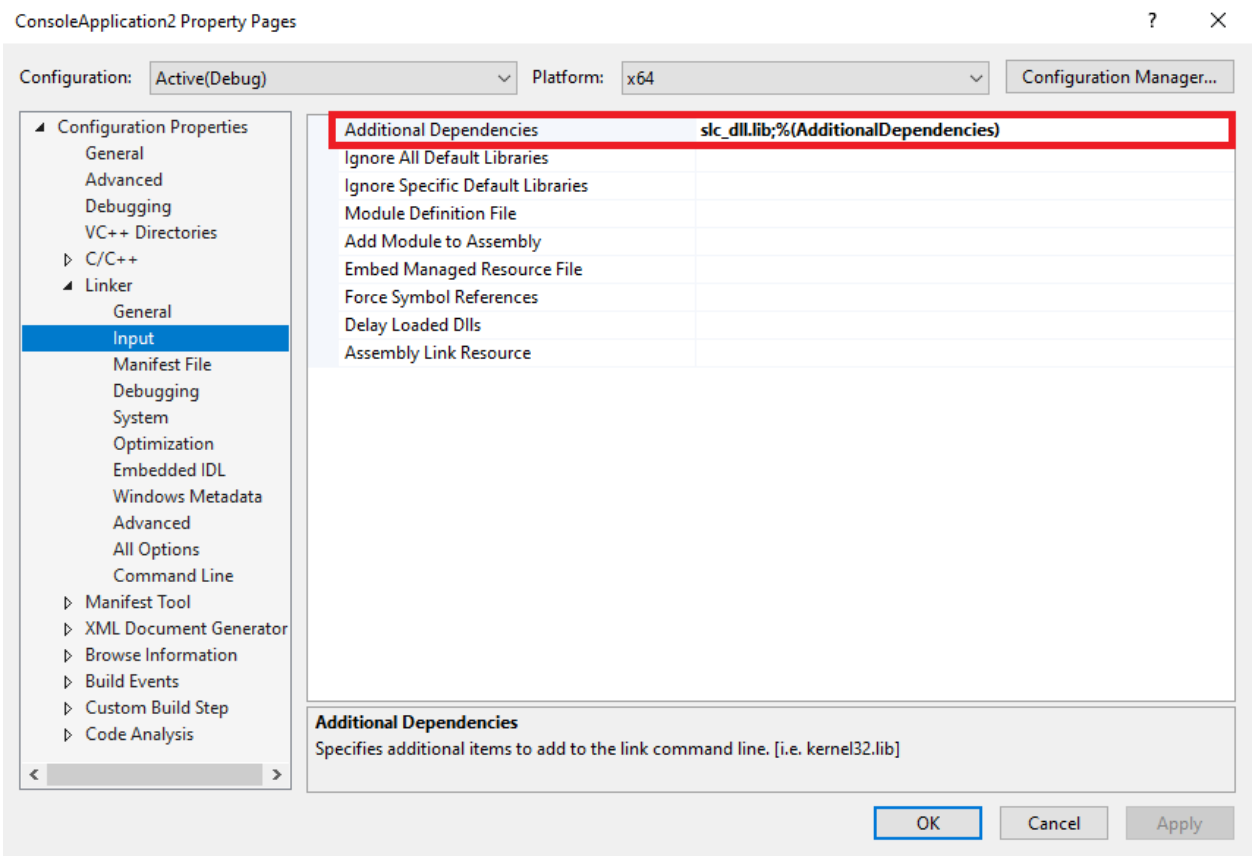
Navigate to Linker → General, click on **Additional Library Directories** dropdown arrow and click Edit, and add the path to the SDK lib directory, which contains the .lib library object files that need to be linked. Usually is <path to SLC>/SDK/lib.

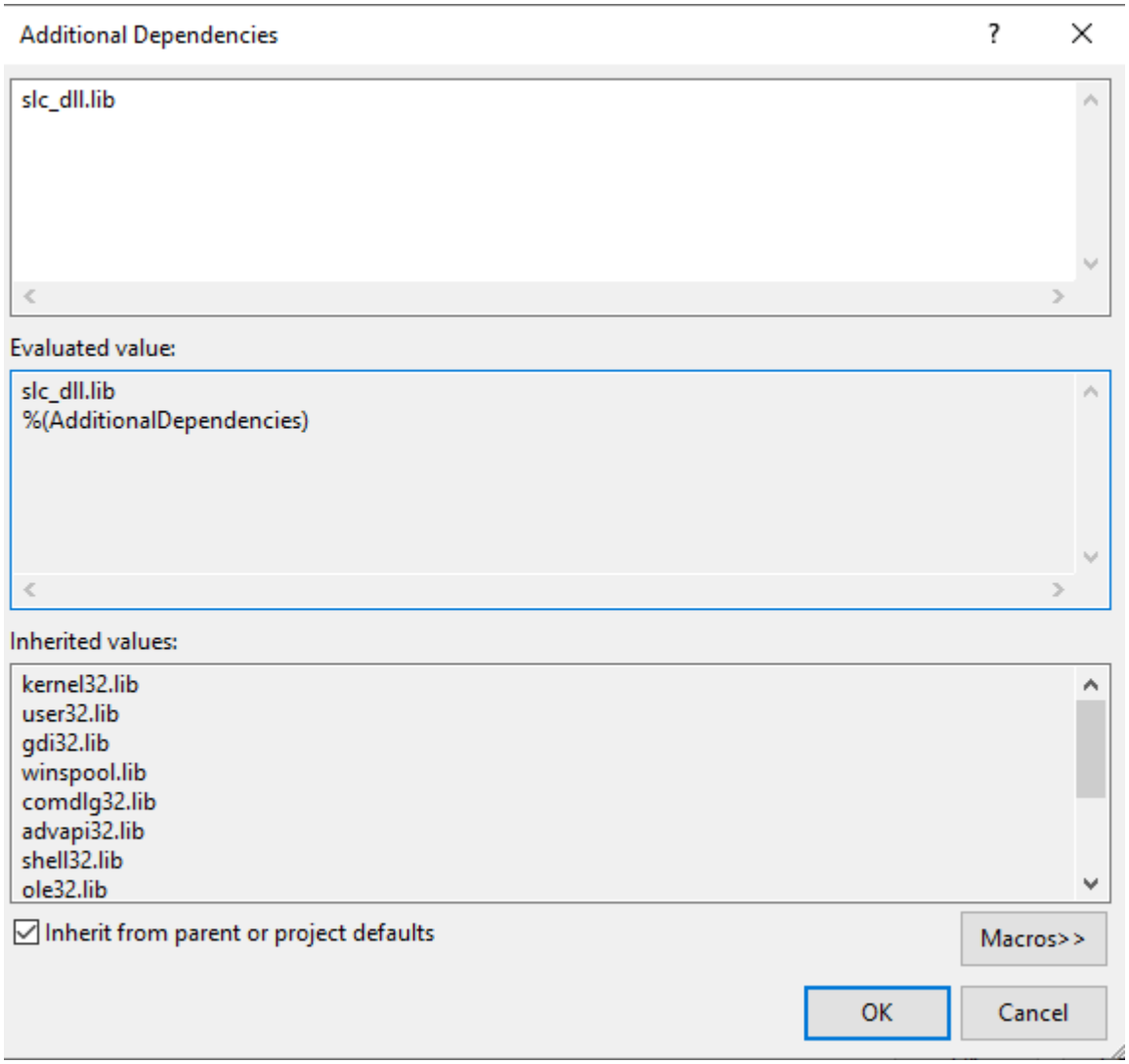

ConsoleApplication2 Property Pages
Configuration: Active(Debug)
Platform: x64
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
Output File $(OutDir)$(TargetName)$(TargetExt)
Show Progress Not Set
Version
Enable Incremental Linking Yes (/INCREMENTAL)
Incremental Link Database File $(IntDir)$(TargetName).ilk
Suppress Startup Banner Yes (/NOLOGO)
Ignore Import Library No
Register Output No
Per-user Redirection No
Additional Library Directories C:\Program Files %28x86%29\DaoAI Studio\SDK\lib;%(AdditionalL
Link Library Dependencies Yes
Use Library Dependency Inputs No
Link Status
Prevent Dll Binding
Treat Linker Warning As Errors
Force File Output
Create Hot Patchable Image
Specify Section Attributes
Output File
The /OUT option overrides the default name and location of the program that the linker creates.
OK
Cancel
Apply



Step 4. Add Dependency

Navigate to Linker → Input, click on **Additional Dependencies** dropdown arrow and click Edit, and add slc_dll.lib as an entry.


ConsoleApplication2 Property Pages
Configuration: Active(Debug)
Platform: x64
Configuration Manager...
Configuration Properties
General
Advanced
Debugging
VC++ Directories
C/C++
Linker
General
Input
Manifest File
Debugging
System
Optimization
Embedded IDL
Windows Metadata
Advanced
All Options
Command Line
Manifest Tool
XML Document Generator
Browse Information
Build Events
Custom Build Step
Code Analysis
Additional Dependencies
slc_dll.lib;%(AdditionalDependencies)
Ignore All Default Libraries
Ignore Specific Default Libraries
Module Definition File
Add Module to Assembly
Embed Managed Resource File
Force Symbol References
Delay Loaded Dlls
Assembly Link Resource
Additional Dependencies
Specifies additional items to add to the link command line. [i.e. kernel32.lib]
OK
Cancel
Apply



When you are finished these steps, apply the changes and click OK.

CHAPTER

FIVE

DAOAI SLC CAMERA WITH MATROX DESIGN ASSISTANT

This document will guide you through setting up to capture real image, depth map images and point cloud from your DaoAI SLC Camera into Matrox Design Assistant.

5.1 Install the DaoAI SLC Camera Software

Install the latest DaoAI SLC Camera software.

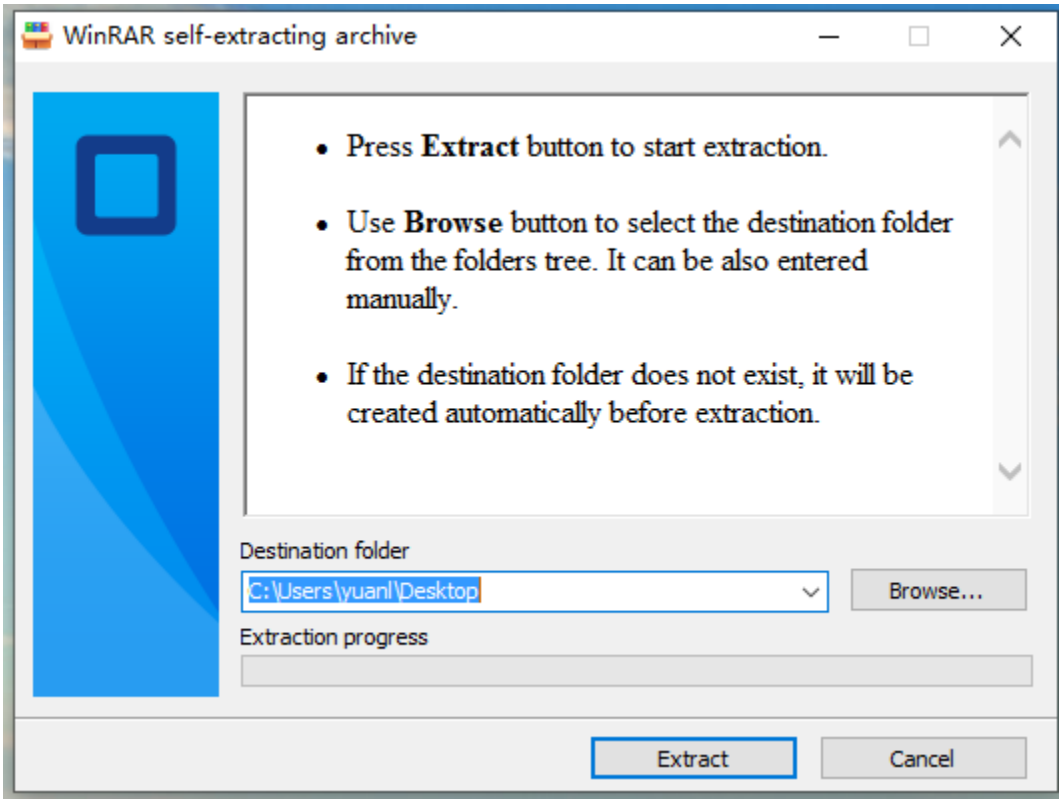
After installation, go to the Application folder, and open DA3rdPartyCamera folder.

5.2 Install the Matrox Design Assistant 2109(8.0)

Download the installer of Matrox Design Assistant . You will need a key for it.

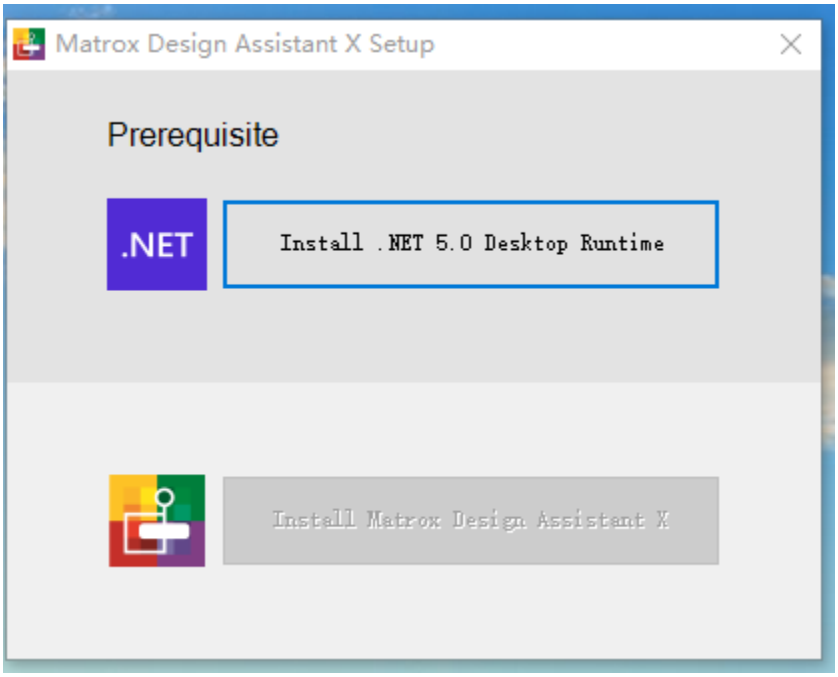
Note: Before you start, go to Control Panel and uninstall "Matrox Imaging" and DaoAI's Vision software first if you already have it installed.

1. When you run the exe above, go with default settings. It will unzip a fodler called "DAXV2109_8_0_291".



2. Open this folder and run “DesignAssistantSetup.exe“

3. Click the first button “Install .NET5.0 Desktop Runtime”



Warning: If there is no "Install .NET5.0 Desktop Runtime" avaliable, you need to first install Windows Desktop .NET 5.0 installation

4. After installing Runtime, Install Matrox Design Assistant

5. Next you will see the component selection, check IDE and RTE

6. At the end of installation, disable "Confiture automatic updates"

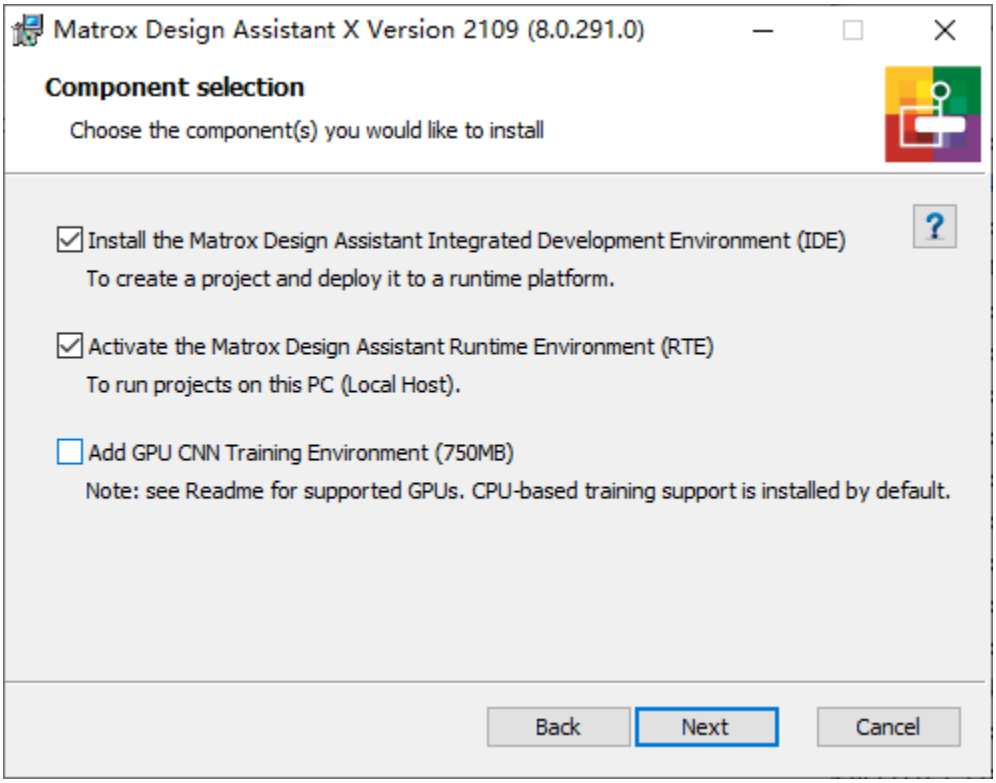
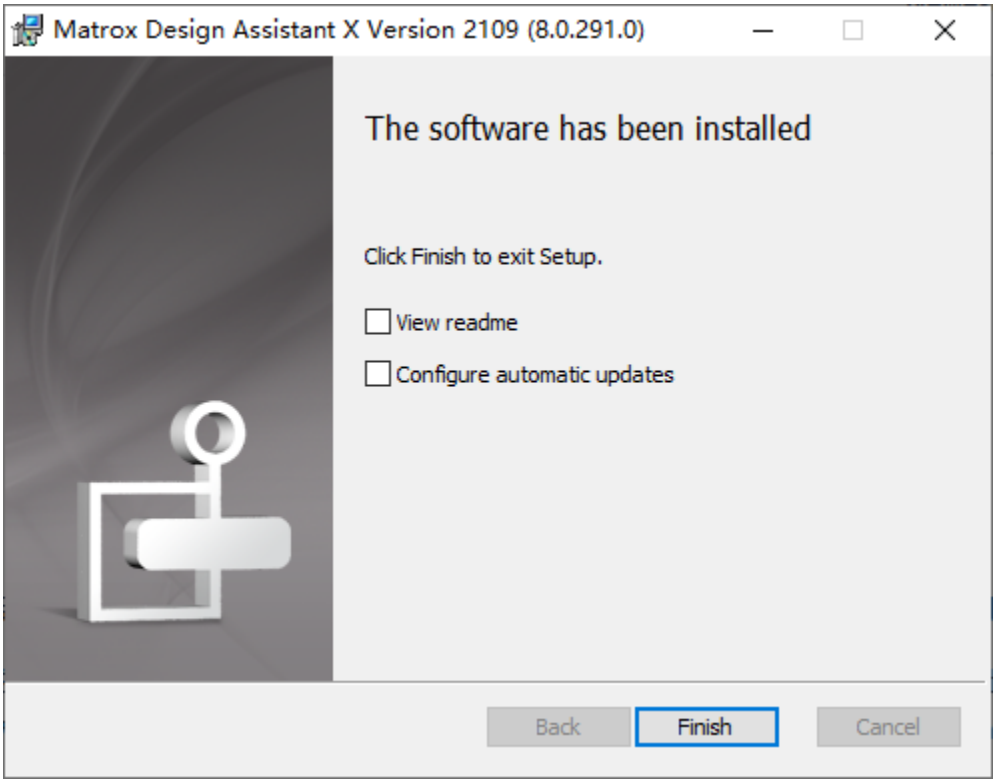
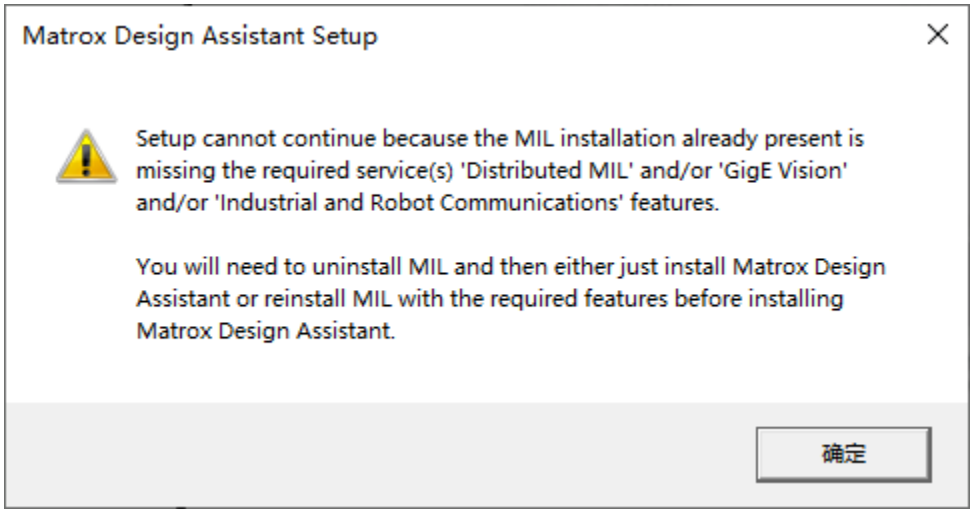


Fig. 1: Check first two options

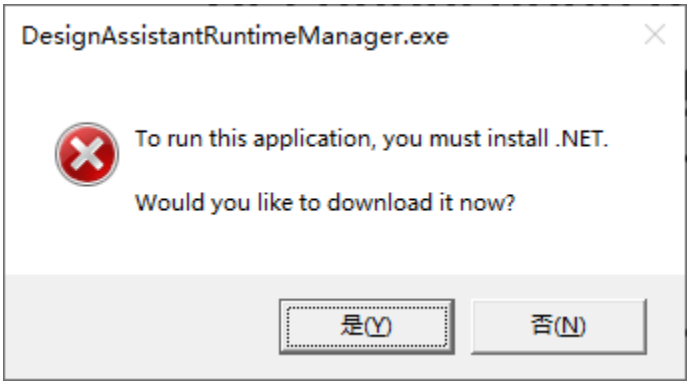


7. Restart PC.

Note: If you see this error, you need to go to Control Panel and make sure you uninstalled Matrox Imaging before the installer started.



If you see this error box, it means you don't have .NET runtime 5.0 installed correctly on your PC.

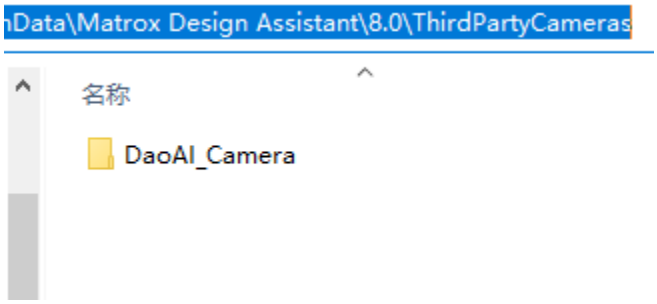


5.3 Configure the Matrox Design Assistant 2109(8.0)

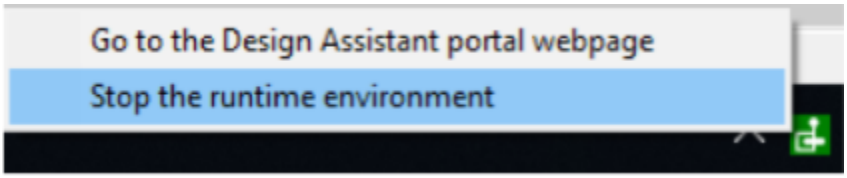
After installing DA from installer above, do the following to load DaoAI SLC Camera into DA system.

1. Open folder C:ProgramDataMatrox Design Assistant8.0ThirdPartyCameras

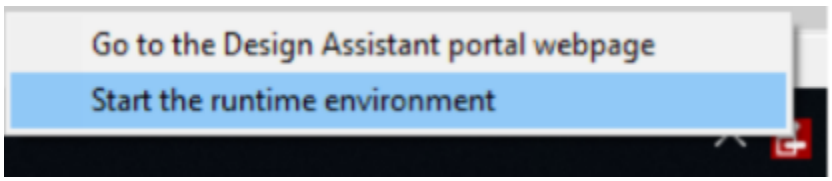
2. Copy the (SLC Installation folder)/DA3rdPartyCamera/DaoAI_Camera folder into ThirdPartyCameras folder opened in step 2.



3. Stop DA runtime environment by right clicking the DA icon.



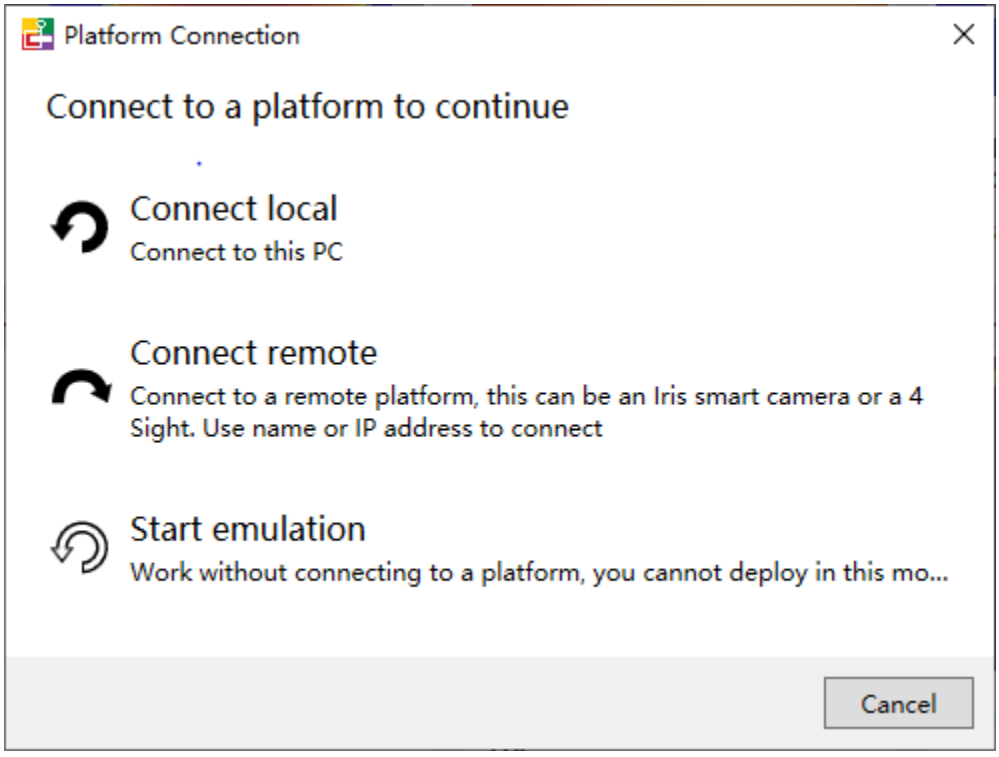
4. Start the DA runtime environment to load new thrid party camera.



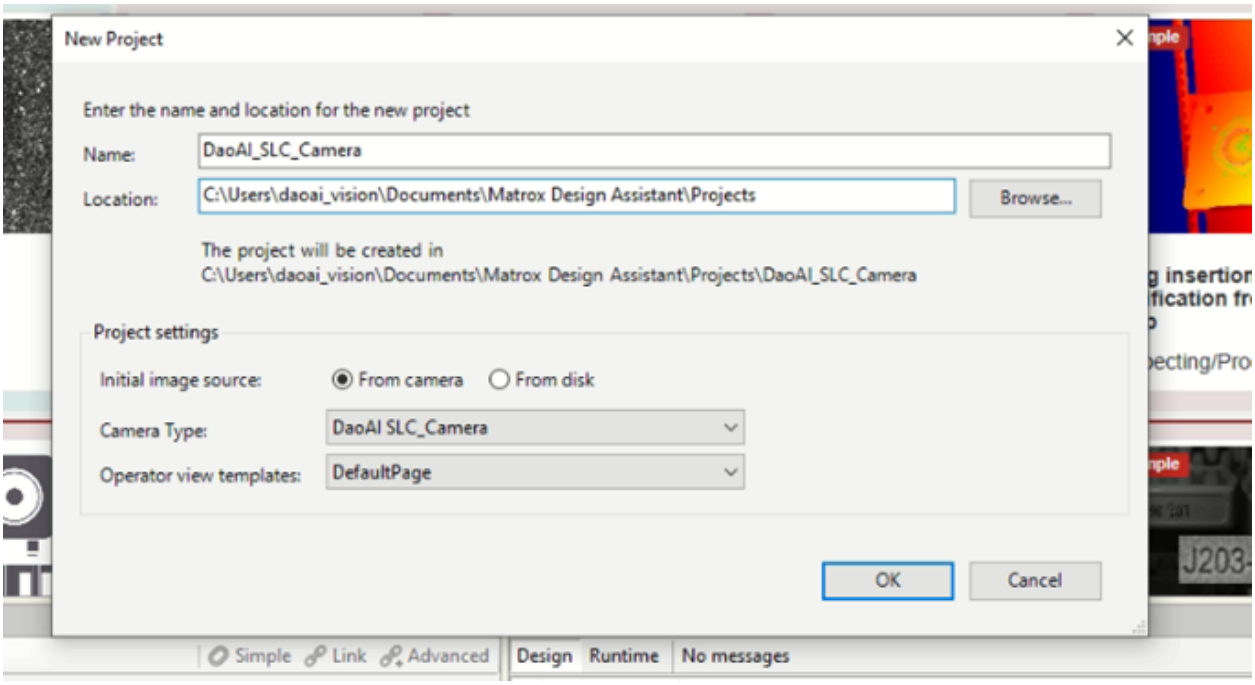
5.4 Creating the Design Assistant Project

1. Open Matrox DA X Version 2109 from start menu

2. Create a new project, choose “Connect Local”



3. Specify the DaoAI SLC_Camera type from 3D camera list as the example image.



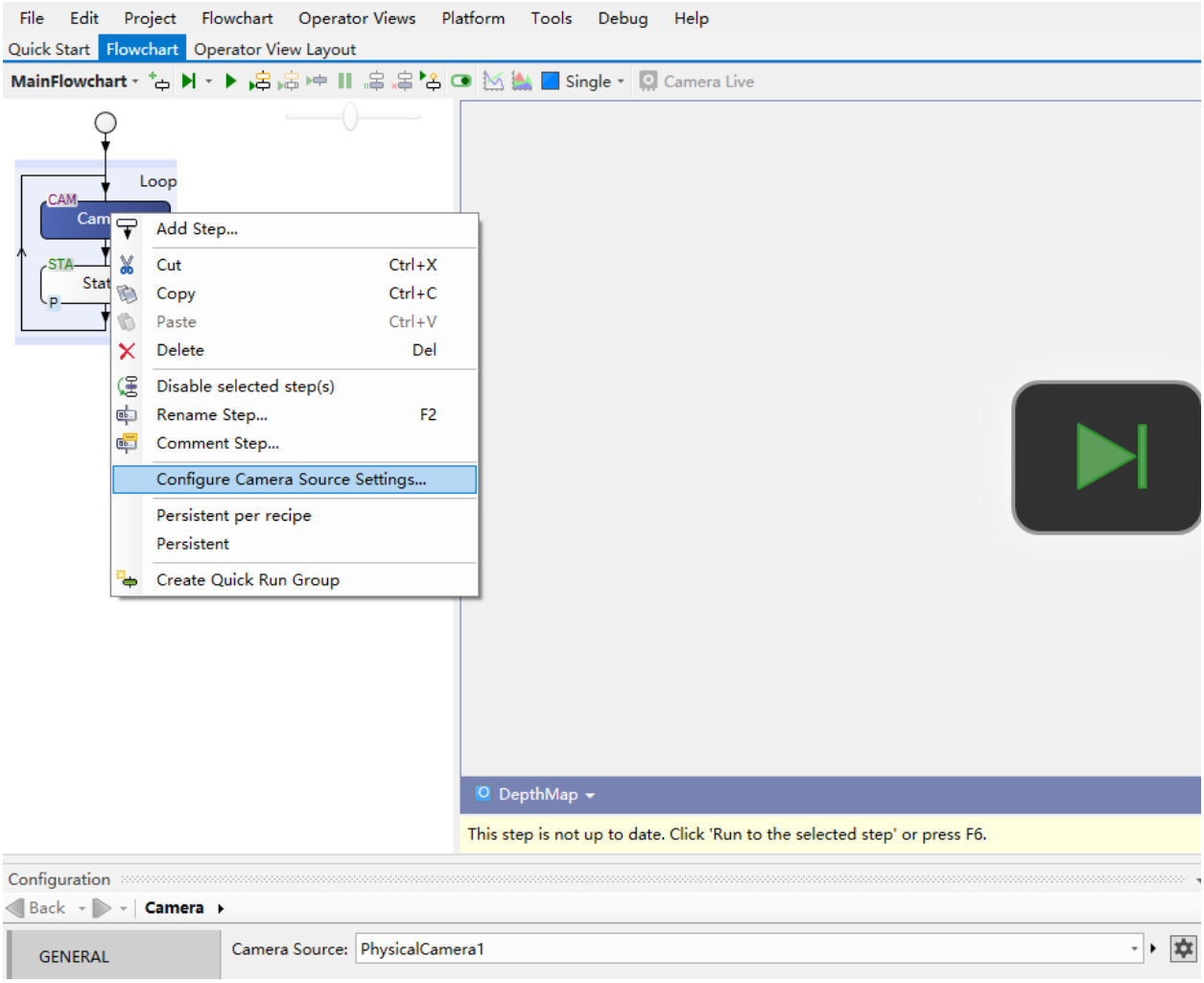
4. Click connect to local in the new pop up.

5.5 Connect to the DaoAI SLC Camera

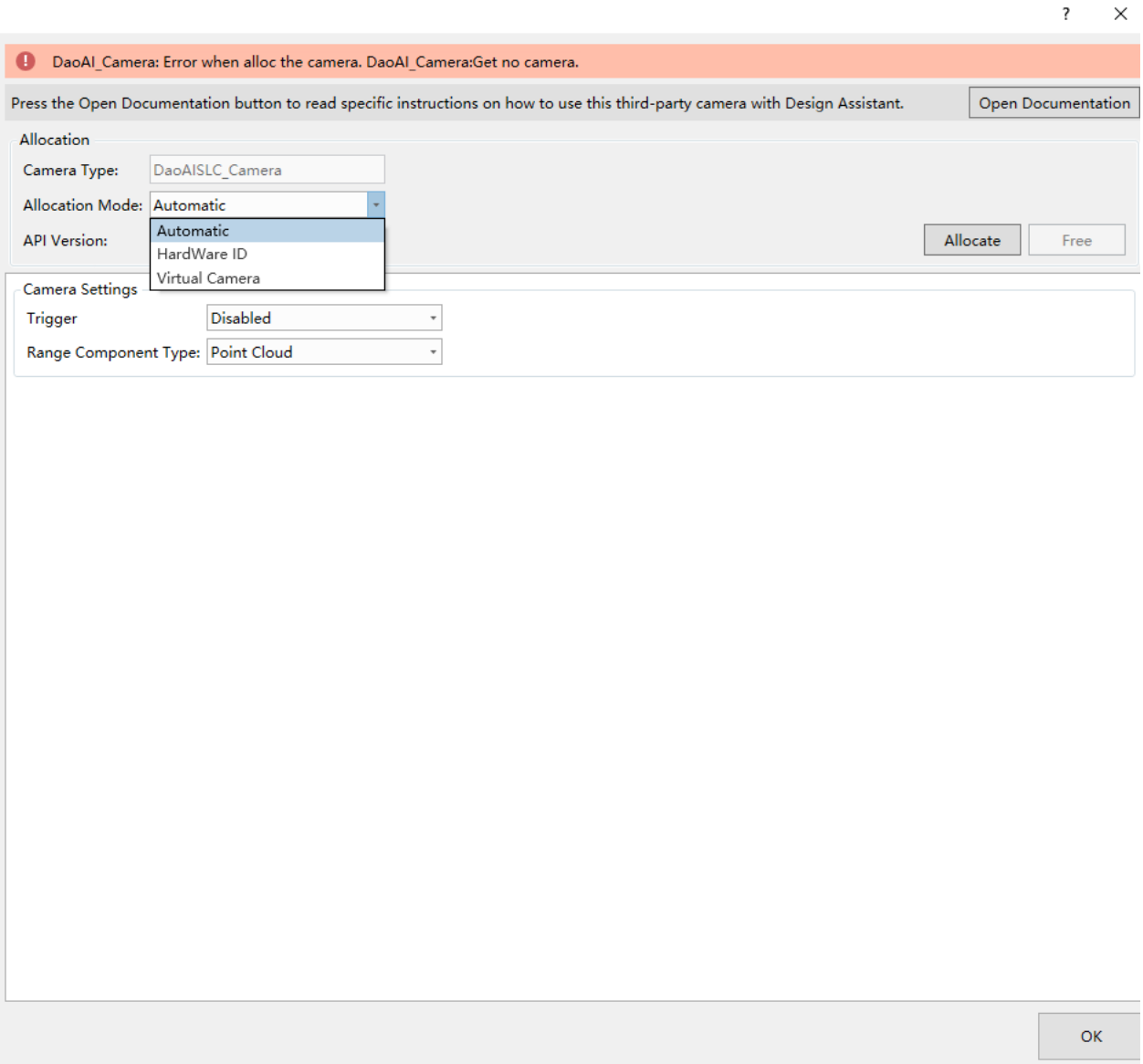
> **Warning:** DaoAI Studio and MIL Design Assistant cannot be connected to the same camera at the same time.

In Matrox Design Assistant, you can choose, connect and disconnect cameras on Platform Configuration dialog.

1. To open Platform Configuration, right click the Camera node, and click "Configure Camera Source Settings…"



2. There are Allocate and Free buttons, which correspond to connection and disconnection. There are three kind of features that you can implement:



a. Automatic : MIL Design Assistant allocates the first DaoAI SLC Camera detected by the API.

b. Hardware ID: The value of the Hardware is the serial number of the camera that you want to connect. The serial number can be found in DaoAI Studio.

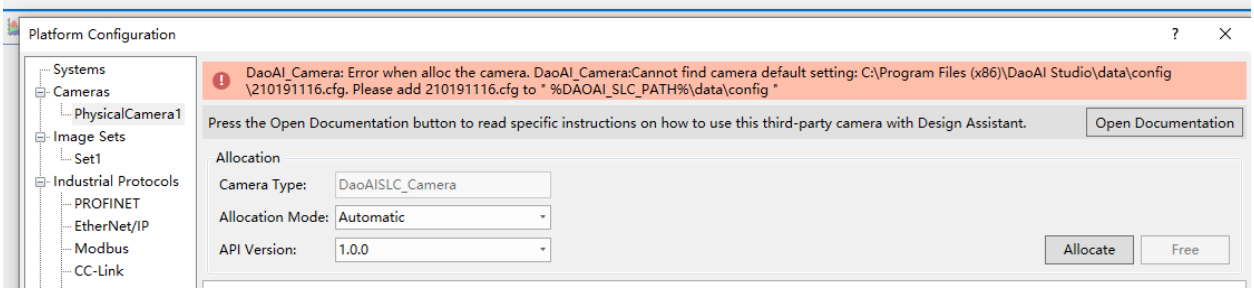
c. Virtual Camera: It will allocate a virtual camera instead of a physical camera. The value you input can be :

i. Empty, which means the virtual camera will load the data in default folder (DaoAI 3D Camera Installation folder)datad3Data.

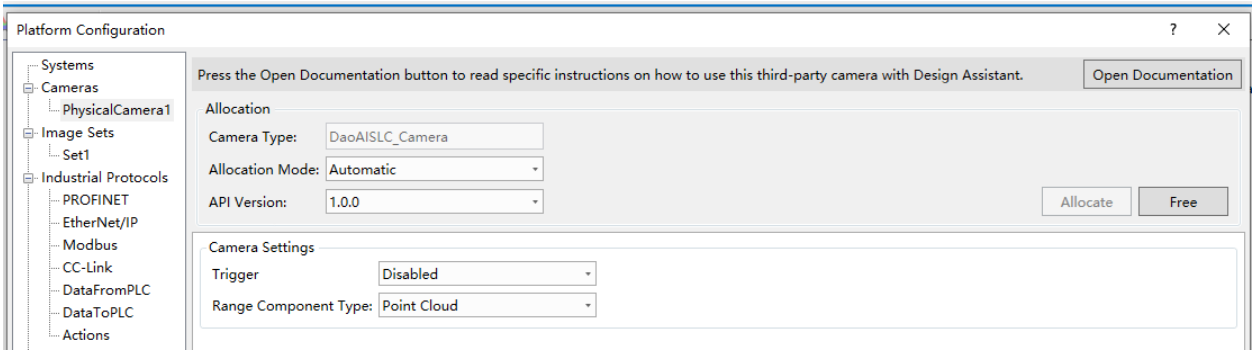
ii. A folder path, which should be a folder contains the .daf and/or .dcf data files that you want the virtual camera to load and display on MIL Design Assistant.

3. Click “Allocate” to connect your camera.

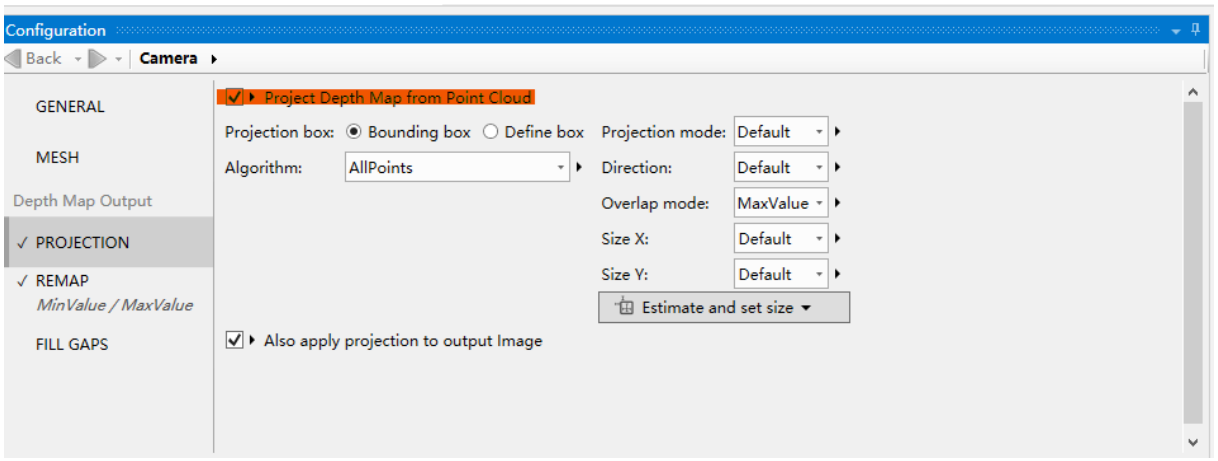
4. For Hardware cameras, you need to prepare a configuration file of the camera. Use “Save Camera Settings” function in DaoAI 3D Camera to save a “.cfg” file. You need to rename the file to the serial number of your camera(The error will prompt you the serial number) and place the cfg file under (DaoAI 3D Camera Installation folder)dataconfig.



5. For a successful connection, you should see the Free button enabled, and no error prompts.



Danger: You must uncheck this “Project Depth Map from Point Cloud” to get a correct 2D Image.

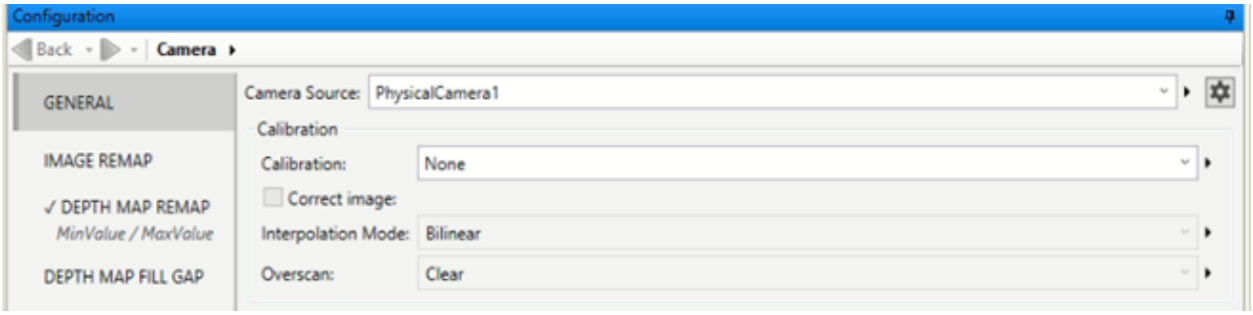


5.6 Further Information

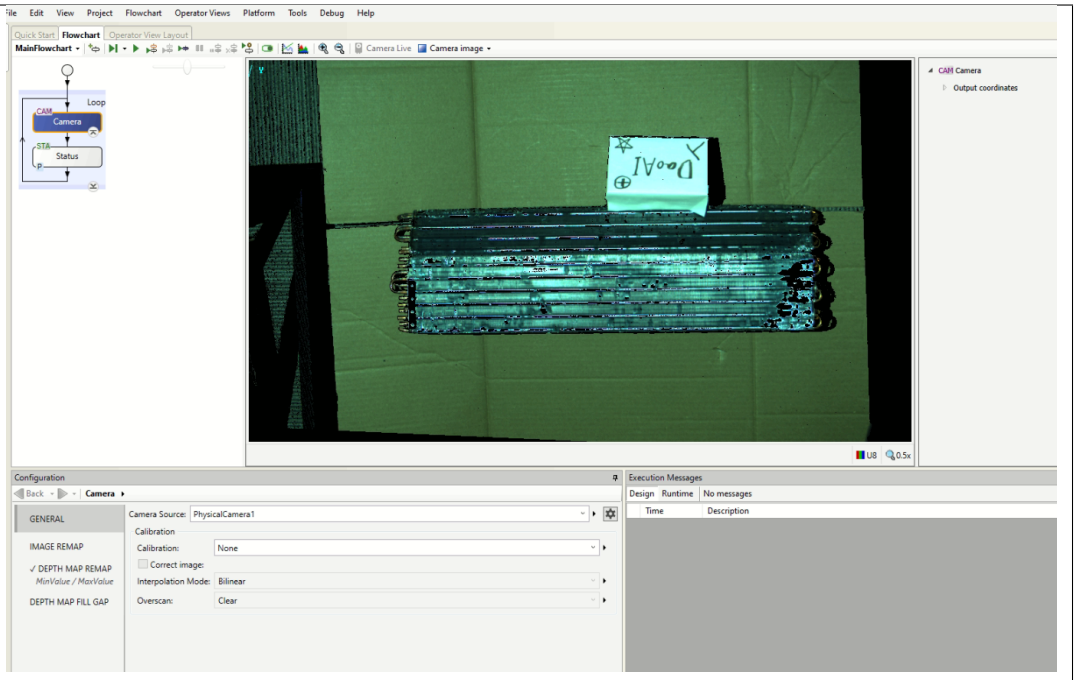
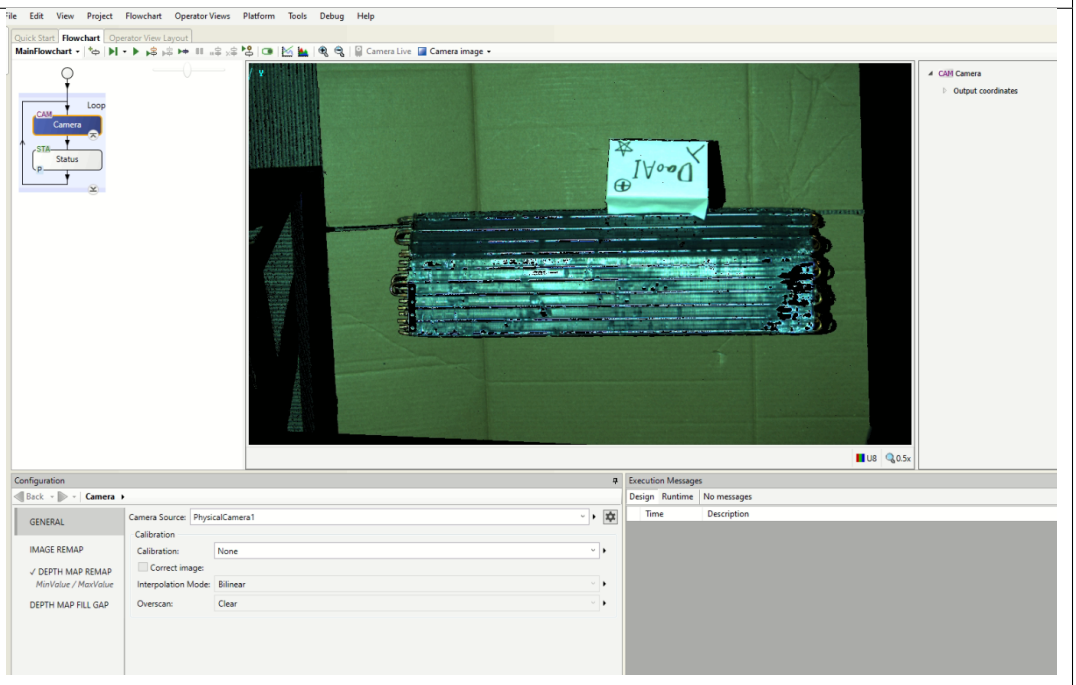
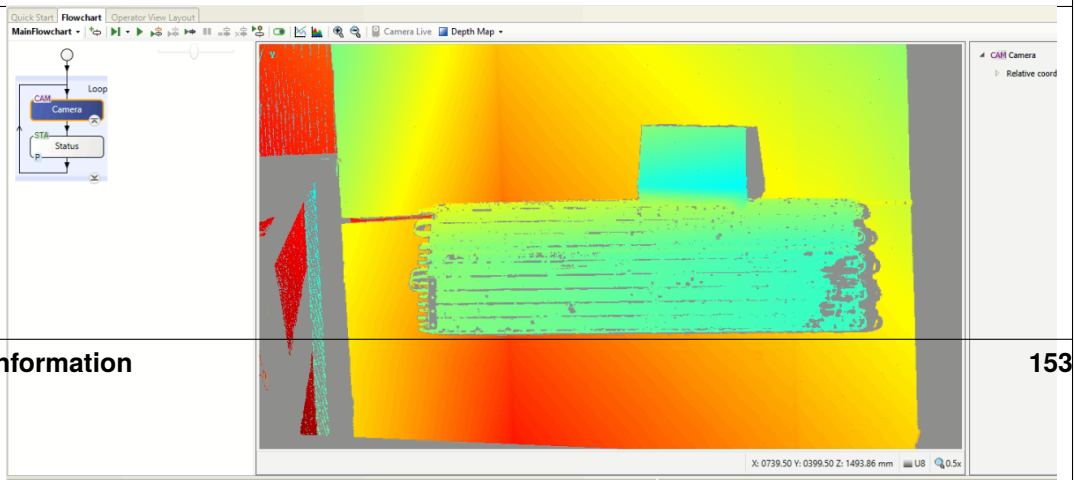
5.6.1 Configuring the Camera step

In the Camera step, verify that the Camera Source input is PhysicalCamera1.

Select camera node and click “Run to Select Step”, then the camera will do the capture.

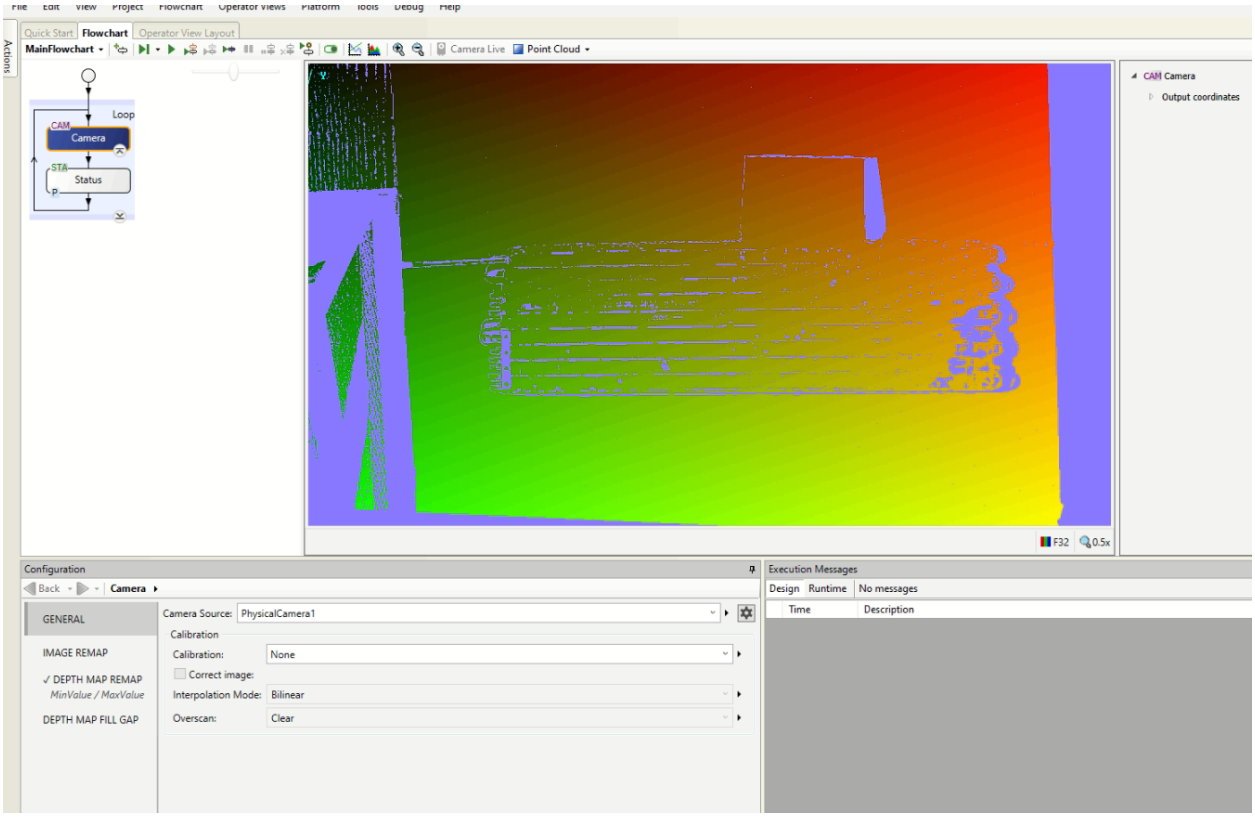


This is the sample result from the capture:

Camera Image	
Depth Image	
	

5.6.2 Change Camera Settings

While using physical camera, there are different camera settings like below:



In DaoAI Studio, you can just change these settings directly, but in MIL Design Assistant, it will load the default settings for the camera you are connecting.

The settings file is named "serial_number.ws", the "serial_number" should be the exact serial number of the camera you want to connect. For example, camera with serial number "202001002" will load settings "202001002.ws"

The setting file "serial_number.ws" shoule be placed in path: %DAOAI_SLC_PATH%dataworkspace

If you want to change the settings file, you need to set the value you want in DaoAI Studio and use "Save As ..." to save the settings with corresponding name and path. Then when you click "Allocate" on Platform Configuration dialog in MIL Design Assistant, it will load the settings you want for the camera.

CHAPTER

SIX

SDK SAMPLE

6.1 Capture tutorial

This tutorial describes how to use the DaoAI SDK to capture point clouds and 2D images.

6.1.1 Prerequisites

C++

- install DaoAI Camera Studio

C#

- install DaoAI Camera Studio

Python

- install Python
- Check your python version by entering the follow command into your command line then choose a SDK package:

```
python --version
```

  - Python version 3.8: DaoAI_SDK-xx-cp38-xx-xx.whl
  - Python version 3.9: DaoAI_SDK-xx-cp39-xx-xx.whl
  - Python version 3.10: DaoAI_SDK-xx-cp310-xx-xx.whl
  - Python version 3.11: DaoAI_SDK-xx-cp311-xx-xx.whl

- install the DaoAI_SDK

```
py -m pip install path/to/DaoAI_SDK-xx-xx.xx.xx.whl
```

- If required dependencies were not bundled into the package on release:
  - Add the following lines of code to the top of your python program to tell python where to find required dependencies (only required once).

```
import os
daoai_slc_path = os.getenv('DAOAI_SLC_PATH') # System enviornment variable␣
↪should point to DaoAI Studio Release path.
os.add_dll_directory(daoai_slc_path + '') # Various dependencies
os.add_dll_directory(daoai_slc_path + '/bin') # Various dependencies
os.add_dll_directory(daoai_slc_path + '/SDK/bin') # Directory including slc_
↪dll.dll
```

- To use the API functions, import the SDK to python.

```
from DaoAI_SDK import *
```

- All set to use the Python API!

6.2 Helper Functions

C++

```
// Helper for checking error information from a returned SlcSdkError object.
bool hasError(DaoAI::SlcSdkError error_info) {
    if (error_info.status() == DaoAI::SlcSdkSuccess) { // A status code of SlcSdkSuccess␣
↪indicates that no error is detected.
        return false;
    }
    else {
        // Consult documentation or header error.h for the meaning of different error␣
↪status codes.
        // Most errors will come with a detailed description, helpful for debugging. See␣
↪SlcSdkError.details().
        //      NOTE: The details section may still include warnings even when the␣
↪status code is SlcSdkSuccess.
        std::cout << "ERROR " << error_info.status() << ": " << error_info.details() <<␣
↪std::endl;
        return true;
    }
}
```

C#

```
// Helper for checking error information from a returned SlcSdkError object.
static bool HasError(DaoAINETError err)
{
   if (err.status() == DaoAINETStatus.SlcSdkSuccess)
   {
         return false;  // A status code of SlcSdkSuccess indicates that no error is␣
↪detected.
   }
   else
   {
         // Consult documentation for the meaning of different error status codes.
         // Most errors will come with a detailed description, helpful for debugging.␣
↪See DaoAINETError.details().
         //      NOTE: The details section may still include warnings even when the␣
↪status code is SlcSdkSuccess.
         Console.WriteLine("ERROR: " + err.status() + ": " + err.details());
         System.Threading.Thread.Sleep(20000);
         return true;
   }
}
```

Python

```
# Helper for checking error information from a returned SlcSdkError object.
def hasError(err):
   if (err.status() == SlcSdkSuccess): # A status code of SlcSdkSuccess indicates that␣
↪no error is detected.
         return False
```

(continues on next page)

(continued from previous page)

```
   else:
         # Consult documentation for the meaning of different error status codes.
         # Most errors will come with a detailed description, helpful for debugging. See␣
↪DaoAINETError.details().
         #      NOTE: The details section may still include warnings even when the␣
↪status code is SlcSdkSuccess.
         print("ERROR: ", err.status(), ": ", err.details())
         return True
```

6.3 Setup

C++

```
// Setup␣
↪=============================================================================================================
// Declare an error return object to check for errors throughout the application.
DaoAI::SlcSdkError ret;

// Create a new DaoAI application instance.
DaoAI::Application* app = new DaoAI::Application();

// Specify directory for logging. Logs contain detailed error and process information.
std::string logging_directory = "../../Logs/";
ret = app->startLogging(logging_directory);
if (hasError(ret)) { return -1; } // Check for errors

// If using remote cameras, specify remote IP address
std::string remote_ip = "192.168.1.2";

// Declare camera map that will be used to fetch all connected DaoAI Cameras.
std::map<std::string, DaoAI::Camera*> cameras;

// Get cameras from application. This step must be completed before attempting to␣
↪connect to any camera.
ret = app->getCameras(cameras, remote_ip);
if (hasError(ret)) { return -1; } // Check for errors

if (cameras.size() == 0) {
   return -1; // Must detect at least one camera.
}
std::cout << cameras.size() << " cameras detected." << std::endl;
for (std::pair<std::string, DaoAI::Camera*> pair : cameras) {
   std::cout << "      " << pair.first << std::endl; // Print serial numbers of detected␣
↪cameras.
}

// Declare pointer to DaoAI Camera object.
DaoAI::Camera* cam;
```

C#

```
// Setup␣
↪==========================================================================================================
// Declare an error return object to check for errors throughout the application.
DaoAINETError err;

// Create a new DaoAI application instance.
Application app = new Application();

// Specify directory for logging. Logs contain detailed error and process information.
string logging_directory = "../../../../../Logs/";

err = app.startLogging(logging_directory);

if (HasError(err)) { return; } // Check for errors


// If using remote cameras, specify remote IP address
string remote_ip = "192.168.1.2";

// Declare a dictionary of cameras that will be used to fetch all connected DaoAI␣
↪Cameras.
// Dictionaries are included in the Systems.Collections.Generic namespace.
Dictionary<string, Camera> cameras = new Dictionary<string, Camera>();

// Get cameras from application. This step must be completed before attempting to␣
↪connect to any camera.
err = app.getCameras(ref cameras, remote_ip);
if (HasError(err)) { return; } // Check for errors

if (cameras.Count == 0)
{
      return; // Must detect at least one camera.
}
Console.WriteLine(cameras.Count + " cameras detected.");

foreach (KeyValuePair<string, Camera> pair in cameras)
{
      Console.WriteLine("   " + pair.Key);  // Print serial numbers of detected cameras.
}
```

Python

```
# Setup␣
↪=============================================================================================
# Create a new DaoAI application instance
app = Application()

#Specify directory for logging. Logs contain detailed error and process information.
logging_directory = "../../Logs/"

# Most functions return an error objet that contains.
err = app.startLogging(logging_directory)
if (hasError(err)): return
```

(continues on next page)

(continued from previous page)

```
# If using remote cameras, specify remote IP address.
remote_ip = "192.168.1.2"

# Get cameras from application. This step must be completed before attempting to connect␣
↪to any camera.
cams, err = app.getCameras(remote_ip) # remote_ip is optional if using a USB camera.
if (hasError(err)): return

if (len(cams) == 0):
   return # At least one camera must be detected.
print(len(cams), " cameras detected: ")
for serial, cam in cams.items(): # Cams is a dictionary of serial number -> camera␣
↪object.
   print("    ", serial) # Print all serial numbers of detected cameras.
```

6.4 Connecting to a Camera

Connecting to camera can have 3 Options.

C++

```
// Connecting to a camera␣
↪=============================================================================================
// A DaoAI Camera must be connected before it can be used for captures.
// OPTION 1: Connecting to the first detected DaoAI Camera.
ret = app->connectCamera(cam);
if (hasError(ret)) { return -1; } // Check for errors
ret = cam->disConnect();
if (hasError(ret)) { return -1; } // Check for errors

// OPTION 2: Connect to specific camera by serial number.
std::string serial_num = cameras.begin()->first; // Grab serial number from first camera␣
↪in map.
// Method A
ret = app->connectCamera(serial_num, cam);
if (hasError(ret)) { return -1; } // Check for errors
ret = app->disconnectCamera(serial_num); // Can also disconnect cam by serial number.
if (hasError(ret)) { return -1; } // Check for errors
// Method B
cam = cameras[serial_num];
ret = cam->connect();
if (hasError(ret)) { return -1; } // Check for errors
ret = app->disconnectCamera(serial_num);
if (hasError(ret)) { return -1; } // Check for errors

// OPTION 3: Connecting any camera found in camera map.
if (cameras.size() > 0) {
   cam = cameras.begin()->second;
}
```

(continues on next page)

(continued from previous page)

```
ret = cam->connect();
if (hasError(ret)) { return -1; } // Check for errors
```

C#

```
// Connecting to a camera␣
↪=============================================================================
// A DaoAI Camera must be connected before it can be used for captures.
// OPTION 1: Connecting to the first detected DaoAI Camera.
err = app.connectCamera(ref cam);
if (HasError(err)) { return; } // Check for errors
err = cam.disConnect();
if (HasError(err)) { return; } // Check for errors

// OPTION 2: Connect to specific camera by serial number.
string serial_num = cameras.Keys.First(); // Grab serial number from first camera in␣
↪dictionary.
      // Method A
err = app.connectCamera(serial_num, ref cam);
if (HasError(err)) { return; } // Check for errors
err = cam.disConnect();
if (HasError(err)) { return; } // Check for errors
      // Method B
cam = cameras[serial_num];
err = cam.connect();
if (HasError(err)) { return; } // Check for errors
err = cam.disConnect();
if (HasError(err)) { return; } // Check for errors

// OPTION 3: Connecting any camera found in camera map.
if (cameras.Count > 0)
{
      cam = cameras.Values.First();
}
err = cam.connect();
if (HasError(err)) { return; } // Check for errors
```

Python

```
# Connecting to a camera␣
↪=================================================================
# A DaoAI Camera must be connected before it can be used for captures.
# OPTION 1: Connecting to the first detected DaoAI Camera.
cam, err = app.connectCamera()
if (hasError(err)): return
cam.disConnect()
if (hasError(err)): return

# OPTION 2: Connect to specific camera by serial number.
serial_number = list(cams.keys())[0] # Grab serial number from first camera in␣
↪dictionary.
# Method A
```

(continues on next page)

(continued from previous page)

```
cam, err = app.connectCamera(serial_number)
if (hasError(err)): return
cam.disConnect()
if (hasError(err)): return
# Method B
cam = cams[serial_number]
err = cam.connect()
if (hasError(err)): return
cam.disConnect()
if (hasError(err)): return

# Option 3: Connecting any camera found in camera map
cam = list(cams.values())[0] # Grab first camera object in dictionary.
err = cam.connect()
if (hasError(err)): return
```

6.5 Camera Actions

Get serial number, camera intrinsic parameters, and camera settings information.

C++

```
// Camera Actions␣
↪==========================================================================================
// Some camera actions will require the camera to be connected, be sure to check␣
↪documentation and error messages.
// Check if a camera is connected.
if (!cam->isConnected()) {
   return -1;
}

// Get serial number of this camera.
serial_num = cam->getSerialNumber();
std::cout << "Serial number of connected camera is " << serial_num << std::endl;

// Get camera intrinsic parameters.
std::vector<float> intrinsic_params;
ret = cam->getIntrinsicParam(intrinsic_params);
if (hasError(ret)) { return -1; } // Check for errors

// Get current settings used by this camera.
DaoAI::Settings settings = cam->getSettings();
```

C#

```
// Camera Actions␣
↪==========================================================================================
// Some camera actions will require the camera to be connected, be sure to check␣
↪documentation and error messages.
// Check if a camera is connected.
if (!cam.isConnected())
```

(continues on next page)

(continued from previous page)

```
{
      return;
}

// Get serial number of this camera.
serial_num = cam.getSerialNumber();
Console.WriteLine("Serial number of connected camera is " + serial_num);

// Get camera intrinsic parameters.
float[] intrinsic_params = new float[] { };
err = cam.getIntrinsicParam(ref intrinsic_params);
if (HasError(err)) { return; } // Check for errors

// Get current settings used by this camera.
Settings settings = cam.getSettings();
```

Python

```
# Canera Actions␣
↪================================================================================
# Some camera actions will require the camera to be connected, be sure to check␣
↪documentation and error messages.
# Check if a camera is connected.
if not cam.isConnected():
   return

# Get serial number of this camera.
serial_num = cam.getSerialNumber()
print("Serial number of connected camera is ", serial_num)

# Get camera intrinsic parameters.
params, err = cam.getIntrinsicParam()
if (hasError(err)): return

# Get current settings used by this camera.
settings = cam.getSettings()
```

6.6 Camera Settings

Create camera settings and load from camera setting file.

C++

```
// Camera Settings␣
↪================================================================================================
// DaoAI Settings can be used with a camera to tweak parameters during capture and the␣
↪reconstruction process.
DaoAI::Settings new_settings;
int icurr, imin, imax; // Use these to inquire integer settings.
double dcurr, dmin, dmax; // Use these to inquire double settings.
bool bcurr; // Use this to inquire boolean settings.
```

(continues on next page)

(continued from previous page)

```
std::string scurr; // Use this to inquire string settings.
bool is_enabled; // Use this to check if a setting is enabled.
int inewval; // Use this to set a new integer value to a setting.
double dnewval; // Use this to set a new double value to a setting.
bool bnewval; // Use this to set a new boolean value to a setting.
// Creating new empty Camera Settings
new_settings = DaoAI::Settings();
// Loading existing Camera Settings from file.
std::string path_to_settings = "../../Examples/sample_settings.cfg";
new_settings = DaoAI::Settings(path_to_settings);
// Cloning settings
new_settings = DaoAI::Settings(settings);
```

C#

```
// Camera Settings␣
↪==========================================================================================
// DaoAI Settings can be used with a camera to tweak parameters during capture and the␣
↪reconstruction process.
Settings new_settings;
int icurr = -1, imin = -1, imax = -1; // Use these to inquire integer settings.
double dcurr = -1.0, dmin = -1.0, dmax = -1.0; // Use these to inquire double settings.
bool bcurr = false; // Use this to inquire boolean settings.
string scurr = ""; // Use this to inquire string settings.
bool is_enabled = false; // Use this to check if a setting is enabled.
int inewval = 0; // Use this to set a new integer value to a setting.
double dnewval = 0.0; // Use this to set a new double value to a setting.
bool bnewval = true ; // Use this to set a new boolean value to a setting.

// Creating new empty Camera Settings
new_settings = new Settings();
// Loading existing Camera Settings from file.
string path_to_settings = "../../../../../Examples/sample_settings.cfg";
new_settings = new Settings(path_to_settings);
// Cloning settings
new_settings = new Settings(settings);
```

Python

```
# Camera Settings␣
↪==========================================================================
# DaoAI Settings can be used with a camera to tweak parameters during capture and the␣
↪reconstruction process.
# Create a new empty settings object.
new_settings = Settings()
# Load existing camera settings from file.
path_to_settings = "../../Examples/sample_settings.cfg"
new_settings = Settings(path_to_settings)
# Clone settings
new_settings = Settings(settings)
```

6.6.1 Acquisition Frames

Configure Acquisition frames parameters.

C++

```cpp
// Acquisition Frames
// Acquisition frames specify parameters to be used during image capture. A settings␣
↪object can support up to 10.
//     Each acquisition frame has three modifiable parameters: Brightness, Gain and␣
↪ExposureStop.
//     See documentation for details.
DaoAI::AcquisitionFrame af;

// Create default AcquisitionFrame
af = DaoAI::AcquisitionFrame();

// Create AcquisitionFrame with initial values
int brightness = 3;
double gain = 2.0;
int exposure_stop = -1;
af = DaoAI::AcquisitionFrame(brightness, gain, exposure_stop);

// View the current value and acceptable bounds for any AcquisitionFrame parameter.
ret = af.inquireSetting(DaoAI::AcquisitionFrame::ExposureStop, icurr, imin, imax);
if (hasError(ret)) { return -1; } // Check for errors
std::cout << "Current exposure stop: " << icurr << ". Exposure stop can be configured to␣
↪any value between " << imin << " - " << imax << std::endl;
ret = af.inquireSetting(DaoAI::AcquisitionFrame::ExposureStop, icurr); // Inquire only␣
↪current value.
if (hasError(ret)) { return -1; } // Check for errors

// Configure any AcquisitionFrame parameter to a custom value.
ret = af.configureSetting(DaoAI::AcquisitionFrame::ExposureStop, 2);
if (hasError(ret)) { return -1; } // Check for errors

// Double parameters can also be retreived and modified with double values.
ret = af.inquireSetting(DaoAI::AcquisitionFrame::Gain, dcurr, dmin, dmax);
if (hasError(ret)) { return -1; } // Check for errors
std::cout << "Current gain: " << dcurr << ". Gain can be configured to any value between
↪" << dmin << " - " << dmax << std::endl;
ret = af.inquireSetting(DaoAI::AcquisitionFrame::Gain, dcurr); // Inquire only current␣
↪value.
if (hasError(ret)) { return -1; } // Check for errors

ret = af.configureSetting(DaoAI::AcquisitionFrame::Gain, 2);
if (hasError(ret)) { return -1; } // Check for errors

// Using the incorrect type to configure or inquire a parameter will be successful but␣
↪will return a warning.
ret = af.inquireSetting(DaoAI::AcquisitionFrame::Gain, icurr, imin, imax);
if (hasError(ret)) { return -1; } // Check for errors
std::cout << ret.details() << std::endl; // Warning about possible data loss, attempting␣
↪to read double as int.
```

(continues on next page)

(continued from previous page)

```
dnewval = 1.5;
ret = af.configureSetting(DaoAI::AcquisitionFrame::ExposureStop, dnewval);
if (hasError(ret)) { return -1; } // Check for errors
std::cout << ret.details() << std::endl; // Warning about possible data loss, attempting␣
↪to set int with double.

// Add acquisition frame to settings.
int index; // Index of added acquisition frame.
ret = new_settings.addAcquisitionFrame(af, index);
if (hasError(ret)) { return -1; } // Check for errors

// Get acquisition frame
DaoAI::AcquisitionFrame returned_af;
ret = new_settings.getAcquisitionFrame(returned_af, 1);
if (hasError(ret)) { return -1; } // Check for errors

// Delete acquisition frame at index.
ret = new_settings.deleteAcquisitionFrame(index);
if (hasError(ret)) { return -1; } // Check for errors

// Add acquisition frame without getting index.
ret = new_settings.addAcquisitionFrame(af);
if (hasError(ret)) { return -1; } // Check for errors

// Modify and replace the acquisition frame at index 1.
ret = af.configureSetting(DaoAI::AcquisitionFrame::Brightness, 2);
if (hasError(ret)) { return -1; } // Check for errors
ret = new_settings.modifyAcquisitionFrame(af, 1);
if (hasError(ret)) { return -1; } // Check for errors

std::map<int, DaoAI::AcquisitionFrame> mofaf;
// Get copy of entire map of acquisition frames.
ret = new_settings.getAcquisitionFrames(mofaf);
if (hasError(ret)) { return -1; } // Check for errors

// Set map of acquisition frames to settings.
mofaf[1] = DaoAI::AcquisitionFrame(1, 0, 1);
mofaf[2] = DaoAI::AcquisitionFrame(2, 2, 2);
ret = new_settings.setAcquisitionFrames(mofaf);
if (hasError(ret)) { return -1; } // Check for errors
```

C#

```
// Acquisition Frames
// Acquisition frames specify parameters to be used during image capture. A settings␣
↪object can support up to 10.
//     Each acquisition frame has three modifiable parameters: Brightness, Gain and␣
↪ExposureStop.
//     See documentation for details.
AcquisitionFrame af;

// Create default AcquisitionFrame
```

(continues on next page)

(continued from previous page)

```
af = new AcquisitionFrame();

// Create AcquisitionFrame with initial values
int brightness = 3;
double gain = 2.0;
int exposure_stop = -1;
af = new AcquisitionFrame(brightness, gain, exposure_stop);

// View the current value and acceptable bounds for any AcquisitionFrame parameter.
err = af.inquireSetting(AcquisitionFrame.AcquisitionFrameSetting.ExposureStop, ref icurr,
↪ ref imin, ref imax);
if (HasError(err)) { return; } // Check for errors
Console.WriteLine("Current exposure stop: " + icurr + ". Exposure stop can be configured␣
↪to any value between " + imin + " - " + imax);
err = af.inquireSetting(AcquisitionFrame.AcquisitionFrameSetting.ExposureStop, ref␣
↪icurr); // Inquire only current value.
if (HasError(err)) { return; } // Check for errors

// Configure any AcquisitionFrame parameter to a custom value.
err = af.configureSetting(AcquisitionFrame.AcquisitionFrameSetting.ExposureStop, 2);
if (HasError(err)) { return; } // Check for errors

// Double parameters can also be retreived and modified with double values.
err = af.inquireSetting(AcquisitionFrame.AcquisitionFrameSetting.Gain, ref dcurr, ref␣
↪dmin, ref dmax);
if (HasError(err)) { return; } // Check for errors
Console.WriteLine("Current gain: " + dcurr + ". Gain can be configured to any value␣
↪between " + dmin + " - " + dmax);
err = af.inquireSetting(AcquisitionFrame.AcquisitionFrameSetting.Gain, ref dcurr); //␣
↪Inquire only current value.
if (HasError(err)) { return; } // Check for errors

err = af.configureSetting(AcquisitionFrame.AcquisitionFrameSetting.Gain, 2.1);
if (HasError(err)) { return; } // Check for errors

// Using the incorrect type to configure or inquire a parameter will be successful but␣
↪will return a warning.
err = af.inquireSetting(AcquisitionFrame.AcquisitionFrameSetting.Gain, ref icurr, ref␣
↪imin, ref imax);
if (HasError(err)) { return; } // Check for errors
Console.WriteLine(err.details()); // Warning about possible data loss, attempting to␣
↪read double as int.
dnewval = 1.5;
err = af.configureSetting(AcquisitionFrame.AcquisitionFrameSetting.ExposureStop,␣
↪dnewval);
if (HasError(err)) { return; } // Check for errors
Console.WriteLine(err.details()); // Warning about possible data loss, attempting to set␣
↪int with double.

// Add acquisition frame to settings.
int index = -1; // Index of added acquisition frame.
err = new_settings.addAcquisitionFrame(af, ref index);
```

(continues on next page)

(continued from previous page)

```
if (HasError(err)) { return; } // Check for errors

// Get acquisition frame
AcquisitionFrame returned_af = new AcquisitionFrame();
err = new_settings.getAcquisitionFrame(ref returned_af, 1);
if (HasError(err)) { return; } // Check for errors

// Delete acquisition frame at index.
err = new_settings.deleteAcquisitionFrame(index);
if (HasError(err)) { return; } // Check for errors

// Add acquisition frame without getting index.
err = new_settings.addAcquisitionFrame(af);
if (HasError(err)) { return; } // Check for errors

// Modify and replace the acquisition frame at index 1.
err = af.configureSetting(AcquisitionFrame.AcquisitionFrameSetting.Brightness, 2);
if (HasError(err)) { return; } // Check for errors
err = new_settings.modifyAcquisitionFrame(af, 1);
if (HasError(err)) { return; } // Check for errors

Dictionary<int, AcquisitionFrame> mofaf = new Dictionary<int, AcquisitionFrame>();
// Get copy of entire dictionary of acquisition frames currently saved in settings.
err = new_settings.getAcquisitionFrames(ref mofaf);
if (HasError(err)) { return; } // Check for errors

// Set map of acquisition frames to settings. Remember that the acquisition frame␣
↪dictionary is one-indexed.
mofaf[1] = new AcquisitionFrame(1, 0, 1);
mofaf[2] = new AcquisitionFrame(2, 2, 2);
err = new_settings.setAcquisitionFrames(mofaf);
if (HasError(err)) { return; } // Check for errors
```

Python

```
# Acquisition Frames.
# Acquisition frames specify parameters to be used during image capture. A settings␣
↪object can support up to 10.
#     Each acquisition frame has three modifiable parameters: Brightness, Gain and␣
↪ExposureStop.
#     See documentation for details.
# Create a new default AcquisitionFrame
af = AcquisitionFrame()

# Create AcquisitionFrame with initial values
brightness = 3
gain = 2.0
exposure_stop = -1
af = AcquisitionFrame(brightness, gain, exposure_stop)

# View the current value and acceptable bounds for any AcquisitionFrame parameter.
curr, min, max, err = af.inquireSetting(ExposureStop)
```

(continues on next page)

(continued from previous page)

```
if (hasError(err)): return
print("Current exposure stop: ", curr, ". Exposure stop can be configured to any value␣
↪between ", min, " - ", max)

# Configure any AcquisitionFrame parameter to a custom value.
err = af.configureSetting(ExposureStop, 2)
if (hasError(err)): return

# Some parameters can be configured/retrieved with decimal values. See documentation for␣
↪details.
curr, min, max, err = af.inquireSetting(Gain)
if (hasError(err)): return
print("Current gain: ", curr, ". Gain can be configured to any value between ", min, " -
↪", max)

err = af.configureSetting(Gain, 2.1)
if (hasError(err)): return

# Using a decimal value to configure an integer-only setting will generate an error.
err = af.configureSetting(ExposureStop, 1.5) # ExposureStop does not support decimal␣
↪values, and will configure to 1.0
if (hasError(err)): return
print(err.details()) # No error is returned, but details will include a warning.

# Add acquisition frame to settings.
idx, err = new_settings.addAcquisitionFrame(af) # Returns the index of the newly added␣
↪acquisition frame.
if (hasError(err)): return

# Get acquisition frame.
returned_af, err = new_settings.getAcquisitionFrame(1) # Get frame at index 1
if (hasError(err)): return

# Delete acquisition frame.
err = new_settings.deleteAcquisitionFrame(idx)
if (hasError(err)): return

# Modify and replace the acquisition frame at index 1.
err = af.configureSetting(Brightness, 2)
if (hasError(err)): return
err = new_settings.modifyAcquisitionFrame(af, 1)
if (hasError(err)): return

# Get copy of entire dictionary of acquisition frames currently saved in settings.
mofaf, err = new_settings.getAcquisitionFrames()
if (hasError(err)): return

# Set map of acquisition frames to settings. Remember that the acquisition frame␣
↪dictionary is one-indexed.
mofaf[1] = AcquisitionFrame(1, 0, 1)
mofaf[2] = AcquisitionFrame(2, 2, 2)
err = new_settings.setAcquisitionFrames(mofaf)
```

(continues on next page)

(continued from previous page)

```
if (hasError(err)): return
```

6.6.2 Capture Assistant

Auto compute acquisition frame settings by analyzing scene given a time buget.

C++

```
// Capture Assistant
// Analyze scene and generate acquisition frame settings, the total time for all␣
↪acquisition frames will be less than the time budget.
//             The higher time budget is, the more acquisition frames will be generated.
std::map<int, DaoAI::AcquisitionFrame> ca_mofaf;
ret = cam->captureAssistant(1.0, ca_mofaf);  // Generate a map of acquisition frames␣
↪with time budget of 1 sec.
if (hasError(ret)) { return -1; }
ret = new_settings.setAcquisitionFrames(ca_mofaf);  // Set the generated acquisition␣
↪frames to camera settings
if (hasError(ret)) { return -1; }
ret = cam->setSettings(new_settings);  // Apply the camera settings to camera
if (hasError(ret)) { return -1; }
DaoAI::Frame ca_frm;
ret = cam->capture(ca_frm);  // Capture point cloud
if (hasError(ret)) { return -1; }
```

C#

```
// Capture Assistant
// Analyze scene and generate acquisition frame settings, the total time for all␣
↪acquisition frames will be less than the time budget.
//             The higher time budget is, the more acquisition frames will be generated.
Dictionary<int, AcquisitionFrame> ca_mofaf = new Dictionary<int, AcquisitionFrame>();
err = cam.captureAssistant(1.0, ref ca_mofaf);  // Generate a map of acquisition frames␣
↪with time budget of 1 sec.
if (HasError(err)) { return; }
err = new_settings.setAcquisitionFrames(ca_mofaf);  // Set the generated acquisition␣
↪frames to camera settings
if (HasError(err)) { return; }
err = cam.setSettings(new_settings);  // Apply the camera settings to camera
if (HasError(err)) { return; }
Frame ca_frm = new Frame();
err = cam.capture(ref ca_frm);  // Capture point cloud
if (HasError(err)) { return; }
```

Python

```
# Capture Assistant
# Analyze scene and generate acquisition frame settings, the total time for all␣
↪acquisition frames will be less than the time budget.
#           The higher time budget is, the more acquisition frames will be generated.
ca_mofaf, err = cam.captureAssistant(1.0) # Generate a set of acquisition frames with␣
↪time budget of 1 sec.
```

(continues on next page)

(continued from previous page)

```
if (hasError(err)): return
err = new_settings.setAcquisitionFrames(ca_mofaf)  # Set the generated acquisition␣
↪frames to camera settings
if (hasError(err)): return
err = cam.setSettings(new_settings)  # Apply the camera settings to camera
if (hasError(err)): return
ca_frame, err = cam.capture()  # Capture point cloud using generated settings
if (hasError(err)): return
```

6.6.3 Filter Settings

Create, read, and modify Filter settings.

C++

```
// Filter Settings
// Filter settings specify parameters that are used during 3D reconstruction. For a full␣
↪list of filter settings
//      and their descriptions consult settings.h and the documentation.
// Enable or Disable filter settings.
ret = new_settings.enableFilterSetting(DaoAI::Settings::OutlierThreshold, true); //␣
↪Enable outlier filter
if (hasError(ret)) { return -1; } // Check for errors
ret = new_settings.enableFilterSetting(DaoAI::Settings::GaussianFilter, false); //␣
↪Disable gaussian filter
if (hasError(ret)) { return -1; } // Check for errors
ret = new_settings.enableFilterSetting(DaoAI::Settings::FillGaps, true); // Enable Fill␣
↪Gaps
if (hasError(ret)) { return -1; } // Check for errors

// Check if a filter setting is enabled.
ret = new_settings.checkEnableFilterSetting(DaoAI::Settings::OutlierThreshold, is_
↪enabled); // Check if outlier filter is enabled.
if (hasError(ret)) { return -1; } // Check for errors
if (is_enabled) { std::cout << "Outlier filter is enabled!" << std::endl; }
ret = new_settings.checkEnableFilterSetting(DaoAI::Settings::GaussianFilter, is_enabled);
↪ // Check if gaussian filter is enabled.
if (hasError(ret)) { return -1; } // Check for errors
if (is_enabled) { std::cout << "Gaussian filter is enabled!" << std::endl; }
ret = new_settings.checkEnableFilterSetting(DaoAI::Settings::FillGaps, is_enabled); //␣
↪Enable Fill Gaps
if (hasError(ret)) { return -1; } // Check for errors
if (is_enabled) { std::cout << "Fill gaps is enabled!" << std::endl; }

// Get the current value and valid range of a filter setting.
ret = new_settings.inquireFilterSetting(DaoAI::Settings::OutlierThreshold, dcurr, dmin,␣
↪dmax);
if (hasError(ret)) { return -1; } // Check for errors
std::cout << "Outlier threshold filter has a current value of " << dcurr << ", with a␣
↪valid range of " << dmin << " - " << dmax << std::endl;
ret = new_settings.inquireFilterSetting(DaoAI::Settings::OutlierThreshold, dcurr); //␣
```

(continues on next page)

(continued from previous page)

```
↪Can also get current value without checking range.
if (hasError(ret)) { return -1; } // Check for errors
ret = new_settings.inquireFilterSetting(DaoAI::Settings::GaussianFilter, icurr, imin,␣
↪imax);
if (hasError(ret)) { return -1; } // Check for errors
std::cout << "Gaussian filter has a current value of " << icurr << ", with a valid range␣
↪of " << imin << " - " << imax << std::endl;
ret = new_settings.inquireFilterSetting(DaoAI::Settings::GaussianFilter, icurr); // Can␣
↪also get current value without checking range.
if (hasError(ret)) { return -1; } // Check for errors
ret = new_settings.inquireFilterSetting(DaoAI::Settings::FillGaps, bcurr);
if (hasError(ret)) { return -1; } // Check for errors

// Configure a filter setting.
inewval = 2;
dnewval = 3.4;
bnewval = true;
ret = new_settings.configureFilterSetting(DaoAI::Settings::OutlierThreshold, dnewval);
if (hasError(ret)) { return -1; } // Check for errors
ret = new_settings.configureFilterSetting(DaoAI::Settings::GaussianFilter, inewval);
if (hasError(ret)) { return -1; } // Check for errors
ret = new_settings.configureFilterSetting(DaoAI::Settings::FillXFirst, bnewval);
if (hasError(ret)) { return -1; } // Check for errors

// For numeric filter settings, using a type mismatch getter or setter will work␣
↪successfully but issue a warning.
ret = new_settings.inquireFilterSetting(DaoAI::Settings::OutlierThreshold, icurr);
if (hasError(ret)) { return -1; } // Expect no error (status = DaoAI::SlcSdkSuccess)
std::cout << ret.details() << std::endl; // Print warning message for using int value to␣
↪retrieve a double parameter.
dnewval = 1.5;
ret = new_settings.inquireFilterSetting(DaoAI::Settings::GaussianFilter, dnewval);
if (hasError(ret)) { return -1; } // Expect no error (status = DaoAI::SlcSdkSuccess)
std::cout << ret.details() << std::endl; // Print warning message for using double value␣
↪to set an integer parameter.
```

C#

```
// Filter Settings
// Filter settings specify parameters that are used during 3D reconstruction. For a full␣
↪list of filter settings
//      and their descriptions consult settings.h and the documentation.
// Enable or Disable filter settings.
err = new_settings.enableFilterSetting(Settings.FilterSetting.OutlierThreshold, true); //
↪ Enable outlier filter
if (HasError(err)) { return; } // Check for errors
err = new_settings.enableFilterSetting(Settings.FilterSetting.GaussianFilter, false); //␣
↪Disable gaussian filter
if (HasError(err)) { return; } // Check for errors
err = new_settings.enableFilterSetting(Settings.FilterSetting.FillGaps, true); // Enable␣
↪Fill Gaps
if (HasError(err)) { return; } // Check for errors
```

(continues on next page)

(continued from previous page)

```
// Check if a filter setting is enabled.
err = new_settings.checkEnableFilterSetting(Settings.FilterSetting.OutlierThreshold, ref␣
↪is_enabled); // Check if outlier filter is enabled.
if (HasError(err)) { return; } // Check for errors
if (is_enabled) { Console.WriteLine("Outlier filter is enabled!"); }
err = new_settings.checkEnableFilterSetting(Settings.FilterSetting.GaussianFilter, ref␣
↪is_enabled); // Check if gaussian filter is enabled.
if (HasError(err)) { return; } // Check for errors
if (is_enabled) { Console.WriteLine("Gaussian filter is enabled!" ); }
err = new_settings.checkEnableFilterSetting(Settings.FilterSetting.FillGaps, ref is_
↪enabled); // Enable Fill Gaps
if (HasError(err)) { return; } // Check for errors
if (is_enabled) { Console.WriteLine("Fill gaps is enabled!"); }

// Get the current value and valid range of a filter setting.
err = new_settings.inquireFilterSetting(Settings.FilterSetting.OutlierThreshold, ref␣
↪dcurr, ref dmin, ref dmax);
if (HasError(err)) { return; } // Check for errors
Console.WriteLine("Outlier threshold filter has a current value of " + dcurr + ", with a␣
↪valid range of " + dmin + " - " + dmax);
err = new_settings.inquireFilterSetting(Settings.FilterSetting.OutlierThreshold, ref␣
↪dcurr); // Can also get current value without checking range.
if (HasError(err)) { return; } // Check for errors
err = new_settings.inquireFilterSetting(Settings.FilterSetting.GaussianFilter, ref icurr,
↪ ref imin, ref imax);
if (HasError(err)) { return; } // Check for errors
Console.WriteLine("Gaussian filter has a current value of " + icurr + ", with a valid␣
↪range of " + imin + " - " + imax);
err = new_settings.inquireFilterSetting(Settings.FilterSetting.GaussianFilter, ref␣
↪icurr); // Can also get current value without checking range.
if (HasError(err)) { return; } // Check for errors
err = new_settings.inquireFilterSetting(Settings.FilterSetting.FillGaps, ref bcurr);
if (HasError(err)) { return; } // Check for errors

// Configure a filter setting.
inewval = 2;
dnewval = 3.4;
bnewval = true;
err = new_settings.configureFilterSetting(Settings.FilterSetting.OutlierThreshold,␣
↪dnewval);
if (HasError(err)) { return; } // Check for errors
err = new_settings.configureFilterSetting(Settings.FilterSetting.GaussianFilter,␣
↪inewval);
if (HasError(err)) { return; } // Check for errors
err = new_settings.configureFilterSetting(Settings.FilterSetting.FillXFirst, bnewval);
if (HasError(err)) { return; } // Check for errors

// For numeric filter settings, using a type mismatch getter or setter will work␣
↪successfully but issue a warning.
err = new_settings.inquireFilterSetting(Settings.FilterSetting.OutlierThreshold, ref␣
↪icurr);
```

(continues on next page)

(continued from previous page)

```
if (HasError(err)) { return; } // Expect no error (status = SlcSdkSuccess)
Console.WriteLine(err.details()); // Print warning message for using int value to␣
↪retrieve a double parameter.
dnewval = 1.5;
err = new_settings.configureFilterSetting(Settings.FilterSetting.GaussianFilter,␣
↪dnewval);
if (HasError(err)) { return; } // Expect no error (status = SlcSdkSuccess)
Console.WriteLine(err.details()); // Print warning message for using double value to set␣
↪an integer parameter.
```

Python

```
# Filter Settings
# Filter settings specify parameters that are used during 3D reconstruction. For a full␣
↪list of filter settings
#      and their descriptions consult settings.h and the documentation.
# Enable or Disable filter settings.
err = new_settings.enableFilterSetting(OutlierThreshold, True) # Enable outlier filter
if (hasError(err)): return
err = new_settings.enableFilterSetting(GaussianFilter, False) # Disable gaussian filter
if (hasError(err)): return
err = new_settings.enableFilterSetting(FillGaps, True) # Enable Fill Gaps
if (hasError(err)): return

# Check if a filter sitting is enabled.
is_enabled, err = new_settings.checkEnableFilterSetting(OutlierThreshold) # Check if␣
↪outlier filter is enabled.
if (hasError(err)): return
if is_enabled : print("Outlier filter is enabled!")
is_enabled, err = new_settings.checkEnableFilterSetting(GaussianFilter) # Check if␣
↪gaussian filter is enabled.
if (hasError(err)): return
if is_enabled : print("Gaussian filter is enabled!")
is_enabled, err = new_settings.checkEnableFilterSetting(FillGaps) # Enable Fill Gaps
if (hasError(err)): return
if is_enabled : print("Fill gaps is enabled!")

# Get the current value and valid range of a filter setting.
curr, min, max, err = new_settings.inquireFilterSetting(OutlierThreshold)
if (hasError(err)): return
print("Outlier threshold filter has a current value of ", curr, ", with a valid range of
↪", min, " - ", max)
curr, min, max, err = new_settings.inquireFilterSetting(GaussianFilter)
if (hasError(err)): return
print("Gaussian filter has a current value of ", curr, ", with a valid range of ", min,
↪" - ", max)
fillgaps, err = new_settings.inquireFilterSetting(FillGaps)
if (hasError(err)): return
if(fillgaps): print("Fill Gaps is turned on!")

# Configure a filter setting
err = new_settings.configureFilterSetting(OutlierThreshold, 3.4)
```

(continues on next page)

(continued from previous page)

```
if (hasError(err)): return
err = new_settings.configureFilterSetting(GaussianFilter, 2)
if (hasError(err)): return
err = new_settings.configureFilterSetting(FillXFirst, True)
if (hasError(err)): return

# For numeric filter settings, using a type mismatch setter will work successfully but␣
↪issue a warning.
# Check documentation for which settings support decimal values.
err = new_settings.configureFilterSetting(GaussianFilter, 1.6)
if (hasError(err)): return # Expect no error (status = SlcSdkSuccess)
print(err.details()) # Print warning message for using double value to set an integer␣
↪parameter.
```

6.6.4 System Settings

Create, read, and export System settings.

C++

```
// System Settings
// System settings are miscellaneous parameters that describe and affect the DaoAI␣
↪System. For a full list of system
//      settings and their descriptions consult settings.h and the documentation.
//      NOTE: Many of these system settings are read-only, and may not be accurate for␣
↪current camera system
//            unless getting the updated settings object directly from a camera␣
↪[DaoAI::Camera.getSettings()].
// Enable or Disable System Setting
ret = new_settings.configureSystemSetting(DaoAI::Settings::ExtraWhitePatternEnable,␣
↪false);
if (hasError(ret)) { return -1; } // Check for errors
ret = new_settings.configureSystemSetting(DaoAI::Settings::TemperatureRegulationEnable,␣
↪true);
if (hasError(ret)) { return -1; } // Check for errors

// Check if a system setting is enabled.
ret = new_settings.checkEnableSystemSetting(DaoAI::Settings::ExtraWhitePatternEnable, is_
↪enabled);
if (hasError(ret)) { return -1; } // Check for errors
if (is_enabled) { std::cout << "Extra white pattern is enabled!" << std::endl; }
ret = new_settings.checkEnableSystemSetting(DaoAI::Settings::TemperatureRegulationEnable,
↪ is_enabled);
if (hasError(ret)) { return -1; } // Check for errors
if (is_enabled) { std::cout << "Temperature regulation is enabled!" << std::endl; }

// Get the current value of a system setting.
ret = new_settings.inquireSystemSetting(DaoAI::Settings::GPUAvailable, bcurr);
if (hasError(ret)) { return -1; } // Check for errors
if (bcurr) { std::cout << "GPU is Available on your system!" << std::endl; }
ret = new_settings.inquireSystemSetting(DaoAI::Settings::CameraModel, scurr);
```

(continues on next page)

(continued from previous page)

```
if (hasError(ret)) { return -1; } // Check for errors
std::cout << "This camera has model " << scurr << std::endl;

// Save and export settings.
std::string save_settings_path = "../../Examples/example_setting_save.cfg";
ret = new_settings.exportSettings(save_settings_path);
if (hasError(ret)) { return -1; } // Check for errors
```

C#

```
// System Settings
// System settings are miscellaneous parameters that describe and affect the DaoAI␣
↪System. For a full list of system
//      settings and their descriptions consult settings.h and the documentation.
//      NOTE: Many of these system settings are read-only, and may not be accurate for␣
↪current camera system
//            unless getting the updated settings object directly from a camera [Camera.
↪getSettings()].
// Enable or Disable System Setting
err = new_settings.configureSystemSetting(Settings.SystemSetting.ExtraWhitePatternEnable,
↪ false);
if (HasError(err)) { return; } // Check for errors

// Check if a system setting is enabled.
err = new_settings.checkEnableSystemSetting(Settings.SystemSetting.
↪ExtraWhitePatternEnable, ref is_enabled);
if (HasError(err)) { return; } // Check for errors
if (is_enabled) { Console.WriteLine("Extra white pattern is enabled!"); }
err = new_settings.checkEnableSystemSetting(Settings.SystemSetting.
↪TemperatureRegulationEnable, ref is_enabled);
if (HasError(err)) { return; } // Check for errors
if (is_enabled) { Console.WriteLine("Temperature regulation is enabled!"); }

// Get the current value of a system setting.
err = new_settings.inquireSystemSetting(Settings.SystemSetting.GPUAvailable, ref bcurr);
if (HasError(err)) { return; } // Check for errors
if (bcurr) { Console.WriteLine("GPU is Available on your system!"); }
err = new_settings.inquireSystemSetting(Settings.SystemSetting.CameraModel, ref scurr);
if (HasError(err)) { return; } // Check for errors
Console.WriteLine("This camera has model " + scurr);

// Save and export settings.
string save_settings_path = "../../../../../Examples/example_setting_save.cfg";
err = new_settings.exportSettings(save_settings_path);
if (HasError(err)) { return; } // Check for errors
```

Python

```
# System Settings
# System settings are miscellaneous parameters that describe and affect the DaoAI System.
↪ For a full list of system
#      settings and their descriptions consult settings.h and the documentation.
```

(continues on next page)

(continued from previous page)

```
#      NOTE: Many of these system settings are read-only, and may not be accurate for␣
↪current camera system
#            unless getting the updated settings object directly from a camera [Camera.
↪getSettings()].
# Enable or Disable System Setting
err = new_settings.configureSystemSetting(ExtraWhitePatternEnable, False)
if (hasError(err)): return # Expect no error (status = SlcSdkSuccess)


# Check if a system setting is enabled.
is_enabled, err = new_settings.checkEnableSystemSetting(ExtraWhitePatternEnable)
if (hasError(err)): return
if (is_enabled) : print("Extra white pattern is enabled!")
is_enabled, err = new_settings.checkEnableSystemSetting(TemperatureRegulationEnable)
if (hasError(err)): return
if (is_enabled) : print("Temperature regulation is enabled!")

# Get the current value of a system setting.
available, err = new_settings.inquireSystemSetting(GPUAvailable)
if (hasError(err)): return
if (available): print("GPU is Available on your system!")
model, err = new_settings.inquireSystemSetting(CameraModel)
if (hasError(err)): return
print("This camera has model " + model)

# Save and export settings.
save_settings_path = "../../Examples/example_setting_save.cfg"
err = new_settings.exportSettings(save_settings_path)
if (hasError(err)): return
```

6.7 Capture

Capture image.

C++

```
// Camera Captures␣
↪==========================================================================================
// Declare a DaoAI Frame object to which capture data will be written
DaoAI::Frame frm;
// Capture with default settings (assuming no settings has been set to camera).
ret = cam->capture(frm);
if (hasError(ret)) { return -1; } // Check for errors

// Capture with custom settings
// OPTION 1: Capture with settings. Settings saved by camera for future captures.
ret = cam->capture(new_settings, frm);
if (hasError(ret)) { return -1; } // Check for errors
// OPTION 2: Set settings object to camera to use in capture.
ret = cam->setSettings(new_settings);
```

(continues on next page)

(continued from previous page)

```
if (hasError(ret)) { return -1; } // Check for errors
ret = cam->capture(frm);
if (hasError(ret)) { return -1; } // Check for errors
// OPTION 3: Load settings from file to camera to use in capture.
ret = cam->setSettings("../../Examples/sample_settings.cfg");
if (hasError(ret)) { return -1; } // Check for errors
ret = cam->capture(frm);
if (hasError(ret)) { return -1; } // Check for errors

// Use HDR image as captured frame's color
ret = new_settings.enableFilterSetting(DaoAI::Settings::ShowHDR, true);
if (hasError(ret)) { return -1; }
ret = cam->setSettings(new_settings);
if (hasError(ret)) { return -1; }
ret = cam->capture(frm);
if (hasError(ret)) { return -1; }
// Use the first acquisition frame image as captured frame's color
ret = new_settings.enableFilterSetting(DaoAI::Settings::ShowHDR, false);
if (hasError(ret)) { return -1; }
ret = cam->setSettings(new_settings);
if (hasError(ret)) { return -1; }
ret = cam->capture(frm);
if (hasError(ret)) { return -1; }

// Enable computation using local GPU (for BP-AMR and USB interface 3D cameras only)
ret = cam->enableGPU(true);
if (hasError(ret)) { return -1; }
ret = cam->capture(frm);
if (hasError(ret)) { return -1; }
// Disable computation using local GPU, use CPU instead (for BP-AMR and USB interface 3D␣
↪cameras only)
ret = cam->enableGPU(false);
if (hasError(ret)) { return -1; }
ret = cam->capture(frm);
if (hasError(ret)) { return -1; }

// Enable temperature regulation
ret = cam->enableTempRegulation(true);
if (hasError(ret)) { return -1; }
// Disable temperature regulation
ret = cam->enableTempRegulation(false);
if (hasError(ret)) { return -1; }
```

C#

```
// Camera Captures␣
↪=============================================================================================
// Declare a DaoAI Frame object to which capture data will be written
Frame frm = new Frame();
// Capture with default settings (assuming no settings has been set to camera).
err = cam.capture(ref frm);
if (HasError(err)) { return; } // Check for errors
```

(continues on next page)

(continued from previous page)

```
// Capture with custom settings
// OPTION 1: Capture with settings. Settings saved by camera for future captures.
err = cam.capture(new_settings, ref frm);
if (HasError(err)) { return; } // Check for errors
// OPTION 2: Set settings object to camera to use in capture.
err = cam.setSettings(new_settings);
if (HasError(err)) { return; } // Check for errors
err = cam.capture(ref frm);
if (HasError(err)) { return; } // Check for errors
// OPTION 3: Load settings from file to camera to use in capture.
err = cam.setSettings("../../../../../Examples/sample_settings.cfg");
if (HasError(err)) { return; } // Check for errors
err = cam.capture(ref frm);
if (HasError(err)) { return; } // Check for errors

// Use HDR image as captured frame's color
err = new_settings.enableFilterSetting(Settings.FilterSetting.ShowHDR, true);
if (HasError(err)) { return; }
err = cam.setSettings(new_settings);
if (HasError(err)) { return; }
err = cam.capture(ref frm);
if (HasError(err)) { return; }
// Use the first acquisition frame image as captured frame's color
err = new_settings.enableFilterSetting(Settings.FilterSetting.ShowHDR, false);
if (HasError(err)) { return; }
err = cam.setSettings(new_settings);
if (HasError(err)) { return; }
err = cam.capture(ref frm);
if (HasError(err)) { return; }
// Check if local GPU is available
Settings temp_settings = cam.getSettings();
bool is_available = false;
err = temp_settings.inquireSystemSetting(Settings.SystemSetting.GPUAvailable, ref is_
↪available);
if (HasError(err)) { return; }
// Enable computation using local GPU (for BP-AMR and USB interface 3D cameras only)
if (is_available)
{
   err = cam.enableGPU(true);
   if (HasError(err)) { return; }
   err = cam.capture(ref frm);
   if (HasError(err)) { return; }
}
// Disable computation using local GPU, use CPU instead (for BP-AMR and USB interface 3D␣
↪cameras only)
if (is_available)
{
   err = cam.enableGPU(false);
   if (HasError(err)) { return; }
   err = cam.capture(ref frm);
   if (HasError(err)) { return; }
```

(continues on next page)

(continued from previous page)

```
}
// Enable temperature regulation
err = cam.enableTempRegulation(true);
if (HasError(err)) { return; }
// Disable temperature regulation
err = cam.enableTempRegulation(false);
if (HasError(err)) { return; }
```

Python

```
# Camera Captures␣
↪============================================================================
# Captures are returned as a DaoAI Frame object.
# Capture with default settings (assuming no settings has been set to camera).
frame, err = cam.capture()
if (hasError(err)): return

# Capture with custom settings
# OPTION 1: Capture with settings. Settings saved by camera for future captures.
frame, err = cam.capture(new_settings)
if (hasError(err)): return
# OPTION 2: Set settings object to camera to use in capture.
err = cam.setSettings(new_settings)
if (hasError(err)): return
frame, err = cam.capture()
if (hasError(err)): return
# OPTION 3: Load settings from file to camera to use in capture.
err = cam.setSettings("../../Examples/sample_settings.cfg")
if (hasError(err)): return
frame, err = cam.capture()
if (hasError(err)): return

# Use HDR image as captured frame's color
err = new_settings.enableFilterSetting(ShowHDR, True)
if (hasError(err)): return
err = cam.setSettings(new_settings)
if (hasError(err)): return
frame, err = cam.capture()
if (hasError(err)): return
# Use the first acquisition frame image as captured frame's color
err = new_settings.enableFilterSetting(ShowHDR, False)
if (hasError(err)): return
err = cam.setSettings(new_settings)
if (hasError(err)): return
frame, err = cam.capture()
if (hasError(err)): return
# Check if local GPU is available
temp_settings = cam.getSettings()
is_available, err = temp_settings.inquireSystemSetting(GPUAvailable)
if (hasError(err)): return
# Enable computation using local GPU (for BP-AMR and USB interface 3D cameras only)
if (is_available):
```

(continues on next page)

(continued from previous page)

```
    err = cam.enableGPU(True)
    if (hasError(err)): return
    frame, err = cam.capture()
    if (hasError(err)): return
# Disable computation using local GPU, use CPU instead (for BP-AMR and USB interface 3D␣
↪cameras only)
if (is_available):
    err = cam.enableGPU(False)
    if (hasError(err)): return
    frame, err = cam.capture()
    if (hasError(err)): return
# Enable temperature regulation
err = cam.enableTempRegulation(True)
if (hasError(err)): return
# Disable temperature regulation
err = cam.enableTempRegulation(False)
if (hasError(err)): return
```

6.8 Frames

Save and load image.

C++

```
// Frames␣
↪=============================================================================================
DaoAI::Frame new_frame;
// Create new empty frame
new_frame = DaoAI::Frame();
// Copy constructor
new_frame = DaoAI::Frame(frm);

// Check if frame has data
if (!new_frame.isEmpty()) { std::cout << "Success: Frame contains data from 3D capture!"
↪<< std::endl; }

// Save a frame. File extension .dcf is the preferred DaoAI frame format, but saving␣
↪also supports .pcd and .ply formats.
std::string save_frame_path = "../../Examples/example_frame_save.dcf";
ret = new_frame.save(save_frame_path);
if (hasError(ret)) { return -1; } // Check for errors

// Load a frame from file. Supports .dcf files.
ret = new_frame.load("../../Examples/sample_frame.dcf");
if (hasError(ret)) { return -1; } // Check for errors

// Get point cloud data.
DaoAI::PointCloud pcl;
ret = frm.getPointCloud(pcl);
if (hasError(ret)) { return -1; } // Check for errors
```

C#

```
// Frames␣
↪==========================================================================================================
Frame new_frame;
// Create new empty frame
new_frame = new Frame();
// Copy constructor
new_frame = new Frame(frm);

// Check if frame has data
if (!new_frame.isEmpty()) { Console.WriteLine("Success: Frame contains data from 3D␣
↪capture!"); }

// Save a frame. File extension .dcf is the preferred DaoAI frame format, but saving␣
↪also supports .pcd and .ply formats.
string save_frame_path = "../../../../../Examples/example_frame_save.dcf";
err = new_frame.save(save_frame_path);
if (HasError(err)) { return; } // Check for errors

// Load a frame from file. Supports .dcf files.
err = new_frame.load("../../../../../Examples/sample_frame.dcf");
if (HasError(err)) { return; } // Check for errors

// Get point cloud data.
PointCloud pcl = new PointCloud();
err = frm.getPointCloud(ref pcl);
if (HasError(err)) { return; } // Check for errors
```

Python

```
# Frames␣
↪==========================================================================
# Create new empty frame
new_frame = Frame()
# Copy constructor
new_frame = Frame(frame)

# Check if frame has data
if (not new_frame.isEmpty()) : print("Success: Frame contains data from 3D capture!")

# Save a frame. File extension .dcf is the preferred DaoAI frame format, but saving also␣
↪supports .pcd and .ply formats.
save_frame_path = "../../Examples/example_frame_save.dcf"
err = new_frame.save(save_frame_path)
if (hasError(err)): return

# Load a frame from file. Supports .dcf files.
err = new_frame.load("../../Examples/sample_frame.dcf")
if (hasError(err)): return

# Get point cloud data from frame.
pcl, err = frame.getPointCloud()
if (hasError(err)): return
```

6.9 Point Cloud

Create, get and read Point Cloud data.

C++

```
// Point Cloud␣
↪==========================================================================================
// Point cloud contains the coordinate and color information from the 3D Capture Frame.
DaoAI::PointCloud new_pcl;
// Create new point cloud.
new_pcl = DaoAI::PointCloud(); // Empty point cloud.
new_pcl = DaoAI::PointCloud(100, 100); // Specify dimensions of created point cloud.
new_pcl = DaoAI::PointCloud(pcl); // Copy point cloud.
// Clone a point cloud.
new_pcl = pcl.clone();
// Get point cloud structure information.
int size = new_pcl.getSize();
int height = new_pcl.getHeight(); // Number of rows.
int width = new_pcl.getWidth(); // Number of columns.
if (!new_pcl.isEmpty()) { std::cout << "Point cloud contains capture data!" << std::endl;
↪ }
// Get point cloud data information.
std::vector<float> x_values = new_pcl.getVecX(); // 2D vector of all the x-coordinates␣
↪in the point cloud.
std::vector<float> y_values = new_pcl.getVecX(); // 2D vector of all the y-coordinates␣
↪in the point cloud.
std::vector<float> z_values = new_pcl.getVecX(); // 2D vector of all the z-coordinates␣
↪in the point cloud.
std::vector<float> confident_values = new_pcl.getVecConfident(); // 2D vector of point␣
↪cloud confidence values.
std::vector<uint32_t> rgba_values = new_pcl.getVecRgba(); // 2D vector of all the RGBA␣
↪values in the point cloud. 0xAARRGGBB format.
std::vector<uint8_t> r_values = new_pcl.getVecR(); // 2D vector of all the r-values in␣
↪the point cloud.
std::vector<uint8_t> g_values = new_pcl.getVecG(); // 2D vector of all the g-values in␣
↪the point cloud.
std::vector<uint8_t> b_values = new_pcl.getVecB(); // 2D vector of all the b-values in␣
↪the point cloud.
std::vector<uint8_t> a_values = new_pcl.getVecA(); // 2D vector of all the a-values in␣
↪the point cloud.
// Get individual point from point cloud.
DaoAI::Point pt;
int idx = rand() % size;
pt = new_pcl(idx); // Get any point using a 1D index between [0, size).
int row = rand() % height; int col = rand() % width;
pt = new_pcl(row, col); // Get any point using a 2D index pair (row, column).
// Get pointer to first point in the point cloud.
DaoAI::Point* first_pt = new_pcl.getDataPtr();
```

C#

```
// Point Cloud␣
```

(continues on next page)

(continued from previous page)

```
↪=============================================================================================
// Point cloud contains the coordinate and color information from the 3D Capture Frame.
PointCloud new_pcl;
// Create new point cloud.
new_pcl = new PointCloud(); // Empty point cloud.
new_pcl = new PointCloud(100, 100); // Specify dimensions of created point cloud.

// Clone a point cloud.
new_pcl = pcl.clone();

// Get point cloud structure information.
int size = (int) new_pcl.getSize();
int height = (int) new_pcl.getHeight(); // Number of rows.
int width = (int) new_pcl.getWidth(); // Number of columns.
if (!new_pcl.isEmpty()) { Console.WriteLine("Point cloud contains capture data!"); }
// Get point cloud data information.
List<float> x_values = new_pcl.getVecX(); // 2D vector of all the x-coordinates in the␣
↪point cloud.
List<float> y_values = new_pcl.getVecX(); // 2D vector of all the y-coordinates in the␣
↪point cloud.
List<float> z_values = new_pcl.getVecX(); // 2D vector of all the z-coordinates in the␣
↪point cloud.
List<float> confident_values = new_pcl.getVecConfident(); // 2D vector of point cloud␣
↪confidence values.
List<uint> rgba_values = new_pcl.getVecRgba(); // 2D vector of all the RGBA values in␣
↪the point cloud. 0xAARRGGBB format.
List<byte> r_values = new_pcl.getVecR(); // 2D vector of all the r-values in the point␣
↪cloud.
List<byte> g_values = new_pcl.getVecG(); // 2D vector of all the g-values in the point␣
↪cloud.
List<byte> b_values = new_pcl.getVecB(); // 2D vector of all the b-values in the point␣
↪cloud.
List<byte> a_values = new_pcl.getVecA(); // 2D vector of all the a-values in the point␣
↪cloud.
                                                    // Get individual point from point␣
↪cloud.
Random rnd = new Random();
int idx = rnd.Next(0, size);

Point pt;
pt = new_pcl.getPoint((uint) idx); // Get any point using a 1D index between [0, size).
int row = rnd.Next(0, height); int col = rnd.Next(0, width);
pt = new_pcl.getPoint((uint) row, (uint) col); // Get any point using a 2D index pair␣
↪(row, column).
```

Python

```
# Point Cloud␣
↪=============================================================================
# Point cloud contains the coordinate and color information from the 3D Capture Frame.
# Create new point cloud.
new_pcl = PointCloud() # Empty point cloud.
```

(continues on next page)

(continued from previous page)

```
# Create point cloud with specific dimensions
new_pcl = PointCloud(100, 100) # Specify dimensions of created point cloud.

# Clone a point cloud.
new_pcl = pcl.clone()

# Get point cloud structure information.
size = new_pcl.getSize()
height = new_pcl.getHeight() # Number of rows.
width = new_pcl.getWidth() # Number of columns.
if (not new_pcl.isEmpty()): print("Point cloud contains capture data!")
# Get point cloud data information.
x_values = new_pcl.getVecX() # 2D vector of all the x-coordinates in the point cloud.
y_values = new_pcl.getVecX() # 2D vector of all the y-coordinates in the point cloud.
z_values = new_pcl.getVecX() # 2D vector of all the z-coordinates in the point cloud.
confident_values = new_pcl.getVecConfident() # 2D vector of point cloud confidence␣
↪values.
rgba_values = new_pcl.getVecRgba() # 2D vector of all the RGBA values in the point cloud.
↪ 0xAARRGGBB format.
r_values = new_pcl.getVecR() # 2D vector of all the r-values in the point cloud.
g_values = new_pcl.getVecG() # 2D vector of all the g-values in the point cloud.
b_values = new_pcl.getVecB() # 2D vector of all the b-values in the point cloud.
a_values = new_pcl.getVecA() # 2D vector of all the a-values in the point cloud.

# Get individual point from point cloud.
idx = random.randint(0, size)
pt = new_pcl(idx) # Get any point using a 1D index between [0, size).
row = random.randint(0, height)
col = random.randint(0, width)
pt = new_pcl(row, col) # Get any point using a 2D index pair (row, column).
```

6.10 Point

Get and read Point data.

C++

```
// Point␣
↪=============================================================================================
// Point contains the coordinate and color information of an individual point.
// Get point data.
float x = pt.getX();
float y = pt.getY();
float z = pt.getZ();
float confident = pt.getConfident();
uint8_t r = pt.getR();
uint8_t g = pt.getG();
uint8_t b = pt.getB();
uint8_t a = pt.getA();
uint32_t rgba = pt.getRgba(); // 0xAARRGGBB format (ARGB)
```

(continues on next page)

(continued from previous page)

```
// Set point data.
DaoAI::Point new_point;
new_point.setX(1);
new_point.setY(2);
new_point.setZ(3);
new_point.setConfident(0.4);
new_point.setRgba(0x00FF0000); // Set to red.
new_point.setRgb(0x00, 0xFF, 0x00); // Set to green.
new_point.setRgba(0x00, 0x00, 0xFF, 0x00); // Set to blue.
```

C#

```
// Point␣
↪==========================================================================================
// Point contains the coordinate and color information of an individual point.
// Get point data.
float x = pt.getX();
float y = pt.getY();
float z = pt.getZ();
float confident = pt.getConfident();
byte r = pt.getR();
byte g = pt.getG();
byte b = pt.getB();
byte a = pt.getA();
uint rgba = pt.getRgba(); // 0xAARRGGBB format (ARGB)
                          // Set point data.
Point new_point = new Point();
new_point.setX(1);
new_point.setY(2);
new_point.setZ(3);
new_point.setConfident(0.4f);
new_point.setRgba(0x00FF0000); // Set to red.
new_point.setRgb(0x00, 0xFF, 0x00); // Set to green.
new_point.setRgba(0x00, 0x00, 0xFF, 0x00); // Set to blue.
```

Python

```
# Point␣
↪=============================================================================
# Point contains the coordinate and color information of an individual point.
# Get point data.
x = pt.getX()
y = pt.getY()
z = pt.getZ()
confident = pt.getConfident()
r = pt.getR()
g = pt.getG()
b = pt.getB()
a = pt.getA()
rgba = pt.getRgba() # rgba is in 0xAARRGGBB format (ARGB)
# Set point data.
new_point = Point()
```

(continues on next page)

(continued from previous page)

```
err = new_point.setX(1)
err = new_point.setY(2)
err = new_point.setZ(3)
err = new_point.setConfident(0.4)
err = new_point.setRgba(0x00FF0000)  # Set to red.
err = new_point.setRgb(0x00, 0xFF, 0x00) # Set to green.
err = new_point.setRgba(0x00, 0x00, 0xFF, 0x00) # Set to blue.
```

6.11 Clean Up

C++

```
// Clean Up␣
↪==========================================================================================================
ret = cam->disConnect();
if (hasError(ret)) { return -1; } // Check for errors
delete cam;

ret = app->stopLogging();
if (hasError(ret)) { return -1; } // Check for errors

std::cout << "End of sample program!" << std::endl;
return 1;
```

C#

```
// Clean Up␣
↪==========================================================================================================
err = cam.disConnect();
if (HasError(err)) { return; } // Check for errors

err = app.stopLogging();
if (HasError(err)) { return; } // Check for errors

Console.WriteLine("End of sample program!");

System.Threading.Thread.Sleep(20000);
```

Python

```
# Clean Up␣
↪=============================================================================
err = cam.disConnect()
if (hasError(err)): return

err = app.stopLogging()
if (hasError(err)): return

print("End of sample program!")
```

CHAPTER

SEVEN

API REFERENCE

DaoAI Camera Studio API reference.

7.1 Namespace

C++

```
namespace DaoAI
```

C#

```
namespace DaoAI_NET
```

7.2 Classes

7.2.1 *Class* Application

C++

```
#include "application.h"
```

C#

```
#include "application.h"
```

Public Member Functions

Constructor:

C++

```
DAOAI_API Application();
```

C#

```
Application();
```

Destructor:

C++

```
DAOAI_API ~Application();
```

C#

```
~Application();
```

getCameras:

Get a list of all USB cameras

C++

Parameters:

- [out] cameras: A map of all connected DaoAI-supported cameras keyed by serial number.
- [in] [OPTIONAL] remote_address: A map of all connected DaoAI-supported cameras keyed by serial number.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError getCameras(std::map<std::string, Camera*>& cameras,␣
↪std::string remote_address = "");
```

C#

Parameters:

- [out] cameras: A map of all connected DaoAI-supported cameras keyed by serial number.
- [in] [OPTIONAL] remote_address: A map of all connected DaoAI-supported cameras keyed by serial number.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ getCameras(Dictionary<System::String^, Camera^>^% cameras,␣
↪System::String^ remote_address);

DaoAINETError^ getCameras(Dictionary<System::String^, Camera^>^% cameras);
```

connectCamera:

Connect to the next available DaoAI camera.

C++

Parameters:

- [in] camera: Camera to connect.
- [in] [OPTIONAL] settings: Connect to the camera with this settings. Must contain at least one frame.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError connectCamera(Camera*& camera, const Settings&␣
↪settings = {});
```

C#

Parameters:

- [in] camera: Camera to connect.
- [in] [OPTIONAL] settings: Connect to the camera with this settings. Must contain at least one frame.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ connectCamera(Camera^% camera, Settings^ settings);

DaoAINETError^ connectCamera(Camera^% camera);
```

connectCamera:
Connect to the DaoAI camera with serial number.

C++

Parameters:

- [in] serial_number: Connect to the camera with this serial number.
- [out] camera: Pointer to the connected camera.
- [in] [OPTIONAL] settings: Connect to the camera with this settings. Must contain at least one frame.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError connectCamera(const std::string serial_number,␣
↪Camera*& camera, const Settings &settings = {});
```

C#

Parameters:

- [in] serial_number: Connect to the camera with this serial number.
- [out] camera: Pointer to the connected camera.
- [in] [OPTIONAL] settings: Connect to the camera with this settings. Must contain at least one frame.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ connectCamera(System::String^ serial_number, Camera^%␣
↪camera, Settings^ settings);

DaoAINETError^ connectCamera(System::String^ serial_number, Camera^%␣
↪camera);
```

disconnectCamera:
Disconnect the DaoAI camera with serial number.

C++

Parameters:

- [in] serial_number: Serial number of camera to disconnect.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError disconnectCamera(const std::string serial_number);
```

C#

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ disConnect();
```

startLogging:
Enable writing camera log to a file

C++

Parameters:

- [in] [OPTIONAL] log_path: Specify directory to write logs

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError startLogging(std::string log_path = "");
```

C#

Parameters:

- [in] [OPTIONAL] log_path: Specify directory to write logs

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ startLogging(System::String^ log_path);

DaoAINETError^ startLogging();
```

stopLogging:
Disable writing camera log to a file

C++

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError stopLogging();
```

C#

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ stopLogging();
```

7.2.2 *Class* Version

C++

```
#include "application.h"
namespace Version
```

C#

```
#include "application.h"
```

Public Member Functions

getSDKVersion:
get the DaoAI SDK version.

C++

Return:

- string: String containing DaoAI SDK version.

```
DAOAI_API std::string getSDKVersion();
```

C#

Return:

- System::String: String containing DaoAI SDK version.

```
System::String^ getSDKVersion();
```

7.2.3 *Class* Camera

C++

```
#include "camera.h"
```

C#

```
#include "camera.h"
```

Public Member Functions

Constructor:

C++

```
DAOAI_API Camera();
```

C#

```
Camera();
```

Copy Constructor:

C++

```
DAOAI_API explicit Camera(const std::shared_ptr<Camera>& other);
```

C#

```
Camera(Camera^ other);
```

Move Constructor:

C++

```
DAOAI_API explicit Camera(class CameraImpl &&other);
```

Destructor:

C++

```
DAOAI_API ~Camera();
```

C#

```
~Camera();
```

connect:

Connect the camera.

C++

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError connect();
```

C#

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ connect();
```

disConnect:

Disconnect the camera.

C++

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError disConnect();
```

C#

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ disconnectCamera(System::String^ serial_number);
```

isConnected:

Check if the camera is connected.

C++

Return:

- bool: returns True if camera is connected, false otherwise.

```
DAOAI_API bool isConnected() const;
```

C#

Return:

- bool: returns True if camera is connected, false otherwise.

```
System::Boolean isConnected();
```

capture:

Capture a single frame.

C++

Parameters:

- [out] Frame: Capture results will be written to this DaoAI Frame object.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError capture(Frame& frame);
```

C#

Parameters:

- [out] Frame: Capture results will be written to this DaoAI Frame object.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ capture(Frame^% frame);
```

capture:

Capture a single frame with settings.

C++

Parameters:

- [in] settings: DaoAI Settings to use for the capture. Must contain at least one frame.
- [out] Frame: Capture results will be written to this DaoAI Frame object.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError capture(Settings settings, Frame& frame);
```

C#

Parameters:

- [in] settings: DaoAI Settings to use for the capture. Must contain at least one frame.
- [out] Frame: Capture results will be written to this DaoAI Frame object.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ capture(Settings^ settings, Frame^% frame);
```

captureAssistant:

Analyze scene and generate acquisition frame settings, the total time for all acquisition frames will be less than the time budget. The higher time budget is, the more acquisition frames will be generated.

C++

Parameters:

- [in] time_budget: Time budget for acquisition frames, in range of (0.0, 5.0]
- [in,out] mofaf: A map of AcquisitionFrame settings

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError captureAssistant(double time_budget, std::map<int,␣
↪AcquisitionFrame> &mofaf);
```

C#

Parameters:

- [in] time_budget: Time budget for acquisition frames, in range of (0.0, 5.0]
- [in,out] mofaf: A map of AcquisitionFrame settings

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ captureAssistant(System::Double time_budget,␣
↪System::Collections::Generic::Dictionary<System::Int32, AcquisitionFrame^
↪>^% mofaf);
```

setSettings:

Set Settings for camera with file path.

C++

Parameters:

- [in] file_path: Path to load settings file.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError setSettings(std::string file_path);
```

C#

Parameters:

- [in] file_path: Path to load settings file.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ setSettings(System::String^ file_path);
```

setSettings:

Set Settings for camera with settings object.

C++

Parameters:

- [in] settings: Settings object for this camera.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError setSettings(Settings settings);
```

C#

Parameters:

- [in] settings: Settings object for this camera.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ setSettings(Settings^ settings);
```

getSettings:

Get Settings for camera.

C++

Return:

- Settings: Current settings structure used by this camera.

```
DAOAI_API Settings getSettings() const;
```

C#

Return:

- Settings: Current settings structure used by this camera.

```
Settings^ getSettings();
```

getSerialNumber:
Get serialNumber of the camera.

C++

Return:

- std::string: Serial number of this camera.

```
DAOAI_API std::string getSerialNumber() const;
```

C#

Return:

- System::String: Serial number of this camera.

```
System::String^ getSerialNumber();
```

getIntrinsicParam:
brief Get IntrinsicParameter of the camera.

C++

Parameters:

- [out] params: A vector of float containing camera intrinsic parameters.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError getIntrinsicParam(std::vector<float>& params) const;
```

C#

Parameters:

- [out] params: A vector of float containing camera intrinsic parameters.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ getIntrinsicParam(array<System::Single>^% params);
```

enableGPU:
Enable or disable using GPU on local PC for computation.

C++

Parameters:

- toggle[in]: Enable or disable.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError enableGPU(bool toggle);
```

C#

Parameters:

- toggle[in]: Enable or disable.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ enableGPU(System::Boolean toggle);
```

enableTempRegulation:

Enable or disable temperature regulation.

C++

Parameters:

- toggle[in]: Enable or disable.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError enableTempRegulation(bool toggle);
```

C#

Parameters:

- toggle[in]: Enable or disable.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ enableTempRegulation(System::Boolean toggle);
```

7.2.4 *Class* Settings

C++

```
#include "settings.h"
```

C#

```
#include "settings.h"
```

AcquisitionFrame Class

Public Members & Functions

AcquisitionFrameSetting:
The acquisition frame settings data structure.

C++

```
enum AcquisitionFrameSetting {
        Brightness, // Int {0, 3}.
        Gain, // Double {0, 3}.
        ExposureStop // Int {-1, 4}.
    };
```

C#

```
enum AcquisitionFrameSetting {
        Brightness, // Int {0, 3}.
        Gain, // Double {0, 3}.
        ExposureStop // Int {-1, 4}.
    };
```

Constructor:
Constructor with initial inputs for acquisition frame setting.

C++

Parameters:

- brightness[in]: brightness for the setting.
- gain[in]: gain for the setting.
- exposure_stop[in]: exposure stop for the setting.

```
DAOAI_API AcquisitionFrame(int brightness, double gain, int exposure_stop);
```

C#

Parameters:

- brightness[in]: brightness for the setting.
- gain[in]: gain for the setting.
- exposure_stop[in]: exposure stop for the setting.

```
AcquisitionFrame(System::Int32 brightness, System::Double gain,␣
↪System::Int32 exposure_stop);
```

Constructor:

Constructor with default values for acquisition frame setting.

C++

```
DAOAI_API AcquisitionFrame();
```

C#

```
AcquisitionFrame();
```

inquireSetting:

Get the current acquisition setting value and range.

C++

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- curr[out]: the current value for the field.
- min[out]: the min value for the field.
- max[out]: the max value for the field.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireSetting(AcquisitionFrameSetting setting, int&␣
↪curr, int& min, int& max);
```

C#

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- curr[out]: the current value for the field.
- min[out]: the min value for the field.
- max[out]: the max value for the field.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireSetting(AcquisitionFrameSetting setting,␣
↪System::Int32% curr, System::Int32% min, System::Int32% max);
```

inquireSetting:

Get the current acquisition setting value.

C++

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- curr[out]: current value for the field.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireSetting(AcquisitionFrameSetting setting, int&␣
↪curr);
```

C#

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- curr[out]: current value for the field.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireSetting(AcquisitionFrameSetting setting,␣
↪System::Int32% curr);
```

inquireSetting:

Get the current acquisition setting value and range in double (gain).

C++

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- curr[out]: the current value for the field, type double (for field gain).
- min[out]: the min value for the field.
- max[out]: the max value for the field.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireSetting(AcquisitionFrameSetting setting,␣
↪double& curr, double& min, double& max);
```

C#

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- curr[out]: the current value for the field, type double (for field gain).
- min[out]: the min value for the field.
- max[out]: the max value for the field.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireSetting(AcquisitionFrameSetting setting,␣
↪System::Double% curr, System::Double% min, System::Double% max);
```

inquireSetting:
Get the current acquisition setting value in double (gain).

C++

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- curr[out]: the current value for the field, type double (for field gain).

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireSetting(AcquisitionFrameSetting setting,␣
↪double& curr);
```

C#

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- curr[out]: the current value for the field, type double (for field gain).

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireSetting(AcquisitionFrameSetting setting,␣
↪System::Double% curr);
```

configureSetting:
Set the acquisition setting value.

C++

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- new_val[in]: the new value to assign to.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError configureSetting(AcquisitionFrameSetting setting,␣
↪int new_val);
```

C#

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- new_val[in]: the new value to assign to.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ configureSetting(AcquisitionFrameSetting setting,␣
↪System::Int32 new_val);
```

configureSetting:
Set the acquisition setting value in double (gain).

C++

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- new_val[in]: the new value (double for gain) to assign to.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError configureSetting(AcquisitionFrameSetting setting,␣
↪double new_val);
```

C#

Parameters:

- AcquisitionFrameSetting[in]: acquisition frame setting to inquire [brightness, gain, exposure_stop].
- new_val[in]: the new value (double for gain) to assign to.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ configureSetting(AcquisitionFrameSetting setting,␣
↪System::Double new_val);
```

Settings Class

Public Members & Functions

FilterSetting:
The filter settings data structure.

C++

```
enum FilterSetting {
   IntensityThreshold, // Double {0, 3}. Enable/Disable. Get/Set.
   OutlierThreshold, // Double {0, inf}. Enable/Disable. Get/Set.
   SaturationFilter, // Bool. Enable/Disable. Get/Set.
   ContrastDistortionMode, // Int {0, 2}.         0: Off, 1: Remove␣
↪Distortion, 2: Correct Distortion. Get/Set.
   ContrastDistortionStrength, // Int {0, 15000}. Get/Set.
   GaussianFilter, // Int {0, 5}. Enable/Disable. Get/Set.
   MedianFilter, // Int {0, 1}. Enable/Disable. Get/Set.
   FaceNormalFilter, // Double {0, 40}. Enable/Disable. Get/Set.
   SmoothFilter, // Int {0, 6}. Enable/Disable. Get/Set.
   FillGaps, // Bool. Enable/Disable. Get/Set.
   WidthThreshold, // Double {0, 500}. Get/Set.
   SlopeThreshold, // Double {0, inf}. Get/Set.
   DepthThreshold, // Double {0, 500}. Get/Set.
   FillXFirst, // Bool. Get/Set.
   FillBidirectional, // Bool. Get/Set.
   PhaseQualityThreshold, // Double {0, 50}. Enable/Disable. Get/Set.
   ConnectedAreaFilter, // Double {0, 10}. Enable/Disable. Get/Set.
   ShowHDR // Bool. Enable/Disable. Get/Set.
};
```

C#

```
enum class FilterSetting {
         IntensityThreshold, // Double {0, 3}. Enable/Disable. Get/Set.
         OutlierThreshold, // Double {0, inf}. Enable/Disable. Get/Set.
         SaturationFilter, // Bool. Enable/Disable. Get/Set.
         ContrastDistortionMode, // Int {0, 2}.    0: Off, 1: Remove␣
↪Distortion, 2: Correct Distortion. Get/Set.
         ContrastDistortionStrength, // Int {0, 15000}. Get/Set.
         GaussianFilter, // Int {0, 5}. Enable/Disable. Get/Set.
         MedianFilter, // Int {0, 1}. Enable/Disable. Get/Set.
         FaceNormalFilter, // Double {0, 40}. Enable/Disable. Get/Set.
         SmoothFilter, // Int {0, 6}. Enable/Disable. Get/Set.
         FillGaps, // Bool. Enable/Disable. Get/Set.
         WidthThreshold, // Double {0, 500}. Get/Set.
         SlopeThreshold, // Double {0, inf}. Get/Set.
         DepthThreshold, // Double {0, 500}. Get/Set.
         FillXFirst, // Bool. Get/Set.
         FillBidirectional, // Bool. Get/Set.
         PhaseQualityThreshold, // Double {0, 50}. Enable/Disable. Get/Set.
         ConnectedAreaFilter, // Double {0, 10}. Enable/Disable. Get/Set.
         ShowHDR // Bool. Enable/Disable. Get/Set.
      };
```

SystemSetting:

The system settings data structure.

C++

```
enum SystemSetting {
   CameraModel, // String. Get only.
   TemperatureSensorAvailable, // Bool. Get only.
```

(continues on next page)

(continued from previous page)

```
    TemperatureRegulationEnable, // Bool. Get only.
    GPUAvailable, // Bool. Get only.
    GPUEnable, // Bool. Get only.
    Version, // String. Get only.
    ExtraWhitePatternEnable // Bool. Enable/Disable. Get/Set.
};
```

C#

```
enum class SystemSetting {
    CameraModel, // String. Get only.
    TemperatureSensorAvailable, // Bool. Get only.
    TemperatureRegulationEnable, // Bool. Get only.
    GPUAvailable, // Bool. Get only.
    GPUEnable, // Bool. Get only.
    Version, // String. Get only.
    ExtraWhitePatternEnable // Bool. Enable/Disable. Get/Set.
};
```

Constructor:

C++

```
DAOAI_API Settings();
```

C#

```
Settings();
```

Copy Constructor:

C++

Parameters:

- other[out]: another setting object to copy to.

```
DAOAI_API explicit Settings(const std::shared_ptr<Settings>& other);
```

C#

Parameters:

- other[out]: another setting object to copy to.

```
Settings(Settings^ other);
```

Destructor:

C++

```
DAOAI_API ~Settings();
```

C#

```
~Settings();
```

Constructor:

Constructor load settings from settings workspace.

C++

Parameters:

- file_path[in]: The path to the workspace

```
DAOAI_API Settings(const std::string& file_path);
```

C#

Parameters:

- file_path[in]: The path to the workspace

```
Settings(System::String^ file_path);
```

exportSettings:

Export current camera settings.

C++

Parameters:

- file_name[in]: The path to save the camera settings.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError exportSettings(const std::string& file_path);
```

C#

Parameters:

- file_name[in]: The path to save the camera settings.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ exportSettings(System::String^ file_path);
```

addAcquisitionFrame:

Add an aquisition frame to settings to use in 3D capture with index.

C++

Parameters:

- af[in]: AcquisitionFrame object to write.
- index[out]: Index where this frame is written.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError addAcquisitionFrame(AcquisitionFrame af, int& index);
```

C#

Parameters:

- af[in]: AcquisitionFrame object to write.
- index[out]: Index where this frame is written.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ addAcquisitionFrame(AcquisitionFrame^ af, System::Int32%␣
↪index);
```

addAcquisitionFrame:
Add an aquisition frame to settings to use in 3D capture.

C++

Parameters:

- af[in]: AcquisitionFrame object to write.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError addAcquisitionFrame(AcquisitionFrame af);
```

C#

Parameters:

- af[in]: AcquisitionFrame object to write.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ addAcquisitionFrame(AcquisitionFrame^ af);
```

getAcquisitionFrame:
Retreive the aquisition frame object from the given index.

C++

Parameters:

- af[out]: AcquisitionFrame object to which data will be written.
- index[in]: Index to retrieve frame.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError getAcquisitionFrame(AcquisitionFrame& af, int index);
```

C#

Parameters:

- af[out]: AcquisitionFrame object to which data will be written.
- index[in]: Index to retrieve frame.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ getAcquisitionFrame(AcquisitionFrame^% af, System::Int32␣
↪index);
```

modifyAcquisitionFrame:
Modify an aquisition frame to settings at a given index.

C++

Parameters:

- af[in]: AcquisitionFrame object to write.
- index[in]: Index of AcquisitionFrame to modify data.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError modifyAcquisitionFrame(AcquisitionFrame af, int␣
↪index);
```

C#

Parameters:

- af[in]: AcquisitionFrame object to write.
- index[in]: Index of AcquisitionFrame to modify data.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ modifyAcquisitionFrame(AcquisitionFrame^ af, System::Int32␣
↪index);
```

deleteAcquisitionFrame:
Delete an aquisition frame at a given index.

C++

Parameters:

- index[in]: Index of AcquisitionFrame to delete.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError deleteAcquisitionFrame(int index);
```

C#

Parameters:

- index[in]: Index of AcquisitionFrame to delete.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ deleteAcquisitionFrame(System::Int32 index);
```

setAcquisitionFrames:
Write a map of AcquisitionFrames to settings to be used in D3 Capture.

C++

Parameters:

- mofaf[in]: Map of int to AcquisitionFrame objects to write.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError setAcquisitionFrames(std::map<int, AcquisitionFrame>␣
↪mofaf);
```

C#

Parameters:

- mofaf[in]: Map of int to AcquisitionFrame objects to write.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^␣
↪setAcquisitionFrames(System::Collections::Generic::Dictionary
↪<System::Int32, AcquisitionFrame^>^ mofaf);
```

getAcquisitionFrames:
Retrieve the current map of AcquisitionFrames from settings.

C++

Parameters:

- mofaf[out]: Current acquisition frame map is written to this map.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError getAcquisitionFrames(std::map<int, AcquisitionFrame>&
↪ mofaf);
```

C#

Parameters:

- mofaf[out]: Current acquisition frame map is written to this map.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^␣
↪getAcquisitionFrames(System::Collections::Generic::Dictionary
↪<System::Int32, AcquisitionFrame^>^% mofaf);
```

enableFilterSetting:
Enable or disable a filter setting.

C++

Parameters:

- setting[in]: The filter to toggle.
- toggle[in]: Enable or disable.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError enableFilterSetting(FilterSetting setting, bool␣
↪toggle);
```

C#

Parameters:

- setting[in]: The filter to toggle.
- toggle[in]: Enable or disable.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ enableFilterSetting(FilterSetting setting, System::Boolean␣
↪toggle);
```

checkEnableFilterSetting:
Check if a filter setting is enabled.

C++

Parameters:

- setting[in]: The filter to check.
- is_enabled[out]: Enable status written to this.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError checkEnableFilterSetting(FilterSetting setting, bool&
↪ is_enabled);
```

C#

Parameters:

- setting[in]: The filter to check.
- is_enabled[out]: Enable status written to this.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ checkEnableFilterSetting(FilterSetting setting,␣
↪System::Boolean% is_enabled);
```

inquireFilterSetting:

Get the current value and valid range of a filter setting.

C++

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.
- min[out]: The minimum valid value that this setting can be configured with.
- max[out]: The maximum valid value that this setting can be configured with.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireFilterSetting(FilterSetting setting, int&␣
↪curr, int& min, int& max);
```

C#

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.
- min[out]: The minimum valid value that this setting can be configured with.
- max[out]: The maximum valid value that this setting can be configured with.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireFilterSetting(FilterSetting setting, System::Int32%␣
↪curr, System::Int32% min, System::Int32% max);
```

inquireFilterSetting:

Get the current value and valid range (double) of a filter setting.

C++

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.
- min[out]: The minimum valid value that this setting can be configured with.
- max[out]: The maximum valid value that this setting can be configured with.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireFilterSetting(FilterSetting setting, double&␣
↪curr, double& min, double& max);
```

C#

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.
- min[out]: The minimum valid value that this setting can be configured with.
- max[out]: The maximum valid value that this setting can be configured with.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireFilterSetting(FilterSetting setting, System::Double%␣
↪curr, System::Double% min, System::Double% max);
```

inquireFilterSetting:
Get the current value of a filter setting.

C++

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireFilterSetting(FilterSetting setting, int&␣
↪curr);
```

C#

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireFilterSetting(FilterSetting setting, System::Int32%␣
↪curr);
```

inquireFilterSetting:
Get the current value(double) of a filter setting.

C++

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireFilterSetting(FilterSetting setting, double&␣
↪curr);
```

C#

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireFilterSetting(FilterSetting setting, System::Double%␣
↪curr);
```

inquireFilterSetting:
Get the current value(bool) of a filter setting.

C++

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireFilterSetting(FilterSetting setting, bool&␣
↪curr);
```

C#

Parameters:

- setting[in]: The filter to check.
- curr[out]: The current value of this setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireFilterSetting(FilterSetting setting, System::Boolean
↪% curr);
```

configureFilterSetting:
Configure a filter setting with the given value.

C++

Parameters:

- setting[in]: The filter to configure.
- new_val[in]: The value to write to this setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError configureFilterSetting(FilterSetting setting, int␣
↪new_val);
```

C#

Parameters:

- setting[in]: The filter to configure.
- new_val[in]: The value to write to this setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ configureFilterSetting(FilterSetting setting, System::Int32␣
↪new_val);
```

configureFilterSetting:
Configure a filter setting with the given value (double).

C++

Parameters:

- setting[in]: The filter to configure.
- new_val[in]: The value to write to this setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError configureFilterSetting(FilterSetting setting, double␣
↪new_val);
```

C#

Parameters:

- setting[in]: The filter to configure.
- new_val[in]: The value to write to this setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ configureFilterSetting(FilterSetting setting,␣
↪System::Double new_val);
```

configureFilterSetting:
Configure a filter setting with the given value (bool).

C++

Parameters:

- setting[in]: The filter to configure.
- new_val[in]: The value to write to this setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError configureFilterSetting(FilterSetting setting, bool␣
↪new_val);
```

C#

Parameters:

- setting[in]: The filter to configure.
- new_val[in]: The value to write to this setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ configureFilterSetting(FilterSetting setting,␣
↪System::Boolean new_val);
```

configureFilterSetting:
Enable or disable a system setting.

C++

Parameters:

- setting[in]: The system setting to toggle.
- toggle[in]: Enable or disable.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError enableSystemSetting(SystemSetting setting, bool␣
↪toggle);
```

C#

Parameters:

- setting[in]: The system setting to toggle.
- toggle[in]: Enable or disable.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ enableSystemSetting(SystemSetting setting, System::Boolean␣
↪toggle);
```

checkEnableSystemSetting:

Check if a system setting is enabled or disabled.

C++

Parameters:

- setting[in]: The system setting to check.
- is_enabled[out]: Enable status written to this.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError checkEnableSystemSetting(SystemSetting setting, bool&
↪ is_enabled);
```

C#

Parameters:

- setting[in]: The system setting to check.
- is_enabled[out]: Enable status written to this.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ checkEnableSystemSetting(SystemSetting setting,␣
↪System::Boolean% is_enabled);
```

inquireSystemSetting:

Check the current value and valid range of a system setting.

C++

Parameters:

- setting[in]: The system setting to check.
- curr[out]: Current value of system setting.
- min[out]: Minimum configurable value of system setting.
- max[out]: Maximum configurable value of system setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireSystemSetting(SystemSetting setting, int&␣
↪curr, int& min, int& max);
```

C#

Parameters:

- setting[in]: The system setting to check.

- curr[out]: Current value of system setting.
- min[out]: Minimum configurable value of system setting.
- max[out]: Maximum configurable value of system setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireSystemSetting(SystemSetting setting, System::Int32%␣
↪curr, System::Int32% min, System::Int32% max);
```

inquireSystemSetting:
Check the current value of a system setting.

C++

Parameters:

- setting[in]: The system setting to check.
- curr[out]: Current value of system setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireSystemSetting(SystemSetting setting, int&␣
↪curr);
```

C#

Parameters:

- setting[in]: The system setting to check.
- curr[out]: Current value of system setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireSystemSetting(SystemSetting setting, System::Int32%␣
↪curr);
```

inquireSystemSetting:
Check the current value (bool) of a system setting.

C++

Parameters:

- setting[in]: The system setting to check.
- curr[out]: Current value of system setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireSystemSetting(SystemSetting setting, bool&␣
↪curr);
```

C#

Parameters:

- setting[in]: The system setting to check.
- curr[out]: Current value of system setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireSystemSetting(SystemSetting setting, System::Boolean
↪% curr);
```

inquireSystemSetting:
Check the current value (string) of a system setting.

C++

Parameters:

- setting[in]: The system setting to check.
- curr[out]: Current value of system setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError inquireSystemSetting(SystemSetting setting,␣
↪std::string& curr);
```

C#

Parameters:

- setting[in]: The system setting to check.
- curr[out]: Current value of system setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ inquireSystemSetting(SystemSetting setting, System::String^
↪% val);
```

configureSystemSetting:
Configure a system setting with given value.

C++

Parameters:

- setting[in]: The system setting to configure.
- new_val[in]: Value to write to system setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError configureSystemSetting(SystemSetting setting, int␣
↪new_val);
```

C#

Parameters:

- setting[in]: The system setting to configure.
- new_val[in]: Value to write to system setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ configureSystemSetting(SystemSetting setting, System::Int32␣
↪new_val);
```

configureSystemSetting:

Configure a system setting with given value (bool).

C++

Parameters:

- setting[in]: The system setting to configure.
- new_val[in]: Value to write to system setting.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError configureSystemSetting(SystemSetting setting, bool␣
↪new_val);
```

C#

Parameters:

- setting[in]: The system setting to configure.
- new_val[in]: Value to write to system setting.

Return:

- DaoAINETError^: DaoAINETError object handle containing status codes and any error messages.

```
DaoAINETError^ configureSystemSetting(SystemSetting setting,␣
↪System::Boolean new_val);
```

7.2.5 *Class* SlcSdkError

C++

```
#include "error.h"
```

C#

```
#include "error.h"
```

Public Members & Functions

SlcSdkStatus:
The DaoAI SDK status data structure.

C++

```
enum DAOAI_API SlcSdkStatus : int
   {
      SlcSdkSuccess,
      SlcSdkErrorInvalidValue,
      SlcSdkErrorGPUMemoryAllocation,
      SlcSdkErrorVirtualFunctionCalled,
      SlcSdkErrorImageAcquisition,
      SlcSdkErrorFileOperation,
      SlcSdkErrorDeviceConnection,
      SlcSdkErrorDeviceOperation,
      SlcSdkErrorTemperatureRegulation,
      SlcSdkErrorWorkspaceVersion,
      SlcSdkErrorRemoteConnection,
      SlcSdkErrorRemoteVersion
   };
```

C#

```
public enum class DaoAINETStatus : int {
   SlcSdkSuccess,
   SlcSdkErrorInvalidValue,
   SlcSdkErrorGPUMemoryAllocation,
   SlcSdkErrorVirtualFunctionCalled,
   SlcSdkErrorImageAcquisition,
   SlcSdkErrorFileOperation,
   SlcSdkErrorDeviceConnection,
   SlcSdkErrorDeviceOperation,
   SlcSdkErrorTemperatureRegulation,
   SlcSdkErrorWorkspaceVersion,
   SlcSdkErrorRemoteConnection,
   SlcSdkErrorRemoteVersion
};
```

Constructor:
Construct the error class with inital value.

C++

```
DAOAI_API SlcSdkError(SlcSdkStatus status, std::string detail_text);
```

C#

```
DaoAINETError(DaoAINETStatus status, System::String^ detail_text);
```

Constructor:
Construct the error class with default (invalid error) value.

C++

```
DAOAI_API SlcSdkError();
```

C#

```
DaoAINETError();
```

status:
Get the status code of the error.

C++

Return:

- SlcSdkStatus: status code of the error.

```
DAOAI_API SlcSdkStatus status();
```

C#

Return:

- DaoAINETStatus: status code of the error.

```
DaoAINETStatus status();
```

details:
Get the detailed description of the error.

C++

Return:

- std::string: detailed description of the error.

```
DAOAI_API std::string details();
```

C#

Return:

- System::String: detailed description of the error.

```
System::String^ details();
```

7.2.6 *Class* Frame

C++

```
#include "frame.h"
```

C#

```
#include "frame.h"
```

Public Members & Functions

Constructor:
Construct the error class with inital value.

C++

```
DAOAI_API Frame();
```

C#

```
Frame();
```

Copy Constructor:

C++

```
DAOAI_API explicit Frame(const std::shared_ptr<Frame>& other);
```

C#

```
Frame(Frame^ a);
```

getPointCloud:
Get the Point Cloud from the Frame.

C++

Parameters:

- [in] pc: PointCloud object to write contained point cloud data to.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError getPointCloud(PointCloud& pc);
```

C#

Parameters:

- [in] pc: PointCloud object to write contained point cloud data to.

Return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ getPointCloud(PointCloud^% pc);
```

save:
Save the frame.

C++

Parameters:

- file_name[in]: The path to save the frame. Supports .dcf (DaoAI data format), .ply, .pcd and .daf file suffixes.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError save(const std::string &file_name);
```

C#

Parameters:

- file_name[in]: The path to save the frame. Supports .dcf (DaoAI data format), .ply, .pcd and .daf file suffixes.

Return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ save(System::String^ file_name);
```

load:
Save the frame.

C++

Parameters:

- file_name[in]: The path to load the frame.

Return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError load(const std::string &file_name);
```

C#

Parameters:

- file_name[in]: The path to load the frame.

Return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ load(System::String^ file_name);
```

isEmpty:
Check whether the frame is empty.

C++

Return:

- bool: True if frame is empty, false otherwise.

```
DAOAI_API bool isEmpty();
```

C#

Return:

- System::Boolean: True if frame is empty, false otherwise.

```
System::Boolean isEmpty();
```

7.2.7 *Class* Point Cloud

C++

```
#include "point_cloud.h"
```

C#

```
#include "point_cloud.h"
```

Public Members & Functions

Constructor:

C++

```
DAOAI_API PointCloud();
```

C#

```
PointCloud();
```

Copy Constructor:

C++

```
DAOAI_API explicit PointCloud(const std::shared_ptr<PointCloud>& other);
```

C#

```
PointCloud(PointCloud^ other);
```

Destructor:

C++

```
DAOAI_API ~PointCloud();
```

C#

```
~PointCloud();
```

PointCloud:
Allocate an organized point cloud with a given number of rows and columns

C++

Parameters:

- rows[in]: the height of the point cloud.
- cols[in]: the width of the point cloud.

```
DAOAI_API PointCloud(size_t rows, size_t cols);
```

C#

Parameters:

- rows[in]: the height of the point cloud.
- cols[in]: the width of the point cloud.

```
PointCloud(System::UInt64 rows, System::UInt64 cols);
```

isEmpty:
Return whether point cloud is empty

C++

return:

- bool: whether point cloud is empty.

```
DAOAI_API bool isEmpty() const;
```

C#

return:

- System::Boolean: whether point cloud is empty.

```
System::Boolean isEmpty();
```

getWidth:
Return width (number of columns) of point cloud

C++

return:

- int: width (number of columns) of point cloud.

```
DAOAI_API int getWidth() const;
```

C#

return:

- System::UInt64: width (number of columns) of point cloud.

```
System::UInt64 getWidth();
```

getHeight:
Return height (number of rows) of point cloud

C++

return:

- int: height (number of rows) of point cloud.

```
```

DAOAI_API int getHeight() const;

C#

return:

- System::UInt64: height (number of rows) of point cloud.

```
System::UInt64 getHeight();
```

getSize:
Return number of points in point cloud

C++

return:

- int: size (number of points) of point cloud.

```
```

DAOAI_API int getSize() const;

C#

return:

- int: size (number of points) of point cloud.

```
System::UInt64 getSize();
```

getPoint:
Obtain a reference to a point given by a 1D linear index (from 0 to number of points).

C++

Parameters:

- idx[in]: index value

return:

- Point: reference to a point

```
```

DAOAI_API Point &operator()(size_t idx);

C#

Parameters:

- idx[in]: index value

return:

- Point: reference to a point

```
Point^ getPoint(System::UInt64 idx);
```

getPoint:
Obtain a constant reference to a point given by a 1D linear index (from 0 to number of points).

C++

Parameters:

- idx[in]: index value

return:

- Point: reference to a point

```

```

DAOAI_API const Point &operator()(size_t idx) const;

getPoint:
Obtain a reference to a point given by row and column, i is Row & j is Column

C++

Parameters:

- i: row
- j: col

return:

- Point: reference to a point

```
DAOAI_API Point operator()(size_t i, size_t j);
```

C#

Parameters:

- i: row
- j: col

return:

- Point: reference to a point

```
Point^ getPoint(System::UInt64 i, System::UInt64 j);
```

getPoint:
Obtain a constant reference to a point given by row and column, i is Row & j is Column

C++

Parameters:

- i: row
- j: col

return:

- Point: reference to a point

```
DAOAI_API const Point &operator()(size_t i, size_t j) const;
```

resize:
Resize the point cloud to the given number of rows and columns

C++

Parameters:

- rows: new row number
- cols: new col number

return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API SlcSdkError resize(size_t rows, size_t cols);
```

C#

Parameters:

- rows: new row number
- cols: new col number

return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ resize(System::UInt64 rows, System::UInt64 cols);
```

getDataPtr:
Obtain a pointer to the first point in the point cloud

C++

return:

- Point*: a pointer to the first point in the point cloud.

```
DAOAI_API Point* getDataPtr() const;
```

getVecX:
Return a vector of all x coordinate data

C++

return:

- std::vector<float>: a vector of all x coordinate data

```
DAOAI_API std::vector<float> getVecX() const;
```

C#

return:

- List<System::Single>: a vector of all x coordinate data

```
List<System::Single>^ getVecX();
```

getVecY:
Return a vector of all y coordinate data

C++

return:

- std::vector<float>: a vector of all y coordinate data

```
DAOAI_API std::vector<float> getVecY() const;
```

C#

return:

- List<System::Single>: a vector of all y coordinate data

```
List<System::Single>^ getVecY();
```

getVecZ:
Return a vector of all z coordinate data

C++

return:

- std::vector<float>: a vector of all z coordinate data

```
DAOAI_API std::vector<float> getVecZ() const;
```

C#

return:

- List<System::Single>: a vector of all z coordinate data

```
List<System::Single>^ getVecZ();
```

getVecRgba:
Return a vector of all rgba color data

C++

return:

- std::vector<uint32_t>: a vector of all rgba color data.

```
DAOAI_API std::vector<uint32_t> getVecRgba() const;
```

C#

return:

- List<System::UInt32>: a vector of all rgba color data.

```
List<System::UInt32>^ getVecRgba();
```

getVecConfident:
Return a vector of all confident data

C++

return:

- std::vector<float>: a vector of all confident data

```
DAOAI_API std::vector<float> getVecConfident() const;
```

C#

return:

- List<System::Single>: a vector of all confident data;

```
List<System::Single>^ getVecConfident();
```

getVecR:
Return a vector of all red channel data

C++

return:

- std::vector<uint8_t>: a vector of all r data

```
DAOAI_API std::vector<uint8_t> getVecR() const;
```

C#

return:

- List<System::Byte>: a vector of all r data

```
List<System::Byte>^ getVecR();
```

getVecG:
Return a vector of all green channel data

C++

return:

- std::vector<uint8_t>: a vector of all g data

```
DAOAI_API std::vector<uint8_t> getVecG() const;
```

C#

return:

- List<System::Byte>: a vector of all g data

```
List<System::Byte>^ getVecG();
```

getVecB:
Return a vector of all blue channel data

C++

return:

- std::vector<uint8_t>: a vector of all b data

```
DAOAI_API std::vector<uint8_t> getVecB() const;
```

C#

return:

- List<System::Byte>: a vector of all b data

```
List<System::Byte>^ getVecB();
```

getVecA:
Return a vector of all Alpha-channel data

C++

return:

- std::vector<uint8_t>: a vector of all a data

```
DAOAI_API std::vector<uint8_t> getVecA() const;
```

C#

return:

- List<System::Byte>: a vector of all a data

```
List<System::Byte>^ getVecA();
```

clone:
Make a deep copy of the point cloud

C++

return:

- PointCloud: Copied point cloud.

```
DAOAI_API PointCloud clone();
```

C#

return:

- PointCloud: Copied point cloud.

```
PointCloud^ clone();
```

7.2.8 *Class* Point

C++

```
#include "point.h"
```

C#

```
#include "point.h"
```

Public Member Functions

Constructor:

C++

```
DAOAI_API Point();
```

C#

```
DAOAI_API Point();
```

Destructor:

C++

```
DAOAI_API ~Point() {}
```

C#

```
DAOAI_API ~Point();
```

isNaN:
Return bool if the point is NaN

C++

return:

- bool: bool if the point is NaN

```
DAOAI_API bool isNaN();
```

C#

return:

- System::Boolean: bool if the point is NaN

```
System::Boolean isNaN();
```

getX:
Get x value from the point.

C++

return:

- float: x value from the point.

```
DAOAI_API inline float getX() const {
   return this->x_;
}
```

C#

return:

- System::Single: x value from the point.

```
System::Single getX();
```

getY:
Get y value from the point.

C++

return:

- float: y value from the point.

```
DAOAI_API inline float getY() const {
   return this->y_;
}
```

C#

return:

- System::Single: y value from the point.

```
System::Single getY();
```

getZ:
Get z value from the point.

C++

return:

- float: z value from the point.

```
DAOAI_API inline float getZ() const {
   return this->z_;
}
```

C#

return:

- System::Single: z value from the point.

```
System::Single getZ();
```

getR:
Get r value from the point.

C++

return:

- uint8_t: r value from the point.

```
DAOAI_API inline uint8_t getR() const {
   return ((rgba_ >> 16) & 0xff);
}
```

C#

return:

- System::Byte: r value from the point.

```
System::Byte getR();
```

getG:
Get g value from the point.

C++

return:

- uint8_t: g value from the point.

```
DAOAI_API inline uint8_t getG() const {
   return ((rgba_ >> 8) & 0xff);
}
```

C#

return:

- System::Byte: g value from the point.

```
System::Byte getG();
```

getB:
Get b value from the point.

C++

return:

- uint8_t: b value from the point.

```
DAOAI_API inline uint8_t getB() const {
   return ((rgba_) & 0xff);
}
```

C#

return:

- System::Byte: b value from the point.

```
System::Byte getB();
```

getA:
Get a value from the point.

C++

return:

- uint8_t: a value from the point.

```
DAOAI_API inline uint8_t getA() const {
   return ((rgba_ >> 24) & 0xff);
}
```

C#

return:

- System::Byte: a value from the point.

```
System::Byte getA();
```

getRgba:
Get rgba value from the point. NOTE: RGBA value is stored in the form 0xAARRGGBB (ARGB format)

C++

return:

- uint32_t: rgba value from the point.

```
DAOAI_API inline uint32_t getRgba() const {
   return this->rgba_;
}
```

C#

return:

- System::UInt32: rgba value from the point.

```
System::UInt32 getRgba();
```

getConfident:
Get confident value from the point.

C++

return:

- float: confident value from the point.

```
DAOAI_API inline float getConfident() const {
   return this->confident_;
}
```

C#

return:

- System::Single: confident value from the point.

```
System::Single getConfident();
```

setX:
Assign x value to the point.

C++

parameters:

- x[in]: value to assign

return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API inline SlcSdkError setX(const float x) {
   x_ = x;
   return SlcSdkError(SlcSdkSuccess, "Successfully modified point data.");
}
```

C#

parameters:

- x[in]: value to assign

return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ setX(System::Single x)
```

setY:
Assign y value to the point.

C++

parameters:

- y[in]: value to assign

return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API inline SlcSdkError setY(const float y) {
   y_ = y;
   return SlcSdkError(SlcSdkSuccess, "Successfully modified point data.");
}
```

C#

parameters:

- y[in]: value to assign

return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ setY(System::Single y)
```

setZ:
Assign z value to the point.

C++

parameters:

- z[in]: value to assign

return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API inline SlcSdkError setZ(const float z) {
   z_ = z;
   return SlcSdkError(SlcSdkSuccess, "Successfully modified point data.");
}
```

C#

parameters:

- z[in]: value to assign

return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ setZ(System::Single z)
```

setRgba:
Assign rgba value to the point.

C++

parameters:

- r[in]: red value to assign
- g[in]: green value to assign
- b[in]: blue value to assign
- a[in]: alpha value to assign

return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API inline SlcSdkError setRgba(const uint8_t r, const uint8_t g, const␣
↪uint8_t b, const uint8_t a)
{
   rgba_ = static_cast<uint32_t>((a << 24) | (r << 16) | (g << 8) | b);
   return SlcSdkError(SlcSdkSuccess, "Successfully modified point data.");
}
```

C#

parameters:

- r[in]: red value to assign
- g[in]: green value to assign
- b[in]: blue value to assign
- a[in]: alpha value to assign

return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ setRgba(System::Byte r, System::Byte g, System::Byte b,␣
↪System::Byte a);
```

setRgba:
Assign rgba value to the point.

C++

parameters:

- rgba[in]:value to assign

return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API inline SlcSdkError setRgba(const uint32_t rgba)
{
   rgba_ = rgba;
   return SlcSdkError(SlcSdkSuccess, "Successfully modified point data.");
}
```

C#

parameters:

- rgba[in]:value to assign

return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ (System::UInt32 rgba);
```

setRgb:
Assign rgb value to the point, and alpha channel will be set to 255.

C++

parameters:

- r[in]: red value to assign
- g[in]: green value to assign
- b[in]: blue value to assign

return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API inline SlcSdkError setRgb(const uint8_t r, const uint8_t g, const␣
↪uint8_t b) {
   this->setRgba(r, g, b, 255);
   return SlcSdkError(SlcSdkSuccess, "Successfully modified point data.");
}
```

C#

parameters:

- r[in]: red value to assign
- g[in]: green value to assign
- b[in]: blue value to assign

return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ setRgb(System::Byte r, System::Byte g, System::Byte b);
```

setConfident:

Assign confident value to the point.

C++

parameters:

- confident[in]: confident value to assign

return:

- SlcSdkError: struct containing status codes and any error messages.

```
DAOAI_API inline SlcSdkError setConfident(const float confident) {
   confident_ = confident;
   return SlcSdkError(SlcSdkSuccess, "Successfully modified point data.");
}
```

C#

parameters:

- confident[in]: confident value to assign

return:

- DaoAINETError^: DaoAINETError class handle containing status codes and any error messages.

```
DaoAINETError^ setConfident(System::Single confident);
```

operator =:

C++

parameters:

- point[in]: another Point.

return:

- Point &: return a reference of a Point.

```
DAOAI_API Point &operator=(const Point& point);
```

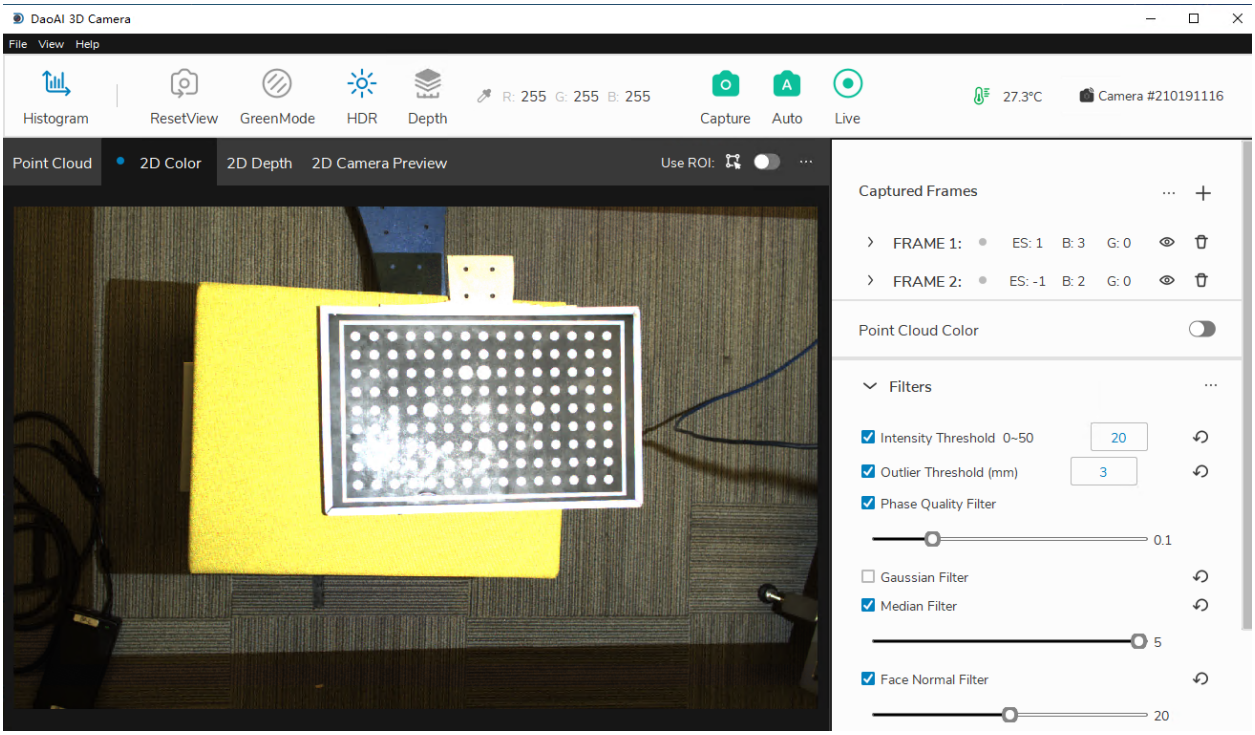
CHAPTER

EIGHT

CASE STUDIES

8.1 Using ROI to Allow Auto Capture Better Generate Frame Parameters

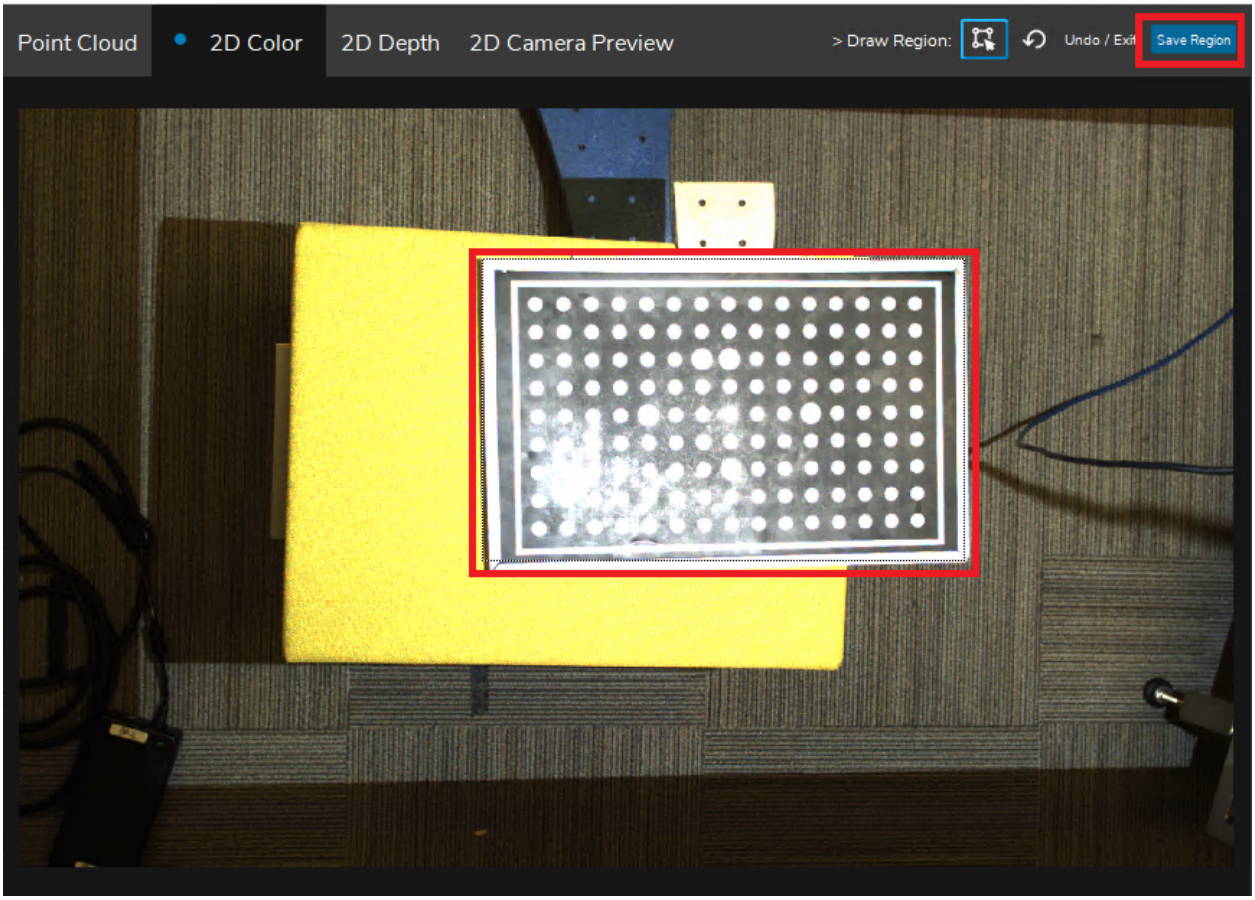
“Why is the image so bright when I use Auto Capture? “



This is because Auto Capture by defualt takes account of the entire image, trying to have all part visible. The background is darker than the foreground, and Auto Capture tries to make the background visible so the entire image becomes too bright.

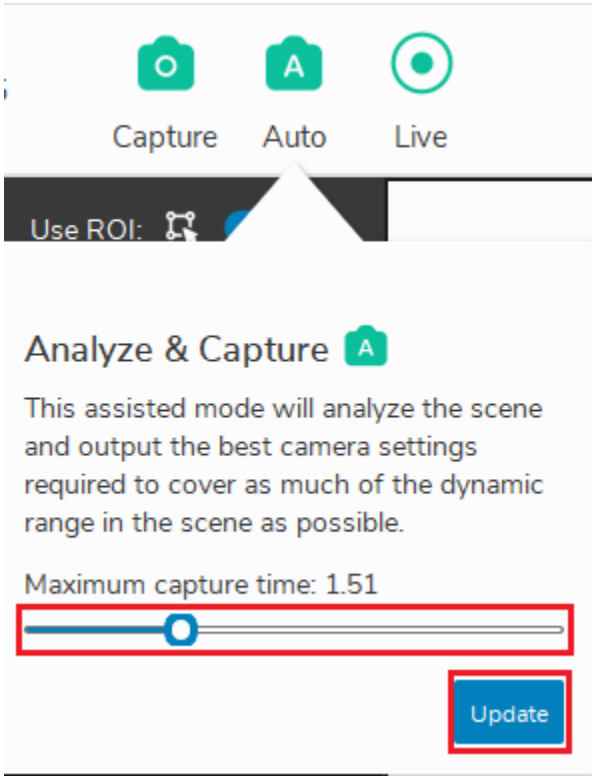
Hence, we need to contrain an Region of Interest so that Auto Capture will only consider this area when generating frame parameters.

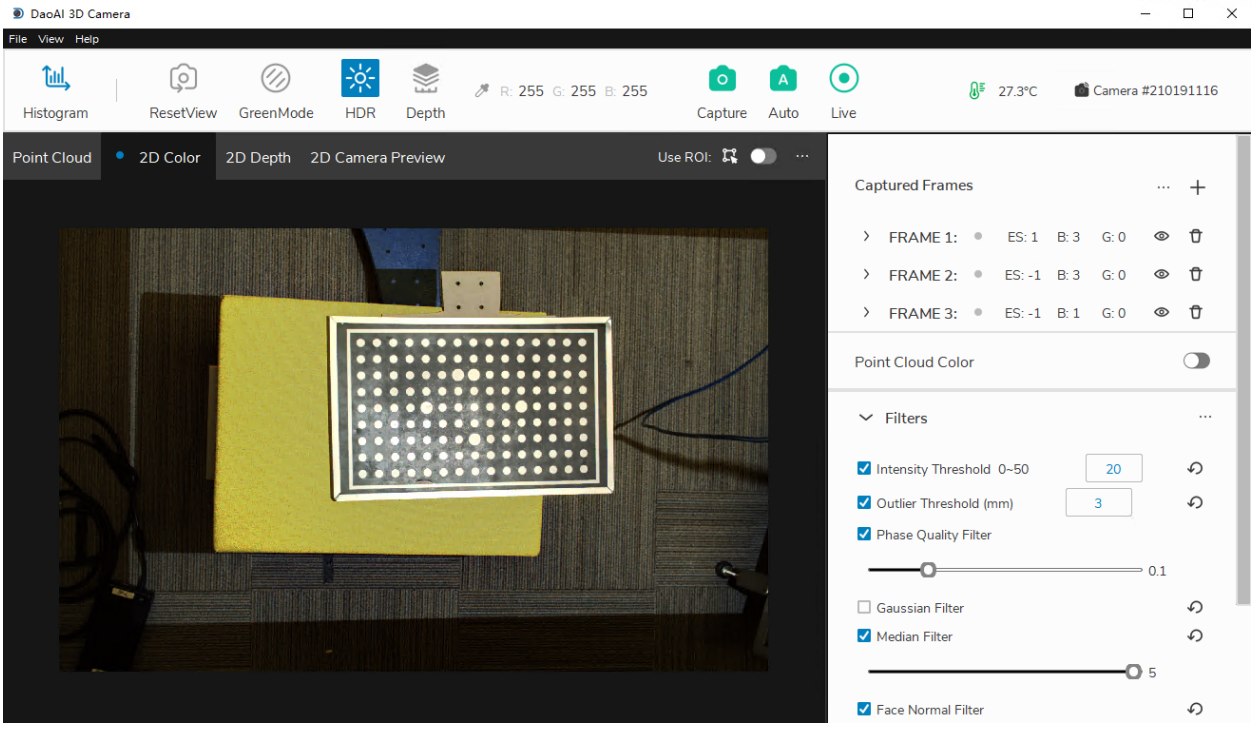
1. Enable ROI and drag to select a region then click save.



2. Hover mouse over Auto and allocate capture time, then click auto to generate frame parameters.

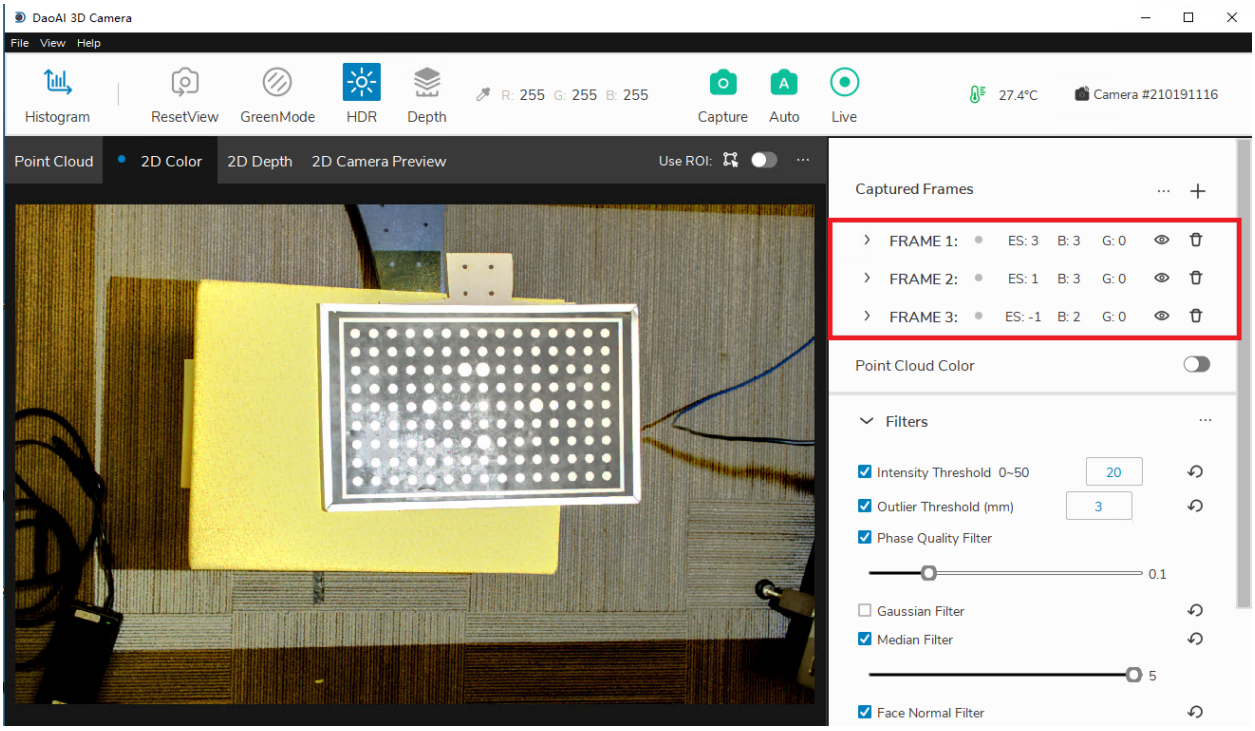
3. You may disable the ROI later as the frame parameters is already generated.


Capture
Auto
Live
Use ROI:
Analyze & Capture
This assisted mode will analyze the scene and output the best camera settings required to cover as much of the dynamic range in the scene as possible.
Maximum capture time: 1.51
Update


DaoAI 3D Camera
File View Help
Histogram
ResetView
GreenMode
HDR
Depth
R: 255 G: 255 B: 255
Capture
Auto
Live
27.3°C
Camera #210191116
Point Cloud
2D Color
2D Depth
2D Camera Preview
Use ROI:
Captured Frames
FRAME 1: ES: 1 B: 3 G: 0
FRAME 2: ES: -1 B: 3 G: 0
FRAME 3: ES: -1 B: 1 G: 0
Point Cloud Color
Filters
Intensity Threshold 0~50
20
Outlier Threshold (mm)
3
Phase Quality Filter
0.1
Gaussian Filter
Median Filter
5
Face Normal Filter

8.2 Delete a Frame in the Least Impactful Way

“When using multiple frames and the image acquisition time is too long, which frame should I delete?””



1. If you have More than 3 frames, delete the frame which its brightness is the median of the frames. This way the brightness is least affected after deletion.

2. If you have 2 frames, combine the two frames by taking the average of their frame settings and replace the first frame, then delete the second frame. This way the brightness is least affected after deletion.


Captured Frames
FRAME 1: ES: 1 B: 3 G: 0
FRAME 2: ES: -1 B: 2 G: 0
Captured Frames
FRAME 1: ES: -1 B: 3 G: 0

CHAPTER

NINE

CAPTURING HIGH QUALITY POINT CLOUDS

DaoAI camera studio have three ways to change the exposure:

- Projector Brightness
- Exposire Time
- Gain

In this tutorial, we will first explore what the different exposure variables are, how they work, and which considerations should be taken when using them. Then we will apply these principles to establish a simple 3D imaging technique that aims at acquiring high-quality point clouds.

Projector Brightness

Projector brightness controls the output power and thus the amount of light that is emitted by the projector. Using the projector brightness is the most efficient way to maximize signal-to-noise ratio (SNR). Maximizing projector brightness will maximize the amplitude of the signal received by the camera. This minimizes impact from noise, as long as the reflected light from the projector does not over-saturate the pixel. At the same time as improving the peak signal amplitude, increased brightness also affects the mean intensity of the image. This also means that projector brightness can be used to control exposure, measured in stops.

Exposure Time

The exposure time also known as shutter speed is the amount of time that a single camera image is exposed to light. In other words, for how long the shutter remains open.

Gain

Gain is a parameter that lets the user configure the pre-amplification of the read-out circuit in the imaging sensor pixels. In photography, it is commonly referred to as ISO. Increasing the gain increases the pre-amplification of the sensor which will increase its sensitivity to light and thus result in brighter images.

9.1 3D imaging technique

In this part of the tutorial, we will present a 3D imaging technique. This technique utilizes the histogram to evaluate our point cloud in a predictable and step-by-step manner, in order to acquire good 3D point clouds. The procedure can be divided into these steps:

9.1.1 Working Distance and Camera Positioning

Introduction

In this tutorial, we will learn how to go about positioning the camera correctly for a given application. We will cover different considerations such as Field-of-View (FOV), working distance, and image blooming.

Find the right working distance

In many applications the camera will be stationary mounted to capture point clouds with a constant FOV. In other applications the camera may be mounted on the robot arm. Robot mounted camera gives a lot more freedom as to how the camera can be positioned to capture good point clouds. In both alternatives, the camera still needs to be positioned such that it is optically optimized for the given scene.

When finding the correct working distance, there are a few things to take into consideration:

1. What is the area or volume that the camera needs to see?
2. What is the required spatial resolution and precision in the working area or volume?
3. Is the given field of view within the working range of the DaoAI camera?

Once we know what region we want to image, we can check if it satisfies the recommended working distance of the camera and the requirements for our algorithm by:

- Using our *Calculator*.
- Check the pages *Working Distance and Field-of-View*. The page will estimate the relationship between camera distance to the scene, FOV, spatial resolution and precision.
- Using the datasheets.
- Manual validation by capturing and inspecting a point cloud using e.g. DaoAI Camera Studio.

Note: In the datasheet and the FOV pages we can find information about DaoAI camera point precision and spatial resolution as functions of the working distance.

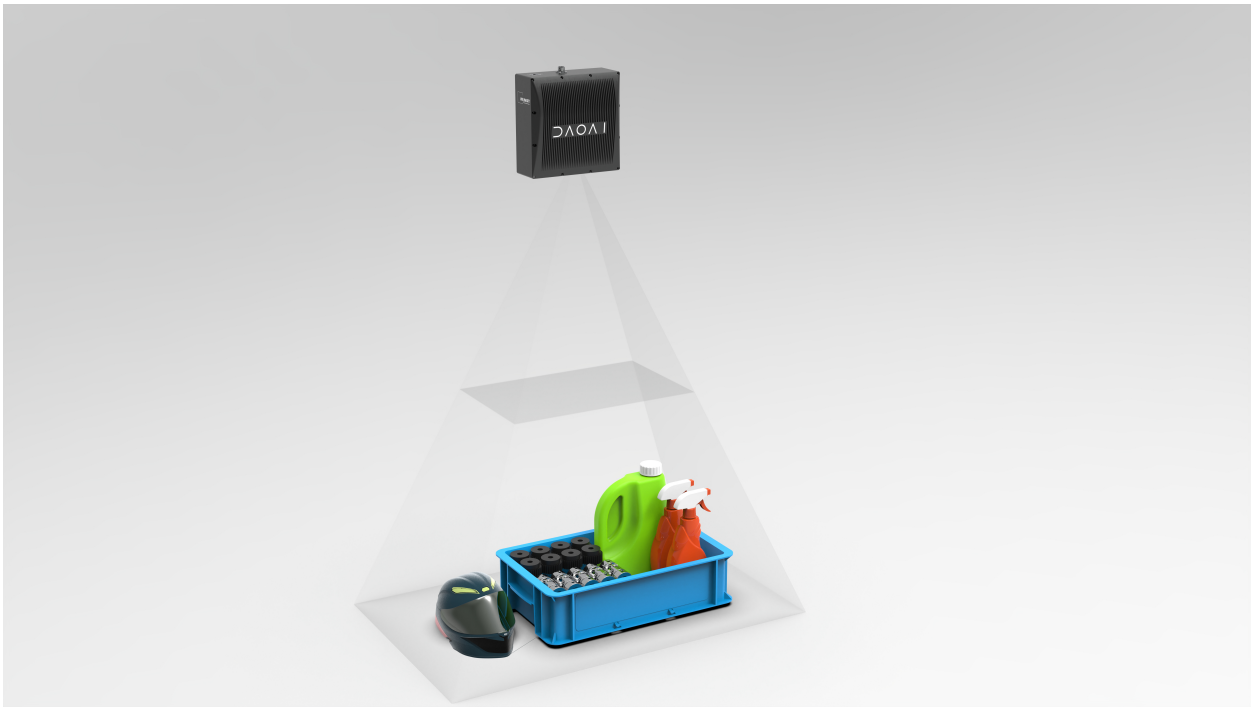
Warning: Be aware that the noise is proportional to the working distance, and that the spatial resolution is inversely proportional to the working distance.

We may proceed if all requirements for working distance, FOV, resolution and precision can be satisfied with a single camera position. If not, we need to consider using robot mounting, multiple cameras and so on. In that case we recommend that you contact support@daoai.com and we will help you find a solution.

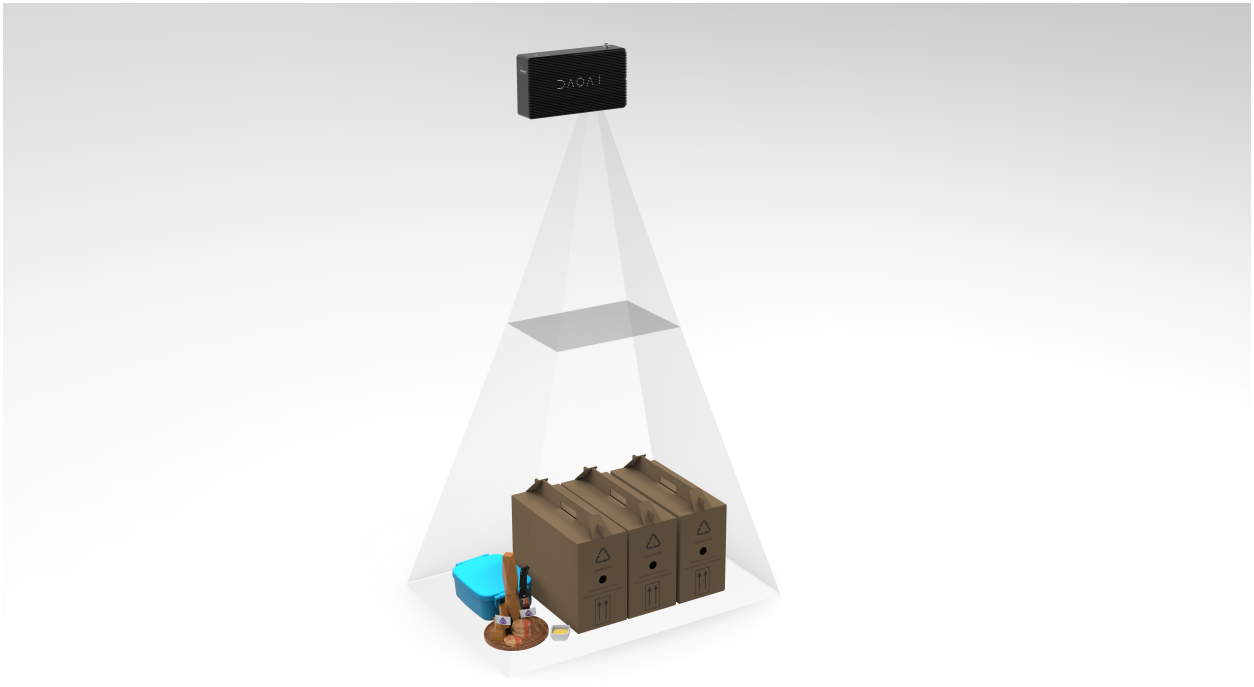
Angle the camera

The imaging sensor inside DaoAI cameras is offset at a slight pan angle in the azimuth direction (y-axis). This should be considered if it is desired to have the camera perpendicular to the scene.

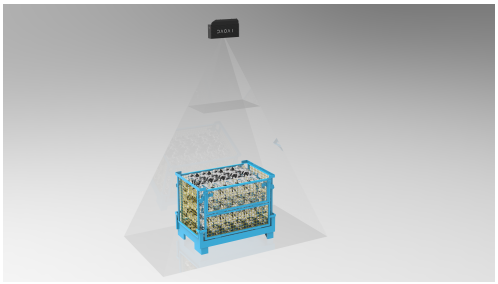
BP SMALL



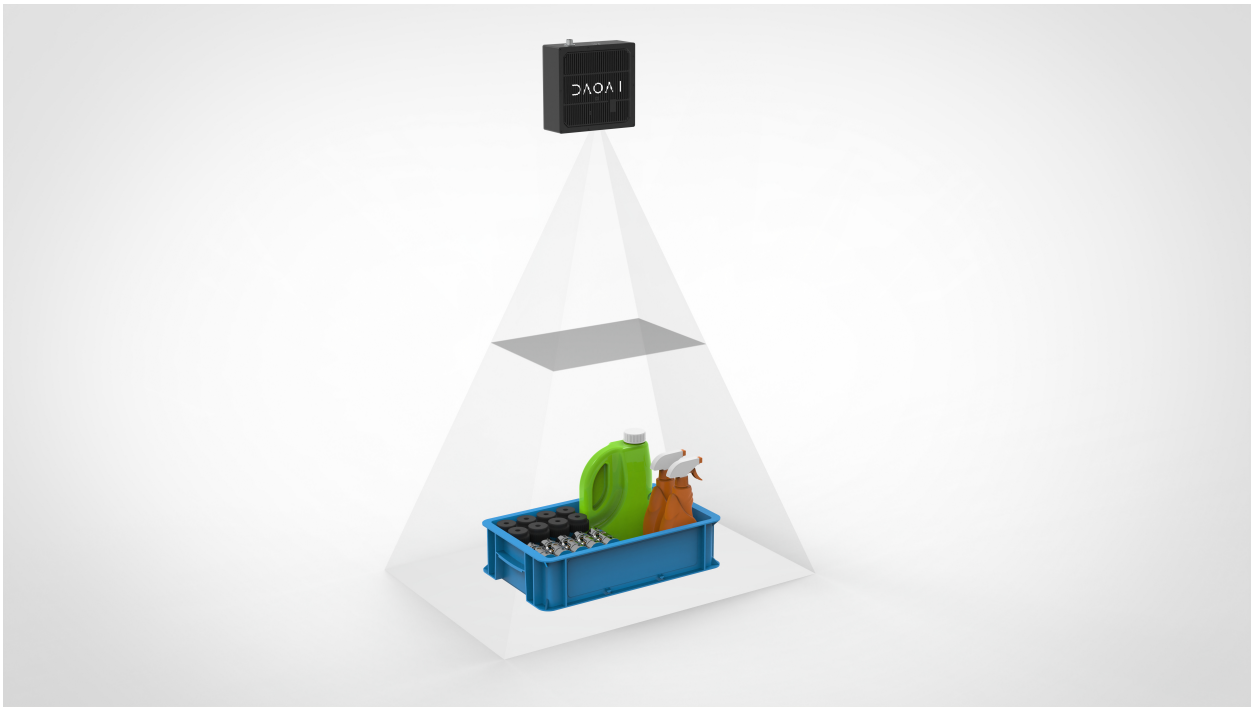
BP MEDIUM



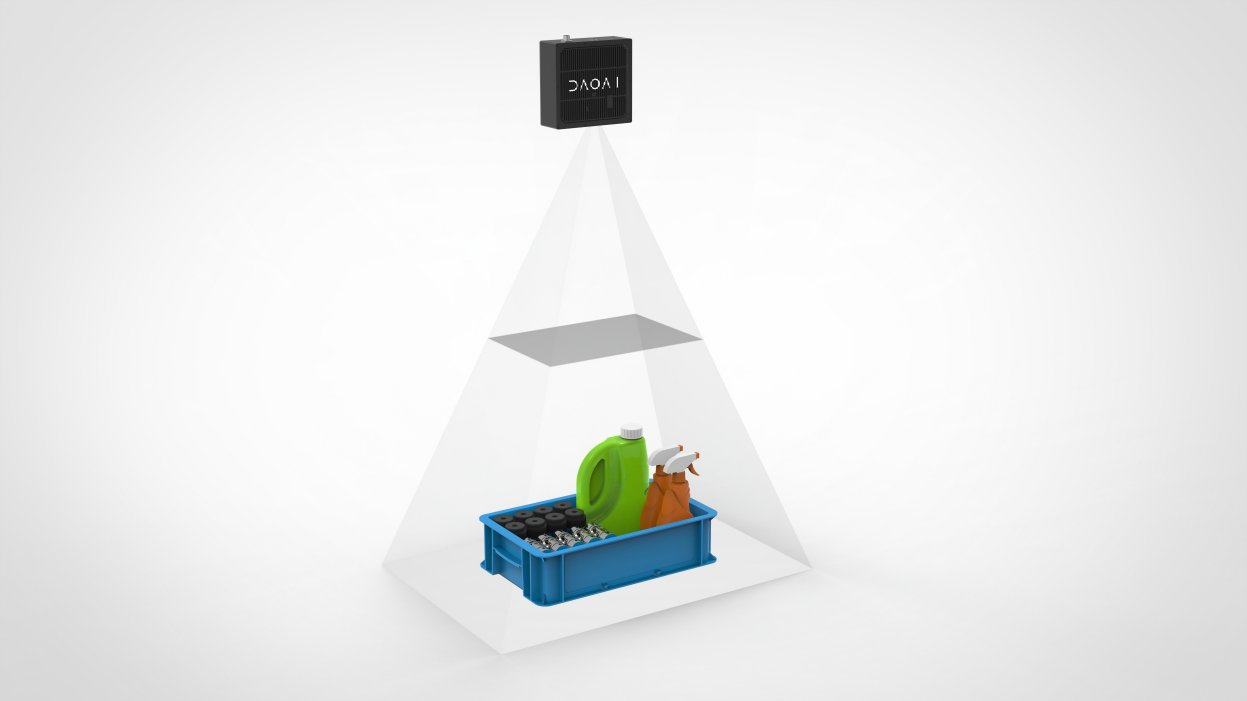
BP LARGE



BP AMR

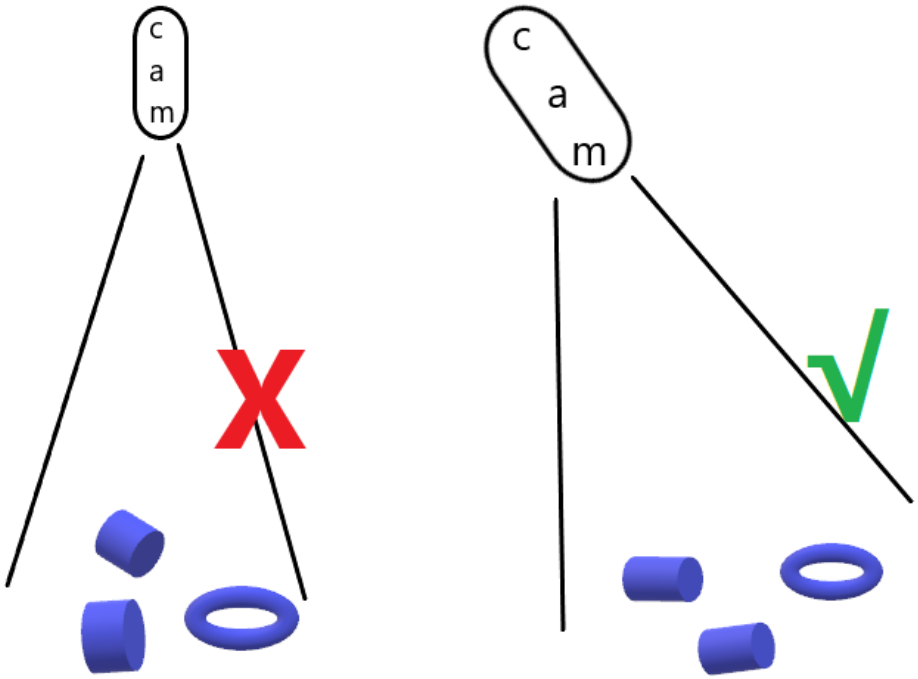


BP AMR-GPU

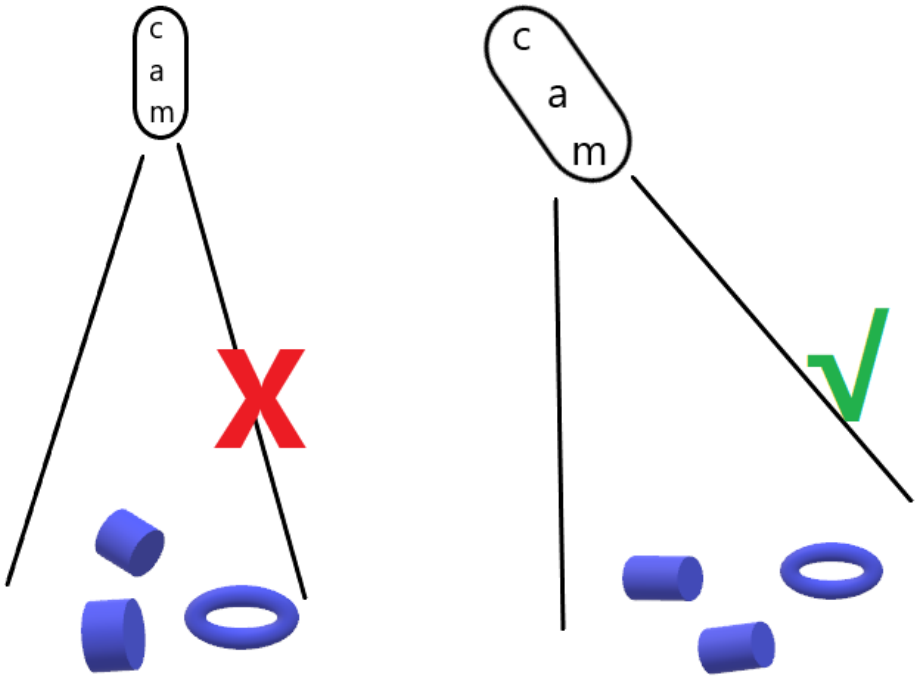


Although it is perhaps most intuitive to mount the camera perpendicular to a scene, this is not the best way. If possible, mount the camera at a slight tilt angle to avoid reflections from the background, as explained in Blooming - Bright Spots in the Point Cloud. This also frees up space above the scene for easier access for tools and robots. Check out available DaoAI Mounting.

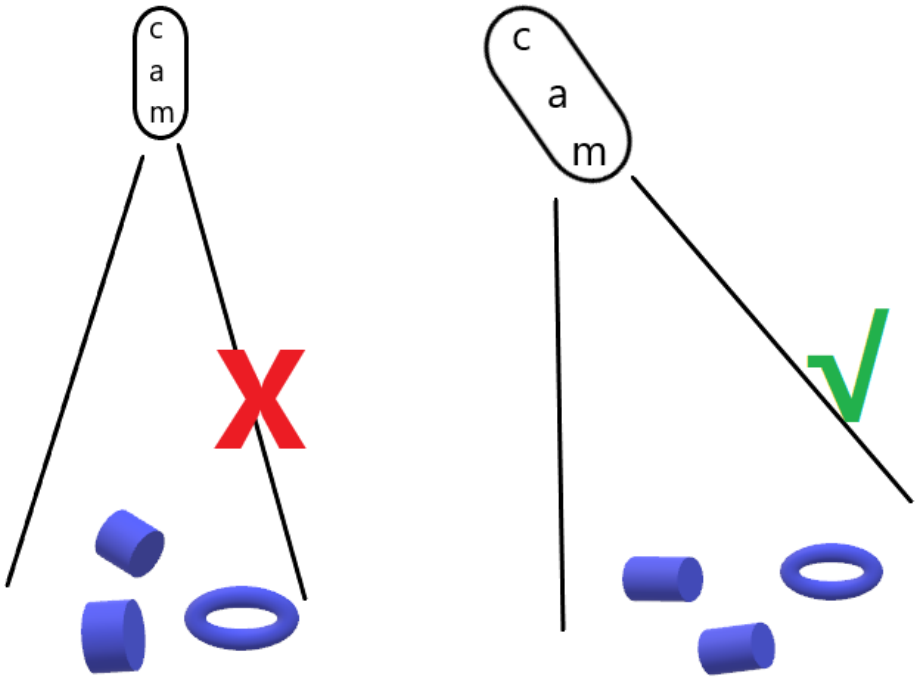
BP SMALL



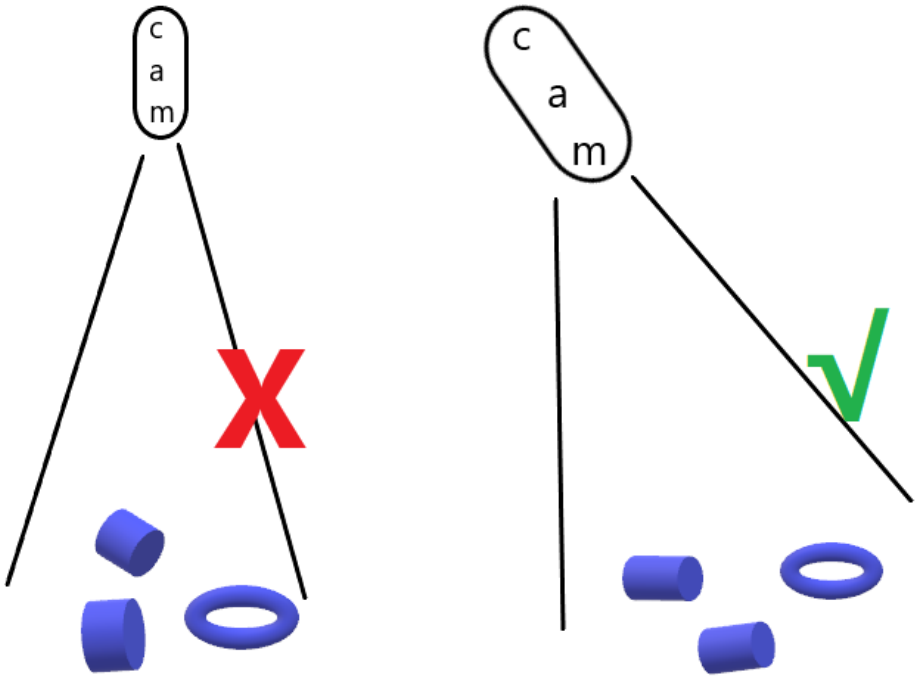
BP MEDIUM



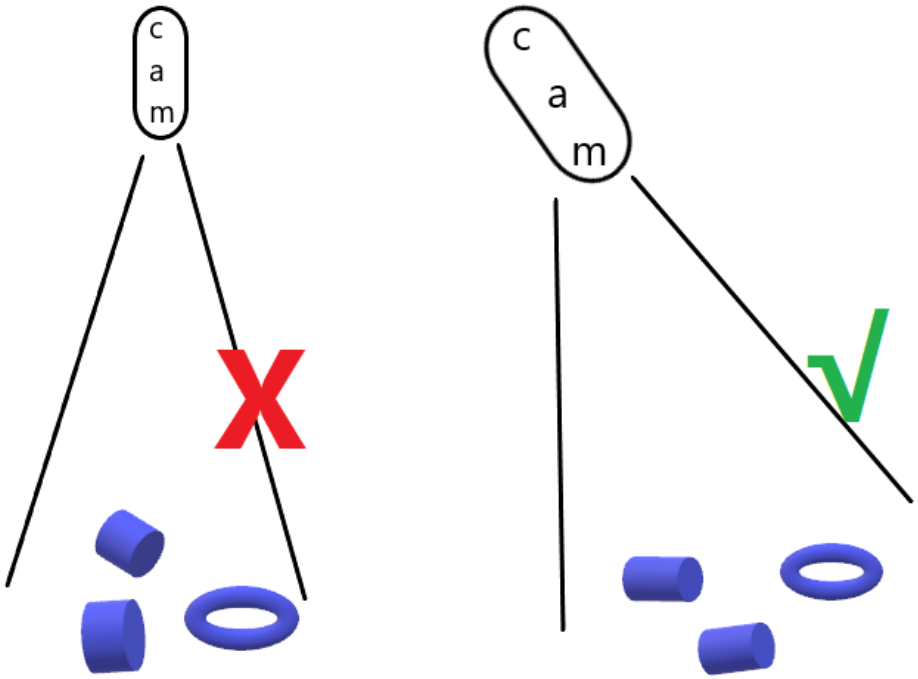
BP LARGE



BP AMR



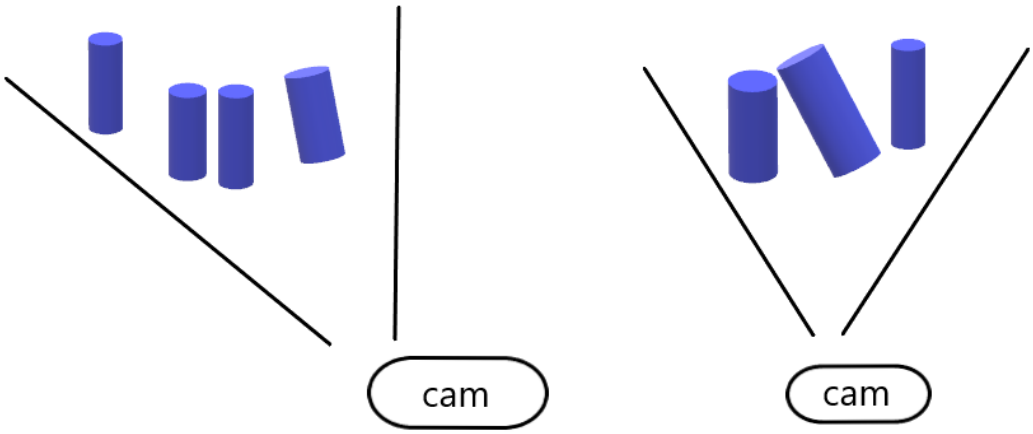
BP AMR-GPU



Note: Camera tilting is more important if the scene contains specular surfaces.

In bin-picking applications

For bin-picking applications, place the DaoAI BP camera projector above the back edge or above the rear corner of the bin (see images below). Pan and tilt it so that the 2D camera is looking at the center of the bin. The projector rays should not fall on the inner surfaces of the two walls closest to the projector; they should almost be parallel to those two walls. Mounting the camera this way minimizes inter reflections from the bin walls.



Find the required depth of focus

DaoAI cameras are very robust against defocus, but to maximize the precision of the point cloud, the depth of focus should be taken into consideration. This step is only required if the algorithm that will work on the point cloud requires it in order to be successful.

Alternatively use the Depth of Focus *Calculator* by selecting the camera model and inputting the closest and farthest working distance.

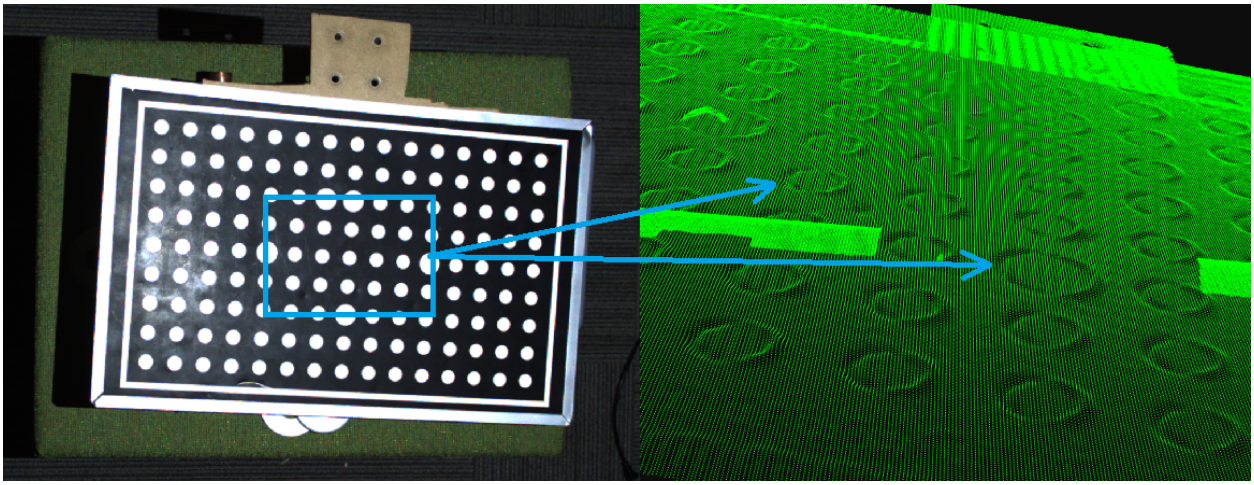
9.1.2 Dealing With HightLights and Shiny Objects

Introduction

This article will discuss the specific challenge of dealing with highlights that may occur when capturing point clouds of very shiny objects and how to deal with it.

In the presence of a shiny object, there is a chance that certain regions of that object cause a direct reflection from the DaoAI camera's projector to the imaging sensor. If this object is extremely specular, then the amount of reflected light can be thousands of times stronger than other light sources in the image. This is then likely to cause the pixel to become oversaturated. An example of such regions can be seen in the images below.

Oversaturated pixels often "bleed" light onto their surrounding pixels. This lens blurring causes results in a Contrast Distortion Artifact. This effect is evident on black to white transitions and shiny cylinders; see the image below.



Check if the projector causes the highlight

Some simple methods can be applied to determine if a highlight is caused by the DaoAI projector, as exemplified in the image above.

1. Study the 2D image and consider the different angles in the surfaces where highlights occur. This method requires some practice.
2. Perform a differential measurement by taking an additional image. Try to kill the projector light by setting projector brightness to 0, or cover the projector lens with, e.g., your hand and capture a second image. If the highlights disappear in the second image, you can conclude that the source is from the projector.

Maximize the dynamic range of the 3D sensor to capture highlights

Getting good point clouds of shiny objects requires that you can capture both highlights and lowlights. The DaoAI 3D camera has a wide dynamic range, making it possible to take images of both dark and bright objects.

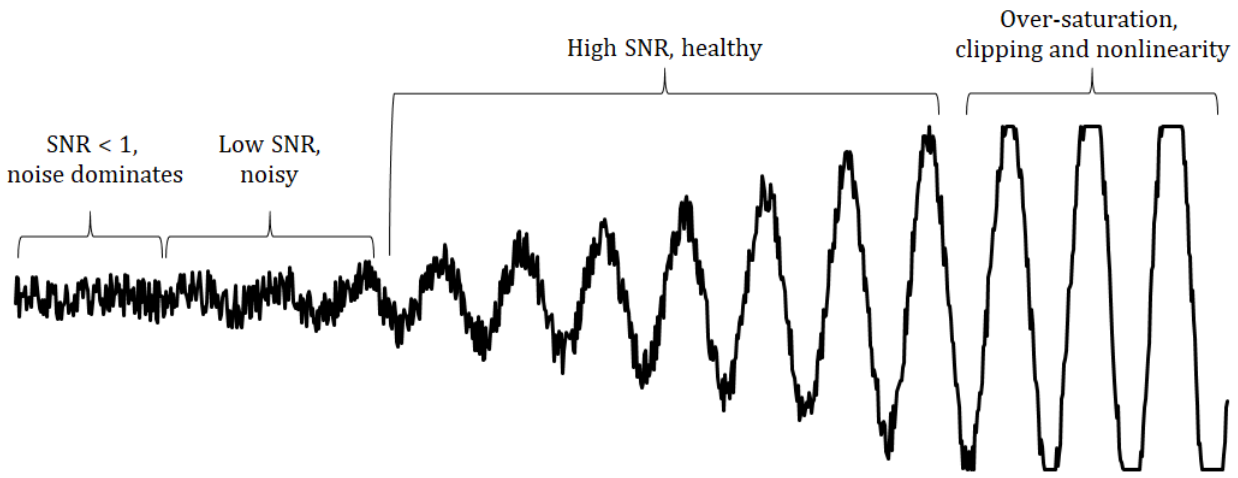
Very challenging scenes (as the one below) typically require 3 HDR acquisition frames or more. Challenging scenes should typically have:

- 1-2 acquisition frames to cover the strongest highlights (very low exposure).
- 1-2 acquisition frames to cover most of the scene (medium exposure).
- 1-2 acquisition frames to cover the darkest regions (very high exposure).

The following two principles should be applied:

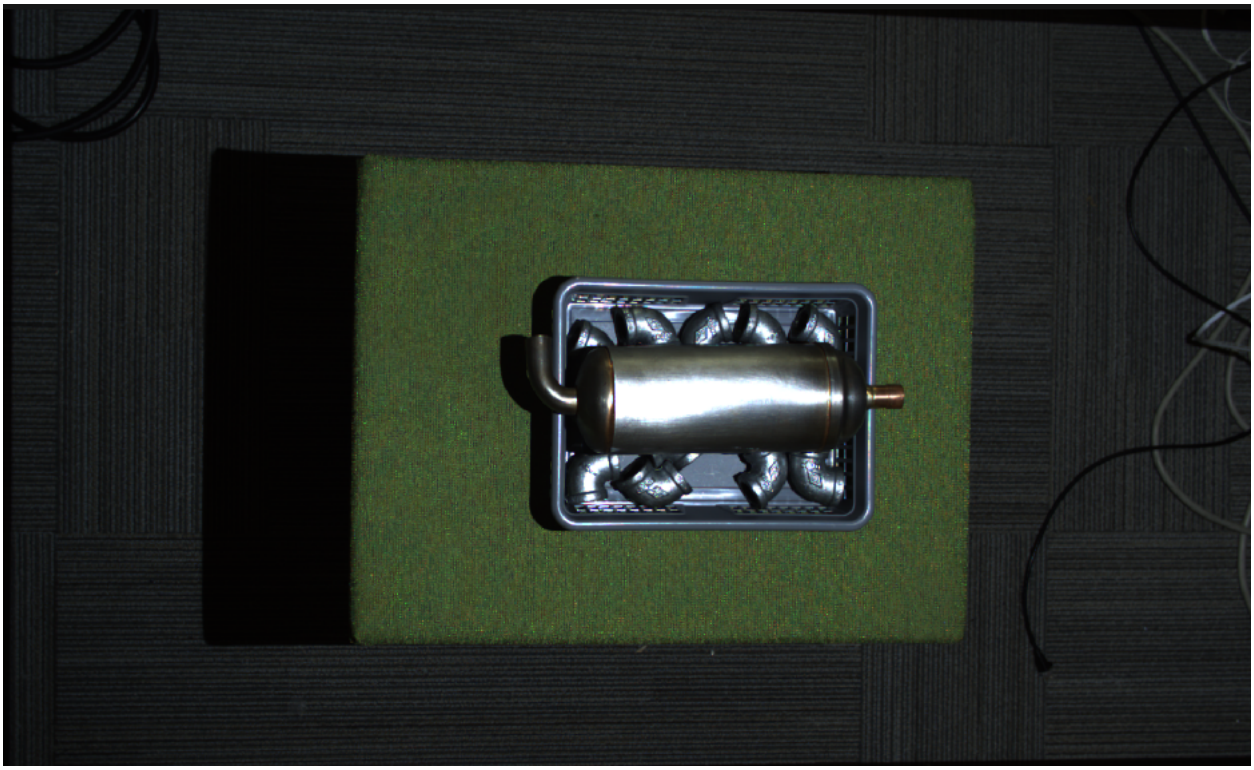
1. Keep the exposure very low.
2. It may be necessary to reduce the projector brightness to keep the projected pattern amplitude within the imaging sensor's dynamic range.

In principle, what we are trying to achieve, by limiting projector brightness, is to bring the signal back, from the over-saturated region, to the healthy region. This is illustrated in the figure below.



We will now focus on capturing the highlights. We assume that we have identified extreme highlights in our scene by using the method described above.

Let us assume the scene below.

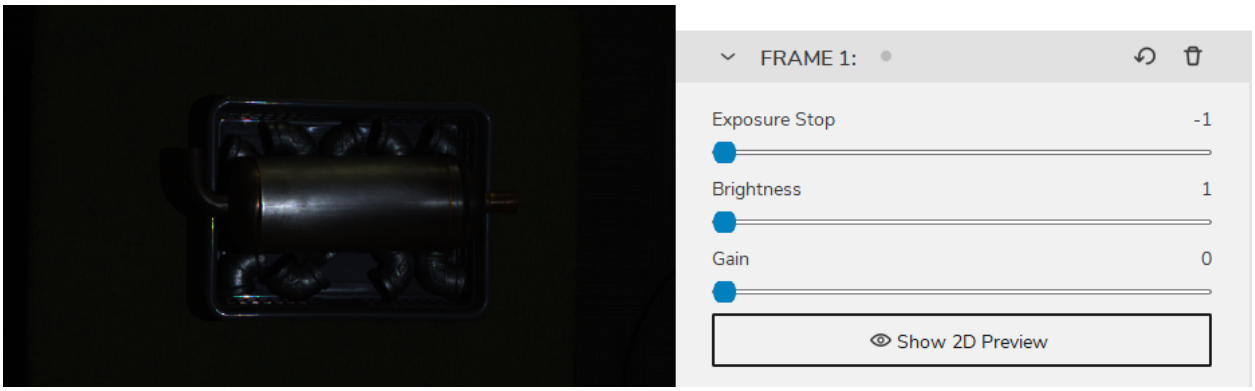


We can clearly see strong highlights in the scene.

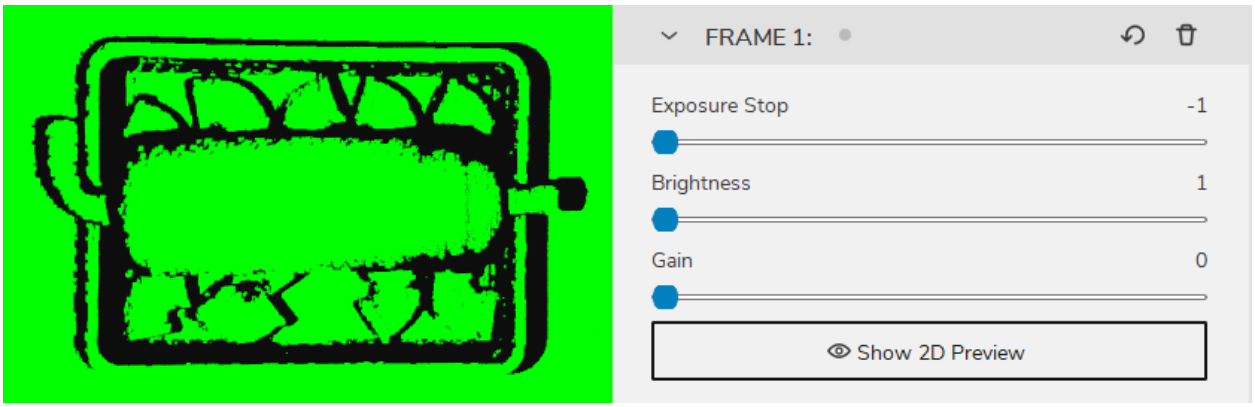
Highlight Frames

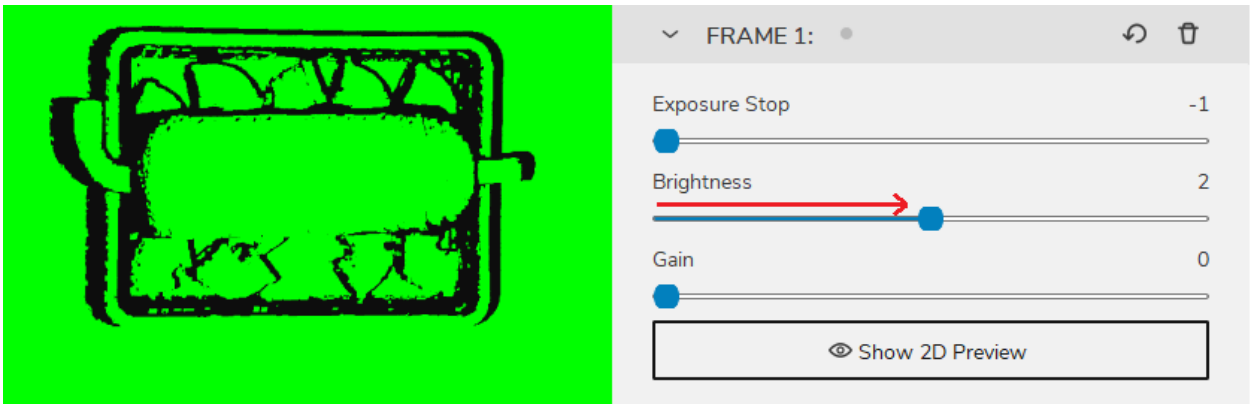
First we begin by adjusting 1-2 acquisition frame settings to best capture the highlight area of the object. In this case, we want to minimize the amount of projector light that shines on the object.

We begin with a single acquisition frame of lowest brightness settings.



Then gradually increase the brightness until the highlight area of the object is best captured and without introducing missing pointcloud and contrast distortion artifacts

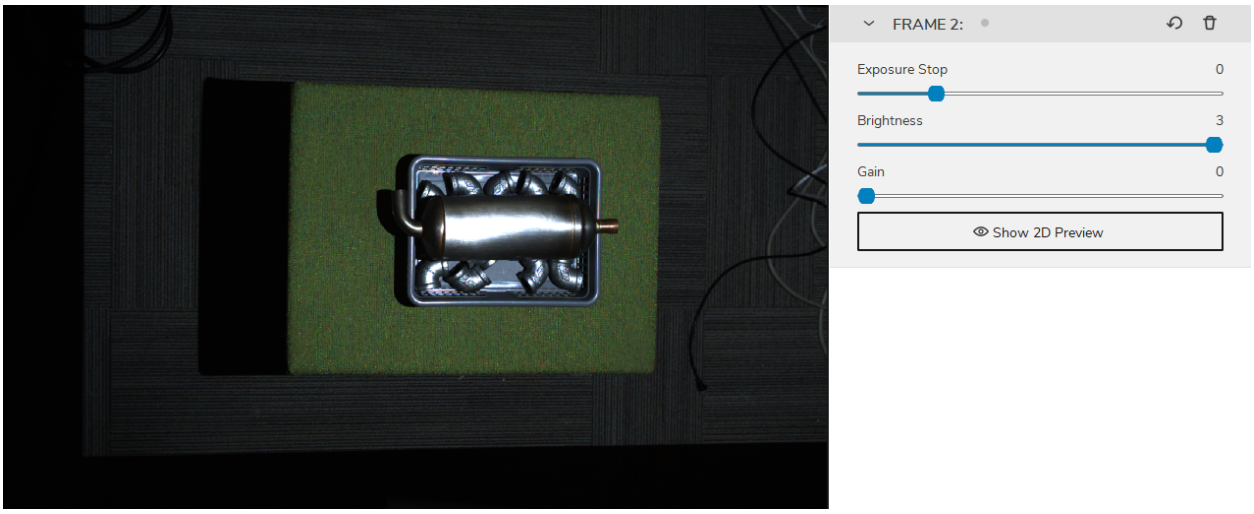




And this will be our highlight frame. Note down this frame settings and delete it for now, and we begin to find our main frame.

Main Frames

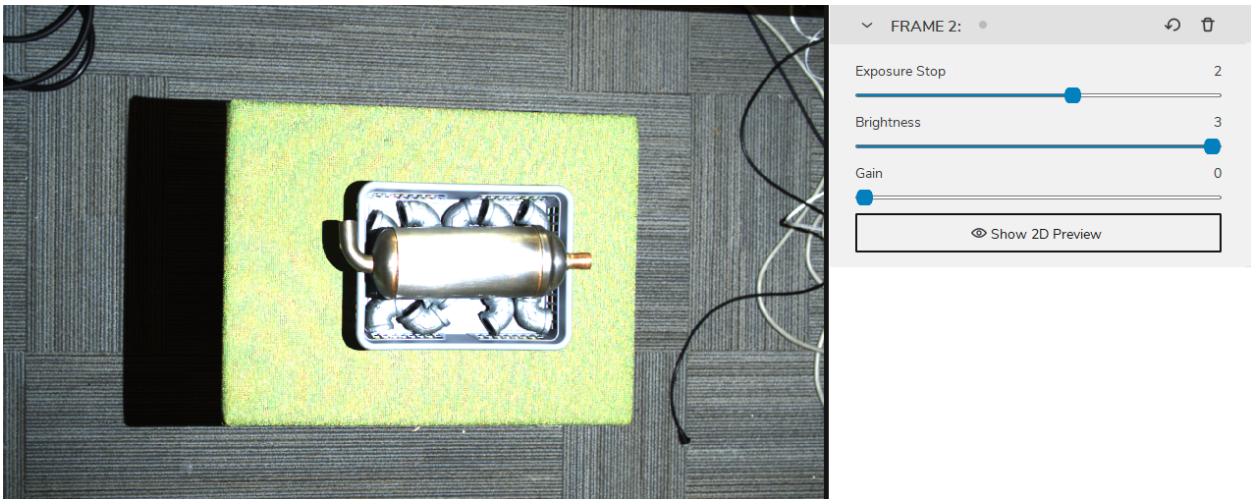
Next we use 1-2 acquisition frames to best capture the main areas of the object. So that every detail of the object, apart from the highlight area can be best captured.



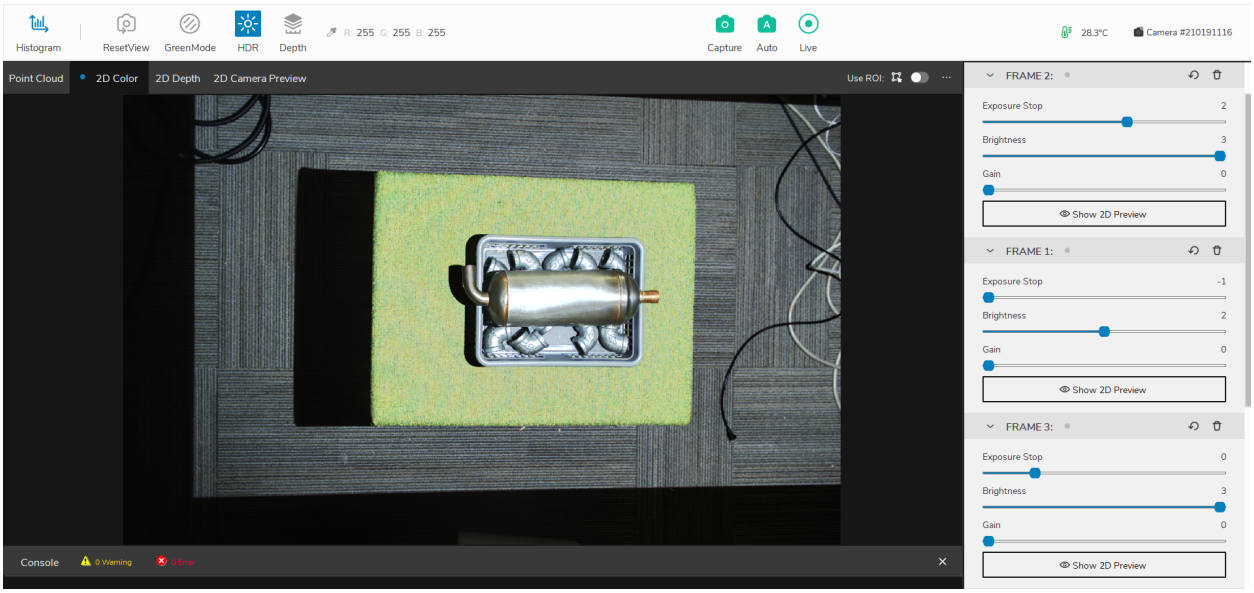
And this will be our main frame. Note down this frame settings and delete it for now, and we begin to find our background frame.

Background frames

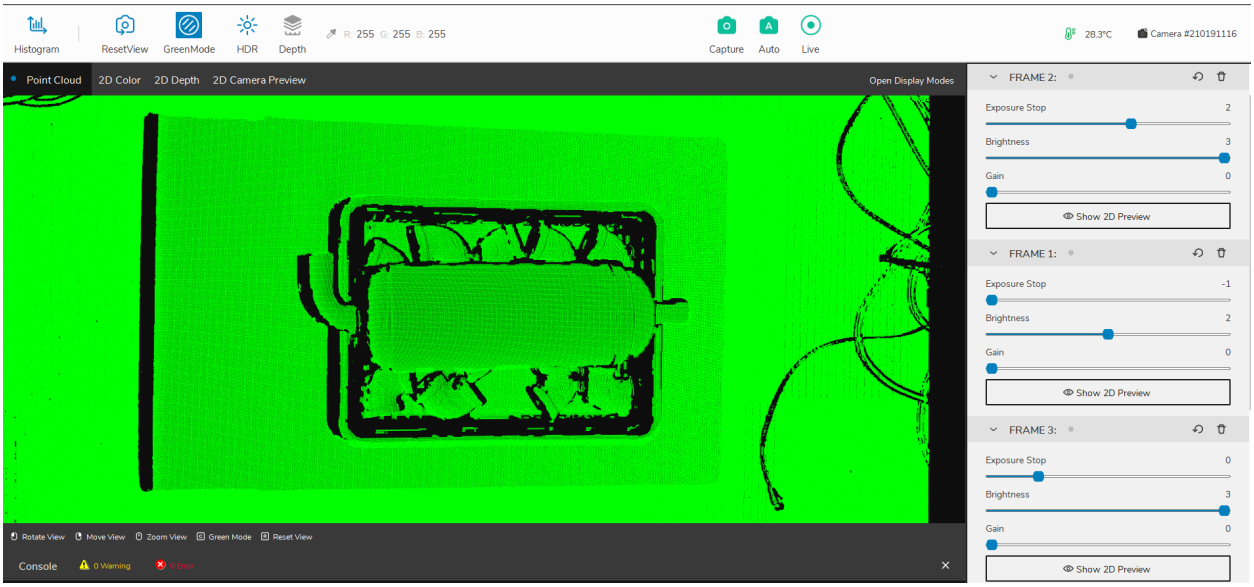
Then we can add a acquisition frame to cover the rest of the scene and the background.



Then finally, we add back our highlight frame and main frame, and enable HDR mode.



and the point cloud is as the following

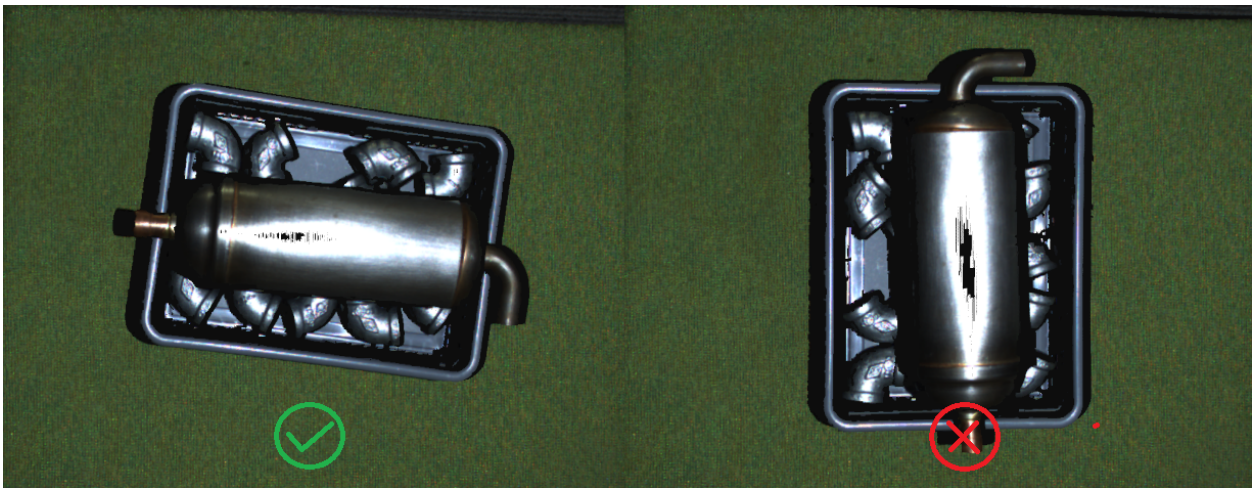


Deal with the contrast distortion

There are mainly two ways to deal with Contrast Distortion. We can reduce the effect by maximizing the dynamic range of our camera and place the camera in strategic areas. Then we can use the Contrast Distortion Filter to correct/remove the remaining points that are affected.

Rotate and align objects in the scene

The first thing to remember is that this is an effect that occurs in the 3D sensor's x-axis. The Contrast distortion effect can be greatly mitigated if your application allows for rotating troublesome regions in the camera's y-axis to its x-axis. By rotating, for instance, a shiny cylinder 90°, the overexposed region along the cylinder follows the camera's baseline, as illustrated in the figure below.



Match the background's reflectivity to the specific object's reflectivity

A good rule of thumb is to try to use similar brightness or color for the background of the scene as the objects that you're imaging:

- For a bright object, use a bright background (ideally white Lambertian).
- For a dark object, use a dark background (e.g., black rubber as used by most conveyor belts).
- For most colored, non-glossy objects, use a background of similar reflectivity (e.g., for bananas, use a grey or yellow background).
- For shiny metallic objects, especially cylindrical, conical and spherical objects, use a dark absorptive background such as black rubber. This is because the target light is typically reflected away from the object near its visible edges, making them appear very dark (see image below). At the same time, light from surrounding regions may be reflected onto the cylinder edge.

Use Contrast Distortion filter

The filter corrects and/or removes these surface elevation artifacts caused by Contrast Distortion - defocusing and blur in high contrast regions. This results in a more realistic geometry of objects, specifically observable on planes and cylinders.

If you want to learn more about this filer and tune its parameters, check out the *Contrast Distortion Filter*.

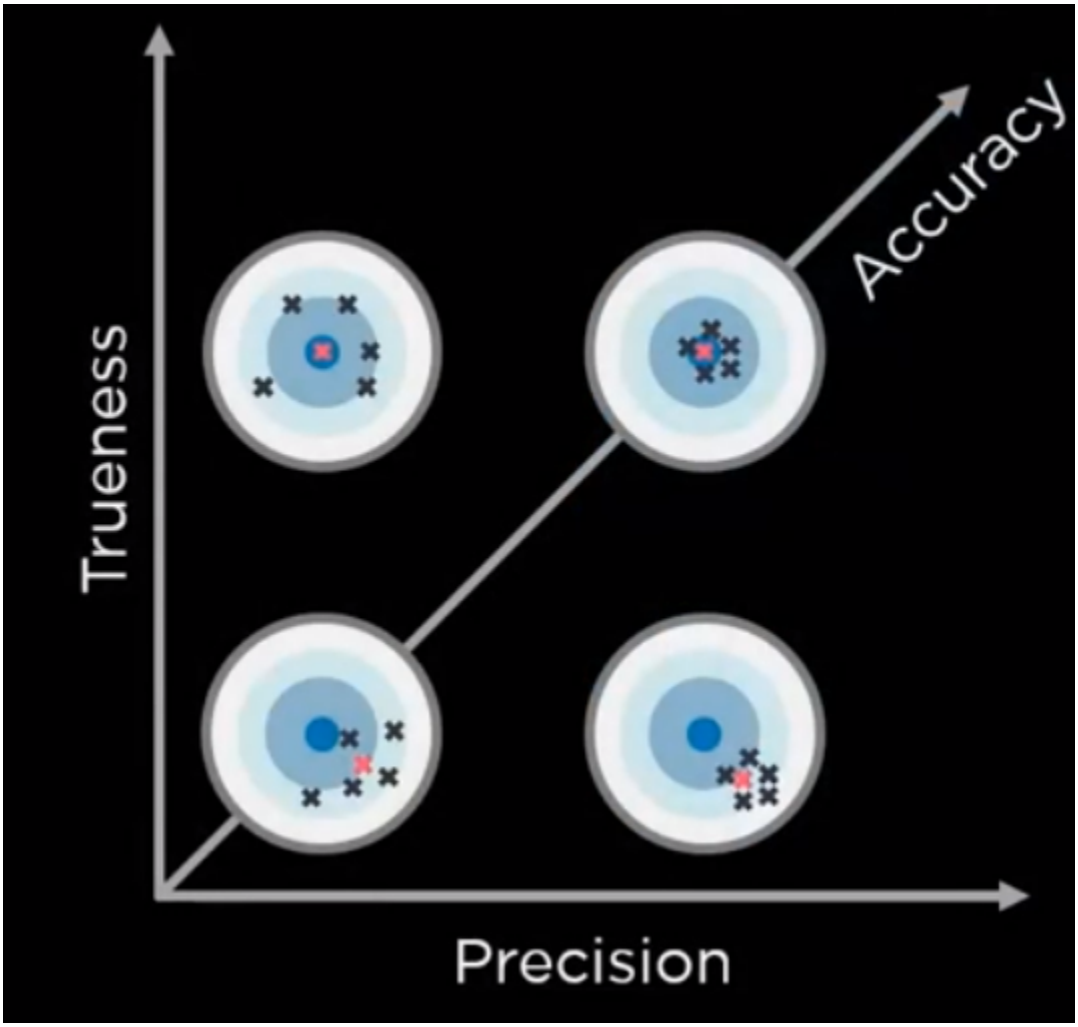
- *Working Distance and Camera Positioning*
- *Filters*
- *Dealing With HightLights and Shiny Objects*

CHAPTER

TEN

EVALUATE ACCURACY

10.1 Camera Accuracy



Camera **accuracy** is the comprehesive measure of the camera **trueness** and camera **precision**.

High camera accuracy represents that both camera trueness and camera precision is high: captured image/point cloud is at a good quality, low in noise, and the reflected object position in camera is the same as the object's acutal position in space.

In oral communication, the term camera **accuracy** is usually refered to camera **trueness**.

The accuracy of a camera is affected by:

- 3D camera hardware and optical quality
- Quality of camera calibration
- Position of the object in the calibrated field of view
- Camera temperature
- Camera aging
- Physical impacts, vibrations, and pressure

10.2 Camera Trueness

Camera trueness reflects the error between the position of the object in space and the position captured in the camera. Camera trueness determines whether the picking process can successfully pick objects in all positions and times within the field of view.

Camera trueness is highest at the camera's focus point, and gradualy decrease as the object moves farther away from the focus point. The error of an object's position on the edge of the camera's field of view is higher than the error in the camera's focus.

This increase in error applies to all axis: x, y and z axis, and the x, y axes have a slightly smaller impact relative to the z axis (x and y axis errors <0.1%), while the z axis has a slightly greater impact (z axis errors <0.2%).

10.3 Camera Precision

Precision reflects the noise level on the point cloud. The precision error will be high if the point cloud is noisy, which may be caused by following factors:

- Low contrast
- Ripple-like error caused by object motion
- Ripple-like error caused by overexposure

10.4 DaoAI Camera Accuracy

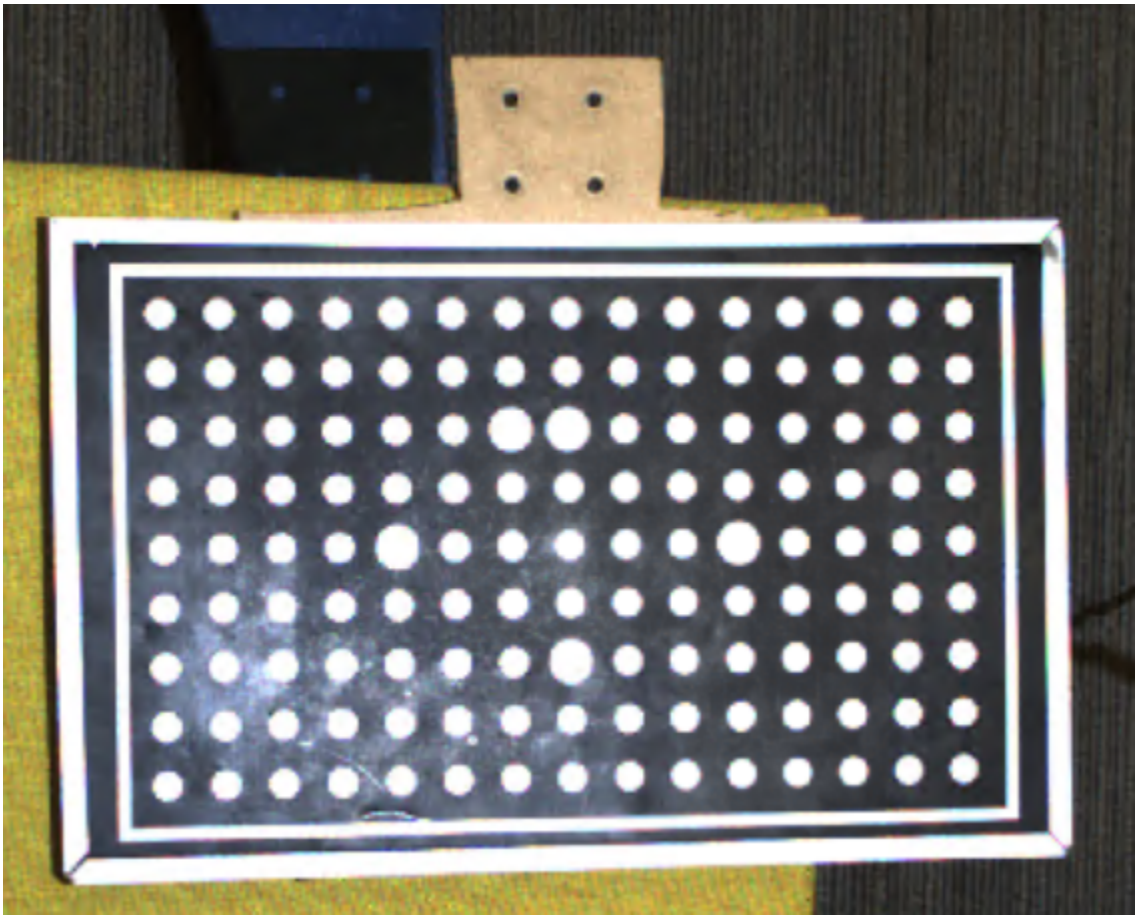
DaoAI camera provides 0.2% accuracy, with x, y-axis accuracy <0.1% and z-axis offset accuracy error <0.2%. This means that, without considering other factors, at a capture distance of 100mm, the camera's accuracy error will be at most 0.2mm. Similarly, at a capture distance of 1000mm, the camera's accuracy error will be at most 2mm.

10.5 How to Validate Camera Trueness

Camera trueness can be effected by physical impacts, tempreature, and time of use. Therefore validating your camera trueness is important in accheiving successful pickings. When camera trueness is greater than the acceptable range (< 0.2%) then the camera will need to be calibrated again.

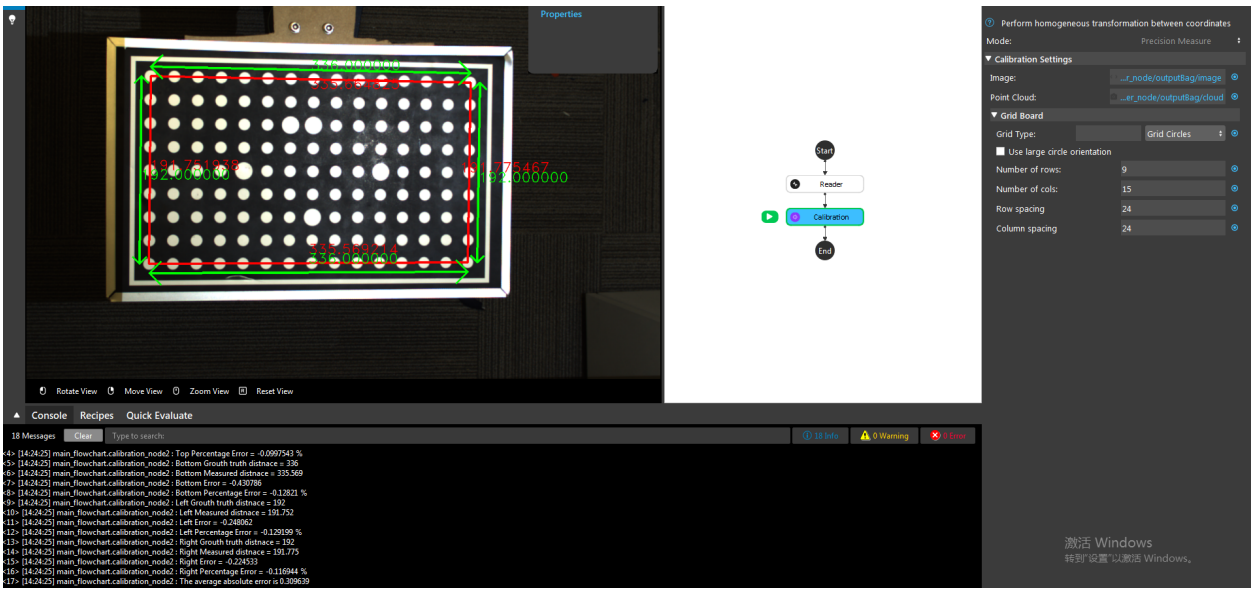
Recall that camera trueness is the error of the captured object position and the actual object position.

To verify this error, we can use a known sized object, the calibration board for example: the distance between the cirlces are known. We can compare the distance captured by the camera with the actual disctance which we know.



For example, the calibration board has 15 cirlces a in a row, and each one is 24mm away from another, that is, the row should have a distance of 336. Suppose this distance measured in camera is 335, then the camera trueness can be calculated as (336.5-335)/336 * 100% = 0.15%.

If you have DaoAI Vision Studio installed, you can use the calibration node's precision measure mode to validate camera trueness.



CHAPTER

ELEVEN

INFIELD CALIBRATION

Infield calibration is a maintenance tool designed to verify and correct for the dimension trueness of DaoAI cameras. The user can check the dimension trueness of the point cloud at different points in the field of view (FOV) and determine if it is acceptable for their application. If the verification shows the camera is not sufficiently accurate for the application, then a correction can be performed to increase the dimension trueness of the point cloud. The average dimension trueness error from multiple measurements is expected to be <0.1%.

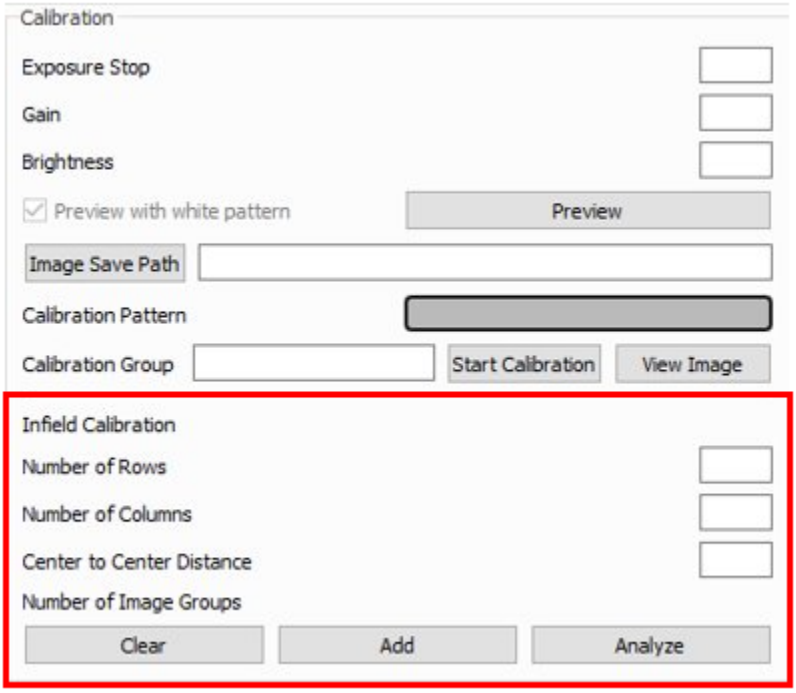
Why is this necessary

Our cameras are made to withstand industrial working environments and continue to return quality point clouds. However, like most high precision electronic instruments, sometimes they might need a little adjustment to make sure they stay at their best performance. When a camera experiences substantial changes in its environment or heavy handling it could require a correction to work optimally in its new setting.

Field calibration function

Note: Requires DaoAI camera studio version 2.22.10.1

If the accuracy of 3D camera is found to be lower than 0.2% error standard during field use, you can use field calibration to optimize the camera parameters and make the camera accuracy meet the standard. Field calibration usually requires only a total of 3-5 sets of photos of calibration plates with different positions, depths and deflections.



Enter the calibration mode: Make sure the 3D camera is connected, click Help → System in the upper left corner, enter the password Robotics_2018 in the new pop-up authentication window and click OK, check the checkbox "Calibrating Mode" (Calibrating Mode) and close the System window. The "Calibration" subgroup box should appear at the bottom of the main window. The lower red section is the Infield Calibration panel.

Table 1: Panel Introduction

Number of Rows	Enter the number of rows of calibrated board circles
Number of Columns	Enter the number of columns of calibrated board circles
Center to Center Distance	Enter the distance from the center of the calibration board to the center of the adjacent circle
Clear	Clear all images that have been captured for field calibration
Add	Add a set of pictures of field calibration
Analyze	Analysis with images that have been collected for field calibration

1. With the current camera parameters, refer to the *Evaluate Accuracy* experiment , analyze the accuracy of the camera, record the current camera accuracy, and use it for comparison with later to judge the effect of field calibration.
2. The calibration board is placed in different positions, depths, and deflections, 3-5 positions in total, throughout the target measurement range. In each position, you need to follow the same requirements as the lab calibration Hand Eye Calibration , adjust the exposure and brightness so that the calibration plate preview photo is bright but not overexposed (no large red areas). Click "Add" button to capture a set of calibration photos, and the number of photo groups will show +1 when finished. 3.
3. After acquiring 3-5 groups of photos, click the "Analyze" button. Camera Studio may be stuck for about 10s, waiting for the data processing to finish.
4. Check the console information. If the field calibration data is processed successfully, the following message will be displayed "Infield calibration done. If this message is not present or error is reported, the analysis failed. You need to click on the "Clear" button and start again from step 2.

```
<23> Frame exposure time: 30 ms. Considering projector dark time, camera exposure: 36
<24> Infield Calibration data added. Group: 1
<25> Frame exposure time: 30 ms. Considering projector dark time, camera exposure: 36
<26> Infield Calibration data added. Group: 2
<27> Frame exposure time: 30 ms. Considering projector dark time, camera exposure: 36
<28> Infield Calibration data added. Group: 3
<29> Finding circles for Group 0 ...
<30> Finding circles for Group 1 ...
<31> Finding circles for Group 2 ...
<32> Analyzing Group 0 ...
<33> Analyzing Group 1 ...
<34> Analyzing Group 2 ...
<35> Analyzing Group 0 ...
<36> Analyzing Group 1 ...
<37> Analyzing Group 2 ...
<38> Initial re-projection error: 0.073626 0.0549011 0.0556188
<39> Iter: 1 ... RMS re-projection error (camera 1, camera 2, projector): 0.0585226 0.181558 0.359734
<40> Iter: 2 ... RMS re-projection error (camera 1, camera 2, projector): 0.0501784 0.0379163 0.0385493
<41> Infield calibration done. Please reconnect camera on GUI to make effect.
<42> Infield Calibration data cleared.
```

5. After the field calibration analysis is complete, disconnect and reconnect the camera in Camera Studio.
6. Similar to step 1, refer to the DaoAI 3D camera spatial distance accuracy experiment to analyze the accuracy of the camera and record the field calibration camera accuracy. If the accuracy is higher than 0.2% error standard, the field calibration is successful.

CHAPTER

TWELVE

USING MULTIPLE DAOAI CAMERAS

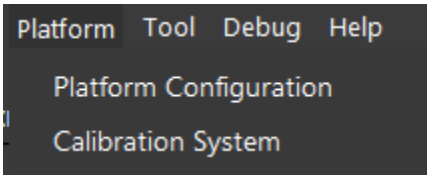
DaoAI Vision allows you to operate multiple DaoAI Cameras of the BP series simultaneously (e.g. connecting both BP-S and BP-L).

However, it comes with some limitations:

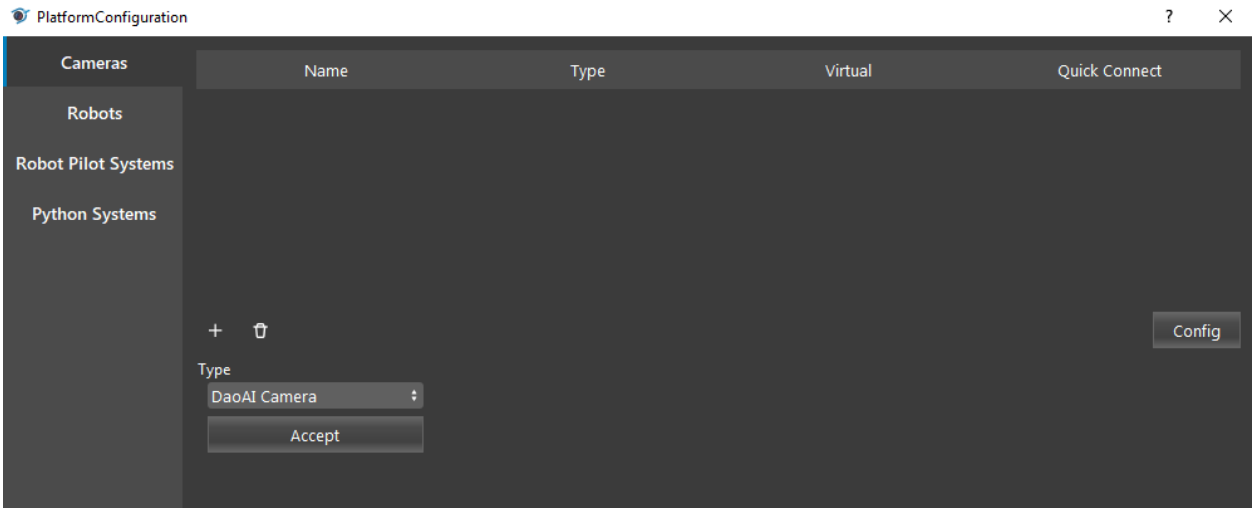
- Each of the DaoAI Cameras must have a unique IP address.
- Each of the DaoAI Cameras have to be connected using separate ethernet ports.
- The same holds for any combination of these API calls (connect, update firmware, list cameras) at the same time from multiple threads or processes.

To connect multiple cameras in Vision:

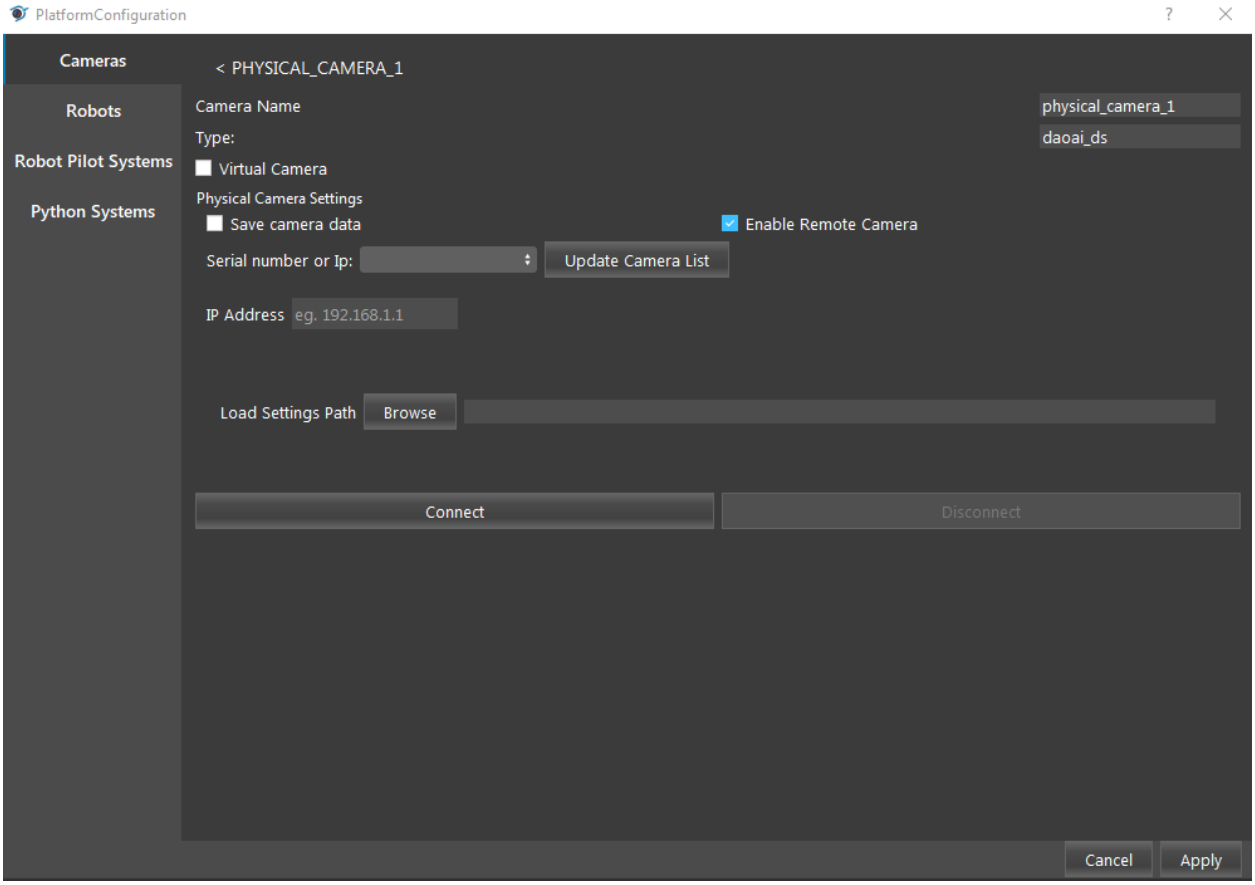
Select **Platform** → **Platform Configuration** in the tool bar.



In the **Camera** tab, click **+** to start adding a new camera, select the appropriate camera model from the dropdown list, then click **Accept** to add the camera.



If it is a remote camera, select the **Enable Remote Camera** checkbox and enter its IP address. Click **Update Camera List**, then select the Camera from the list. Click Connect to connect the camera.



Note: For multiple Remote Cameras, make sure you have configured your cameras' IP addresses (see *Network Configuration*), otherwise you may have issue finding or connecting the cameras.

CHAPTER

THIRTEEN

HAND-EYE CALIBRATION

13.1 Introduction

Hand-eye calibration is used to relate what the camera (“eye”) sees to where the robot arm (“hand”) moves.

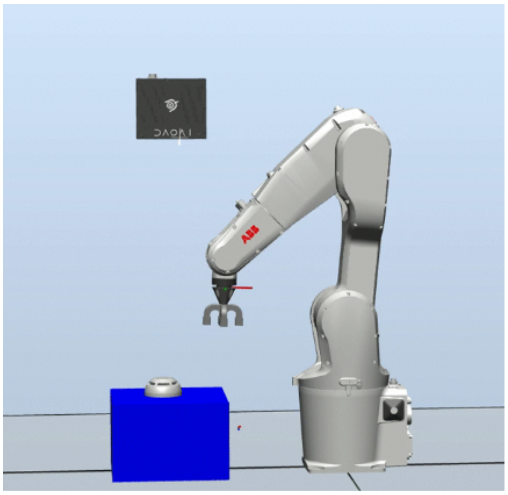
Eye-in-hand calibration is a process for determining the relative position and orientation of a robot-mounted camera with respect to the robot’s end-effector. It is usually done by capturing a set of images of a static object of known geometry with the robot arm located in a set of different positions and orientations.

Eye-to-hand calibration is a process for determining the position and orientation of a statically mounted camera with respect to the robot’s base frame. It is usually done by placing an object of known geometry in the robot’s gripper. Followed by taking a series of images of it in a set of different positions and orientations.

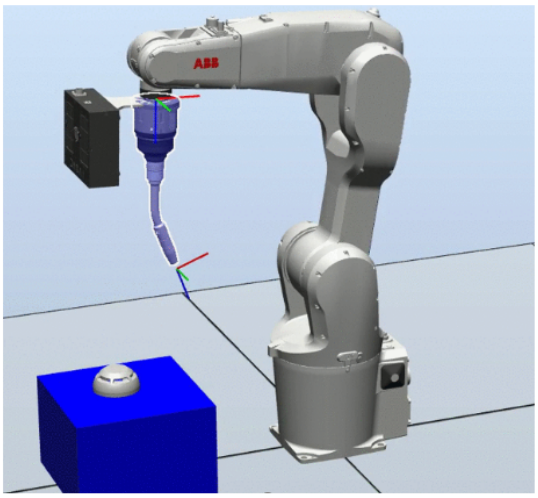
13.2 Further Reading

13.2.1 System Configuration

When it comes to the location of the camera in a vision-guided robotic system, there are two distinct standard configurations, “eye-to-hand” and “eye-in-hand”.



Eye-to-hand



Eye-in-hand

Eye-to-hand

Systems with this configuration have a stationary camera located in the robot's environment, oriented for observing the workspace of the robot arm.

The advantage of this system is that the camera is relatively distant from the robot's picking work area, so it is not likely that the robot will collide with it. Another benefit of this system configuration is that the image acquisition can be performed at the same time while the robot is moving, which implies potentially lower cycle times.

The downside is that the robot arm may occlude the view of the objects by moving between them and the camera. This can then negatively affect the cycle times. To prevent it from happening, it is necessary to use smart motion planning and synchronize robot motion and image acquisition, which is not trivial. This system is also less flexible because the camera is static and has a constant viewpoint. This means that the FOV cannot be changed and the accuracy of image measurements cannot be increased when required. Finally, there is an extra effort in terms of building the robot cell as it requires a mechanical structure for a camera to be mounted on.

Eye-in-hand

This configuration assumes that the camera is rigidly mounted on the robot's end-effector and moves with the robot. Most commonly it is mechanically fixed to a flange or a tool.

The advantage of this system lies in its flexibility. If a larger FOV is required, the robot can pull back and do the imaging from afar. Imaging an object from multiple viewpoints allows a more complete 3D model (possibly a full 360-degree model). It can also change the viewpoint of the camera to look from a different angle. This can also help mitigate problems with specular reflections as these will typically appear in different regions when the viewpoint is changed. Another advantage of this system configuration is the absence of potential object occlusions caused by the robot.

One downside of this configuration is that collision avoidance must be implemented with careful control of the robot's accelerations. This is to avoid damaging the working environment, the camera, and the robot. Another disadvantage of robot mounting is that the camera may not be used for image acquisition when the robot is not in the right position. For instance when it is delivering a part that it has picked up. Also, the robot has to stop moving before grabbing a 3D image. Finally, the weight of the camera reduces the available payload for the gripper and the objects the robot picks.

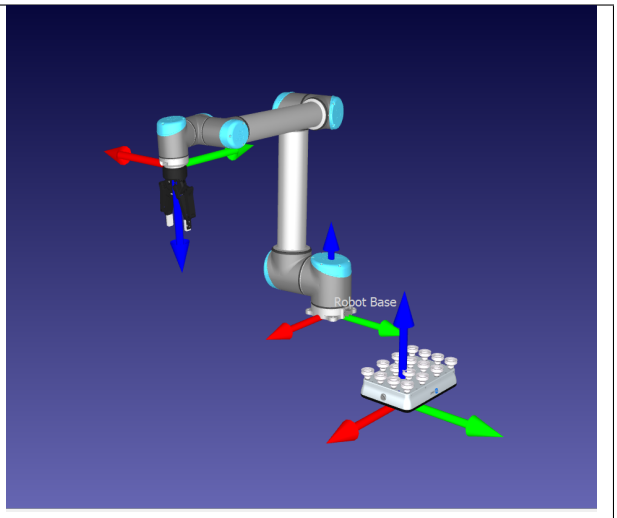
Now that we've introduced the system configurations, next we'll identify the *Hand-Eye Calibration Problem*.

13.2.2 Hand-Eye Calibration Problem

This tutorial aims to describe the problem that the hand-eye calibration solves as well as to introduce robot poses and coordinate systems that are required for the hand-eye calibration. The problem is the same for eye-to-hand systems and eye-in-hand systems. Therefore, we first provide a detailed description for the eye-to-hand configuration. Then, we point out the differences for the eye-in-hand configuration.

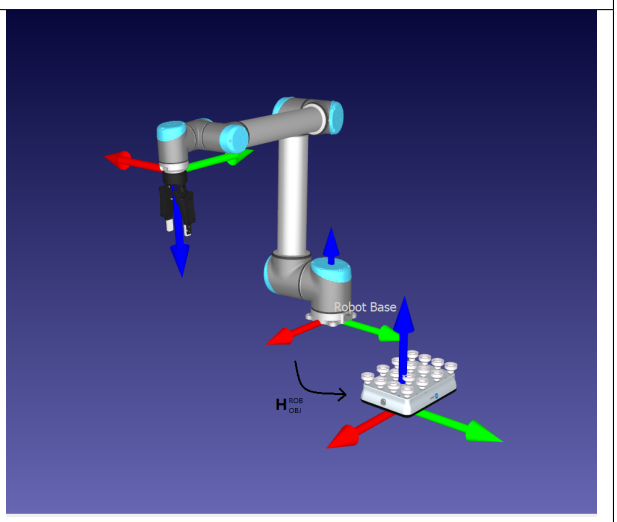
Eye-to-hand

How can a robot pick an object?

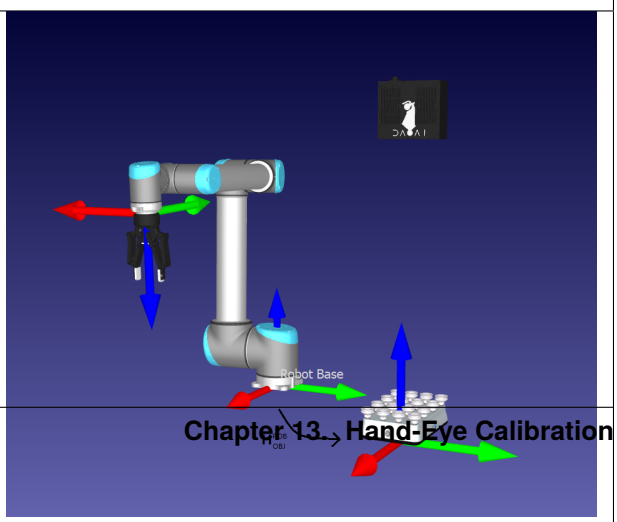


Let's start with a robot that doesn't involve a camera. Its two main coordinate systems are:

1. The robot base coordinate system
2. The end-effector coordinate system

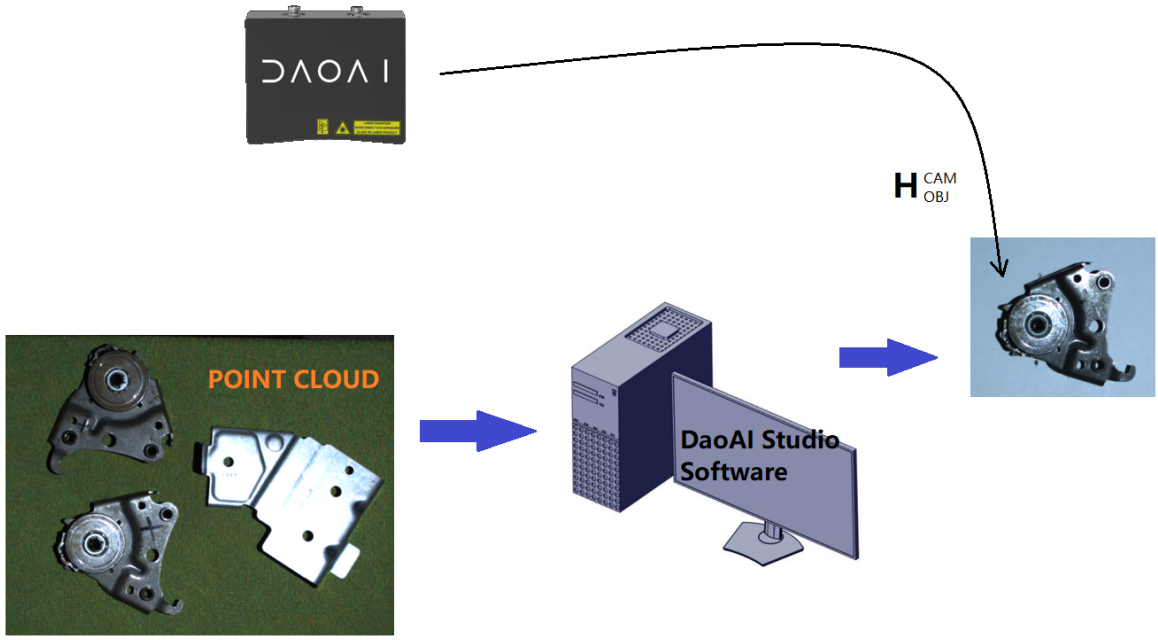


To be able to pick an object, the robot controller needs to know the object's pose (position and orientation) relative to the robot base frame. It also requires knowledge about the robot's geometry. This combined information is sufficient to compute the joint angles that will move the end-effector/gripper towards the object.

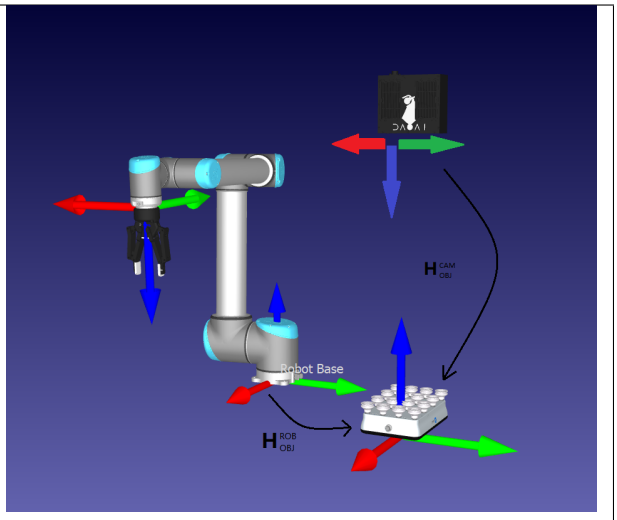


Now, let's assume that the pose of the object relative to the robot is unknown. That's where DaoAI 3D vision

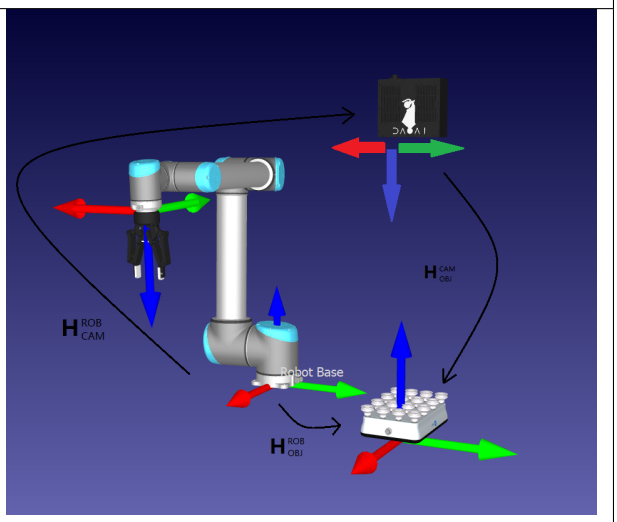
DaoAI point clouds are given relative to the DaoAI camera's coordinate system. The origin in this coordinate system is fixed at the middle of the DaoAI imager lens (internal 2D camera). A machine vision software can run detection and localization algorithms on this collection of data points. It can determine the pose of the object in DaoAI camera's coordinate system ().



DaoAI camera can now see the object in its field of view, but relative to its own coordinate system. To enable the robot to pick the object it is necessary to transform the object's coordinates from the camera coordinate system to the robot base coordinate system.

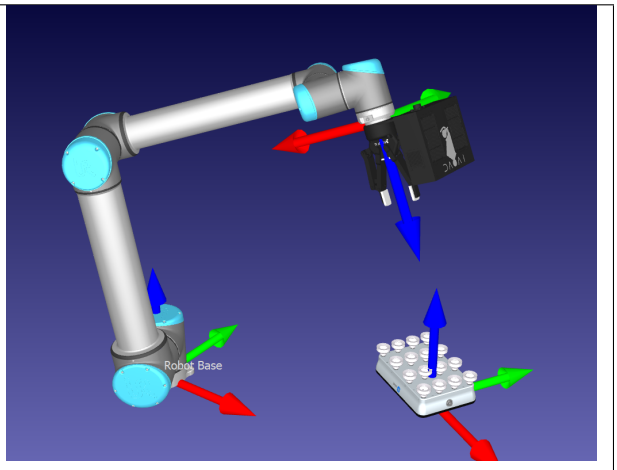
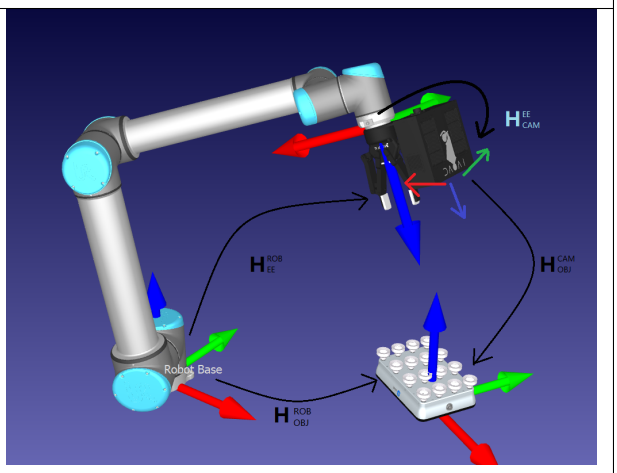


The coordinate transformation that enables this is the result of hand-eye calibration. For eye-to-hand systems, it
is the pose of the camera relative to the robot's base ( ) that is estimated with the hand-eye calibration.
Once the pose circle is closed, it is possible to calculate one pose from the other poses in the circle. In this case, the pose of the object relative to the robot. This is found by post-multiplying the pose of the camera relative to the robot, with the pose of the object relative to the camera:



Eye-in-hand

How can a robot pick an object?

DaoAI camera can now see the object in its field of view, but relative to its own coordinate system. To enable the robot to pick the object it is necessary to transform the object's coordinates from the camera coordinate system to the robot base coordinate system.	
In this case, the transformation is done indirectly: The pose of the end-effector relative to the base of the robot () is known, and is provided by the robot controller. The pose of the camera relative to the end-effector (), which is in this case constant, is estimated from the hand-eye calibration.	

Now that we've defined the hand-eye calibration problem, let's see *Hand-Eye Calibration Solution*.

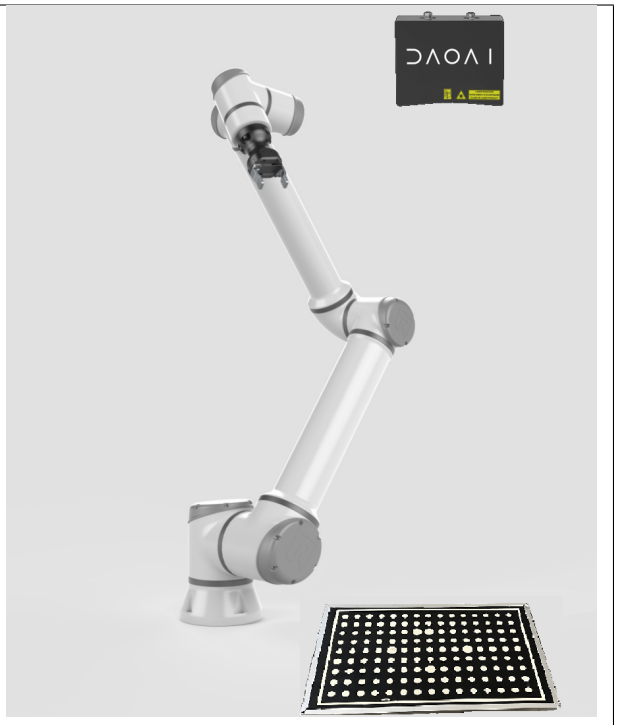
13.2.3 Hand-Eye Calibration Solution

The previous part of this tutorial presented the problem that the hand-eye calibration needs to solve. This tutorial describes the background idea for a solution. The core idea is the same for eye-to-hand systems and eye-in-hand systems. Therefore, we first provide a detailed solution for the eye-to-hand configuration. Then, we point out the differences in the eye-in-hand configuration.

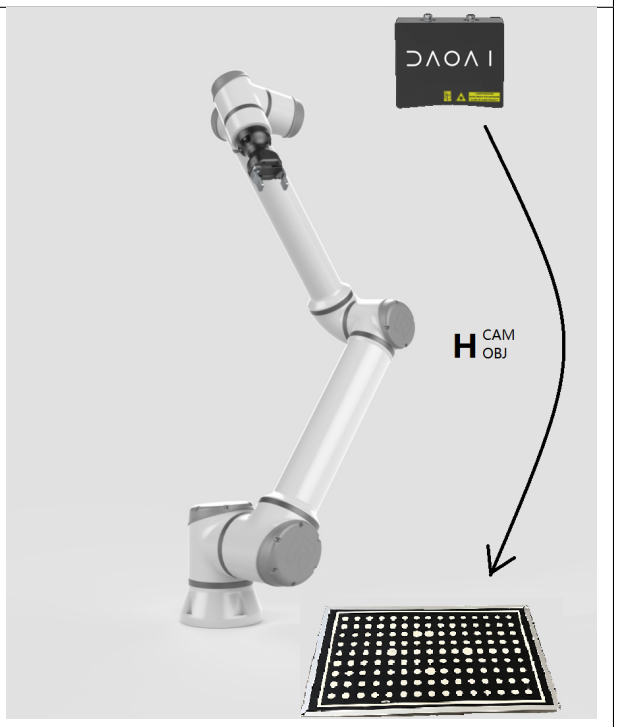
Note: You don't need a tool, or to know its pose (if you have one attached) to do the hand-eye calibration. The Tool Center Point (TCP) value does not affect the hand-eye calibration result. In this article and later tutorials, the end-effector refers to the tool flange/end-link.

Eye-to-hand

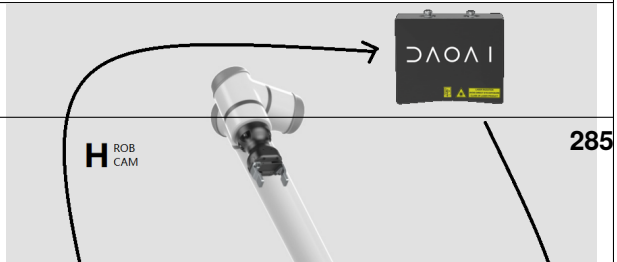
How to solve the eye-to-hand calibration?



The first step is choosing a calibration object, e.g. a checkerboard. DaoAI checkerboards will be covered in the
next part of this tutorial.



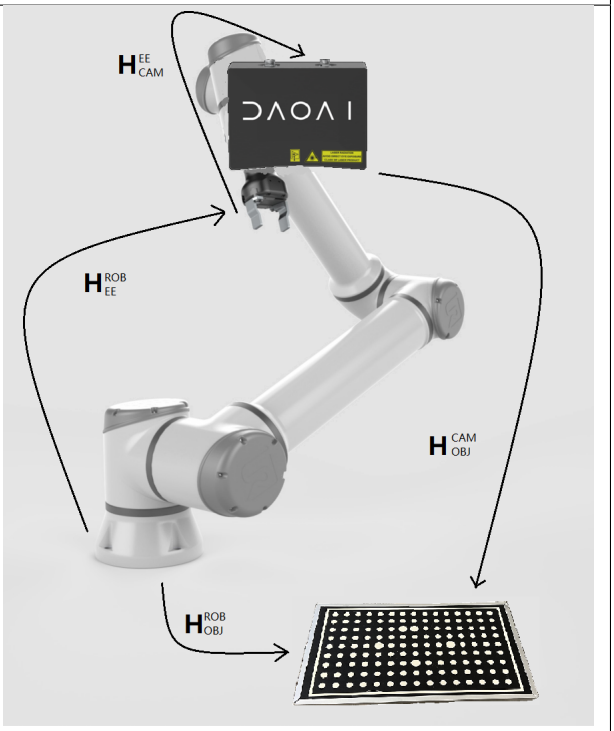
The calibration object is of known geometry. Thus, it can
be detected from the camera image. Further, its pose relative to the camera () can be estimated.



Eye-in-hand

How to solve the eye-in-hand calibration?

The situation is very similar for eye-in-hand systems. In this case the calibration object is fixed to the work environment. Thus, it is ensured that its pose relative to the robot base is constant during the robot motion.



This allows us to express the pose of the camera relative to the end-effector () as a function of two variable, known poses:

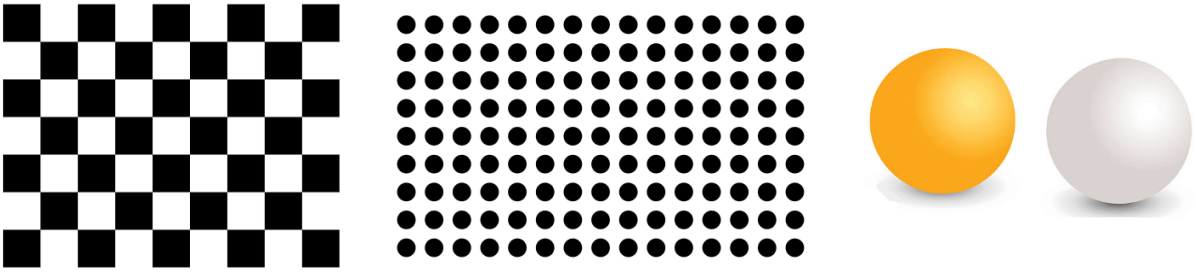
1. Robot to end-effector
2. Camera to calibration object and one constant, unknown pose .

Just as in the eye-to-hand configuration case, we can solve for

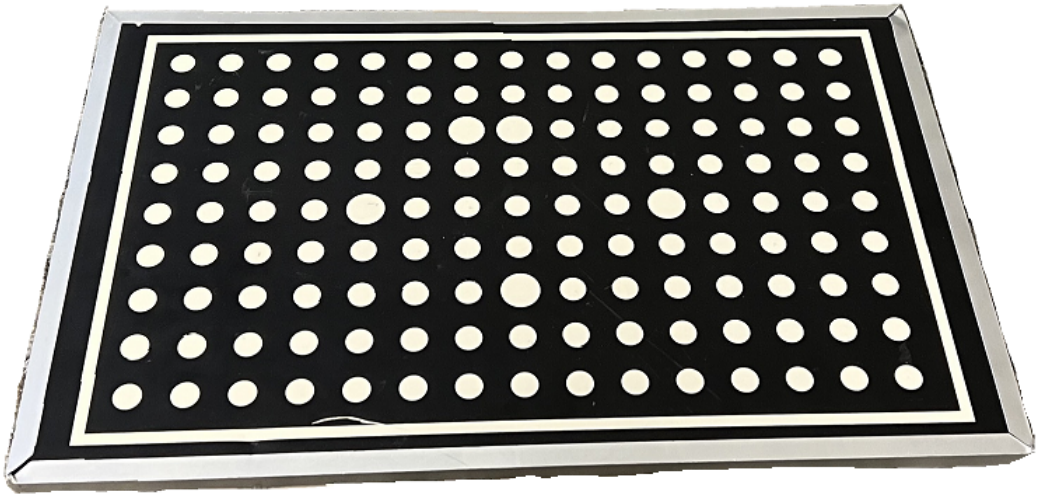
Now that we've explained how to solve the hand-eye calibration problem, let's see learn about *Calibration Object*.

13.2.4 Calibration Object

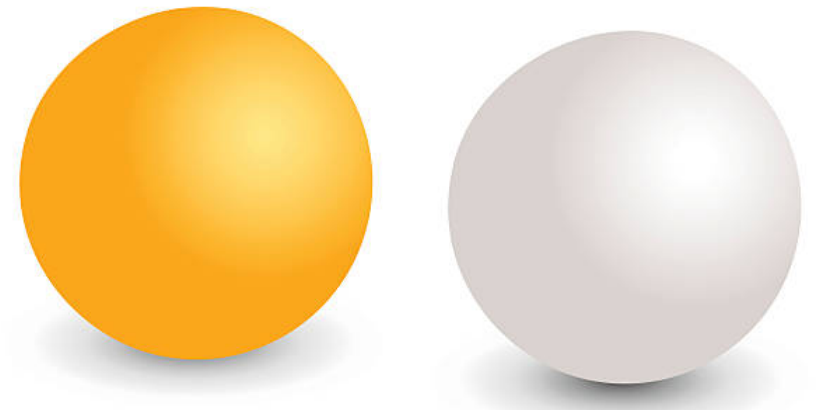
The hand-eye calibration process requires using an object of known geometry that can be detected and localized from the camera image. 2D patterns are most commonly used as calibration objects. OpenCV libraries use a checkerboard, while HALCON software uses a dot pattern. Having a 3D camera enables using a 3D calibration object; an example is spheres.



DaoAI uses a circleboard for the hand-eye calibration. The main DaoAI calibration objects can be found on DaoAI product box . These boards are also used for camera maintenance with infield correction. Each of these circleboards has a 9x15 checkerboard made of 24 mm checkers and a fiducial marker in the center. For these boards to be used in hand-eye calibration, both the circleboard and fiducial marker need to be detectable in each capture.



If the DaoAI calibration object is unavailable, DaoAI hand-eye calibration will also function, it is possible to buy some spherical objects, such as ping pong balls, by the way , the mesh model of the spheres need to be prepared, and the size of the mesh object must be the same as the purchased object.



Continue reading about *How To Get Good Quality Data On DaoAI Calibration Board* .

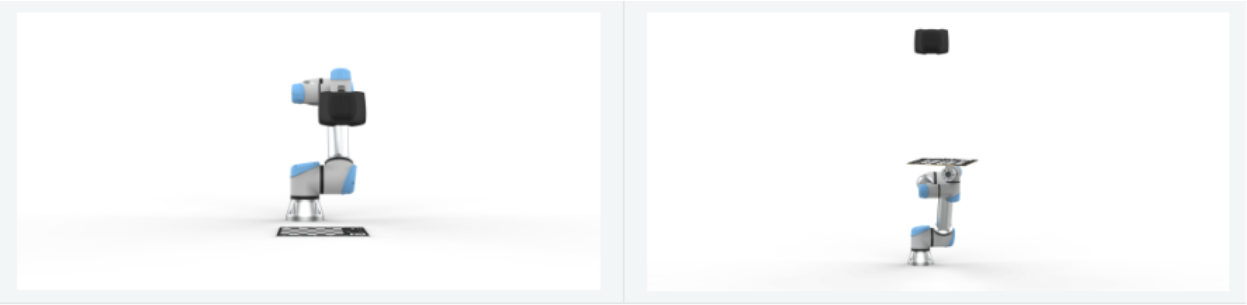
13.2.5 How To Get Good Quality Data On DaoAI Calibration Board

This tutorial aims to present how to acquire good quality point clouds of DaoAI checkerboard for hand-eye calibration. It is a crucial step to get the hand-eye calibration algorithm to work as well as to achieve the desired accuracy. The goal is to configure a DaoAI HDR setting that gives high-quality point clouds regardless of where the DaoAI checkerboard is seen in the working space.

It is assumed that you have already specified the robot poses at which you want to take images of the DaoAI checkerboard. Check out how to select appropriate poses for hand-eye calibration.

Note: When using DaoAI calibration board, the entire calibration board, including fiducial marker, must be visible for each pose.

We will talk about two specific poses, the 'near' pose, and the 'far' pose. The 'near' pose is the robot pose where the imaging distance between the camera and the checkerboard is the least. For eye-in hand systems, that is the pose where the robot mounted camera approaches closest to the checkerboard. For eye-to-hand systems, it is the one where the robot positions the checkerboard closest to the stationary camera.

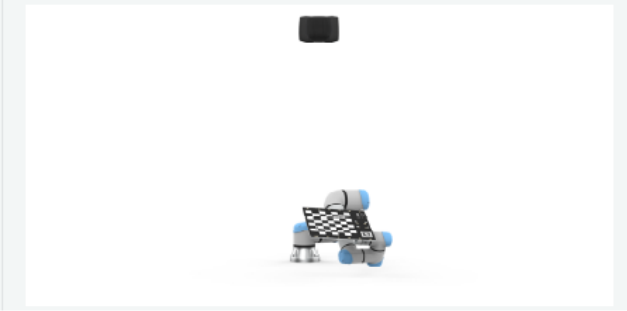


Eye-in-hand robot pose for a close capture of DaoAI calibration board

Eye-to-hand robot pose for a close capture of DaoAI calibration board

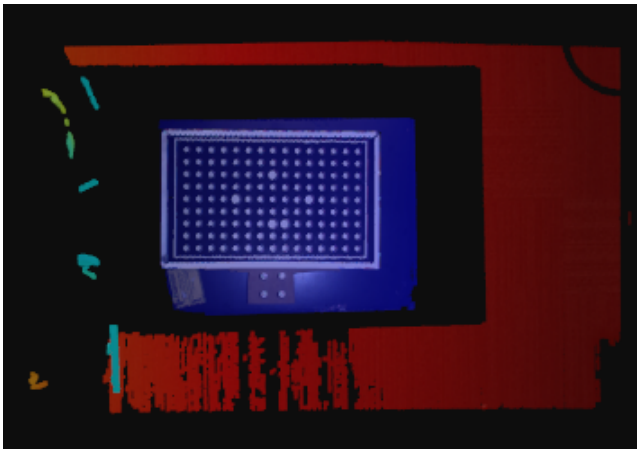
The 'far' pose is the robot pose where the imaging distance between the camera and the checkerboard is the largest. For eye-in hand systems, that is the pose where the robot mounted camera pulls furthest away from the checkerboard. For eye-to-hand systems, it is the one where the robot positions the checkerboard furthest away from the stationary camera.

Eye-in-hand robot pose for a close capture of DaoAI calibration board



Eye-to-hand robot pose for a close capture of DaoAI calibration board

The expected result is illustrated below:



Read *Camera Settings* to get the default setting of different DaoAI Camera.

Let's see how to realize the *Hand-Eye Calibration Process* .

13.2.6 Hand-Eye Calibration Process

To gather the data required to perform the calibration involves a robot making a series of planned movements (10 to 20 are recommended), either human-operated or automatically. At the end of each movement, the camera takes an image of the calibration object. The calibration object pose is extracted from the image, and the robot pose is registered from the controller. To achieve good calibration quality, the robot poses used when the camera takes images of the calibration object should be:

- sufficiently distinct
- using all the robot joints

This results in a diversity of perspectives with different viewing angles. The images below illustrate the required diversity of imaging poses for eye-to-hand and eye-in-hand systems. At the same time, the calibration object should be fully visible in the field of view of the camera.

The task is then to solve homogeneous transformation equations to estimate the rotational and translational components of the locations of the calibration object and those of the hand-eye transformation.

Hand-eye calibration process steps:

Tip: It is recommended to Warm-up the camera and run Infield Correction before running hand-eye calibration. Use

the same capture cycle during warmup, infield correction, and hand-eye calibration as in your application. To further reduce the impact of temperature dependent performance factors, enable Thermal Stabilization.

1. Move the robot to a new posture
2. Register the end-effector pose
3. Image the calibration object (obtain its pose)
4. Repeat steps 1-3 multiple times, e.g. 10 - 20
5. Compute hand-eye transform

13.2.7 Cautions And Recommendations

Image quality

It is important that the camera images of the calibration object are well exposed and that they are in focus. To ensure the calibration object is within the depth of field it is recommended to use higher factors-number values and compensate with larger exposure times. Also, the same aperture setting should be used for all the images to avoid unnecessary fluctuations due to different aperture sizes. It is important not to choose similar views for the hand-eye calibration images, but excite all six degrees of freedom of the calibration object and that of the robot.

DaoAI API

DaoAI point clouds are given in mm. Translation part of the Hand-Eye output pose in DaoAI API is also in mm. Therefore, the translation part of the input robot poses for Hand-Eye must also be in mm.

Detectable fiducial marker

If the calibration object is a DaoAI calibration board ensure that the fiducial/ArUco marker is visible for all poses. Otherwise, the algorithm will fail when it is trying to detect the checkerboard.

Eye-to-hand

During the eye-to-hand calibration process, the calibration object is mounted on the robot end-effector and moves with the robot. It can be fixed directly to a flange or held by a gripper. The exact location for mounting is not important because the relative pose between the calibration object and the end-effector does not have to be known. It is important that the calibration object does not move relative to the flange or the gripper during the motion; it has to be is fixed or held tightly. It is recommended that mounting brackets, as well as the calibration plate itself, are made of rigid materials. It is also crucial that the calibration object is still during image acquisition. A good routine is to wait a couple of seconds after the robot moves with the calibration object. This allows the whole structure to stabilize in case there are post-motion vibrations. Robot motion should be smooth with controlled accelerations to avoid shaking and dislocation of the calibration object.

Eye-in-hand

During the eye-in-hand calibration process, the calibration object is stationary, placed in the robot's workspace where the end-effector mounted camera can see it from different perspectives. The exact location of the calibration object is not important because it is not necessary to know its pose relative to the robot's base. However, the calibration object must not move during the calibration and should, therefore, be tightly fixed.

Environment conditions

Temperature changes have some impact on performance. Hence it is recommended to ensure that the temperature during the hand-eye calibration is somewhat stable. This can be done with Warm-up procedure. It is a good idea to keep it similar to the working conditions when the system is going to be set in operation. To further reduce the impact of temperature dependent performance factors, enable Thermal Stabilization.

Choosing the correct method

Some hand-eye calibration methods compute camera intrinsic parameters along with extrinsic parameters and the relative pose between the camera and the robot frame. We do not recommend these approaches because they treat a DaoAI camera as an uncalibrated 2D camera, rather than a well-calibrated 3D camera.

Each DaoAI camera unit goes through an extensive calibration process, which includes determining the intrinsic parameters of its 2D color camera. Our calibration uses a complex camera model with more intrinsic parameters than some well-known pinhole camera models, e.g., OpenCV camera model. Since DaoAI camera model is proprietary, our internal camera intrinsics are not available in the SDK. However, DaoAI SDK does offer approximations of OpenCV and Halcon models (see Camera Intrinsics) from our camera model. Because information is lost in approximation, using hand-eye calibration methods that utilize OpenCV or Halcon intrinsics is not the best approach either.

Since DaoAI cameras provide 3D data, it is possible to calculate camera extrinsics from the point cloud. DaoAI Hand-Eye Calibration method utilizes this benefit and that is why it is the recommended approach and the one that works best with our cameras. There are alternative, non-DaoAI methods, that utilize the possibility to calculate camera extrinsics from the point cloud. These methods rely purely on point cloud data and an example is hand-eye calibration based on CAD matching.

Accuracy and recalibration

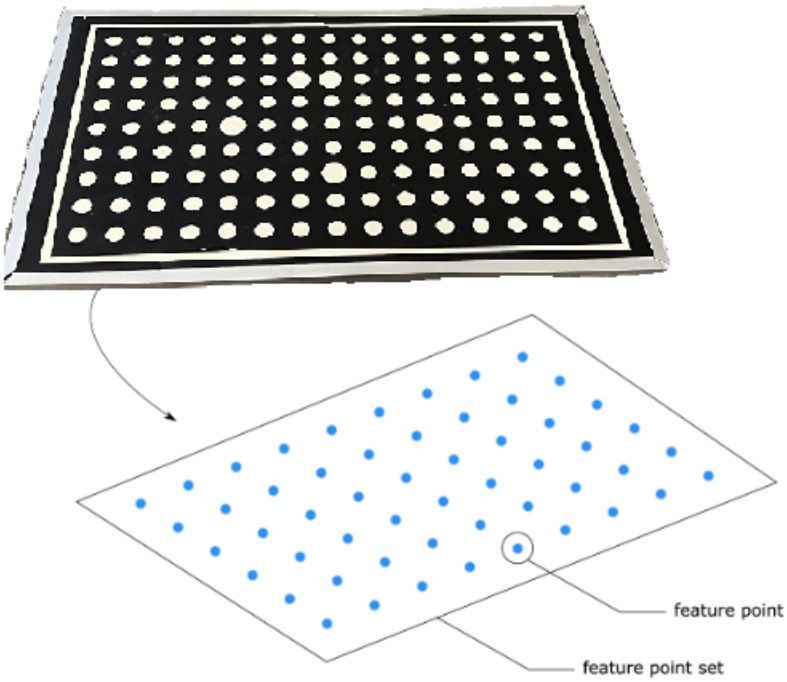
The picking accuracy of a vision-guided robotic system depends on the combined accuracy of the camera, hand-eye calibration, machine vision software, and robot's positioning. Robots are in general highly repeatable but not accurate. Temperature, joint friction, payload, and manufacturing tolerances are some of the factors that cause the robot to deviate from its preprogrammed positioning. However, robot pose accuracy can be improved by calibrating the robot itself, which is highly recommended for complex systems with multiple factors that affect the picking accuracy. If the robot loses the calibration, the picking accuracy will deteriorate. Repeating the calibration (robot and/or hand-eye) can compensate for such deteriorated performance. It is also necessary to repeat the hand-eye calibration after dismounting the camera from a fixed structure or a robot and mounting it back on.

Read more about *Hand-Eye Calibration Residuals*.

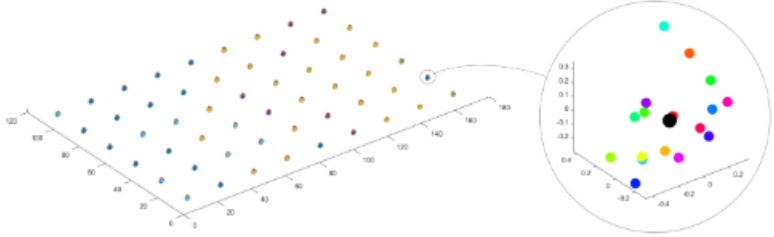
13.2.8 Hand-Eye Calibration Residuals

In order to evaluate the performance of the hand-eye calibration, we need a method to check the residuals. Here we explain what hand-eye calibration residuals represent and how they are calculated.

For each checkerboard point cloud in the dataset, DaoAI software extracts a certain number of feature points. We shall refer to this collection of feature points as a feature point set.



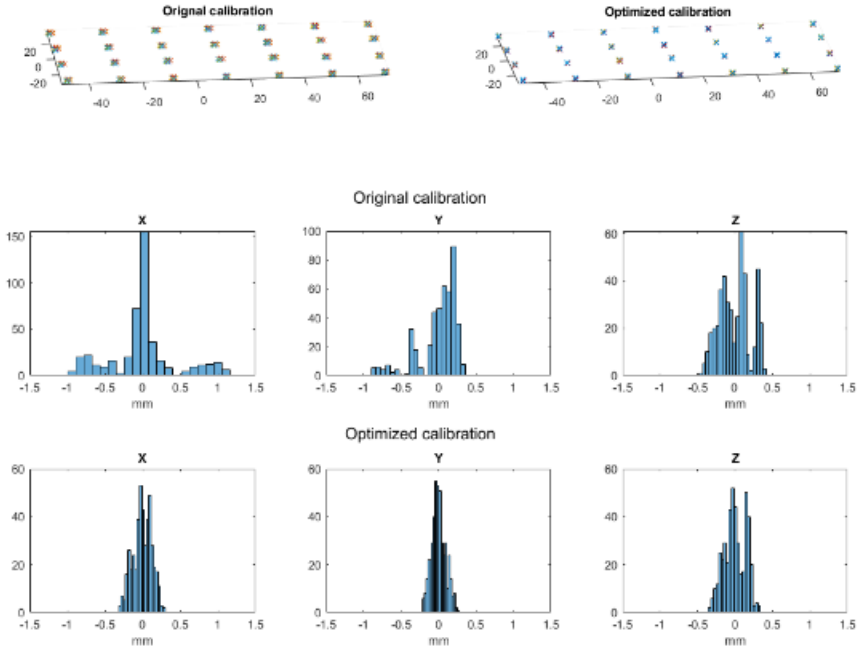
With the result of hand-eye calibration, it is possible to compute a coordinate transformation. This coordinate transformation can convert the feature point sets from the camera coordinate frame to the robot base frame. Assume that each element of the robotic system, i.e. camera, robot, and hand-eye calibration algorithm was perfect. Then feature points from one transformed set would have the same coordinates as their counterparts from the other sets in the dataset. Visually, this means that all the feature point sets in 3D space would overlap. This is never the case in reality and there are always some residuals. Visually, this means that the same feature points from different sets don't fully overlap. This can be seen from the image below, which is the visualization of one of the in-house hand-eye calibration experiments.



We will now explain how DaoAI software calculates the residuals.

With all feature point sets from the dataset, a set of reference feature points is found that represents the arithmetic mean of all other feature point sets. This means that each feature point of the reference set has coordinates so that the sum

of Euclidean distances from its counterpart feature point, from the other sets, is minimized. This is represented by the black enlarged sphere in the zoomed in view. DaoAI software then estimates the pose of each feature point set, including the reference set. Finally, translational and rotational residuals are calculated as the relative position and orientation between the reference feature point set and all other feature point sets. The translational residual is given as the Euclidean distance between the reference frames that represent the two feature point sets. The rotational residual is given as the angle of the angle-axis representation between the two reference frames.



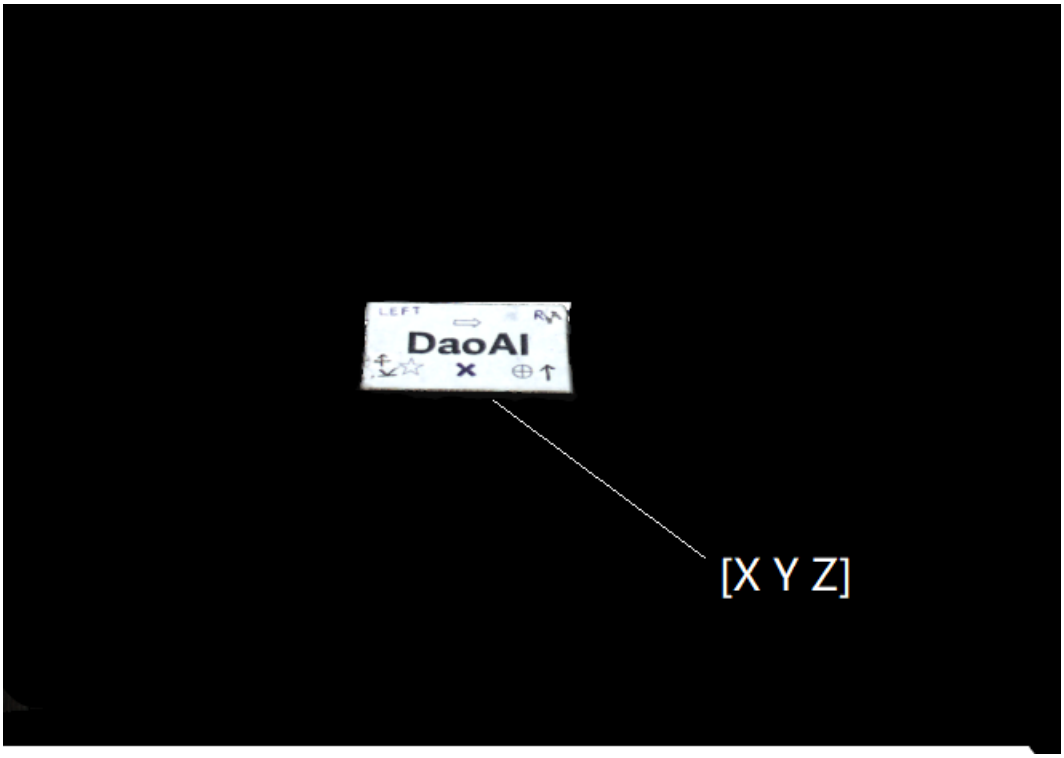
Continue reading on *How To Use The Result Of Hand-Eye Calibration*.

13.2.9 How To Use The Result Of Hand-Eye Calibration

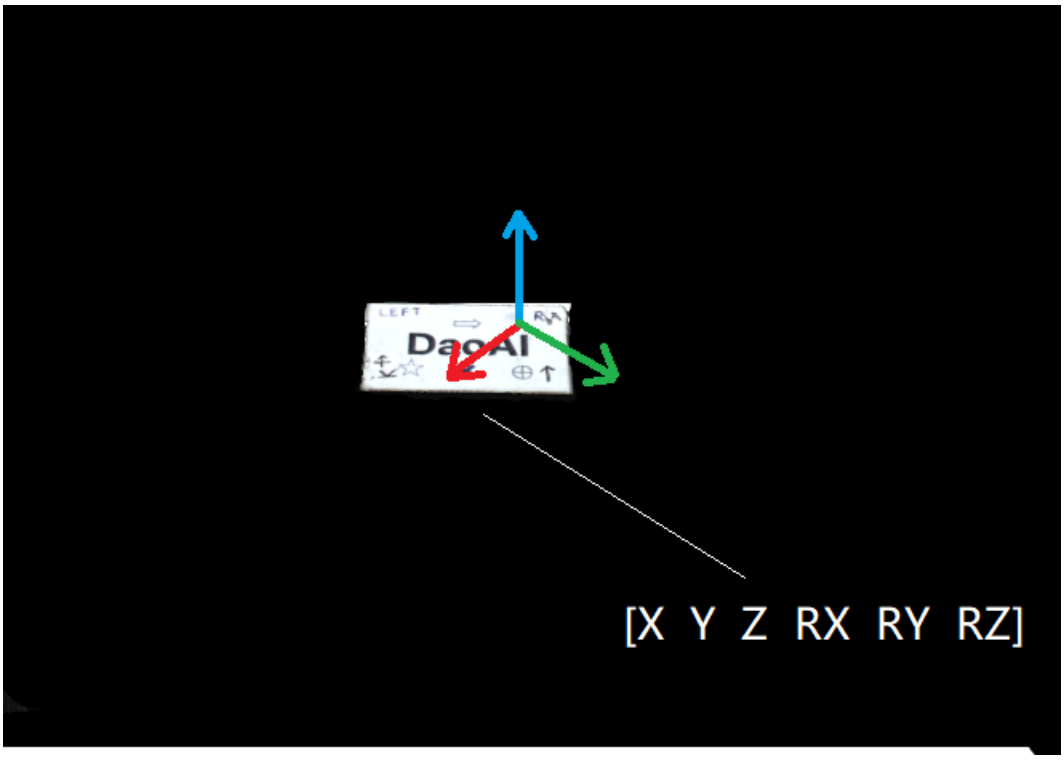
We have analyzed the requirements for a robot that is accompanied by a DaoAI camera and a machine vision software to pick an object. This led to the description of the Hand-Eye Calibration Problem. If you are unsure how to utilize the result of the hand-eye calibration, you are on the right page. This is where we describe how to transform the object's coordinates from the DaoAI camera coordinate system to the robot base coordinate system.

Let's suppose you run machine vision software on a DaoAI point cloud. It detects the object of interest, such as this DaoAI gem, and estimates its position. The x, y, z values describing the picking point are given relative to the DaoAI camera's coordinate system.

Tip: Before running your application it is recommended to Warm-up the camera using the same capture cycle as for hand-eye calibration. To further reduce the impact of temperature dependent performance factors, enable Thermal Stabilization.



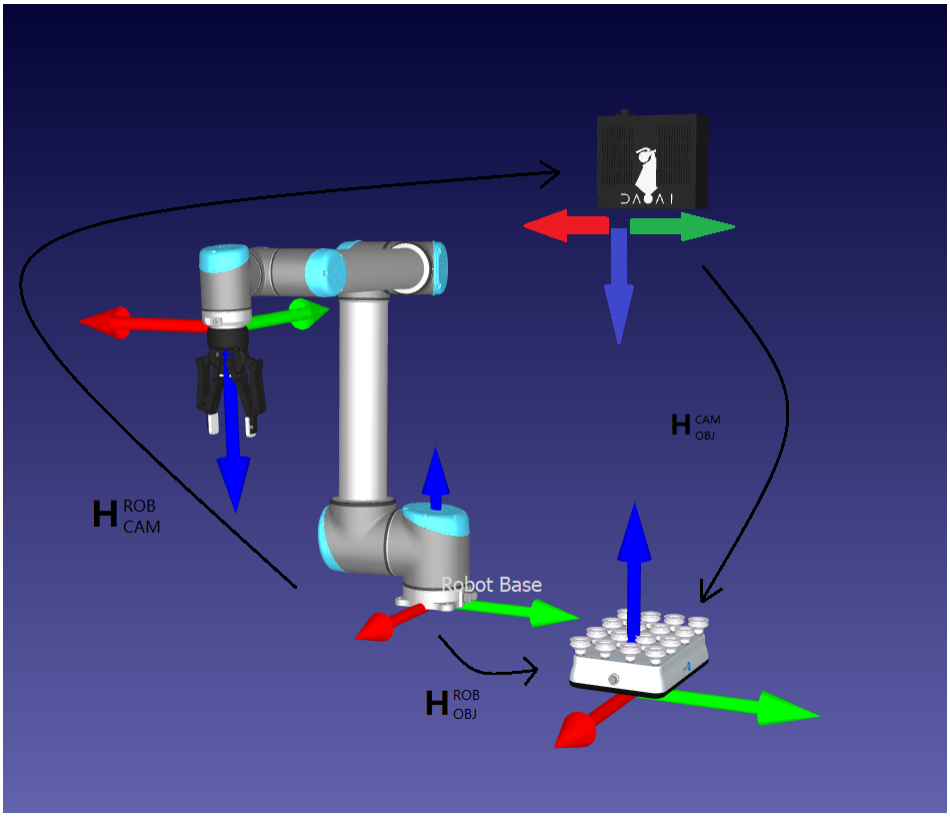
In some cases, your algorithm will also output the object's orientation, e.g. the roll, pitch, and yaw angles. These parameters are also given relative to the DaoAI camera's coordinate system.



The pose (position and orientation) of your object can be described with a homogeneous transformation matrix. If you are not familiar with (robot) poses and coordinate systems.

Below you will see the mathematical theory of transform a single point or an entire point cloud from the camera coordinates to the robot base coordinates. In practice, the easiest way of doing this is to use the DaoAI SDK supported transformation. This transforms the data before it is copied on the CPU and is therefore very fast.

Eye-to-hand



If you are dealing with an eye-to-hand system, this is how a single 3D point can be transformed from the DaoAI camera to the robot base coordinate system:

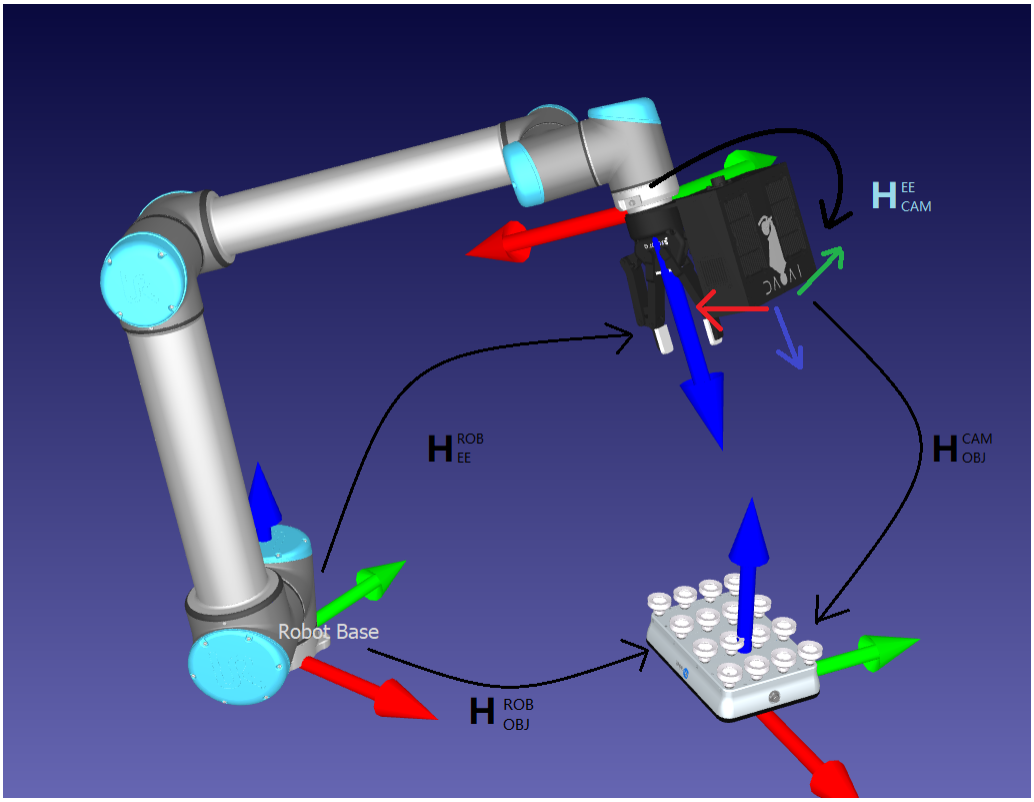
To convert the whole DaoAI point cloud, from the camera coordinate system to the robot base coordinate system, apply the equation above to each point in the point cloud.

On the other hand, to transform the pose of the object relative to the DaoAI camera, apply the following equation:

We assume that your pose is described with a homogeneous transformation matrix.

The resulting pose is the one that the robot Tool Center Point (TCP) should attain for picking. The offset between the TCP and the robot's flange should be accounted for on the robot side.

Eye-in-hand



The approach for eye-in-hand systems is similar. The difference is that the current pose of the robot has to be included in the equations. As with the other poses, we assume that the robot pose is represented with a homogeneous transformation matrix.

The following equation describes how to transform a single 3D point from the DaoAI camera to the robot base coordinate system:

To convert the whole DaoAI point cloud from the camera coordinate system to the robot base coordinate system, apply the equation above to each point in the point cloud.

To transform the pose of the object relative to the DaoAI camera use the following equation:

The resulting pose is the one that the robot Tool Center Point (TCP) should attain for picking. The offset between the TCP and the robot's flange should be accounted for on the robot side.

13.3 Version History

Coming later.....

CHAPTER

FOURTEEN

CAMERA SETTINGS

The table below shows the default settings for different DaoAI camera models.

14.1 Default Settings

The table below shows the default settings for different DaoAI camera models.

Models	BP AMR	BP AMR-GPU	BP-S	BP-M	BP-L
Exposure Time (us)	7500	7500	7500	10000	10000
Gain	0	0	0	0	0
Projector Brightness	3	3	3	3	3
Intensity Filter	Yes	Yes	Yes	Yes	Yes
Intensity Filter Threshold	20	20	20	20	20
Outlier Filter	Yes	Yes	Yes	Yes	Yes
Outlier Threshold (mm)	3	3	3	3	3
Phase Quality Filter	Yes	Yes	Yes	Yes	Yes
Phase Quality Filter Threshold	0.1	0.1	0.1	0.1	0.1
Gaussian Filter	No	No	No	No	No
Median Filter	Yes	Yes	Yes	Yes	Yes
Median Filter Size	5	5	5	5	5
Face Normal Filter	Yes	Yes	Yes	Yes	Yes
Face Normal Filter Threshold	20	20	20	20	20
Remove Small Area	Yes	Yes	Yes	Yes	Yes
Remove Small Area Threshold	2	2	2	2	2
Smooth Filter	No	No	No	No	No
Fill Gaps	No	No	No	No	No
Saturation Filter	No	No	No	No	No
Contrast Distortion Filter	No	No	No	No	No
Red Balance	1	1	1	1	1
Green Balance	1	1	1	1	1
Blue Balance	1	1	1	1	1

CHAPTER

FIFTEEN

CALCULATOR

15.1 Calculate Depth of View

The depth of field (DOF) is the range within which the object can move away from or towards the camera without resulting in a blurred image. The calculator below estimates the depth of field, which can guide you to position your camera and object.

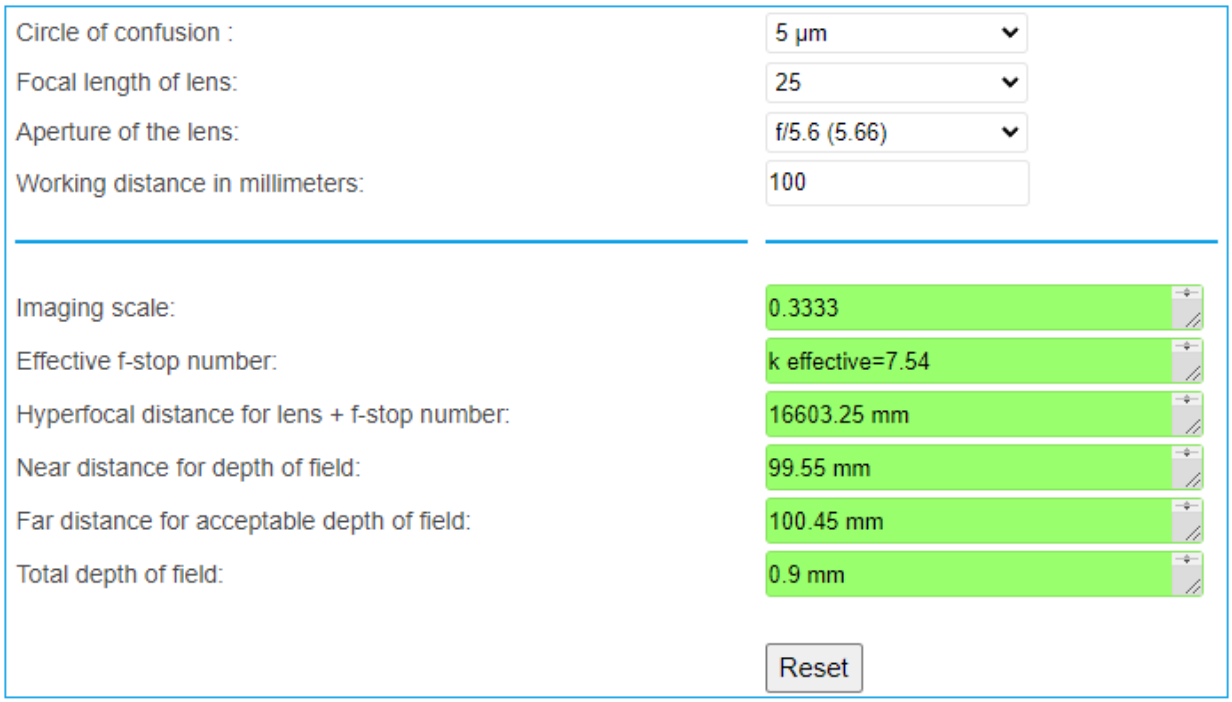
15.1.1 Inputs

- Circle of Confusion (in micrometers, m)
- Focal length of lens (in millimeters, mm)
- aperture of the lens
- Working distance in millimeters

15.1.2 How to Use the Calculator

Goto the Calculator Page

Enter values for all input fields, and the result will be displayed as the following image.



You may refer to camera defualt settings for your inputs.

Models	BP AMR	BP AMR-GPU	BP-S	BP-M	BP-L
Circle of Confusion	3.45	3.45	5.86	5.86	5.86
Focal Length of Lens	6	6	12.5	16	12.5
Aperture of the Lens	f/5.6	f/5.6	f/5.6	f/5.6	f/5.6
Working Distance (mm)	500-1000	500-1000	500-1000	800-1800	1000-3000

15.2 Camera Selection Tool

The following html page will help you figure out which camera is suitable for your need.

Camera Selection Tool

15.3 FOV Calculator

Using our online calculators you can determine which DaoAI camera best fits your application in terms of the FOV and distance of imaging.

The FOV calculator outputs the size of the FOV (width and height) and spatial resolution for a given DaoAI camera model and the imaging working distance.

CHAPTER

SIXTEEN

HARDWARE

This section provides articles releated to the DaoAI hardware.

16.1 Cleaning Instructions

Stains on the optical glass parts can interfere with the light signal passing through and thus affect the point cloud quality. Therefore, check and maintain the optical glass parts regularly.

Caution: Do not touch the glass parts with fingers to avoid staining.

If the glass parts are stained, follow the instructions for cleaning optical components:

1. Remove as much dust and dirt as possible with a blower or soft-bristled brush.
2. Apply a few drops of lens cleaning solution to a lens tissue (lint-free wipes) or cleaning cloth.
3. Using a circular motion, gently remove grease, fingerprints, and grime from the surface, working from the center outward.

16.2 Ethernet cables

What is a Cat6 Ethernet Cable?

Cat6 Ethernet cables consist of four twisted pairs of copper wire and 250 MHz of bandwidth, supporting data transfer speeds of up to 10 Gbps (10GBASE-T) for distances up to approximately 180 feet.

Cat6 cables use the same RJ-45 jack as Cat5 cables and previous generations of Ethernet cables. In fact, Cat6 cables have backwards compatibility with Cat5/5e and Cat3 cables.

When used at 328 feet, the maximum data transfer speed drops to approximately 1 Gbps. They are up-to-spec for applications with substantial data transfer needs, including: Internet of Things (IoT) setups like smart homes, school/enterprise networking setups and data centers. Cat6 cables are now the global standard for Ethernet cables.

What is a Cat6a Ethernet Cable?

Cat6a (the "a" stands for "augmented") cables have thicker, heavier construction than standard Cat6 cables, and individual pairs may also have metal shielding to reduce interference even further. Cat6a cables support 10 Gbps internet up to 328 feet at a maximum bandwidth of 500MHz, double the bandwidth of Cat6. These cables have stringent cable termination requirements and must comply with ANSI/TIA-568 standards. Cat6a have backwards compatibility with Cat6 and Cat6e cables due to their shared RJ-45 jack.

Cat6a cables shine when it comes to applications outside data and telephony; namely, automation and physical security systems like access control and CCTV. They're commonly used in networks with heavy data use but not necessarily needing the more expensive fiber optic cables, including healthcare and higher education.

What are the Advantages of a Cat6 Ethernet Cable? Compared to Cat5/5e cables, Cat6 cables have stricter performance specifications and significantly higher data transfer speeds at greater distances. They are more tightly wound than Cat5 cables, and the cable conductors and cable sheath are thicker as well.

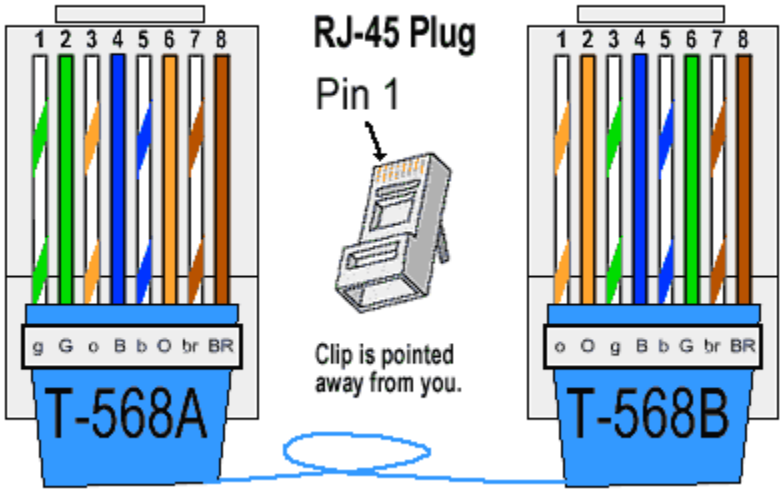
These factors work in tandem to reduce interior and exterior signal/EMI interference to a greater extent than Cat5 cables. This could be a great solution for networking in industrial settings where motors, generators or wireless devices could be causing significant signal interference.

What are the Disadvantages of a Cat6 Ethernet Cable?

Cat6 cables are more expensive than Cat5 cables (usually about 10-20% more than Cat5e) and tend to be more than what most homes need today. However, that doesn't mean they won't be needed 5-10 years from now as connected IoT homes become more commonplace.

For networks transferring terabytes of data or experiencing excess signal noise, Cat6 cables are the way to go. If you want a cable with optimal performance and have the cash and want to futureproof your IT infrastructure, go Cat6. Keep in mind if space is limited, the additional thickness and insulation of Cat6 cables may be of concern: the additional stiffness/thickness also makes the cable less flexible and harder to work with. Cat6a cables are around 40-50% thicker and heavier than Cat6 and even more expensive.

RJ45 Pinout Ethernet Cables



RJ45 Ethernet Cable Pinout

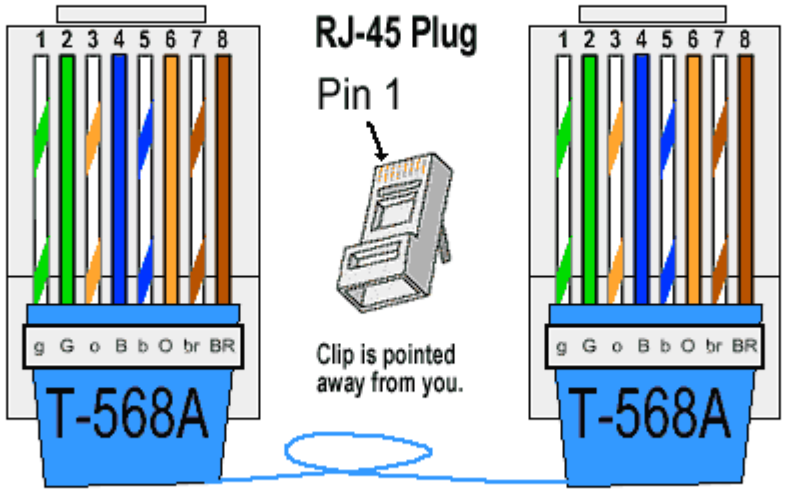
Ethernet LAN cables can come in two types - Crossover or Straight through. Most modern communications equipment can auto-sense which type you are using, but some still need the correct cable pinout. The following are the pinouts for the RJ45 connectors so you can check which one you have or make up your own. It doesn't matter if you make up some Cat5e, Cat6, and Cat7 cables. The pinout is always the same for Ethernet cables.

Straight Through LAN Cable Pinout

Straight through LAN cables are the most common, and the pinout is the same if they are Cat5e, Cat6, or Cat 7. The different types of cables (category or cat) offer increasingly faster transmit and receive speeds. This is achieved by increasing the wire twists, better shielding, drain wire, and increased diameter.

There are two different pinout standards used worldwide, and depending on your location, you will determine which one you should use. An easy way to remember the two different RJ45 connector pinouts is T568A is used in America and Asia, and T568B is used in Britain(UK) and Europe. The different pinouts will still work if you mix them up.

- T-568**A** – Most commonly used in the US**A** and **Asia** – Think A for America
- T-568**B** – **B**ritain (UK) and Europe



Ethernet Cable Pinout T568A and T568B

RJ45 Pinout for a LAN Cable

- Pin 1 → White and Green (Transmit +) wire
- Pin 2 → Green (Transmit -) wire
- Pin 3 → White and Orange (Receive +) wire
- Pin 4 → Blue wire
- Pin 5 → White and Blue wire
- Pin 6 → Orange (Receive -) wire
- Pin 7 → White and Brown wire
- Pin 8 → Brown wire

Straight LAN cable					
RJ45		**Color**		**RJ45**	
Function	*Pin*	*Color*	*Pair*	*Pin*	*Function*
TX+	1	white/green	3	1	RX+
TX-	2	Green	3	2	RX-
RX+	3	white/orange	2	3	TX+
	4	Blue	1	4	
	5	white/blue	1	5	
RX-	6	Orange	2	6	TX-
	7	white/brown	4	7	
	8	Brown	4	8	

RJ45 Ethernet Cable Pinout

How to Crimp an RJ45 Ethernet Cable Follow these steps to make sure you make the perfectly crimped RJ45 connector:

1. Trim the outer sheath back about 10mm to expose the inner conductors.
2. Trim off any nylon strands or wire guides.
3. Straighten the wires.
4. Sort them out to the correct color codes for the pinout.
5. Snip the wires so they are all the same length.
6. Push the wires into the connector.
7. Make sure the outer sheath is inside the RJ45 crimp.
8. Crimp the connector.
9. Test the Ethernet cable.

What is the Maximum Length of an Ethernet Cable? The recommended maximum length for any structured cabling is 100m. This includes 5m of patch cables at either end of the cable run, so the fixed cabling's actual distance is 90m. Anything over this distance will introduce interference and losses on the cable, which will be seen as errors, dropped traffic packets, and reduced throughput.

Structured Cabling = 90m

LAN Patch Cables = 2 x 5m

It's good installation practice to keep the cable runs as short as possible and avoid unnecessary excessive service loops of cable.

16.3 DaoAI Power cable

16.3.1 DaoAI Power Supply

All DaoAI cameras come with a power supply of 3.2 m in length in total. The DaoAI Power Supply consists of an AC/DC adapter with an integrated DC cable (1.2 m) and a separate AC power cord (2 m). The power cord comes with EU/KOR, US/JAP, CN, or UK AC plug types.

BP SMALL

Input voltage	100-240 V AC
Input frequency	50-60 Hz
Output voltage	24 V DC
Output current	10 A
Line regulation	<± 1% of rated input voltage at full load
Rise time	50 ms at 100-240 V AC input with full load from 10% to 90% of output voltage
Dynamic load regulation	±5% excursion for 50%-100% or 100%-50% load change of DC output at any frequency up to 1 KHz (duty 50%)
Ripple/noise	1.5 % max of rated output
Overcurrent protection	110%-170% of rated output current

BP MEDIUM

Input voltage	100-240 V AC
Input frequency	50-60 Hz
Output voltage	24 V DC
Output current	10 A
Line regulation	<± 1% of rated input voltage at full load
Rise time	50 ms at 100-240 V AC input with full load from 10% to 90% of output voltage
Dynamic load regulation	±5% excursion for 50%-100% or 100%-50% load change of DC output at any frequency up to 1 KHz (duty 50%)
Ripple/noise	1.5 % max of rated output
Overcurrent protection	110%-170% of rated output current

BP LARGE

Input voltage	100-240 V AC
Input frequency	50-60 Hz
Output voltage	24 V DC
Output current	15 A
Line regulation	<± 1% of rated input voltage at full load
Rise time	50 ms at 100-240 V AC input with full load from 10% to 90% of output voltage
Dynamic load regulation	±5% excursion for 50%-100% or 100%-50% load change of DC output at any frequency up to 1 KHz (duty 50%)
Ripple/noise	1.5 % max of rated output
Overcurrent protection	110%-170% of rated output current

BP AMR

Input voltage	100-240 V AC
Input frequency	50-60 Hz
Output voltage	24 V DC
Output current	10 A
Line regulation	<± 1% of rated input voltage at full load
Rise time	50 ms at 100-240 V AC input with full load from 10% to 90% of output voltage
Dynamic load regulation	±5% excursion for 50%-100% or 100%-50% load change of DC output at any frequency up to 1 KHz (duty 50%)
Ripple/noise	1.5 % max of rated output
Overcurrent protection	110%-170% of rated output current

BP AMR-GPU

Input voltage	100-240 V AC
Input frequency	50-60 Hz
Output voltage	24 V DC
Output current	10 A
Line regulation	<± 1% of rated input voltage at full load
Rise time	50 ms at 100-240 V AC input with full load from 10% to 90% of output voltage
Dynamic load regulation	±5% excursion for 50%-100% or 100%-50% load change of DC output at any frequency up to 1 KHz (duty 50%)
Ripple/noise	1.5 % max of rated output
Overcurrent protection	110%-170% of rated output current

16.3.2 DaoAI Power Extension Cables

We also offer power extension cables in 5 m, 10 m, and 20 m options.

DaoAI strongly recommends that the cables are carefully checked before use or if run time errors occur. The Power connector needs to be screwed in completely to ensure the watertight IP rating of the camera.

Note: When using DC power extension cables For DaoAI cameras, ensure both the PC and the camera AC/DC adapter are supplied from the same power outlet.

Tip: DaoAI Power Extension Cables are specifically designed high-quality cables suitable for robot applications.

16.4 Recommended Industrial PCs

The table below shows recommended PC specs from High-End to Low-End grades. The expected capture speed increases from the bottom row to the top row.

Grade	CPU	Intergrated GPU	RAM	Ethernet	Dedicated GPU
High-End	Intel i5	Intel UHD 630/750/770	16 GB	1 GigE	Nvidia GTX 1080TI or GTX 1080
Mid-End			8 GB		Nvidia GTX 1070 or GTX 1660
Low-End					Nvidia GTX 1650 or GTX 1050Ti

The PC budget should be first spent on the components in the table. If the budget is higher, your spending should roughly be done in this order:

- Upgrading the GPU to 1060 or similar
- Upgrading the CPU to i7 or similar
- Upgrading the GPU to RTX 2080
- Upgrading the CPU to i9
- Upgrading the RAM to 64 GB

16.5 Laser Safety Facts

Class 3R (IIIa) laser safety information

WHAT IS A CLASS 3R LASER?

Class 3R lasers are considered safe when handled carefully. There is only a small hazard potential for accidental exposure. For visible-light lasers, Class 3R lasers' output power is between 1 and 4.99 milliwatts.

In the United States, both Class 2 and 3R lasers can be sold as "pointers" or for pointing purposes. (In Australia, the U.K., and many other countries, laser pointers are restricted to Class 2 only.)

Class 3R is essentially the same as the Roman numeral "Class IIIa" you may see on some lasers' labels. At this website, we primarily use the Arabic numerals, for convenience.

SAFE USE GUIDANCE - GENERAL

A Class 3R laser is low powered. It normally would not harm eyes during a momentary exposure of less than ¼ second. This is within the aversion response, where a person turns away and/or blinks to avoid bright light.

Do not deliberately look or stare into the laser beam. Laser protective eyewear is normally not necessary. A Class 3R laser is not a skin or materials burn hazard.

However, a Class 3R laser can be a distraction, glare or flashblindness hazard for pilots and drivers. **NEVER aim any laser towards an aircraft or vehicle that is in motion.** This is unsafe and is illegal – you could be arrested and jailed.

ONLY ALLOW USE BY RESPONSIBLE PERSONS

This is not a toy. Children can safely use Class 3R lasers only with continuous adult supervision.

Warning:

CLASS 3R LASER HAZARDS

SAFETY NOTICE: This website is intended for the educational, instructional and informational purposes of the user and is not to be considered a substitute for a knowledgeable and trained Laser Safety Officer (LSO)

with the duties and responsibilities as defined in the ANSI Z136 standard published by the American National Standard Institute.

The hazard distances listed below are intended only as general guidance. This is because 1) your laser may vary from the parameters (power, divergence) listed below, and 2) information on labels or marketing materials may not always be correct. For example, studies have shown that some laser pointers may be falsely labeled to avoid regulations – the actual power may be 10 times or more what the label indicates.

Always err on the side of safety. If your laser has not been measured by a knowledgeable and trained Laser Safety Officer, assume it is more hazardous than the label or marketing materials would indicate.

EYE INJURY HAZARD

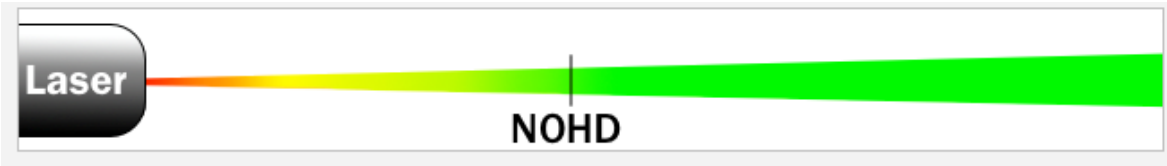
EYE INJURY HAZARD – DIRECT AND REFLECTED BEAM

Class 3R visible-light lasers are considered safe for unintentional eye exposure, because a person will normally turn away or blink to avoid the bright light. Do NOT deliberately look into or stare into the beam – this can cause injury to the retina in the back of the eye.

Be aware of beam reflections off glass and shiny surfaces. Depending on the surface, the reflected beam could be about as strong and as focused as a direct beam.The Nominal Ocular Hazard Distance (NOHD) for the most powerful Class 3R visible-beam laser (4.99 mW) with a tight beam (0.5 milliradian divergence) is **104 ft (32 m)**.

Fig. 1: Color indicates the relative hazard: Red = potential injury, green = unlikely injury. Beyond the Nominal Ocular Hazard Distance, the chance of injury is "vanishingly small" according to safety experts.

For a 4.99 mW Class 3R laser with a less-tight beam that spreads out faster (1 milliradian), the NOHD is 52 feet (16 m). This divergence is more typical of consumer lasers.



If you are closer than the NOHD distance to the laser, there is a possibility of retinal damage if the direct or reflected beam enters your eye longer than about ¼ second. The closer you are to the laser and the longer the beam is in the eye, the greater the chance of injury.

CHAPTER

SEVENTEEN

FREQUENTLY ASKED QUESTIONS (FAQ)

17.1 What depth sensing technology is used?

The DaoAI Cameras calculates the depth information of the scene by projecting structured light into the environment. This technology allows high resistant to ambient light, and ensures high accuracy in any light environment.

17.2 Is the light harmful?

The lights are very safe.

However, please avoid staring into the light directly as it could still cause temporary damages to your eyes and affect your vision.

17.3 Safety Standards

All of the DaoAI Cameras are industrial graded, with IP65 rated enclosure, which protects the DaoAI Cameras from low pressure water jets, condensation, and water spray.

On top of that, the cameras also offer high thermal stability, which allows the camera to function reliably in dynamic working conditions.

17.4 How much does the camera weigh?

The Bin Picking Series Cameras weigh from 1.2kg to 5.2kg depending on the camera model:

- BP-S: 2.3kg
- BP-M: 5kg
- BP-L: 5.2kg
- BP-AMR-GPU: 2.1kg
- BP-AMR: 1.2kg

The Inspection Series Cameras weigh from 3.7kg to 3.8kg depending on the camera model:

- IN-XS: 3.8kg
- IN-S: 3.7kg
- IN-M: 3.7kg

Please refer to our Bin Picking Series page and Inspection Series page for a more detailed field of view comparison graph and table under the SPECS section.

17.5 Field of View of Cameras

The field of view differs from camera to camera.

For the Bin Picking Series:

- BP-S: 665 x 394 mm
- BP-M: 897 x 560 mm
- BP-L: 1725 x 971 mm
- BP-AMR-GPU: 617 x 394 mm
- BP-AMR: 617 x 394 mm

For the Inspection Series:

- IN-XS: 83 x 60 mm
- IN-S: 132 x 106 mm
- IN-M: 225 x 185 mm

Please refer to our Bin Picking Series page and Inspection Series page for a more detailed field of view comparison graph and table under the SPECS section.
Note that the values are in millimeters.

17.6 How accurate are the DaoAI Cameras?

The DaoAI Cameras garauntee 99% point cloud correctness and have fewer outliers.
The Inspection Series capture fine details with 5 megapixels.

17.7 What kind of Ethernet cable should I use?

The CAT-6A Ethernet cable are compatible with the DaoAI Cameras.

17.8 Supported Operating Systems?

We currently support Windows operating system.

17.9 Supported APIs?

- C++
- C#
- Python

17.10 What formats can files be saved?

The camera configuration can be saved and exported as a `.cfg` file; Point clouds can be saved as `.pcd` files.

17.11 What temperatures can I safely run the cameras in?

All the Bin Picking Camera Series and Inspecting Camera Series can be operated in 0C to 40C.
It is recommendated to operate the DaoAI Cameras in this temperature range to ensure the best working quality.

17.12 How can I access the log files to forward to Support?

Please refer to our *Suggest a Feature* page for more details.

CHAPTER

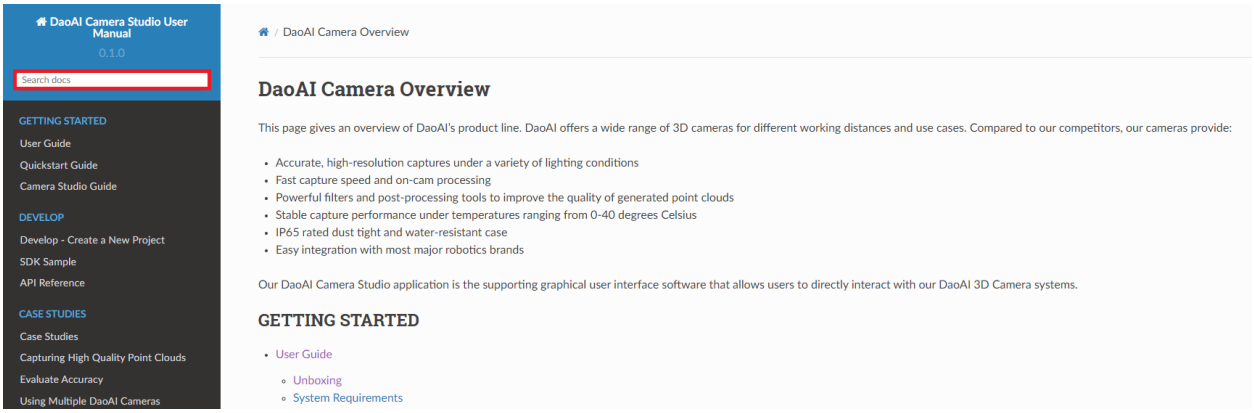
EIGHTEEN

BUG REPORT

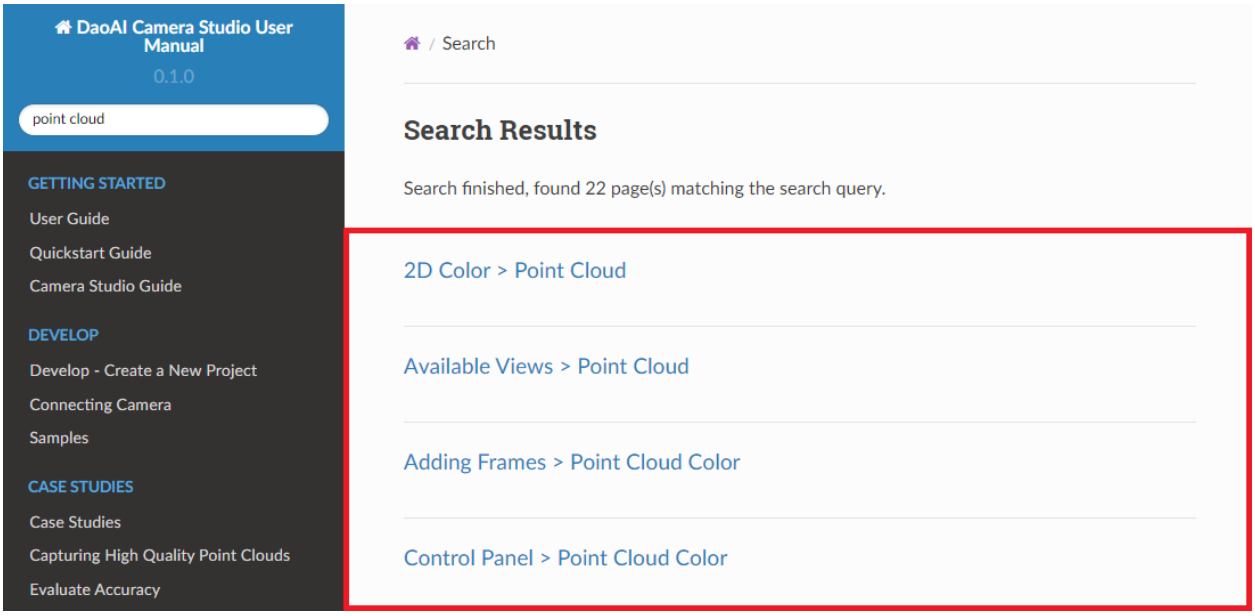
If your issue is not addressed in the user manual nor listed on our FAQ page, you can report your issue at our Help Center.

You can also reach out to us via support@daoai.com and our team will get back to you as soon as possible!

18.1 Search for Related Articles

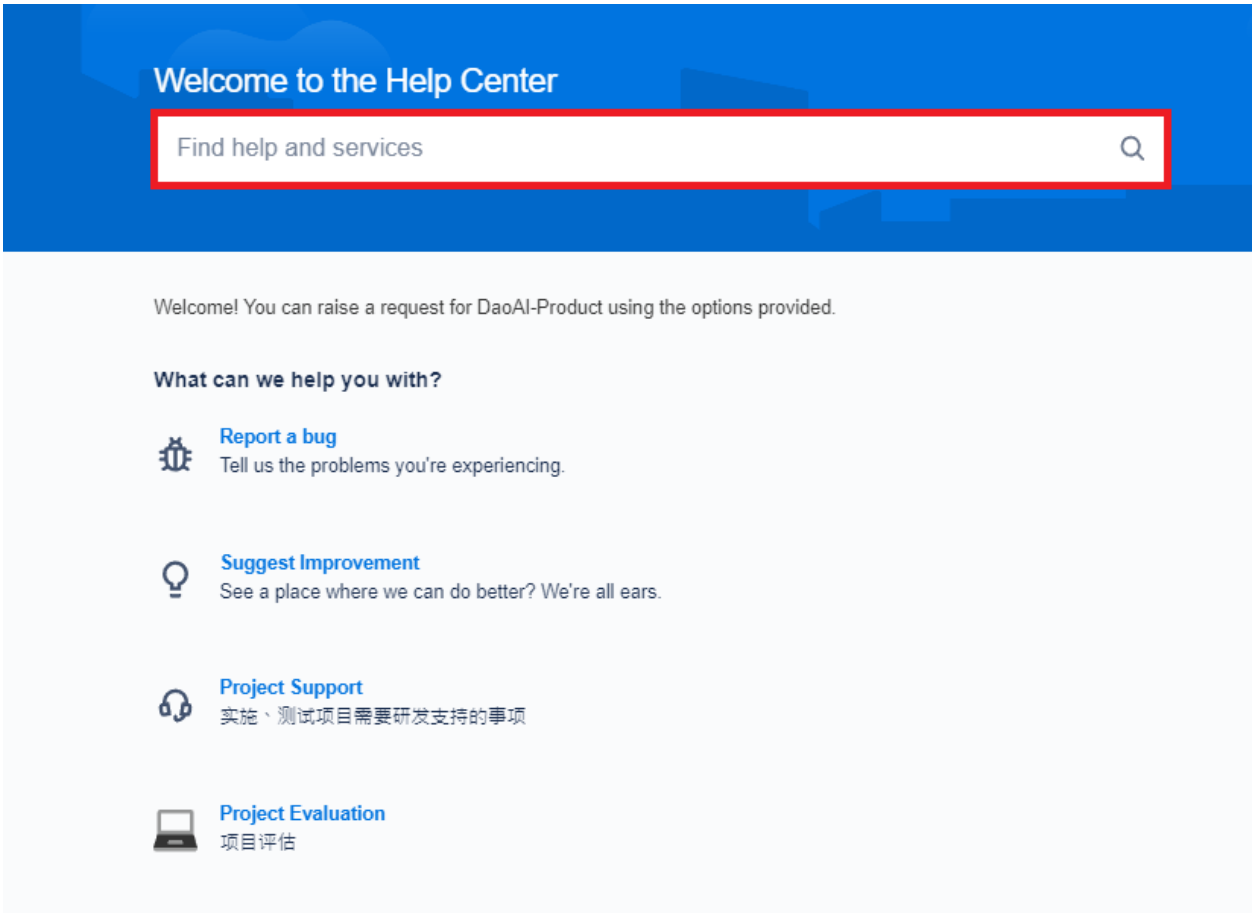


You can type in the keyword in the search bar located in the upper left side to look for related articles on your topic.

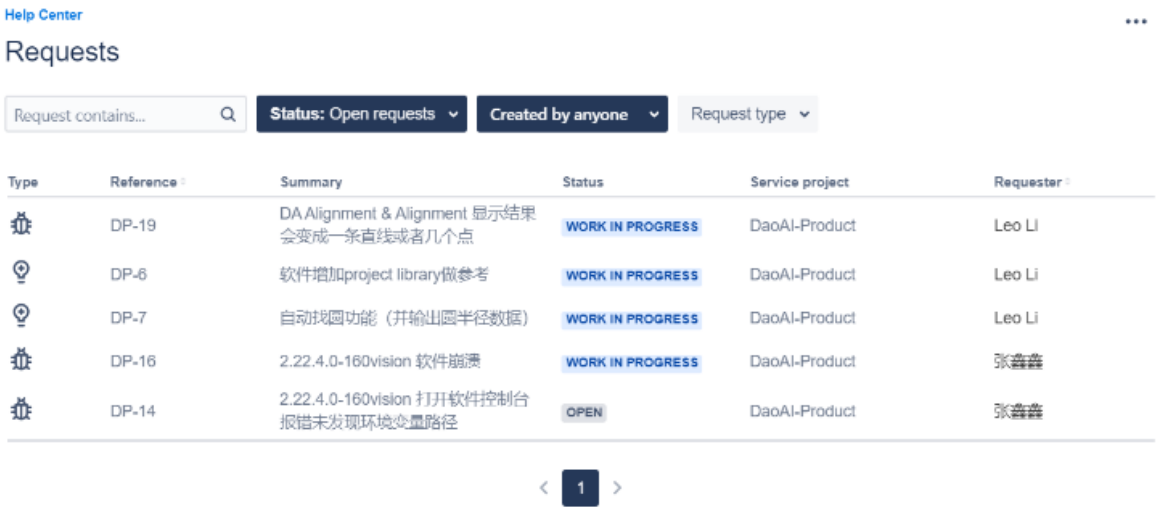


For example, we searched for the keyword “point cloud”, and all articles related to point clouds are listed in the result.

18.2 Search for Related Bug Reports

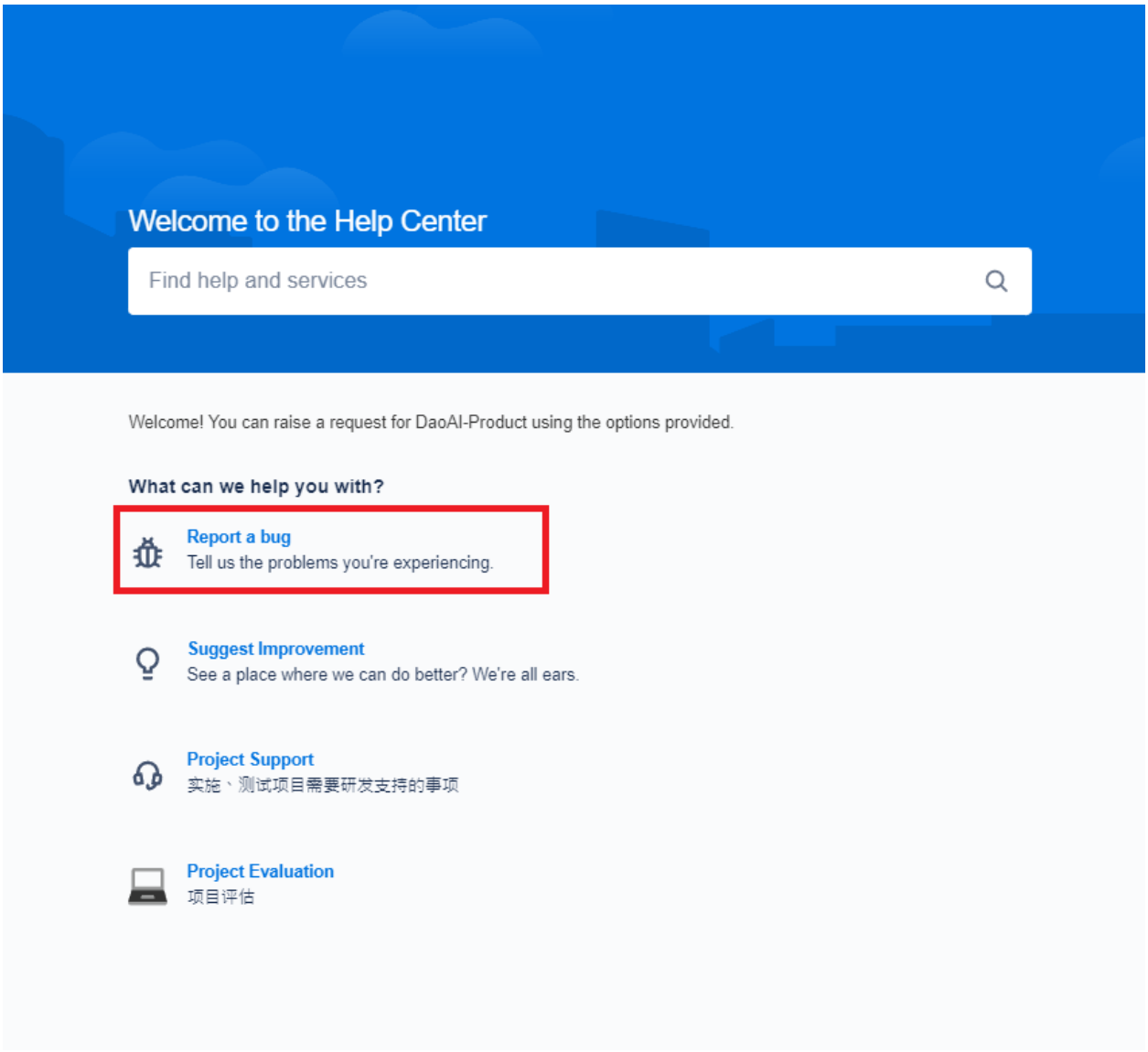


You can search for the keyword, or adjust the filter to see if similar issues have been reported or resolved.

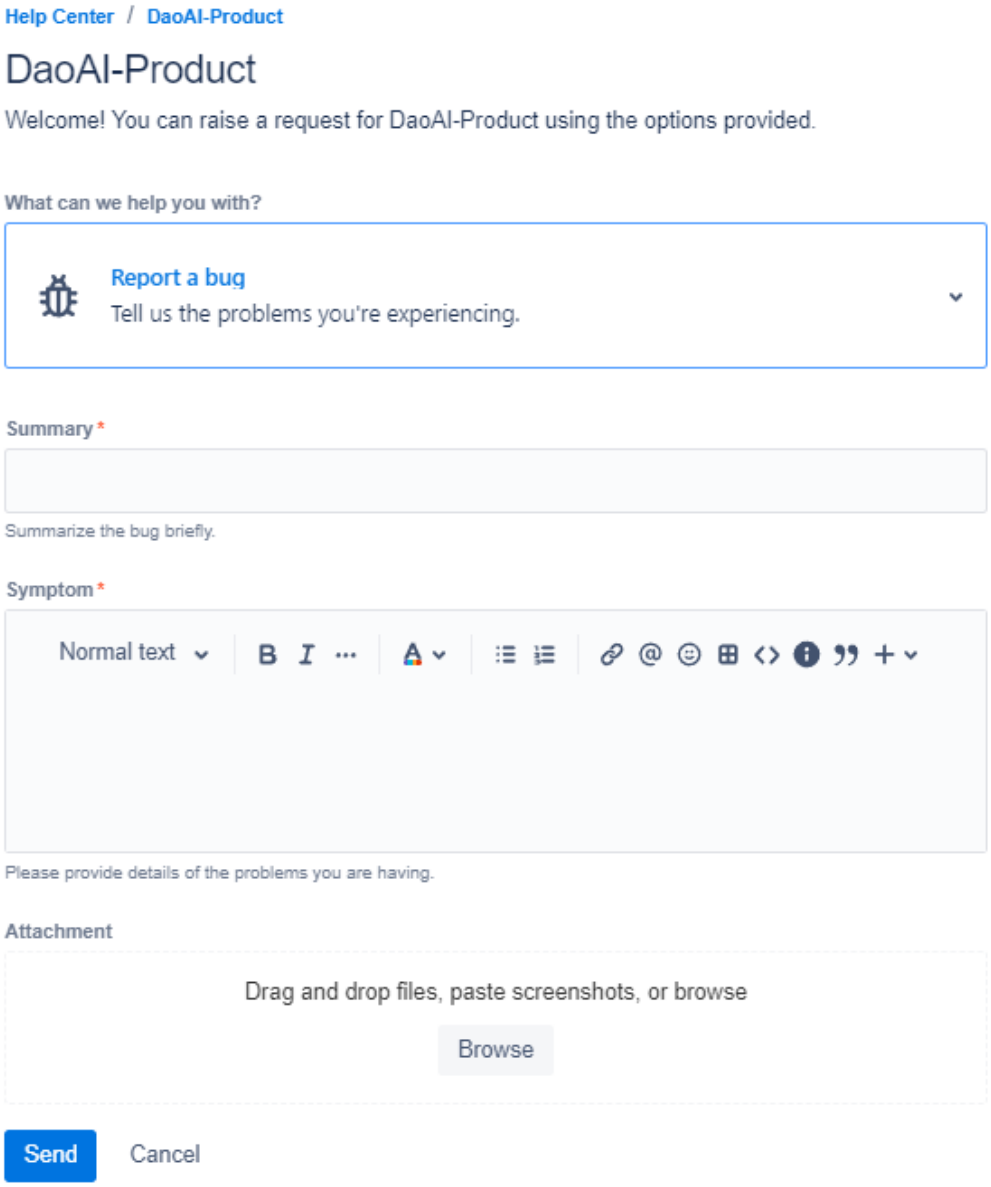


18.3 Report a Bug

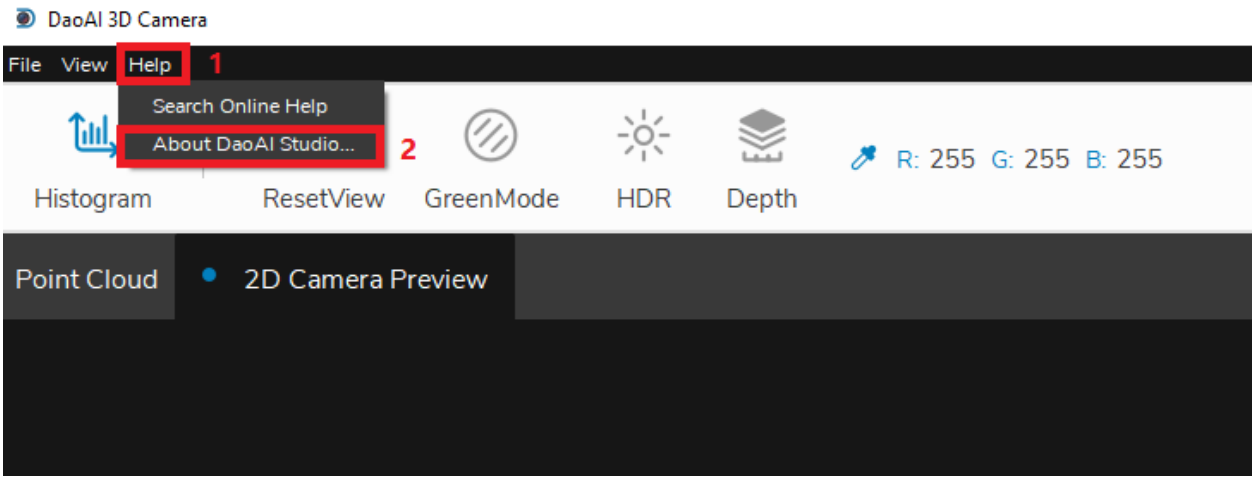
To report a bug, please go the Help Center and click “Report a Bug”.



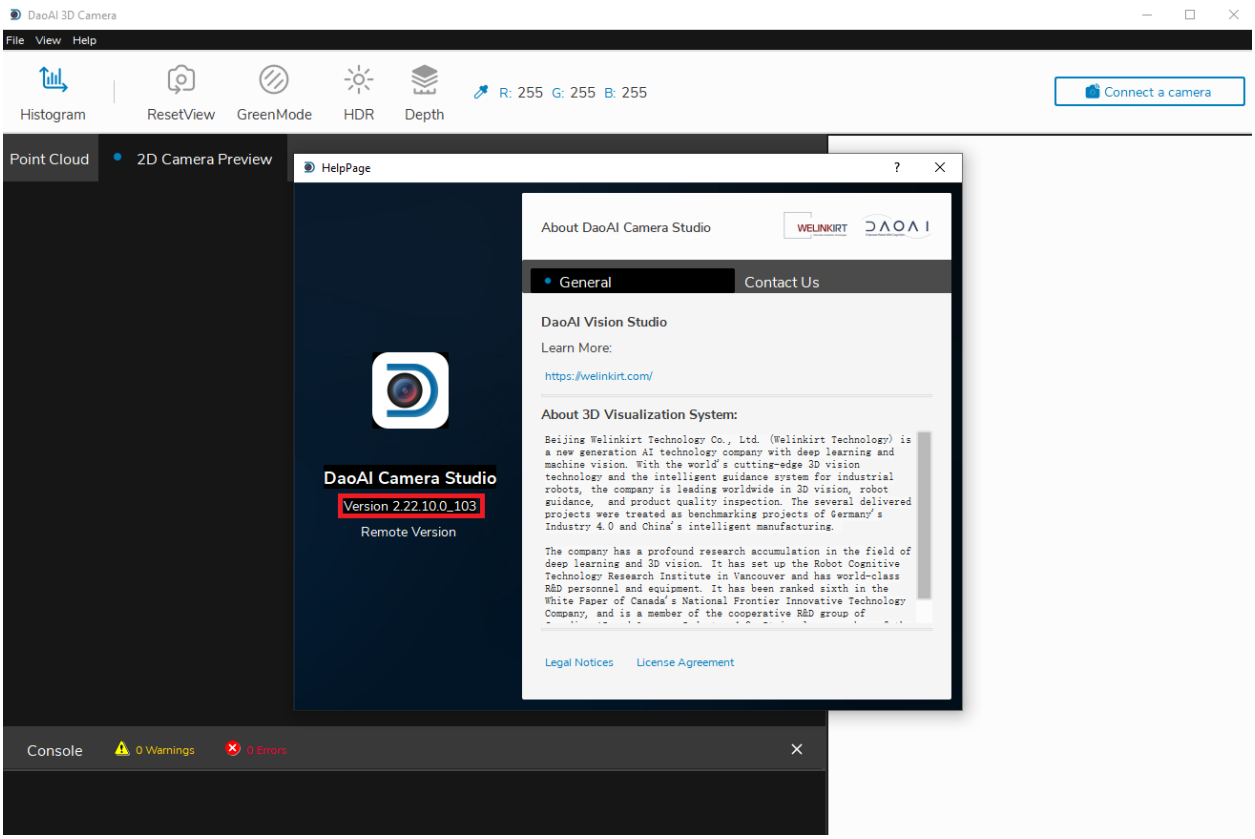
You will be directed to the report a bug page.
Please fill in as much information as possible (e.g. software version) so that we can resolve the problem as quickly as possible.



To check your software version:
Locate the top menu bar → select “Help” → select “About DaoAI Studio”.

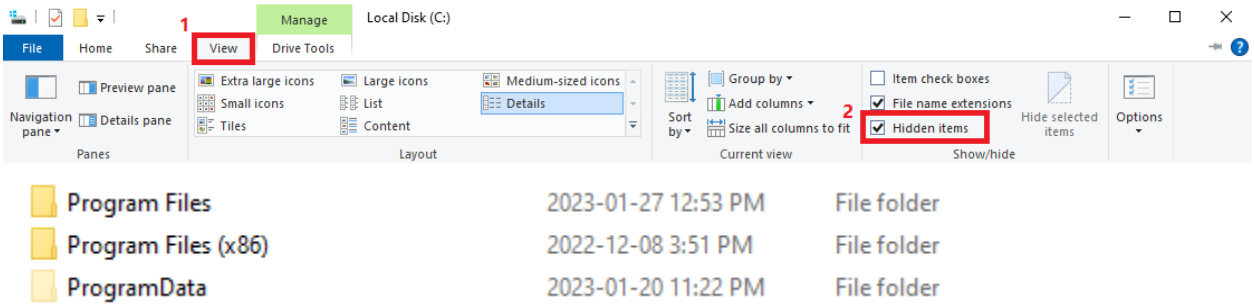


You can find the software version under the DaoAI Studio logo.
In this case, the software version is 2.22.10.0_103.



If you encountered a crash, please attach the dmp file in your report.
You can find the dmp file via the path C:\ProgramData\DaoAI\Camera Studio\Crashpad\db\reports.

If you cannot see the ProgramData folder, you may have to change the view settings:



Here are some other information that would be nice to have in the Description section:

- Steps to reproduce the issue
- Software version, working environment
- Your name
- Contact email address and phone number (so that we can update you on the reported bug)

CHAPTER

NINETEEN

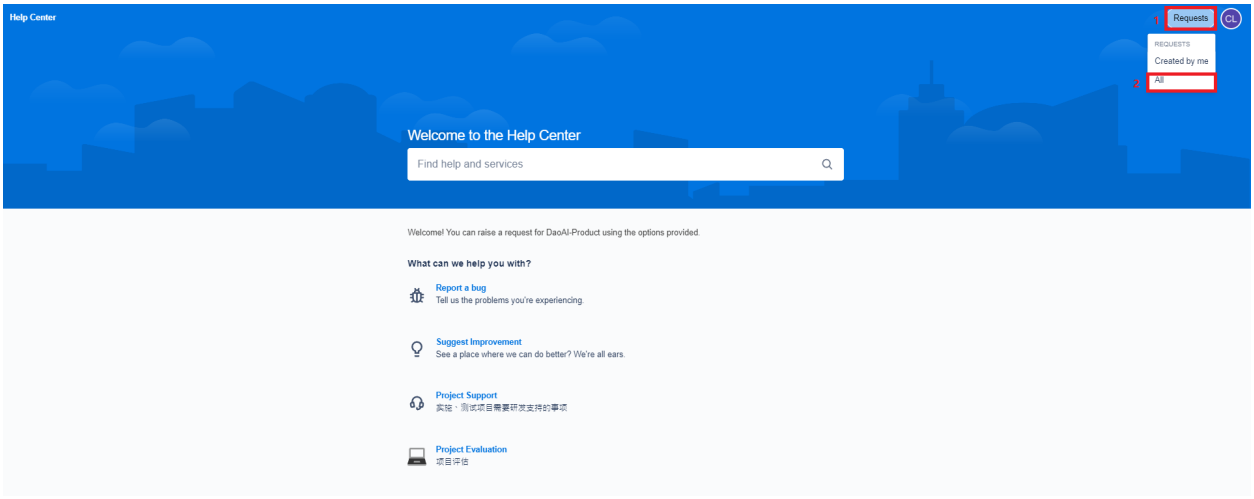
SUGGEST A FEATURE

If you would like to see a new feature added in DaoAI Camera Studio, you suggest it at our Help Center.

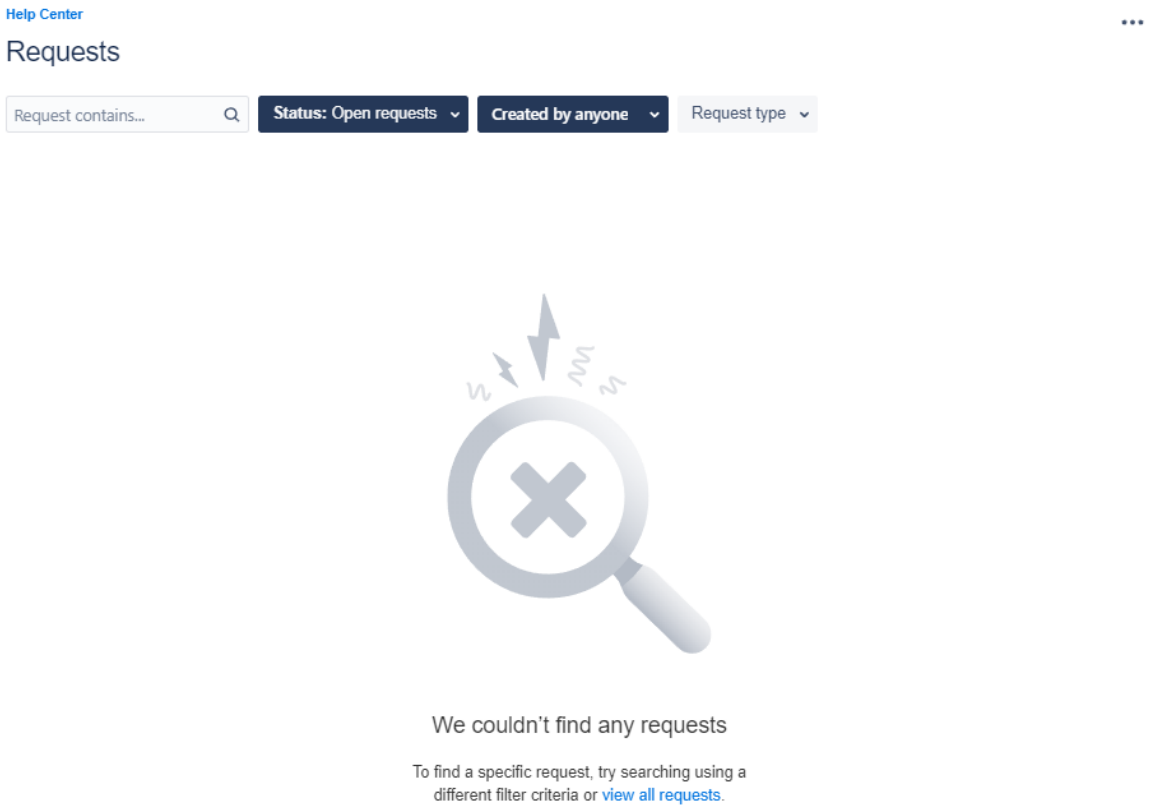
19.1 Search for Related Suggestions

Before you submit your feature suggestion, please check if any other similar feature request has already been submitted.

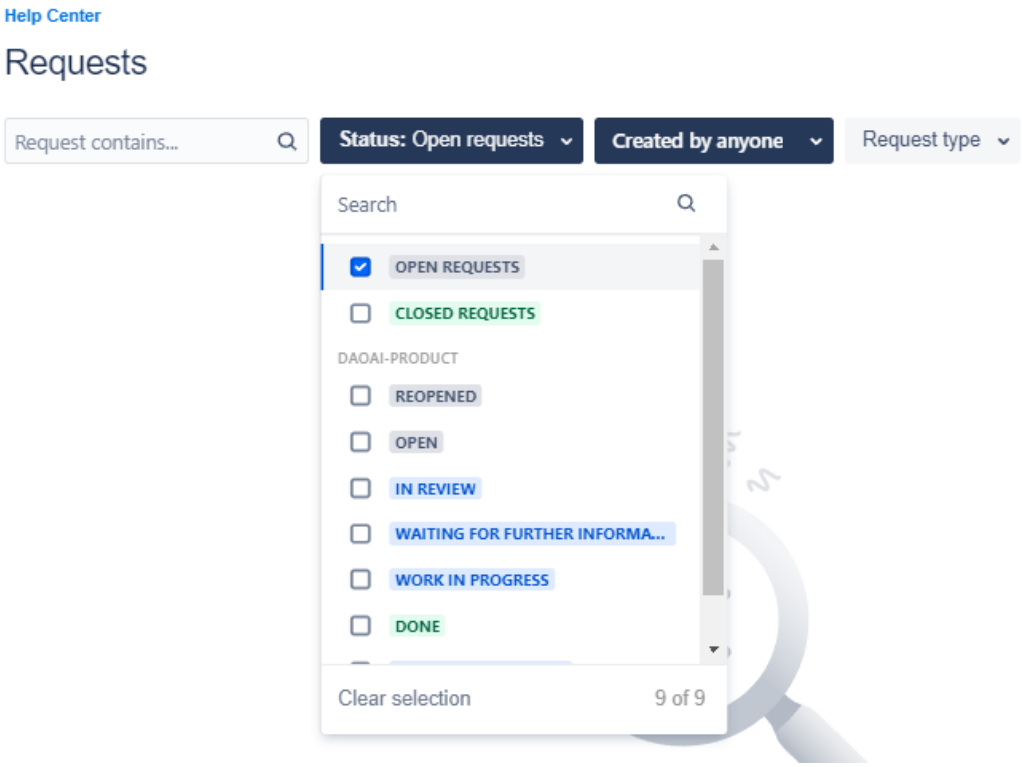
You can see all feature suggestions by clicking “Requests” → All at the top right corner.



All the feature suggestion that are currently being processed will be shown in the list.



You can adjust the filters to search for specific requests.

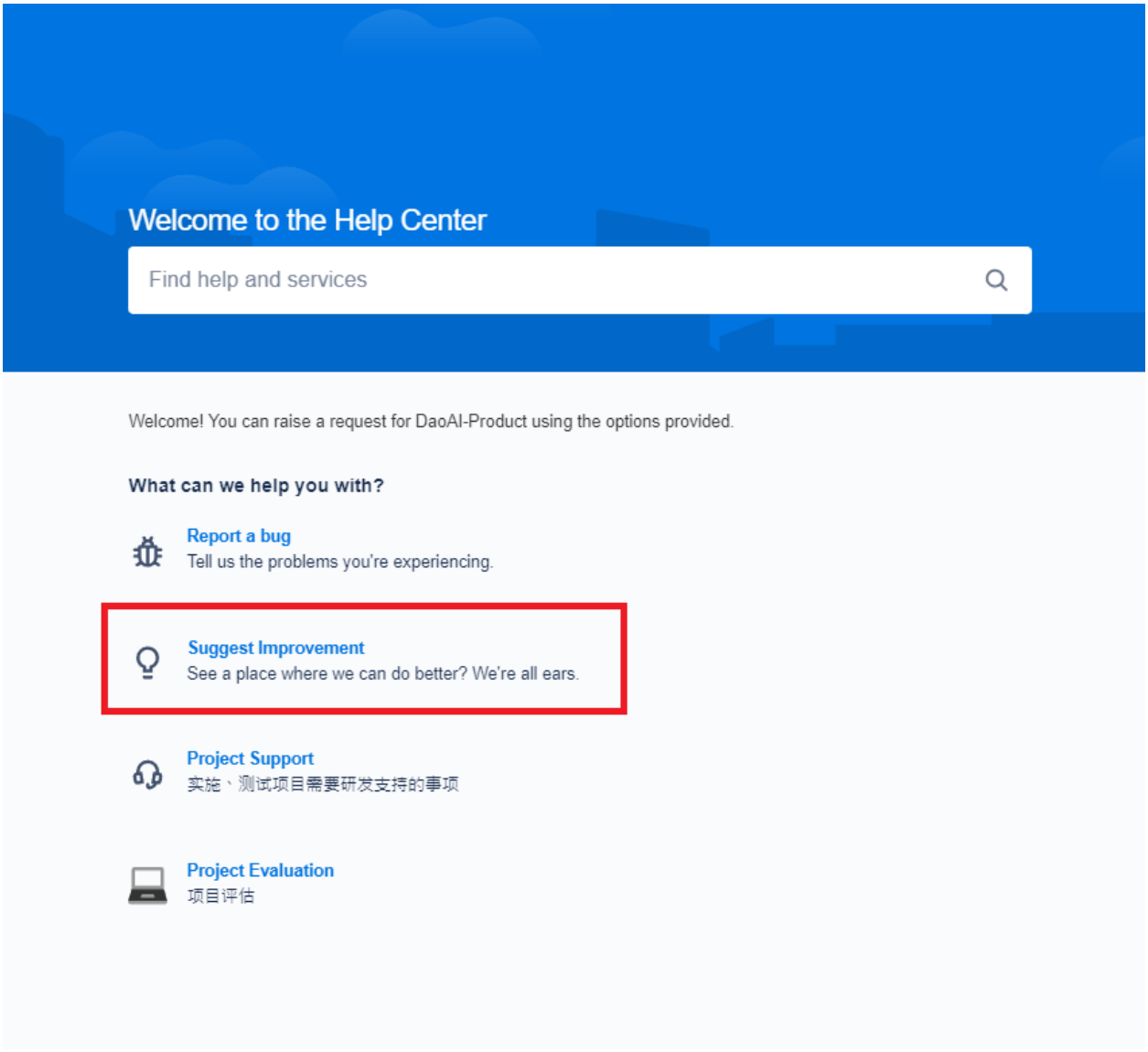


19.2 Feature Request

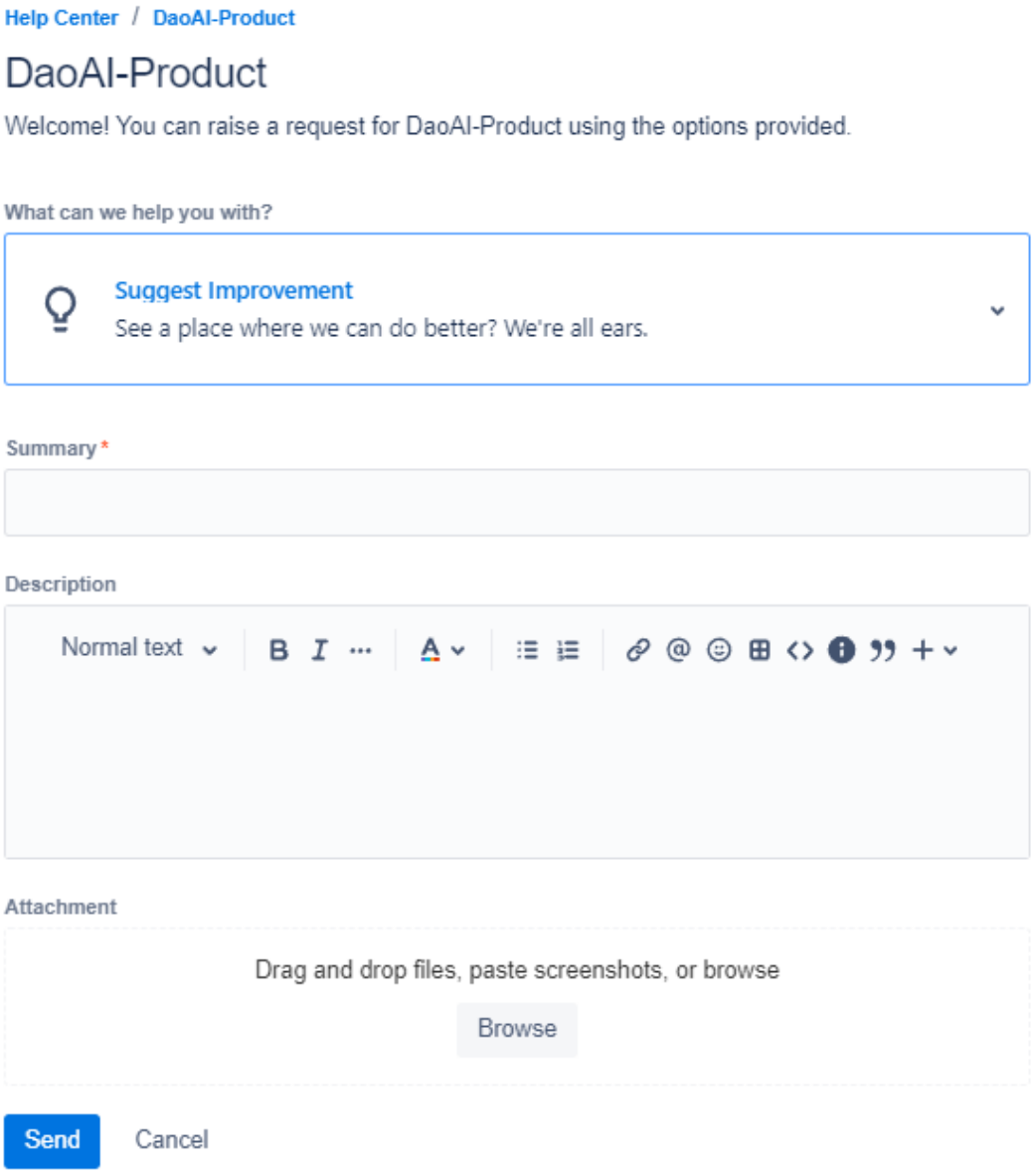
Once you have discovered that you have a novel feature suggestion, you can request it to us!

Go to the Help Center home page.

Select the second option, “Suggest Improvement”.



You will be directed to the Suggest Feature page.



Here are some other information that would be really nice to include in the Description section:

- Feature description
- Why would the feature be a great addition
- Your name
- Contact email address and phone number (so that we can update you on the feature)

Please fill in as much detail as you can regarding your suggestions, so that we can realize your requests as soon as possible!

CHAPTER

TWENTY

RELEASE NOTES

Version update informations will be listed in this page.