Fast and Stereoscopic Imaging System

Dovan Darazi

October 2023

Contents

Chapter 1

Introduction

Polaris is a linear plasma device, located at AUB. The device consists of many complex parts each designed to complete a specific task. In this document, we are only concerned about the imaging system. I will discuss the steps taken to develop the fast stereoscopic imaging in detail. The goal of this document is to provide as much information as possible so the next person that works on this project is brought up to date and can continue the work instead of going back to step one.

Figure 1.1: Polaris the Lebanese linear plasma device

1.1 Motivation

In devices like Polaris, Tokamaks and other plasma devices,the presence of dust, flakes, and unidentified flying objects (UFOs) in the plasma flow can have significant implications for both the operation and safety of the reactor.
Dust can come from various sources, including the erosion of plasma-facing components,

deposition of impurities, and material sputtering from the walls. Its impacts include: Plasma Contamination, Safety Concerns, Diagnostics Interference.
Flakes are typically larger particles that can originate from the delamination or spalling of coatings or from larger pieces of eroded material from the walls and components.
UFOs refer to any unidentified objects observed within the plasma that cannot be immediately classified as dust or flakes. Their biggest impact is unpredictability.

Figure 1.2: Melted Molybdenum tiles from Alcator C-Mod due to UFO disruptions.

In order to predict the effect of dust particles, flakes or UFOs in the plasma, we must first determine their three-dimensional position, their size, and velocity. These particles move at high speeds. We need high-speed imaging cameras which are very expensive. Hence, we look to develop a method to achieve high-speed/fast imaging using less expensive cameras. We discuss this in the second chapter, The position. size and velocity of these particles should enable us to determine whether to interrupt the plasma **BEFORE** a disruption takes place. We discuss this in the third chapter,

Chapter 2

Fast Imaging

2.1 Camera's inner workings

In order to understand how this specific system works and the basis on which the improvements will be based, we need to understand how a digital camera captures video.

2.1.1 Relevant camera parts

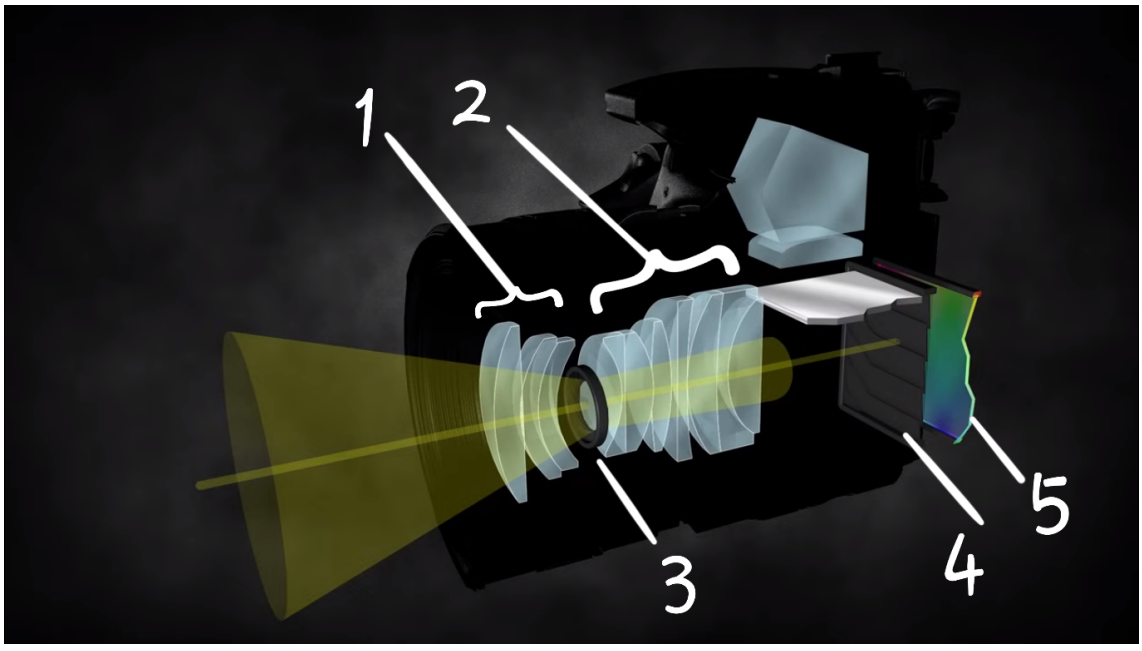


Figure 2.1: An image of a DSLR Camera showing its parts, and enumerating the ones we are interested in.

Body of the camera

The body of a camera is the base itself. It's the part that you hold or mount and it houses most of the significant components. When it comes to DSLR cameras (i.e. Digital Single Lens Reflex cameras, cameras used by photographers and photography enthusiasts), the body is what most people refer to as "the camera". The body is mainly made out of plastic and metal as well as other materials such as optical glass, silicon, aluminum, steel, rubber, leather, fibers and so on. Everything else that may be included, such as lenses or flashes, are accessories that are interchangeable and not part of the camera proper.

Lens

Light enters the camera through the lens, which is interchangeable for some cameras similar to the one in figure 3 above. The parts 1 and 2 in the figure are the lens.

You probably noticed that both 1 and 2 are a series of lenses and not just one, but we are referring to them as the lens. That's simply because we are visualizing the whole part as one lens that's doing the combined work of all its individual parts. The job of the lens is to bend the light by moving either manually or automatically resulting in zooming in and out, as well as playing a role in focusing on certain parts of the image.

Aperture

The aperture is situated in between the lens' components (figure 2). The role of the aperture is to partially close to control the amount of light passing through to be captured by the sensor. When the aperture is wide open, it allows more light to enter the lens and hit the sensor from multiple angles. This creates a shallow depth of field, the subject might be in focus but the background will be out of focus. Narrowing down the opening of the aperture will result in less light entering. The light will be more or less directly from the front in one direction and therefore we get a deeper depth of field, both the subject and the background will be in focus.

Shutter

The shutter (part 4 figure 2) is located inside the body of the camera. Its job is to block the light coming into the camera through the lens from reaching the camera's image sensor.The shutter release button controls it. Once you press this button, the shutter opens, allowing the light to strike the image sensor and capture the desired image.

Sensor

The sensor detects the light and records it to create your image. It measures the intensity of light hitting it as the shutter opens. The sensor is made up of individual units, which are called pixels. Each pixel measures the intensity of light by detecting the number of photons that reach the pixel. This information is relayed to the camera as a voltage value which can then be recorded by the camera.

Image Processor

The image processor is the component that takes all of the information from the camera sensor and uses it to create a visual image that we perceive on the screen. Without the image processor, all we would have is a bunch of coded voltage values that would not look anything like the picture we took

Storage System

After capturing the image and processing it we need to store it somewhere in order to access it at another time. This is the job of the storage system. Most cameras will use what is known as an SD card, which stands for Secure Digital Card. An SD card is a small removable memory card that emerged out of a group of memory cards that hit the market when digital cameras first came of age. We can also use a hard disk drive (HDD) or a Solid State Drive (SSD) to store the data. In most cases an HDD or SSD is used for storing the SD card's data long term. In other cases where the camera is fixed e.g. security cameras or more importantly cameras used for experiments and collecting too much data we have the option to connect the camera directly to an HDD/SSD linked to a computer ready for analysis.

2.1.2 Capturing an image

Light enters the camera through the lens passes through the aperture and proceeds toward the closed shutter. The shutter opens then closes for a time Δt called the shutter speed. When the shutter is open the sensor detects the light in sections through it's pixels and relays the captured data to the image processor that proceeds to process the data sent in the form of voltage values and relays it to be stored. This entire process is done in the time taken by the shutter to open and close. Increasing Δt or not setting it at all before starting to take a photo can be used to capture more light over time which creates motion in a still image. This method is mostly used in light painting.

P.S.: Light painting is the art of creating photos by setting long exposure times on a camera and using a moving light source to "paint". A light painting photographer opens a camera's shutter and keeps it open as they draw in the air with a light source figure 3.

2.1.3 Recording video

"Video" refers to the visual aspect of what most people call a video or a recording. The concept of a video is really simple. Similar to how in mechanics we draw two points on a graph showing the variation of position as a function of time which allows us to deduce the velocity (i.e. motion) of the object, we only need to capture multiple images in succession, called frames, (i.e. the position points on the graph) of some event that's happening and play them one after the other in a given time interval, called the frame rate expressed in frames per second (FPS), (i.e. the variation of time) to create motion. All we need in order to take multiple shots is to alternate between allowing and preventing the light from reaching the sensor.

Figure 2.2: Photograph created by light painting

This process can be achieved by the shutter. It opens allowing the sensor to capture light, then it closes preventing the light from crossing which results in the successful capture of one frame. The shutter opens again and the process is repeated n times. The time taken by the shutter to complete one cycle of opening and closing is called the shutter speed, which is inversely proportional to the frame rate.

This process can also be achieved by varying the intensity of the incident light itself. This idea was developed in 1987. Due to shining a bright infrared laser through a noble gas, a light at different frequencies to the initial laser pulse will be produced. These frequencies will interfere with each other. The main idea is to cause a specific interference that consists of a series of constructive and destructive interference. The period of this wave now becomes what we called the shutter speed. This technique is really difficult in comparison to the previous mechanical technique, because even for light it is hard to make pulses that are so short. The difficulty of creating such short pulses earned the physicists who could achieve a breakthrough in that field a Nobel prize, once in 2018 for finding a way of generating a femtosecond pulse ($10^{-15}s$), and more recently in 2023 by creating an attosecond pulse ($10^{-18}s$). You might be wondering "why are physicists going to such extremes to create these pulses instead of sticking to the good old shutter?". Physicists don't just love complicating simple things. The mechanical aspect of the shutter method limits it's potential to relatively slow speeds. Even if a way to make the shutter go extremely fast was discovered and the the heating problems that arise were ignored, matter is still limited to speeds less than the speed of light and inertia would cause catastrophic problems.

In our case we're gonna use a method somewhat in between the two discussed above, working around the limited shutter speed of the first and the difficulty of the second one. Check this section to learn more about it.

NOTE: A high frame rate results in a smoother playback. A really high frame rate results in a smooth slow motion. The faster the object we're observing, the higher the frame rate should be. That's why we need a slower shutter speed.

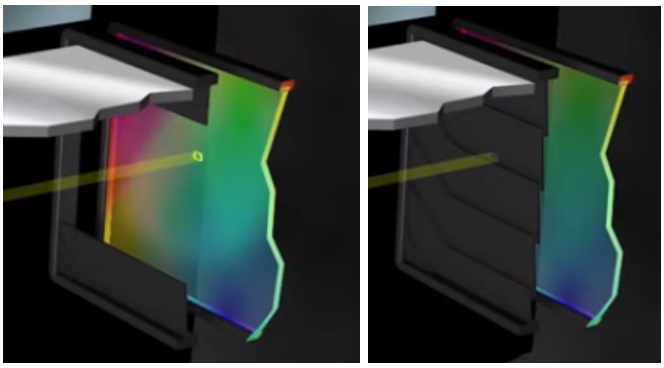


Figure 2.3: Left: an open shutter allowing light to reach the sensor.
Right: closed shutter blocking the path of the light ray.

2.2 Fast-Imaging camera

The Chronos 1.4 High-Speed Camera (figure 4) is the ideal choice for research and development, engineering, and manufacturing work that requires relatively high frame rates for review and analysis. It has the capacity for high frame rates at lower resolutions and is the best choice for lower budgets. Video is saved in compressed H.264 (MPEG-4) or uncompressed RAW format to removable media. You can save hundreds of shots on a single SD card.

2.2.1 Main Features

1.4 Gpx/s, 1.3-megapixel image sensor captures 1280x1024 @ 1069 Frames Per Second (FPS), and up to 40413FPS at lower resolution. Available in color and monochrome. The monochrome option has higher effective resolution and is twice as sensitive as the color option. 8GB, 16GB, and 32GB high-speed RAM buffer options for 4, 8, and 16 second record time respectively. High sensitivity of ISO 320-5120 (Color) and 740-11840 (Monochrome) enables shooting with modest lighting. Completely standalone, untethered operation. Field-swappable internal battery lasts for up to 1.5 hours of recording. Runs indefinitely on AC adapter or external power source. Continuous record mode records normal rate video (60FPS) continuously to storage devices while simultaneously recording bursts of high-speed video. I/O ports enable synchronization and remote triggering through electrical signals, audio, and web interface. An open source, REST-based Application Programming Interface (API) is also included for integration into custom software or control environments. Focus peaking highlights sharp edges for quick and perfect focus. Zebra lines help you set correct exposure.

2.2.2 More Details

For more information about Chronos 1.4 check the manual "CHRONOS 1.4 DATASHEET" either provided by Dr. Antar or found online through the link.

Figure 2.4: The Chronos 1.4, camera of choice for Polaris

2.3 Two-Camera setup with the same field of view

2.3.1 Main Idea

It's time to discuss the last paragraph of section 2.3 (i.e. the workaround of the shutter speed). As mentioned before the mechanical shutter has it limitations. Adding to that, better hardware equates to higher price. There must be a better budget friendly way of achieving lower shutter speeds (reason stated in the NOTE of section 2.3), and using the camera to its full potential. The secret lies in using two cameras that are "seeing" the same image at different time intervals. In other terms we want a setup of two cameras recording from the exact same point of view, but with a phase shift ϕ such that $\phi = \pi + d\theta$ where $d\theta$ is a differential phase shift that will help in separating the two consecutive frames taken by the two cameras. By using this method, one camera will be capturing a frame when the other is resetting to capture the next. One of the cameras' shutters will be closed when the other is open and then they switch back and forth periodically. If the frames were numbered consecutively the odd numbered frame would be taken by camera A and the even numbered frames would be taken by camera B.

2.3.2 Building the setup

Two cameras, one point of view

The key to achieving the outcome described in the previous section is the use of a 50-50 beam splitter shown in figure 6 labeled 'S'. The incident light enters the camera chamber

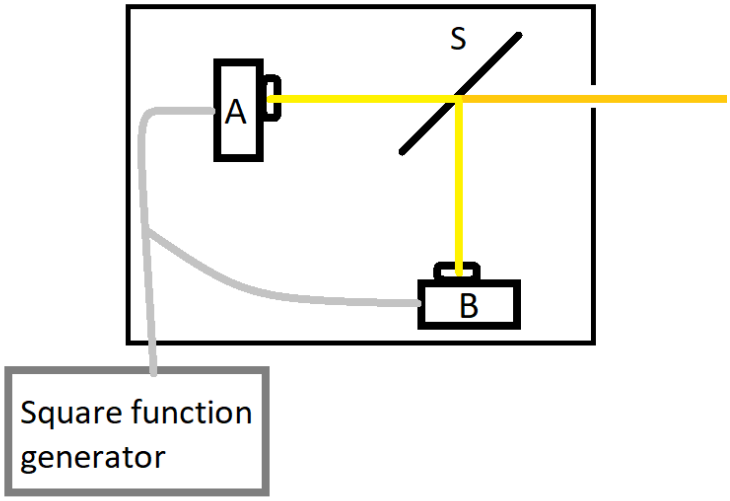


Figure 2.5: A diagram showing the experimental two-camera setup.

through an opening in its wall. The beam then reaches the beam splitter 'S' which transmits the beam with 50% of its intensity toward the camera 'A', and reflects the other 50% in the direction of the camera 'B'. Now both cameras are "seeing" the same image at the same time, with the same intensity: $I_A = I_B = \frac{1}{2} I_{incident}$.

2.3.3 Setting the phase shift

Before creating a phase difference between the two cameras we need to sync them up first. In order for both cameras to perceive the same light at the same time, they need to be positioned the same distance away from 'S' (i.e. $d(S, A) = d(S, B)$).

Now that there is no (measurable) phase difference between the two cameras $\phi \approx 0$ we can add our desired phase difference. We start by installing a square function generator, which will generate a function like the one in figure 7. We relay the outputted function to the cameras 'A' and 'B'. Now both cameras are receiving the same function at the same time. Finally one camera should be set to open the shutter when the function is rising and close when the function is falling, vice versa.

2.4 The work done by Osama

The work done by Osama discusses the Chronos 1.4 Gated Set Up (July 2019). The document written discusses the Two-Camera setup used and how it's operated. This section is similar to the original document due its compactness and density of important information.

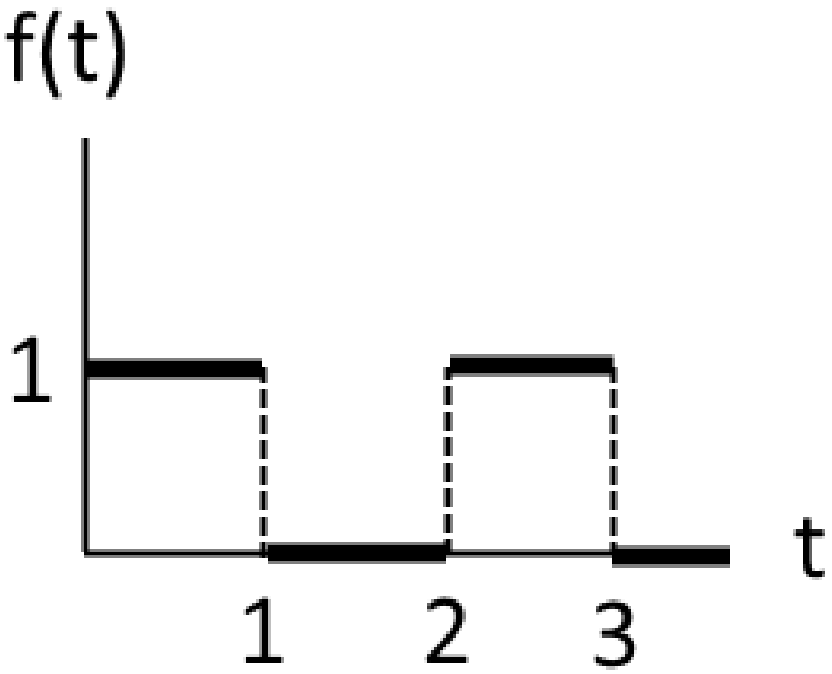


Figure 2.6: a square function/a periodic Heaviside function generated by the function generator.

2.4.1 Camera Setup

This section details how to set up the camera for a gated trigger configuration.

1. On the main screen go to Record Settings and choose the desired resolution. Note down the Max rate for this resolution and Min Period.

2. On the same screen go to Record Modes and choose Normal. Choose the duration of the recording.

3. Press OK until the main screen is reached.

4. Go to Trigger/IO Settings and choose Shutter Gating under IO 1 BNC. Ensure Invert, 20 mA Pullup, 1mA Pullup, Debounce are unchecked/off.

5. Set the Threshold to 2.50 and ensure under IO 2 the None is checked/on.

   on the main screen the recording should freeze as soon as the trigger is selected (as the camera is not recording).

2.4.2 Function/Pulse Generator Set Up

Gated Trigger

1. Function Generator In the Shutter Gating trigger mode the trigger input decides both the frame rate and the exposure time (how long the shutter stays open).

   - Choose a Square Wave on the function generator.
   - The frequency of the signal is the frame rate. The $\frac{Duty\ Cycle\ Percentage}{Frequency}$ is the exposure time.
   - . Ensure the frequency is at least 1 less than the Max rate to avoid dropped frames. Example: Max rate = 1057 then Max frequency = 1056.
   - Set the amplitude to 5.00 Vpp and ensure Offset is 0.00V
   - Press Record on the camera and connect the BNC from the FG to the camera. To stop recording use the shutter button or the button on screen.

2. Pulse Generator

   Set up your desired pulse with the following minimum limitations (set by the camera). The Pulse Period $\geq 3\mu s$.. The pulse delay $\geq 2\mu s$. The pulse width $\geq 1\mu s$. These are ideal numbers.

   The actual minimum Pulse Period $\sim 26\mu s$ is set by the camera frame limit of 38565 fps at 336×96. Ensure that the amplitude (the height from 0.00 V) is around 2.5V to avoid damage to the camera. Trigger the camera for the FG.

Dual Cameras Single Input

In the Exposure Trigger mode, the trigger input decides only the frame rate. The exposure time (how long the shutter stays open) is set by the camera on the Record Settings screen. Two cameras can be setup with a phase difference using only one input (such as the sole output of the Pulse Generator).

1. For both cameras follow the Camera Setup but now in step (1) choose Record Exp. In step (4) choose Exposure Trigger.
2. For one of the cameras (the one you wish to delay) check/turn on Invert. This camera will now trigger on the falling input. Finish the camera setup. Ensure the total BNC length to each camera is the same.
3. Follow the Function Generator setup. Split the single output of the function generator using a BNC T and connect individually to each camera.

Now the Duty Cycle of the input controls the phase. The Duty Cycle does NOT control the exposure time in the Exposure Trigger mode.

2.4.3 Plasma Imaging at Low Exposure

NOTE: a section was skipped here because it was discussed in section 4.

During Trial 3 (17 July 2019) the plasma imaged at successively higher frame rates with lower exposure times. The plasma properties were Magnetic Field Current = 200A, Power = 200W, Pressure = 2.31×10^{-3} Torr and Bluish Mode. During the trial there was a significant effort made to keep the plasma source at the center of the camera screen/image.

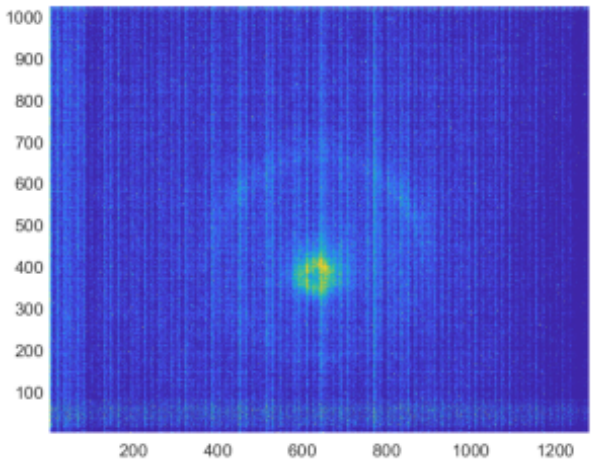


Figure 2.7: Average Image (c1a22), 1024 × 1280, 1057fps, 10.42µs exposure

Figure (7) shows the average image of a 10µs exposure taken at the normal camera resolution. The central bright region is the plasma source.

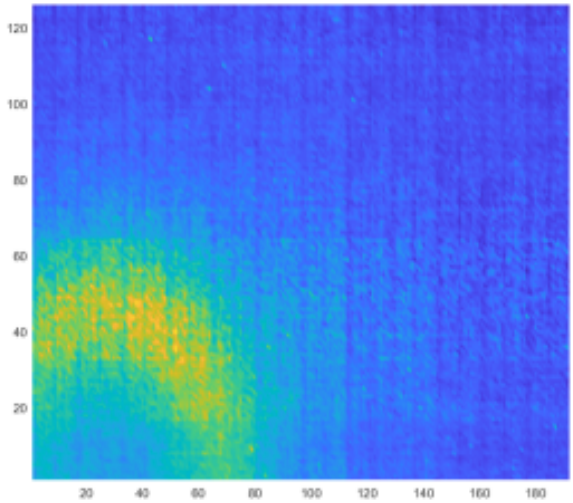


Figure 2.8: Average Image (c1a32), 126 × 192, 29711fps, 20.56µs exposure

Meanwhile, Figure (8) shows a 20µs exposure at the non-standard resolution of 126 × 192. The plasma source is at the corner in this case. These figures prove that at high frame rates and low exposure times the camera obtains enough light to form a good image of the plasma source. Therefore, now centering the source in the image needs to be worked on and the plasma properties have to be optimized for imaging.

2.5 New Design of the Camera Box

The previous design had a few issues that we're attempting to solve. The new design is composed of the base, cameras and their mounts, new lenses, guiding rods, the beam splitter and its mount, and the lid. This box will be as compact as possible while being more efficient than the previous one. For precise measurements check the CAD designs. The cameras need to be easily accessible, resulting in a short calibration time.

2.5.1 Base

The base is the most important component of the housing unit. An aluminum sheet will be the bottom side of the box, all the components will be secured in place on the aluminum sheet by screws. The wall of base will contain 5 openings: 1 opening for the light to enter, 2 for the cameras' screens and 2 for the cameras' wiring.

2.5.2 Camera Mounts

The box will contain two camera mounts, each one is made out of three parts. Firstly, we'll have the dedicated area of the camera. This area will be surrounded by a small edge that will hold the camera and stop it from moving in the plane of the base. Secondly, a permanent screw will secure the camera from the bottom. Lastly, an other screw, that

won't be so permanent, will secure the top side of the camera when it's fully calibrated and the lid is secured in place.

2.5.3 Lens

One of the key aspects of this upgrade/redesign is the lens. The lens used in the previous design was bigger than what is needed, which isn't a problem on its own until you try to set both lenses to the same exact focal length. A simple solution to this problem is to use a different type of lens that smaller and opening/closing it all the way. Adding to that the lens being used has a mechanism that locks the focal length after setting it up. The chosen lens is the 50mm Pentax C25011KP.

Features:

- Standard machine vision lens series
- Optical performance optimized for compatibility with VGA - XGA cameras
- Compatible with 1/2" - 1" format CCD and CMOS Cameras
- Lockable focus & iris rings

Figure 2.9: The Pentax C25011KP 50mm lens

2.5.4 Guiding Rods

The idea of adding guiding rods was a solution for the first design of the beam splitter's mount. Even though the problem was solved by redesigning the mount, the idea of guiding the lid and closing it perfectly every time is still a valid idea that will help avoid any undesired outcomes. The guiding system consists of four semi-threaded rods. The rods will be threaded from both ends only. One side will be permanently joined to the base and the other end will be used to secure the lid after closure.

2.5.5 Beam splitter mount

The beam splitter dimensions are 80x80x1 mm. Two designs for the mount were considered. The first design was composed of two small slits installed one on the base and the other on the lid. The slits were supposed to hold the lower and upper edge of the beam splitter. A problem arose with this idea, it was the difficulty of making sure that the

beam splitter will fit perfectly between the two slits every time the lid wad closed. The second idea involves a U-shaped frame where the lens will slide in (similar to a paper in a plastic sleeve). A small lid could be added, closing the U-shaped frame to further secure the glass.

2.5.6 Lid

The lid will have 4 holes in it aligned with the guiding rods so they can guide the lid (as their name suggests). Two additional holes will be drilled, those correspond to the cameras. The Lid-Base intersection might have some carving, when joined the lid and base will fit together like lego.

2.6 Software

2.6.1 Camera settings

Two of the the camera settings should be changed to work with the function generator.

- Access the “trigger/IO settings”. in the IO 1 section, we must select the “Shutter Gating” in order for the camera to trigger using the provided input. IO 2 and IO 3 are not selected. The ”none option” must be checked like the figure below. Note: if the phase difference is provided by the frequency generator, both cameras should have the same settings. Otherwise, the invert button should be selected in one of the cameras only.
- Back to the main menu we go to ”Record Settings”. Here we must change the ”Frame Rate” such that it is the lowest possible value that is greater than the frequency of the generated function. As the ”Shutter Gating” is On, the frame rate won’t be the one used to capture the video but the rate at which the camera checks for a change in the function. Therefore, we should set it higher than the frequency of the function in order for the camera to not miss a period.

2.6.2 Function Generator

The function generator is the tool used to sync the two cameras in order to make the imaging system work. Following the previous section concerning the camera settings, the camera is now using a square function as an input. Two function generators are available for this project, this section discusses how to set each of the up.

FeelTech

After turning on the function generator, a list of variables is shown on the screen. Use the F1-F5 keys next to the screen to switch between them. the arrows on the right are used to move the cursor from one digit to the next. After having the cursor on the desired digit, we change its value using the dial above the arrows.

1. Make sure the function generator is **not** connected to the camera.

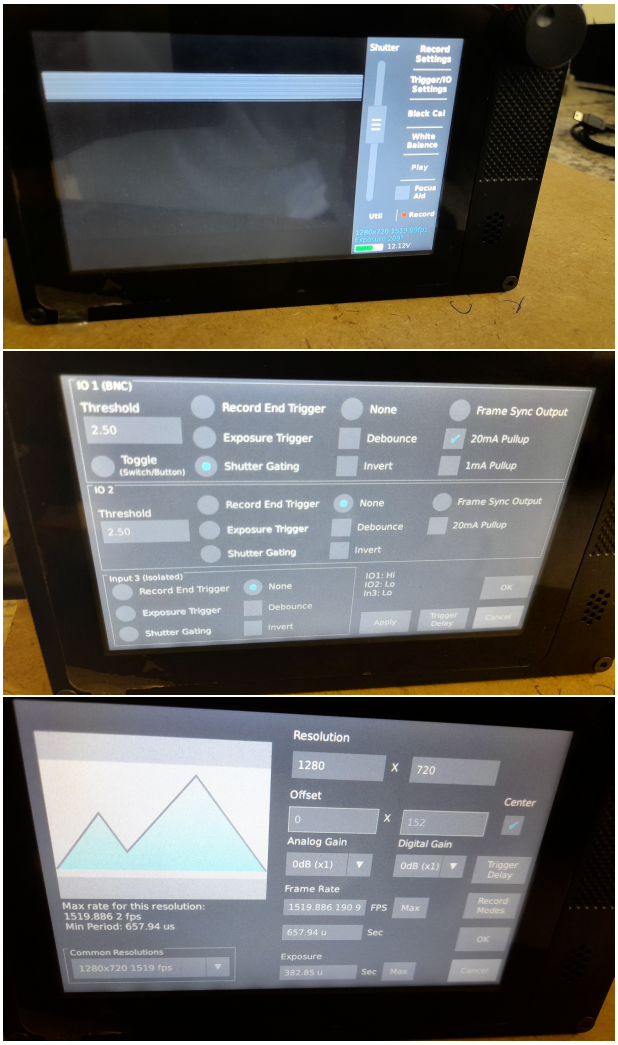


Figure 2.10: Photos showing the camera settings.

2. Plug the function generator in and turn it on.

3. The light below the CH1 and CH2 should be on. Choose CH1: the top menu shown on the screen should be the color of CH1 (yellow).

4. Press the "Wave" button on the far top-left to change the wave to a square wave.

5. The frequency of the wave is the frame rate of the video such that 1KHz corresponds to 1000 FPS. Set the frequency to the desired value. Note: the frequency is in KHz.

6. The amplitude should be 5V, changing that value could damage the camera. This value should be the default value.

7. The offset should be 0V. This value should be the default value.

8. The Duty cycle is going to set the exposure i.e. the duration the shutter will stay open. $Exposure = \frac{Duty\,Cycle}{Frequency}$. Smaller Duty cycle corresponds to a darker image with less motion blur.

9. The phase angle should be zero for CH1 and 180° For CH2. Note: in that case both cameras will have the same settings without one being inverted.

10. Select CH2 and go through everything again.

11. Before connecting the cameras, make sure the the amplitude is 5V and the offset is 0 on both channels.

12. The function generator can be turned off using the on/off button next to the screen without changing the settings.

13. Finally, connect the cameras and start recording. Noting that when the function generator is off the shutter will not move at all.

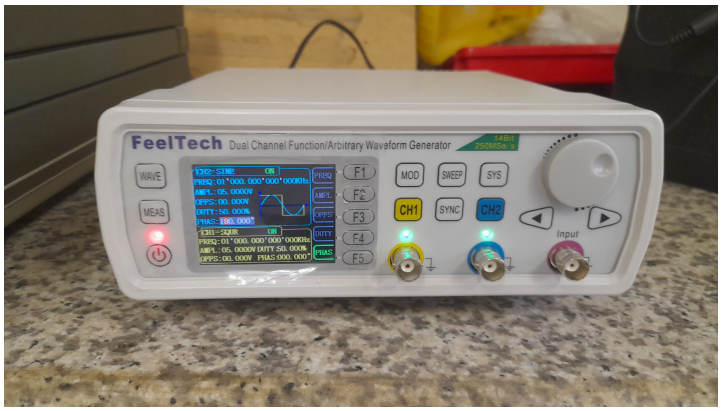


Figure 2.11: FeelTech function generator.

HP 8082A

2.6.3 One Camera

A (USB A - USB mini) cable is used to connect the camera to the computer. The camera can be accessed at the IP address 192.168.12.1 using the browser of choice. After accessing the camera, it can be controlled via the web page. For better monitoring and recording VLC media player is used. After installing and running VLC Media player, Access the “Network Stream” either by selecting t Media-¿Open Network Stream or by using the shortcut ctrl+N.

In the network settings paste the following in the URL section ”rtsp://192.168.12.1”. Check the more options box and change the caching to 500 ms for a smoother playback. Hit play to start streaming. In order to record the stream Advanced Controls needs to be enabled. Go to View > Advanced Controls to do so. Now you can see a record button on top of the play options at the bottom. Turn on the record when the stream starts and turn it off after. The recorded video will be saved to the computers ”videos” file.

2.6.4 Two Cameras

Connecting the system

As it turns out, in a two camera system a USB cable cannot be used to connect both cameras to one computer as the computer will ”see” them as having the same ID. The solution to that is to use a router connecting the computer to the cameras. The setup is done in the following manner. The cameras are connected to a switch/router via Ethernet cables, and so is the the used pc. Make sure to use a 1GB switch to be able to move big data from the camera to the pc while using higher frame rates. from the camera go to util > Network > ip address > turn off the DHCP and set a static IP address that is different for both cameras > hit apply and check in the Network status window that the camera is connect. the ip address should be shown after ”inet addr: ”. In case that did not work, simply factory reset the camera and try again.

SMB Share Setup

The following explains how to setup the system so that the cameras save the video taken directly to the specified folder on the pc. In this system we are using a 1GB switch so it can handle transferring the data as it is recorded at high frame rates and from both cameras at once.

Prerequisites:

- Windows 10
- Chronos 1.4 or 2.1
- Software version 0.4.0 or greater
- Connection over ethernet
- SMB Port 445 open on any applicable firewalls

Setup

1. Open windows features.
2. Find “SMB 1.0/CIFS File Sharing Support” expand and enable “SMB 1.0/CIFS Server” .
3. Create the receiving folders in the root directory of the main drive.
   Note: Do not use spaces or special characters in the folders names (one for each camera).
4. Share the folder with the user whose Windows login details will be used to connect from the cameras.
5. Determine your pc’s ”IPv4 Address” by going to command line (windows+R > cmd) . Type ”ipconfig /all” and find the ”IPv4 Address”.
6. On your cameras. Go to util > Network. In the ”Windows/SMB Network Storage” window enter the following:
   - **SMB Share:** $// < IPv4Address > / < Foldername >$
   - **Username:** $< Username(fromstep4) >$
   - **Password:** $< UsersPassword >$
7. If the camera is successfully set up ”SMB share” will be automatically present as a save location.
8. Test the system to make sure it works

Final setup

The switch we are using transfers up to 2GB/s. However, it is a layer 1 switch i.e. it doesn’t give IP addresses to our devices. We need to include a router in our setup in order to solve this problem. We connect the router, both cameras and the pc to the switch. All the data transfer will be done through the high speed switch.

Now on our pc we open the command line and type ”ipconfig”. we check the Etherent-Ethernet connection. We are interested in two parameters: the ”subnet mask” and the ”Default Gateway”. On our cameras we navigate to the Network tab and set a new static IP for each with the same subnet mask and Default Gateway obtained before. from cmd ping the cameras’ IP address to make sure they’re connected. The cameras can be accessed separately from any browser using the ip address or together using the multicam extension available on github. The camera control to start and stop recording is done through the web page and after the recording is over we must save it on the SD and download it on the pc as an mp4 file one camera at a time. The multicam extension will distinguish the files saved by camera. We need to find a way to save the files directly to the pc skipping the use of the SD card. However, at the moment we don’t have a way.

More info

More information can be found here.

Combining the footage

The process of combining the footage is done through 3 steps.

1- The first step is to extract the frames of the mp4 file. NOTE: we are saving the files as an mp4 and not as TIFF(Images) due to the fact that mp4 compresses the files to take less space and we don't lose any important data. The extraction is done through FFmpeg that you can download here. Open cmd/terminal and navigate to the directory where your video file is located, or provide the full path to your video file. Create three directories in the directory where the video file is. Call them '1', '2' and 'Done'. Run the following command specifying your video's name where it it specified.

```
ffmpeg -i <your_video.mp4> 1\frame_%09d.png
```

For the second video do the same step but change the '1' by '2' in order to save the new pictures in the second directory.

2- The second step is to run the python code which will combine the images flipping the mirrored ones and alternating between the images of camera one and camera two. Note: before running the code make sure that the frames match, i.e. make sure that one camera did not start recording a split second before the other, that can be done by finding a two consecutive frames from the first video and comparing them with the same frames from the other camera they should be continuous (flow perfectly from one to the other). If they weren't continuous, keep a frame from the first video and try to find the frame after it from the other camera's data. after determining the number of extra frames delete that number of frames from the the late video's frames starting from the beginning. Now your python file ,provided in the overleaf file of this document, should be in the same directory as the directories created before. Run the python code.

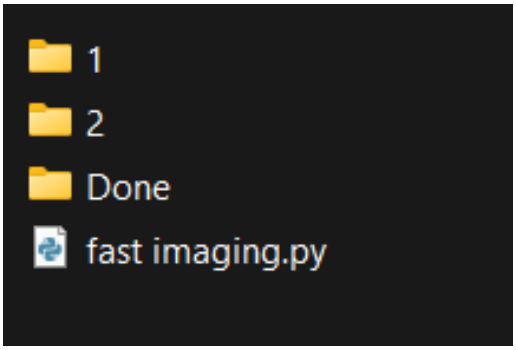


Figure 2.12: Directory before using the python file

3- Navigate to the 'Done' directory in cmd/Terminale and run the following code:

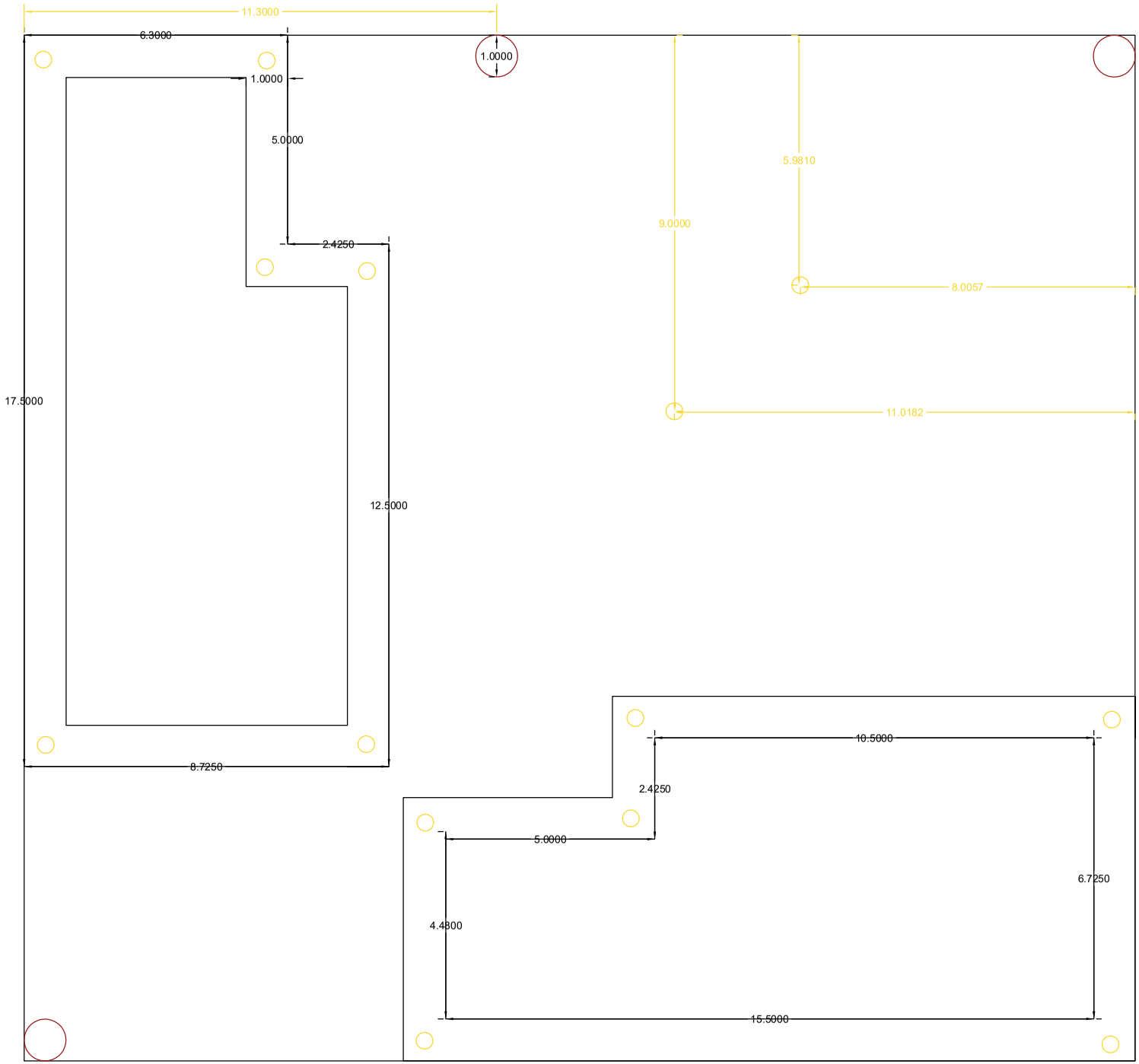
```
ffmpeg -framerate 30 -i %01d.png -c:v libx264 -r 30 -pix_fmt yuv420p output.mp4
```

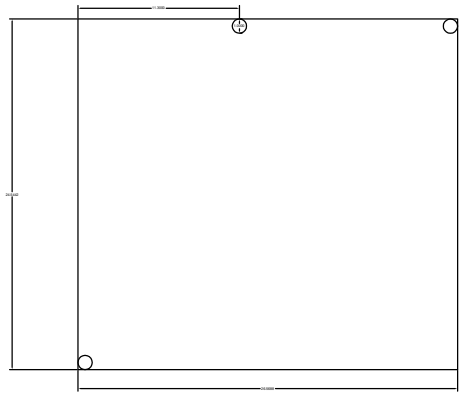
This code will combine the modified images into an mp4 file with a frame rate of 30 fps. you can modify the frame rate as needed by changing the value after "-framerate". Note: if you recorded 1000 Fps per camera you'd get 2000 frames per second, using the 30 fps combination will slow down the taken videos 200:3 which is 67 times slower than the original.

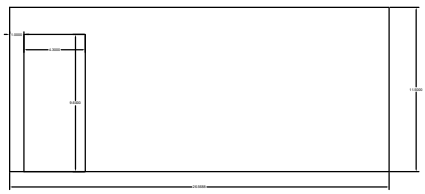
4- After making sure that the output.mp4 is as it should be, delete the contents of the "Done" directory as they take a lot of space and now they are compressed inside the mp4 file.

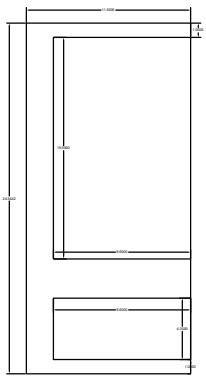
2.7 CAD designs

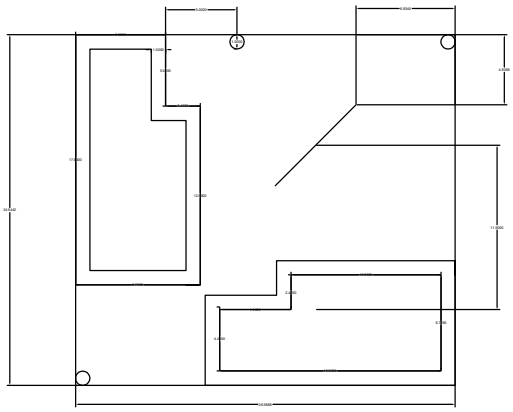
2.7.1 Design 1: Failed

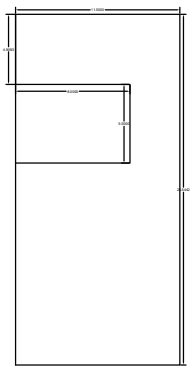

11.3000
6.3000
1.0000
1.0000
5.0000
2.4250
5.9810
9.0000
8.0057
17.5000
11.0182
12.5000
8.7250
10.5000
2.4250
5.0000
6.7250
4.4800
15.5000

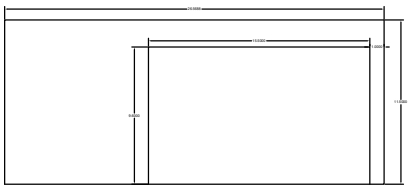




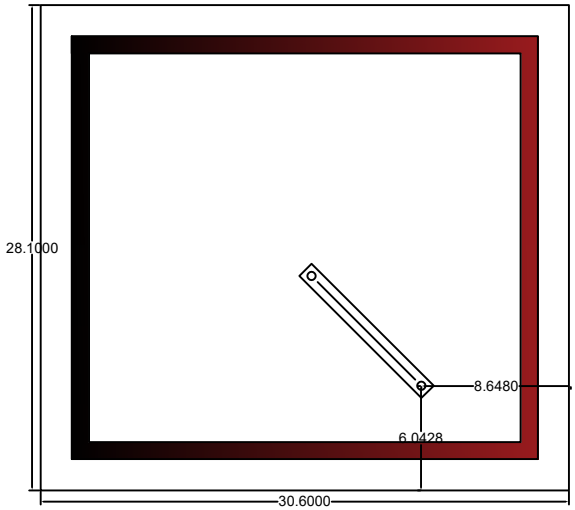


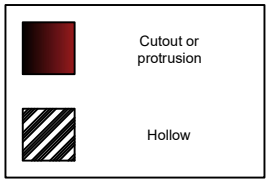


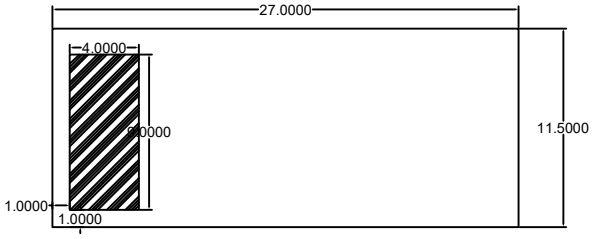


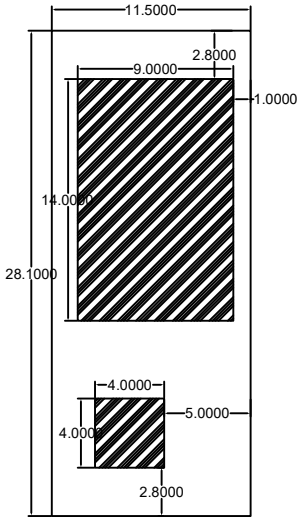


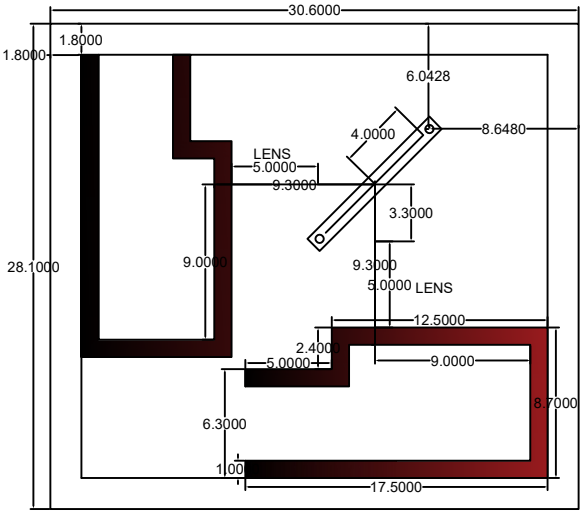
2.7.2 Design 2

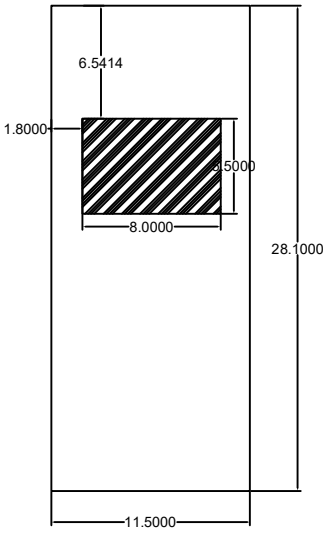

28.1000
8.6480
6.0428
30.6000

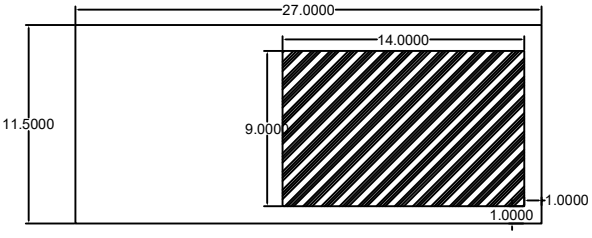

Cutout or
protrusion
Hollow


27.0000
4.0000
11.5000
1.0000
1.0000


11.5000
2.8000
9.0000
1.0000
14.0000
28.1000
4.0000
5.0000
4.0000
2.8000


30.6000
1.8000
1.8000
6.0428
4.0000
8.6480
LENS
5.0000
9.3000
3.3000
28.1000
9.0000
9.3000
5.0000
LENS
12.5000
2.4000
5.0000
9.0000
8.7000
6.3000
1.0000
17.5000


6.5414
1.8000
8.0000
28.1000
11.5000


27.0000
14.0000
11.5000
9.0000
1.0000
1.0000

Chapter 3

Stereoscopic Imaging

3.1 What is stereoscopic Imaging

Stereoscopic Imaging, in our case, is the use of two cameras a certain distance apart from each other to capture images at the same time and using the images in order to obtain the distance of a certain object from the cameras. It is a method of determining the depth. As we will see later, we'll use stereoscopic imaging in order to obtain the three-dimensional coordinates of the desired object which will allow us to determine its size, position, velocity, acceleration and many more parameters important for the assessing the threat posed by the object to our plasma flow.

3.2 Perfectly parallel cameras

We start our formula derivation by drawing a clear diagram and identifying our parameters. We distinguish two cases, one where the object lies between the optical axes and another where the object lies outside.

3.2.1 Object between the optical axes

We start by defining our parameters. $d_0 \equiv$ the distance separating the optical axes. $f \equiv$ the focal length of the used lens. $Z \equiv$ the distance between the lens and the object. $x_l, x_r \equiv$ the coordinates of the object on the image plane of the left and right camera resp. $X_L, X_R \equiv$ the distance between the object and the left and right optical axes resp. Finally $Z \equiv$ the Z coordinate, depth or the distance between the lens and the object. Note that Z is the same for both cameras as they are parallel with a translation in the X-direction only.

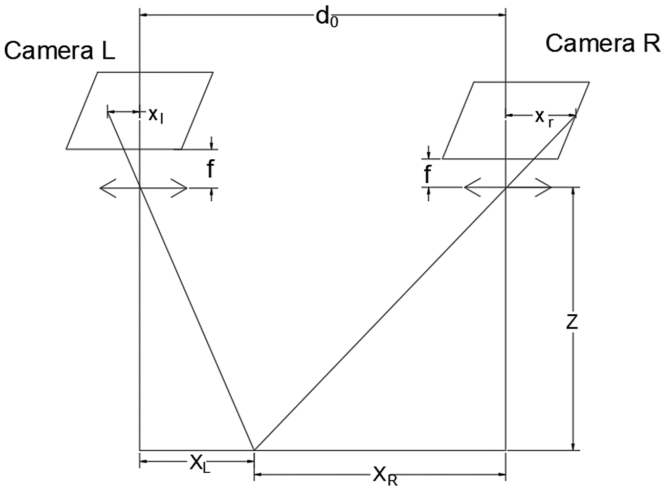


Figure 3.1: Diagram of an object between the optical axes

The goal here is to derive a formula that allows us to calculate Z as a function of x_l&x_r. We start by finding an equation relating them. Geometrically, we can see that:

$$\frac{x_l}{f} = \frac{X_L}{Z} \tag{3.1}$$

$$\frac{x_r}{f} = \frac{X_R}{Z} \tag{3.2}$$

We know that f and Z are the same for both cameras.

$$\Longrightarrow \frac{Z}{f} = \frac{X_L}{x_l} = \frac{X_R}{x_r} \tag{3.3}$$

We also know that the Object is in between the two optical axes which gives us the following:

$$d_0 = X_L + X_R \tag{3.4}$$

Solving the system we get from eq.(3.3) and (3.4) we get:

$$X_R = \frac{d_0}{x_l}\left(\frac{1}{x_l} + \frac{1}{x_r}\right)^{-1}$$

We substitute X_R in eq. (3.2):

$$Z = \frac{d_0\, f}{x_r + x_l} \tag{3.5}$$

This is the formula for Z as a function of x_l&x_r we were looking for.

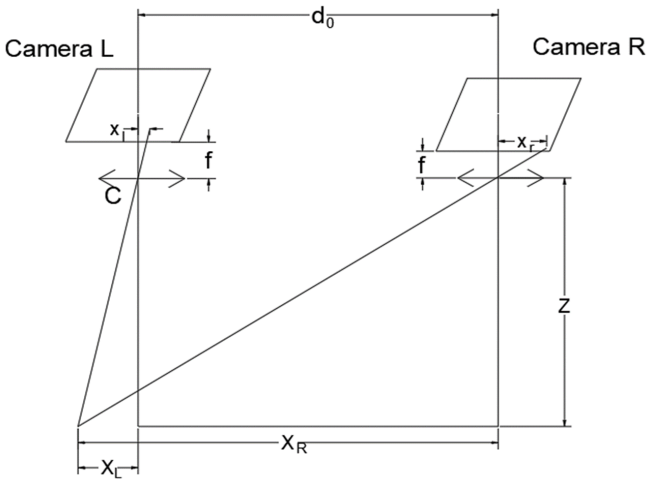


Figure 3.2: Diagram of an object outside the optical axes

3.2.2 Object outside the optical axes

In this case, eq. (3.1), (3.2), (3.3) hold up. However, (3.4) becomes:

$$d_0 = -X_L + X_R \tag{3.6}$$

Solving the system now gives the following:

$$X_L = \frac{d_0}{x_r}\left(\frac{1}{x_l} - \frac{1}{x_r}\right)^{-1}$$

Finally:

$$Z = \frac{d_0\, f}{|x_r - x_l|} \tag{3.7}$$

This equation is also valid for $X_L > X_R$ due to the absolute value of the denominator.

3.2.3 Full Coordinates

Both cameras have the same Z. But they do not have the same X. Therefore, we choose by convention one of the cameras to be our reference frame. From now on **the Left camera will be the our reference frame**.

Eq. (3.1) gives us:

$$X = X_L = \frac{x_l\, Z}{f} \tag{3.8}$$

and similarly for Y:

$$Y = Y_L = \frac{y_l\, Z}{f} \tag{3.9}$$

Our coordinates w.r.t. the Left camera are:

$$\begin{cases} X = X_L = \frac{x_l \, Z}{f} \\ Y = Y_L = \frac{y_l \, Z}{f} \\ Z = \frac{d_0 \, f}{|x_r \pm x_l|} \end{cases} \tag{3.10}$$

3.2.4 Uncertainty Calculation

To proceed with these calculations we need to define some new parameters. Let $\delta x \equiv$ pixel error which is the error on selecting the exact same pixel in both images, in general we take it equal to 450px which is the size of the cursor tip. $p \equiv$ pixel size or more accurately pixel length, this is the size of each pixel on the camera's sensor. the pixel is square which means its size will be AxA which is an area p is equal to the value of A. Finally, $D \equiv$ denominator of Z from eq. (3.10). In this part we are going to assume that there's no error on f and d_0. The error on them can be added later.

Z Uncertainty

the denominator uncertainty is directly related to the pixel error. However, δx is in pixels and we need the uncertainty to be in unit length. Therefore, we need to multiply it by the pixel size which in expressed in the units of $\mu m/px$:

$$\delta D = \delta x \times p$$

The uncertainty on Z becomes:

$$\delta Z = Z \frac{\delta D}{D} = \frac{f \; d_0}{D^2} (p \; \delta x) \tag{3.11}$$

X and Y Uncertainty

From the relation in eq. (3.10) we can determine the uncertainty to be:

$$\begin{cases} \delta X = \frac{\delta x \; p \; \delta Z}{f} \\ \delta Y = \frac{\delta x \; p \; \delta Z}{f} \end{cases} \tag{3.12}$$

Coordinates Uncertainty

$$\begin{cases} \delta X = \frac{\delta x \; p \; \delta Z}{f} \\ \delta Y = \frac{\delta x \; p \; \delta Z}{f} \\ \delta Z = \frac{f \; d_0}{D^2} (p \; \delta x) \end{cases} \tag{3.13}$$

3.3 Parallel Camera Device

3.3.1 Support

A device was designed to hold the cameras in this case we are using the Basler acA3800-14uc. A support was designed to hold both cameras at a distance $d_0 = 10cm$.

The device's stl file is in the extra files folder in the overleaf project.

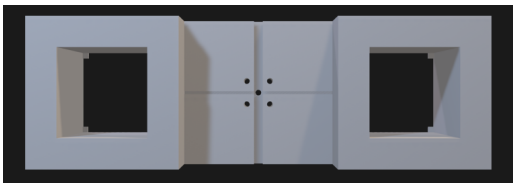
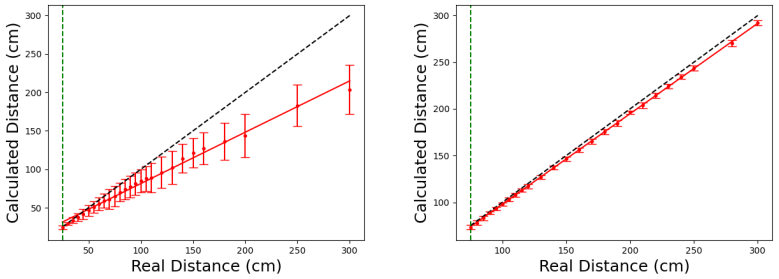


Figure 3.3: The device designed for Stereoscopic Imaging

3.3.2 Results of the Support

Two supports were printed by the time I am writing this. The first had a small place for the camera then we had to sand it down to make the cameras fit. The act of sanding it down resulted in a deviation between the two cameras which produced a lot of error. Another support was printed after fixing the 3D model and the results were much better. The CSV files of the two tests can be found in the extra files folder.

The test consisted of taking multiple images at well known distances and running the code that gave us the calculated distance and compared the results of the calculated distance with the real distance. The test was run 10 times for each distance and the mean values were plotted.



(a) Plot of the first print's data (b) Plot of the second print's data

Figure 3.4

In the figures the green vertical line represents the minimal distance where both cameras will see the object. Any distance less than that the object will appear in the image taken by one camera and not the other. The black diagonal line is the first bisector, i.e. it represents the best case scenario where the calculated distance matches the calculated distance. The red dots are our data points and the red line is the fitting function.
We can clearly see that in the figure (a) the data points diverged drastically away from the first bisector giving a max error at 300 cm $> 30\%$.
However, figure (b) shows a much better dataset where the calculated distance is extremely close to the real values with a maximum error $\approx 4.5\%$.

We believe that the error of the first device is a result of the cameras not being ac-

tually parallel. Due to the first print having a smaller space for the camera, we resorted to sanding the inner part if the device which must have created an angle between the the two cameras. In the next part we are going to discuss a way to account for the angle between the two cameras in our calculation.

Note: 10 tests were run using the python code, available in the extra files folder, and the resulting plots are the of the mean values of the 10 tries. I also included a csv file of a test run from each device.

3.4 Non-parallel Cameras

A problem arose in the last part with the first print. We concluded that the problem was due to the cameras not being parallel. In this section we will develop the mathematics for transforming the coordinates obtained a rotated camera to ones we would get from a camera perfectly parallel to the other. As we discussed in the previous section we will use the reference frame of the left camera. This means that we are going to take the coordinates from the left camera as they are and we apply our matrix that we'll get to the right camera.

3.4.1 Rotation

An important idea to keep in mind is that even though the camera's sensor records images in two dimensions it still is a three dimensional object and can rotates along three axes. The axes are defined as previously mentioned in the coordinates section, i.e. if you are looking through the camera's lens the X-axis is toward the right, the Y-axis is towards the bottom, and the Z-axis going into the image.

Rotation along the X-axis

Note: by a rotation along the X-axis we mean a rotation in the Y-Z plane following the right hand rule rule where the thumb points along X.

We define the angle of rotation along X as θ. We deduce from the diagram that the matrix $R_X(\theta)$ is the following:

$$R_X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \tag{3.14}$$

In order to study the effect of this rotation on the coordinates we get from the camera we apply it to the 1 1 1 vector:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(\theta) & -sin(\theta) \\ 0 & sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ cos(\theta) - sin(\theta) \\ cos(\theta) + sin(\theta) \end{bmatrix}$$

The R_X matrix won't change the X coordinates and it changes the Y and Z coordinates as shown in the resulting vector. Note: we are only interested in how the matrix changes the X and Y coordinates as they are the data we're extracting from the 2D picture.

Rotation along the y-axis

This is a rotation in the $Z - X$-plane as defined above. We define the angle of rotation along y as ϕ. The rotation matrix becomes:

$$R_Y(\phi) = \begin{bmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{bmatrix} \tag{3.15}$$

$$\begin{bmatrix} cos(\phi) & 0 & sin(\phi) \\ 0 & 1 & 0 \\ -sin(\phi) & 0 & cos(\phi) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\phi) + sin(\phi) \\ 1 \\ cos(\phi) - sin(\phi) \end{bmatrix}$$

Rotation along the Z-axis

This is a rotation in the X-Y plane as defined above. We define the angle of rotation along Z as ψ. The rotation matrix becomes:

$$R_Z(\psi) = \begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.16}$$

$$\begin{bmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\psi) - sin(\psi) \\ cos(\psi) + sin(\psi) \\ 1 \end{bmatrix}$$

As we are only interested in the X and Y coordinates we can see how a rotation along the Z-axis can cause the more error than the rest as it changes both the X and Y coordinates.

The General Rotation Matrix

In real life the camera will be in fact rotated along all of the axes as nothing is perfect. In case you haven't noticed yet the angles θ, ϕ and ψ are the so-called Euler angles. We are going to use the Euler convention for computing the general rotation matrix;

$$R \equiv R_Z(\psi)\ R_Y(\phi)\ R_X(\theta)$$

Note: Euler uses different angle naming.

3.4.2 Angle Determination first step

We need a method of accurately determining the three angles. The main idea here is to use print of a square where we know the side angles, the next part is to determine the angles of each camera w.r.t. the test square and use those angles to find the ones of the right camera w.r.t. the left one.

The following will help us determine the angles from the image directly as it is just a scaled projection of real life objects.

Pre-requisites

Recall Eq. (3.10), we can get the projection equation from it which is x_l as a function of X_L. For convenience in this part we will call X_L simply x and x_l will be x'. Similarly, y and y' for Y_L and y_l resp.

$$\implies \begin{cases} x' = \frac{f\ x}{z} \\ \\ y' = \frac{f\ y}{z} \end{cases} \tag{3.17}$$

We will assume a reference frame where the top left corner of our square is the origin (0, 0, 0). The square is shown below with the coordinates of its corners. The square has sides of length a.

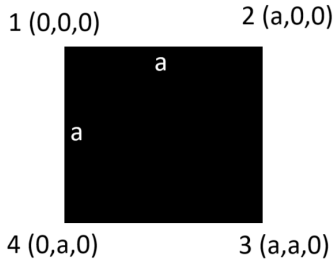


Figure 3.5: Diagram of the test square.

Let us assume a camera at a position (x_c, y_c, z_c) the projection equations become the following:

$$\implies \begin{cases} x' = \frac{f\ (x-x_c)}{z-z_c} \\ \\ y' = \frac{f\ (y-y_c)}{z-z_c} \end{cases} \tag{3.18}$$

In the following calculations we are going to use the following coordinates for the camera (0,0,-d) to make calculations easier. As we will see later, the angles are independent from the camera coordinates.

No Rotation

With no rotation, we should directly apply the projection formulas to obtain the coordinates as they appear on the image.

$$\begin{cases} 1 : (0,0,0) \rightarrow (0,0) \\ 2 : (a,0,0) \rightarrow (\frac{f\ a}{d}, 0) \\ 3 : (a,a,0) \rightarrow (\frac{f\ a}{d}, \frac{f\ a}{d}) \\ 4 : (0,a,0) \rightarrow (0, \frac{f\ a}{d}) \end{cases}$$

We define $L_T \equiv$ Length of the top side as seen by the camera, $L_B \equiv$ that of the bottom side, $L_L \equiv$ that of the left side and $L_R \equiv$ that of the right.

$$\implies \begin{cases} L_T = \frac{f\ a}{d} \\ L_B = \frac{f\ a}{d} \\ L_L = \frac{f\ a}{d} \\ L_R = \frac{f\ a}{d} \end{cases}$$

With no rotation the sides are only affected by the 3D $\rightarrow$ 2D projection.

Rotation along X

For rotation along the x-axis we use the rotation matrix $R_X(\theta)$ on our corner coordinates before we apply the projection.

$$\begin{cases} 1 : (0,0,0) \rightarrow (0,0,0) \rightarrow (0,0) \\ 2 : (a,0,0) \rightarrow (a,0,0) \rightarrow (\frac{f\ a}{d}, 0) \\ 3 : (a,a,0) \rightarrow (a, acos(\theta), asin(\theta)) \rightarrow (\frac{f\ a}{asin(\theta)+d}, \frac{f\ a\ cos(\theta)}{asin(\theta)+d}) \\ 4 : (0,a,0) \rightarrow (0, acos(\theta), asin(\theta)) \rightarrow (0, \frac{f\ a\ cos(\theta)}{asin(\theta)+d}) \end{cases}$$

Calculating the side lengths we get:

$$\implies \begin{cases} L_T = \frac{f\ a}{d} \\ L_B = \frac{f\ a}{asin(\theta)+d} \\ L_L = \frac{f\ a\ cos(\theta)}{asin(\theta)+d} \\ L_R = \frac{f\ a\ cos(\theta)}{asin(\theta)+d} \end{cases}$$

The first thing we notice is that the left and right sides are equal. Even though they are different than what we showed they are for no rotation they still yield the same value. That is due to both sides rotating the same amount which scales both by $cos(\theta)$. The top and bottom sides are different and they are expressed as a function of the angle θ.

$$\begin{cases} \frac{L_L}{L_R} = 1 \\ \frac{L_T}{L_B} = \frac{asin(\theta)+d}{d} \end{cases}$$

From the second equation we can deduce the value of theta to be

$$\theta = arcsin\left[\frac{d}{a}\left(\frac{L_T}{L_B} - 1\right)\right] \tag{3.19}$$

Rotation along Y

For the rotation along the y-axis we use the rotation matrix $R_Y(\phi)$ just like we did for the rotation along X.

$$\begin{cases} 1 : (0,0,0) \rightarrow (0,0,0) \rightarrow (0,0) \\ 2 : (a,0,0) \rightarrow (a\ cos(\phi), 0, -a\ sin(\phi)) \rightarrow (\frac{f\ a\ cos(\phi)}{d - asin(\phi)}, 0) \\ 3 : (a,a,0) \rightarrow (a\ cos(\phi), a, -a\ sin(\phi)) \rightarrow (\frac{f\ a\ cos(\phi)}{d - asin(\phi)}, \frac{f\ a}{d - asin(\phi)}) \\ 4 : (0,a,0) \rightarrow (0,a,0) \rightarrow (0, \frac{f\ a}{d}) \end{cases}$$

Calculating the side lengths we get

$$\Longrightarrow \begin{cases} L_T = \frac{f\ a\ cos(\phi)}{d - a\ sin(\phi)} \\ L_B = \frac{f\ a\ cos(\phi)}{d - a\ sin(\phi)} \\ L_L = \frac{f\ a}{d} \\ L_R = \frac{f\ a}{d - a\ sin(\phi)} \end{cases}$$

In this case top and bottom sides are equal while the left and right sides vary which is opposite to that previous rotation.

$$\begin{cases} \frac{L_T}{L_B} = 1 \\ \frac{L_L}{L_R} = \frac{d - a\ sin(\phi)}{d} \end{cases}$$

$$\phi = arcsin\left[\frac{d}{a}\left(1 - \frac{L_L}{L_R}\right)\right] \tag{3.20}$$

Rotation along Z-axis

The rotation along Z is in the $X-Y$-plane which means we can measure it directly from the image as the the angle between the horizontal and the top side. Note that the angle ψ is positive in the direction opposite to normal convention due to the chosen reference, frame i.e. it is positive in the clockwise direction.

$$\psi = \frac{y2 - y1}{L_T}$$

Where y2 and y1 are the y coordinates of the points 1 and 2.

Combining all of the Rotations

Finally, we have determined a way to determine the angles assuming only one type of rotation took place. Before we proceed any further we need to make sure that the angles do not interfere with the parameters we are using to determine the others.

We should check to see if the rotation along Z effects the ratios of the side lengths. We follow the same steps used in the previous calculations.

$$\begin{cases} 1 : (0,0,0) \rightarrow (0,0,0) \rightarrow (0,0) \\ 2 : (a,0,0) \rightarrow (a\ cos(\psi), a\ sin(\psi), 0) \rightarrow (\frac{f\ a\ cos(\psi)}{d}, f\ a\ sin(\psi)d) \\ 3 : (a,a,0) \rightarrow (a(cos(\psi) - sin(\psi)), a(cos(\psi) + sin(\psi)), 0) \rightarrow (\frac{f\ a\ (cos(\psi) - sin(\psi))}{d}, \frac{f} \\ 4 : (0,a,0) \rightarrow (-a\ sin(\psi), a\ cos(\psi), 0) \rightarrow (\frac{-f\ a\ sin(\psi)}{d}, f\ a\ cos(\psi)d) \end{cases}$$

Calculating the side lengths we get:

$$\Longrightarrow \begin{cases} L_T = \frac{f\ a}{d} \\ L_B = \frac{f\ a}{d} \\ L_L = \frac{f\ a}{d} \\ L_R = \frac{f\ a}{d} \end{cases}$$

A rotation along Z has no effect on the side lengths and the ratios. Therefore, ψ does not effect the parameters we are using to calculate θ & ϕ. And we already proved that θ & ϕ have no effect on the ratios we are using to calculate the other.

But do the x and y rotations effect the Z parameters? The answer is yes. For starters, θ **AND** ϕ effect all side lengths. We need two lengths to calculate the ψ they are L_T which is in our case diagonal

$$\psi = arcsin\left[\frac{y_2 - y_1}{L_T\ cos(\phi)}\right]$$

With that being determined we can finally write the angle as determined from the images:

$$\begin{cases} \theta = arcsin\left[\frac{d}{a}\left(\frac{L_T}{L_B} - 1\right)\right] \\ \phi = arcsin\left[\frac{d}{a}\left(\ 1 - \frac{L_L}{L_R}\right)\right] \\ \psi = arcsin\left[\frac{y_2 - y_1}{L_T\ cos(\phi)}\right] \end{cases} \tag{3.21}$$

A small problem arises from this method. The angle in the Z axis causes the horizontal and vertical lines to be in the other's domain meaning that the rotations along X and Y will affect both the horizontal and vertical lines. This method is a good enough approximation foe small ψ.

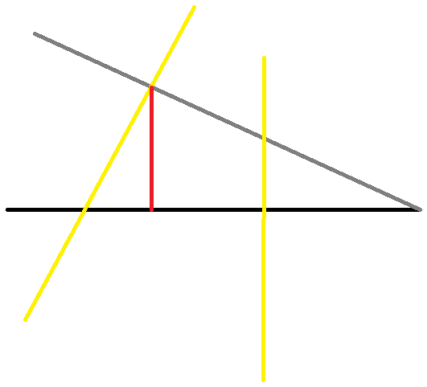


Figure 3.6: Picture showing the angled sensor

3.4.3 Using the Matrix

Now that we have calculated the rotation matrix and we able to use it, in the code we start by calibrating the cameras and finding the angles of rotation, then we proceed to determine the points we want to calculate the distance to in the pictures. We find the Z coordinate from the equation and then apply the R matrix to our coordinates. After running the test the results were not better than before. The problem might be that our matrix is missing a transnational part.

The black line shows the sensor in it is desired position, and the grey line is the angled sensor. If the light was perpendicular to the desired direction the rotation matrix will correct the correction perfectly. However, the light ray is at an angle the the rotation matrix will not work well. it can either lessen the error or make it worse. This method is not complete and we need to account for the translation factor and make the whole process more compact.

3.4.4 A more compact formula

Homogeneous Coordinates

Consider a 2D vector (x,y). Using a 2×2 matrix we can achieve the following operations: Scaling, Shearing, Reflecting, Rotation and any of their combinations. However, we can not translate the object using a matrix. We can achieve it by adding another 2D vector to it with the translation we desire as the components.

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \end{bmatrix}$$

A more convenient approach is to add a new coordinate set to 1 in our vector and incorporating a new column and row in the matrix. By placing the translation values at

the upper right of the matrix, we achieve the exact same outcome.

$$\begin{bmatrix} 1 & 0 & d_x \\ 0 & 1 & d_y \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + d_x \\ y + d_y \\ 1 \end{bmatrix}$$

Now we can put all our transformations in one matrix again. The previous transformation we've seen, i.e. the combination of all the 2×2 matrices can be put at the top left corner of our 3×3 matrix. For example, incorporation the rotation matrix in the 3×3 matrix we'll be the equivalent of rotating first and then translating. We can also add the translation vector to the mix in any order we want if all the matrices were turned into 3×3, which is typically not doable with the vector sum. And in a code we reduce the amount of operations by half which improves performance and makes debugging easier. This is specifically used when we have an object represented in local coordinates and we want to express it in absolute coordinates or another local coordinate system. Which is done by translating the coordinate system.

Another advantage to this is that we can lock some vector in place by setting it's new coordinate to 0 which locks it into place by nullifying the d_x and d_y terms when the matrix is applied to it.

What if the homogeneous coordinate is not 1 nor 0 but an arbitrary real number we call w? The rule here is to divide every component of the vector by the homogeneous coordinate to retrieve the original vector. Bringing the homogeneous coordinate back to 1 while re-scaling the other components.

$$\begin{bmatrix} x \\ y \\ w \end{bmatrix} \implies \begin{bmatrix} x/w \\ y/w \\ 1 \end{bmatrix}$$

Using this convention, we can ,modify the last row of our matrix to achieve division by certain coordinates.

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ y \end{bmatrix} \implies \begin{bmatrix} x/y \\ y/y \\ 1 \end{bmatrix}$$

Using these operations to derive the matrix

We need to find a matrix that takes as input the world coordinates and outputs the one perceived by the camera. Matrix P defined as

$$x = PX$$

where x are the projection coordinates and X the coordinates in the real world. As usual we are going to start in the perfect world scenario and proceed to add operations that are applied before that in order to transform the real world scenario to the perfect one.

We are going to use the homogeneous coordinates from the beginning to be ready for the translation.

From eq (3.17) we can see that both x' and y' are inversely proportional to z. Taking a case where f=1 we can write it in homogeneous coordinates as

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \rightarrow \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix}$$

This is are the coordinates of the projection. Our 3D coordinates of the real world are:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

The first thing we notice is that our input vector has 4 components and the output has 3. Therefore, the matrix we need in order to achieve this projection should be a 3×4 matrix. The last column is for translation, as we discussed in the previous section, so it has components of zeros for the time being. the rest is the transformation 3×3 matrix which in this case is the Identity.

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = [I|0]$$

For a focal length that isn't 1 we need to multiply x and y by f as determined by the formula we found before.

$$\implies P = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Note: one must remember that we are working with 3 coordinate systems not 2. There is the coordinate system of the world, that of the camera and that of the image plane. In general the camera and the image plane have different coordinate systems because the origin of the camera is different than the origin of the sensor. We should account for that in the camera matrix. let's assume the transformation is a shift denoted by the components p_x and p_y:

$$\implies P = \begin{bmatrix} f & 0 & d_x & 0 \\ 0 & f & d_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & d_x \\ 0 & f & d_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = K[I|0]$$

K is the homogeneous transformation from 2D to 2D accounting for focal length and origin shift. $[I|0]$ is the perspective projection 3D to 2D assuming that image plane is at $z = 1$.

Now we need to find the transformation between the world coordinate system and the camera coordinate system. By convention we'll use the "Tilde" on top of a coordinates to indicate that they are taken w.r.t. the real world system. To change the coordinate system from the real world one to that of the camera we can't simply subtract the coordinates of the camera system origin, in the world reference frame, from the value of the coordinates we want to transform. That was demonstrated earlier because we need to account for rotation present between the two reference frames. The transformation becomes:

$$X_c = R.(\tilde{X_w} - \tilde{C})$$

This equation is in heterogeneous coordinates. We need to find it in homogeneous coordinates in order to add it to our P matrix.

We expand the equation and it becomes:

$$X_c = R\tilde{X_w} - R\tilde{C}$$

Note that $-R\tilde{C}$ is the translation vector. In homogeneous, want to find the equation such that:

$$\begin{bmatrix} X_c \\ 1 \end{bmatrix} = T \begin{bmatrix} \tilde{X_w} \\ 1 \end{bmatrix}$$

Where T is a 4×4 matrix accounting for Rotation R and translation -R$\tilde{\text{C}}$. T can be written as:

$$T = \begin{bmatrix} R & -R\tilde{C} \\ 0^T & 1 \end{bmatrix}$$

Where R the 3×3 rotation matrix and 0^T is a 1×3 row vector of zeros.

$$\implies P = \begin{bmatrix} f & 0 & d_x \\ 0 & f & d_y \\ 0 & 0 & 1 \end{bmatrix} [I|0] \begin{bmatrix} R & -R\tilde{C} \\ 0^T & 1 \end{bmatrix} = \begin{bmatrix} f & 0 & d_x \\ 0 & f & d_y \\ 0 & 0 & 1 \end{bmatrix} [R| - R\tilde{C}]$$

Finally we have the equation of P which can be written as:

$$P = KR[I| - C] = K[R|t] \tag{3.22}$$

where $t = -R\tilde{C}$

K contains the intrinsic parameter of the camera and $[R|t]$ contains the extrinsic parameters. The matrix P will help us calibrate the cameras then we can use the equations above in order to calculate the coordinates.

Uncalibrated Stereo Problem

We have a scenario of a 3-dimensional scene and two pictures of this scene captured using two different devices. The first camera took a picture of the left view of a static scene, while the second camera took the right view of the same static scene. The problem is to compute the three-dimensional structure of this scene from these two pictures.

We assume that we know the internal parameters of the two cameras. The internal parameters are the focal lengths f_x, f_y and the location of the principal point O_x, O_y.

We assume that this information (internal parameters) is available to us for each image we use. These internal parameters are known as the intrinsic and are known for both views/cameras.

What we are interested in finding to calibrate our uncalibrated stereo system are the extrinsic parameters, also known as the external parameters of these two cameras. These extrinsic parameters are the relative position and the orientation of one camera with respect to another.

Calibrating the Stereo System

The first step in uncalibrated stereo is to calibrate the stereo system. We have a scenario of a 3-dimensional scene and our two cameras (the left camera and right camera). The left camera has a three-dimensional coordinate frame and is situated at O_l, and the right camera has its own frame at O_r. We do not know the relationship between O_l and O_r. The image coordinate for the left camera is given by $\hat{U}_l$, and the image coordinate for the right camera is given by $\hat{U}_r$.

The first goal is to find the external parameters (extrinsic parameters). We want to find the relationship between the 3-dimensional coordinate frames of the two cameras (the rotation and the translation). To do this, we want to find a small number of reliable correspondences between these two arbitrary views (find a set of corresponding features, at least 8, in the left and right images). By finding these set of correspondences, we can find the rotation and translation of each camera with respect to the other camera. This means we can find the relative position and the orientation. We call the translation T and the rotation R.

Now, this uncalibrated stereo system is being calibrated. Using this information, we find dense correspondences. For every point in the left image, we want to find where the corresponding point lies in the right image. Once we find it, we can use the triangular shape and find the location of points in the scene (computing depth).

Epipolar Geometry

Our goal is to find the relative position and orientation of one camera with respect to the other camera. The relative position and orientation between the two cameras are completely described by the epipolar geometry of the stereo system.

In our scenario, we have left and right cameras with 3D coordinate frames positioned at O_l and O_r respectively. The relationship between these frames is given by the translation T and rotation R. We introduce the epipoles, where the epipole e_l is the projection of the center of the left camera onto the right camera's image, and the epipole e_r is the projection of the center of the right camera onto the left camera's image.

The epipolar plane includes the points P (a scene point), O_l, O_r, and the epipoles e_l and e_r. This plane is called the epipolar plane of the scene point P.

Epipolar Plane and Constraint

We are interested in the epipolar plane. There is a vector that is normal to the epipolar plane, which we call $\mathbf{n}$. This normal vector can be calculated as the cross product of

the unknown translation vector $\mathbf{t}$ with the position of the point P in the left camera's coordinate frame $\mathbf{X}_l$:

$$\mathbf{n} = \mathbf{t} \times \mathbf{X}_l$$

The dot product of $\mathbf{n}$ and $\mathbf{X}_l$ (since they are perpendicular vectors) is zero:

$$\mathbf{X}_l \cdot (\mathbf{t} \times \mathbf{X}_l) = 0$$

This is known as the epipolar constraint.

The epipolar constraint can be written in matrix form as follows:

$$\begin{bmatrix} X_l & Y_l & Z_l \end{bmatrix} \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix} = 0$$

This matrix is called the translation matrix $(\mathbf{t}\times)$ with $\mathbf{X}_l$, the coordinates of the point P in the left camera frame.

We know that $\mathbf{t}$ is the position of the right camera in the left camera's coordinate frame, and $\mathbf{R}$ is the orientation of the left camera in the right camera's coordinate frame. Using $\mathbf{t}$ and $\mathbf{R}$, we can relate the three-dimensional coordinates of a point P in the left camera to the three-dimensional coordinates of the same point in the right camera:

$$\mathbf{X}_l = \mathbf{R}\mathbf{X}_r + \mathbf{t}$$

Expanding this equation, we have:

$$\begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Now, substituting into the epipolar constraint gives:

$$\begin{bmatrix} X_l & Y_l & Z_l \end{bmatrix} \left(\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} + \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right) = 0$$

Essential Matrix

We previously defined the matrix equation:

$$\begin{bmatrix} X_l & Y_l & Z_l \end{bmatrix} \left(\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} + \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \right) = 0$$

Since $\mathbf{t} \times \mathbf{t} = 0$, the second term simplifies to zero. Hence, the equation simplifies to:

$$\begin{bmatrix} X_l & Y_l & Z_l \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = 0$$

where $\mathbf{E}$ is the essential matrix defined by:

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$$

Here, $[\mathbf{t}]_\times$ is the skew-symmetric matrix of the translation vector $\mathbf{t}$, and $\mathbf{R}$ is the rotation matrix. Specifically:

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

The essential matrix $\mathbf{E}$ is computed as:

$$\mathbf{E} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

where $\mathbf{T} \times \mathbf{R}$ represents the product of the skew-symmetric matrix $[\mathbf{t}]_\times$ and the orthonormal rotation matrix $\mathbf{R}$.

Computing the Essential Matrix

To compute the essential matrix $\mathbf{E}$, follow these steps:

1. Obtain the rotation matrix $\mathbf{R}$ and translation vector $\mathbf{t}$.
2. Form the skew-symmetric matrix $[\mathbf{t}]_\times$.
3. Compute the essential matrix using the formula $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$.

Once the essential matrix $\mathbf{E}$ is known, it is possible to recover the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$ using Singular Value Decomposition (SVD) of $\mathbf{E}$.

We know that:

$$\mathbf{X}_l^T \mathbf{E} \mathbf{X}_r = 0$$

where:

$$\mathbf{X}_l = \begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix}$$

$$\mathbf{E} = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix}$$

$$\mathbf{X}_r = \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix}$$

The equation:

$$\begin{bmatrix} X_l & Y_l & Z_l \end{bmatrix} \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix} = 0$$

means that $\mathbf{X}_l$ and $\mathbf{X}_r$ are 3D coordinates of the same scene point. However, we often do not have $\mathbf{X}_l$ and $\mathbf{X}_r$ directly, but rather the projections of these points onto the image planes.

Perspective Projection

For the left camera, the perspective projection equations are:

$$U_l = \frac{f_x^l \cdot X_l}{Z_l} + O_x^l$$

$$V_l = \frac{f_y^l \cdot Y_l}{Z_l} + O_y^l$$

where f_x^l and f_y^l are the focal lengths, and O_x^l and O_y^l are the coordinates of the principal point in the left camera.

Rearranging these equations, we get:

$$Z_l U_l = f_x^l \cdot X_l + Z_l \cdot O_x^l$$

$$Z_l V_l = f_y^l \cdot Y_l + Z_l \cdot O_y^l$$

Using homogeneous coordinates, this can be expressed as:

$$\begin{bmatrix} Z_l \cdot U_l \\ Z_l \cdot V_l \\ Z_l \end{bmatrix} = \begin{bmatrix} f_x^l \cdot X_l + Z_l \cdot O_x^l \\ f_y^l \cdot Y_l + Z_l \cdot O_y^l \\ Z_l \end{bmatrix} = \begin{bmatrix} f_x^l & 0 & O_x^l \\ 0 & f_y^l & O_y^l \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix}$$

This equation simplifies to:

$$\begin{bmatrix} Z_l \cdot U_l \\ Z_l \cdot V_l \\ Z_l \end{bmatrix} = \begin{bmatrix} f_x^l & 0 & O_x^l \\ 0 & f_y^l & O_y^l \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix}$$

which shows that the homogeneous coordinates of the point (X_l, Y_l, Z_l) can be found from its image coordinates (U_l, V_l) in the left camera.

$$\mathbf{K}_l = \begin{bmatrix} f_x^l & 0 & O_x^l \\ 0 & f_y^l & O_y^l \\ 0 & 0 & 1 \end{bmatrix}$$

This matrix is known as the camera matrix $\mathbf{K}_l$:

where $\mathbf{K}_l$ is the camera matrix for the left camera, fully known since we assume that we know all the internal parameters of the two cameras.

The left camera equation can thus be written as:

$$Z_l \begin{bmatrix} U_l \\ V_l \\ 1 \end{bmatrix} = \mathbf{K}_l \begin{bmatrix} X_l \\ Y_l \\ Z_l \end{bmatrix}$$

Similarly, for the right camera, the perspective projection equations are:

$$U_r = \frac{f_x^r \cdot X_r}{Z_r} + O_x^r$$
$$V_r = \frac{f_y^r \cdot Y_r}{Z_r} + O_y^r$$

Rearranging these equations, we get:

$$Z_r U_r = f_x^r \cdot X_r + Z_r \cdot O_x^r$$
$$Z_r V_r = f_y^r \cdot Y_r + Z_r \cdot O_y^r$$

Using homogeneous coordinates, this can be expressed as:

$$\begin{bmatrix} Z_r \cdot U_r \\ Z_r \cdot V_r \\ Z_r \end{bmatrix} = \begin{bmatrix} f_x^r \cdot X_r + Z_r \cdot O_x^r \\ f_y^r \cdot Y_r + Z_r \cdot O_y^r \\ Z_r \end{bmatrix} = \begin{bmatrix} f_x^r & 0 & O_x^r \\ 0 & f_y^r & O_y^r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix}$$

This equation simplifies to:

$$\begin{bmatrix} Z_r \cdot U_r \\ Z_r \cdot V_r \\ Z_r \end{bmatrix} = \begin{bmatrix} f_x^r & 0 & O_x^r \\ 0 & f_y^r & O_y^r \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix}$$

This matrix is known as the camera matrix $\mathbf{K}_r$:

$$\mathbf{K}_r = \begin{bmatrix} f_x^r & 0 & O_x^r \\ 0 & f_y^r & O_y^r \\ 0 & 0 & 1 \end{bmatrix}$$

The right camera equation can thus be written as:

$$Z_r \begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix} = \mathbf{K}_r \begin{bmatrix} X_r \\ Y_r \\ Z_r \end{bmatrix}$$

Compact Form of the Essential Matrix Equation

We can rewrite $\mathbf{X}_l^T \mathbf{E} \mathbf{X}_r = 0$ in a more compact form using the projection equations.

$$\mathbf{X}_l^T = \begin{bmatrix} U_l & V_l & 1 \end{bmatrix} Z_l \mathbf{K}_l^{-1^T}$$

and

$$\mathbf{X}_r = \mathbf{K}_r^{-1} Z_r \begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix}$$

Thus, the essential matrix equation becomes:

$$\begin{bmatrix} U_l & V_l & 1 \end{bmatrix} Z_l \mathbf{K}_l^{-1^T} \mathbf{E} \mathbf{K}_r^{-1} Z_r \begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix} = 0$$

Since Z_l and Z_r are scalars, they can be factored out:

$$\begin{bmatrix} U_l & V_l & 1 \end{bmatrix} \mathbf{K}_l^{-1^T} \mathbf{E} \mathbf{K}_r^{-1} \begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix} = 0$$

This shows how we can compute the essential matrix $\mathbf{E}$ given the known camera matrices and the projection equations.

So rewriting in terms of image coordinates:

$$\begin{bmatrix} U_l & V_l & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix}$$

So:

$$\mathbf{E} = \mathbf{K}_l^T \mathbf{F} \mathbf{K}_r$$

Estimating the Fundamental Matrix

To estimate the fundamental matrix $\mathbf{F}$, we first need to find corresponding features in the two images. Given these correspondences, we can use the epipolar constraint for each pair of corresponding points to derive linear equations.

Epipolar Constraint

For each correspondence i, we write out the epipolar constraint as follows. Given the image coordinates (U_l^i, V_l^i) in the left image and (U_r^i, V_r^i) in the right image, the epipolar constraint can be expressed using the fundamental matrix $\mathbf{F}$ as:

$$\begin{bmatrix} U_l^i & V_l^i & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} U_r^i \\ V_r^i \\ 1 \end{bmatrix} = 0$$

Expanding this matrix equation, we get:

$$(f_{11}U_r^i + f_{12}V_r^i + f_{13})U_l^i + (f_{21}U_r^i + f_{22}V_r^i + f_{23})V_l^i + (f_{31}U_r^i + f_{32}V_r^i + f_{33}) = 0$$

This can be rearranged to form a linear equation:

$$f_{11}U_r^iU_l^i + f_{12}V_r^iU_l^i + f_{13}U_l^i + f_{21}U_r^iV_l^i + f_{22}V_r^iV_l^i + f_{23}V_l^i + f_{31}U_r^i + f_{32}V_r^i + f_{33} = 0$$

Grouping the terms involving $\mathbf{F}$, we get:

$$f_{11}U_r^iU_l^i + f_{12}V_r^iU_l^i + f_{13}U_l^i + f_{21}U_r^iV_l^i + f_{22}V_r^iV_l^i + f_{23}V_l^i + f_{31}U_r^i + f_{32}V_r^i + f_{33} = 0$$

This equation provides a linear relationship between the components of the fundamental matrix $\mathbf{F}$ and the coordinates of the corresponding points.

Forming the Linear System

To estimate the fundamental matrix $\mathbf{F}$, we rearrange the terms of the epipolar constraint to form a linear system.

Linear System Formation

Starting from the epipolar constraint:

$$(f_{11}U_r^i + f_{12}V_r^i + f_{13})U_l^i + (f_{21}U_r^i + f_{22}V_r^i + f_{23})V_l^i + (f_{31}U_r^i + f_{32}V_r^i + f_{33}) = 0$$

we can rearrange the terms to isolate each component of the fundamental matrix $\mathbf{F}$. Expanding and grouping terms gives:

$$f_{11}U_r^iU_l^i + f_{12}V_r^iU_l^i + f_{13}U_l^i + f_{21}U_r^iV_l^i + f_{22}V_r^iV_l^i + f_{23}V_l^i + f_{31}U_r^i + f_{32}V_r^i + f_{33} = 0$$

This can be written in matrix form as:

$$\begin{bmatrix} U_r^iU_l^i & V_r^iU_l^i & U_l^i & U_r^iV_l^i & V_r^iV_l^i & V_l^i & U_r^i & V_r^i & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

Matrix Form

To solve for the fundamental matrix $\mathbf{F}$, we collect multiple such equations for different correspondences into a matrix A and vector $\mathbf{f}$. Let there be N correspondences. Each correspondence gives us one linear equation, so:

$$A \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = \mathbf{0}$$

where A is an $N \times 9$ matrix with rows:

$$\begin{bmatrix} U_r^i U_l^i & V_r^i U_l^i & U_l^i & U_r^i V_l^i & V_r^i V_l^i & V_l^i & U_r^i & V_r^i & 1 \end{bmatrix}$$

and $\mathbf{f}$ is the column vector of the fundamental matrix components.

Solving the Linear System

To solve for the fundamental matrix, use methods like Singular Value Decomposition (SVD) to find the vector $\mathbf{f}$ that satisfies $A\mathbf{f} = \mathbf{0}$. This vector is reshaped into the 3x3 fundamental matrix $\mathbf{F}$.

Fundamental Matrix and Homogeneous Coordinates

The fundamental matrix $\mathbf{F}$ acts on the homogeneous coordinates (image coordinates of the scene point):

$$\begin{bmatrix} U_l & V_l & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix} = 0$$

Multiplying the fundamental matrix $\mathbf{F}$ by a scalar K does not change the epipolar constraint, since the two equations are still equal to zero:

$$\begin{bmatrix} U_l & V_l & 1 \end{bmatrix} \begin{bmatrix} Kf_{11} & Kf_{12} & Kf_{13} \\ Kf_{21} & Kf_{22} & Kf_{23} \\ Kf_{31} & Kf_{32} & Kf_{33} \end{bmatrix} \begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix} = 0$$

Therefore, the fundamental matrix $\mathbf{F}$ and the scaled fundamental matrix $K\mathbf{F}$ describe the same epipolar geometry. This means that the fundamental matrix is defined only up to a scale factor.

To uniquely determine the fundamental matrix, we can set its scale by imposing a normalization condition. A common choice is to set the Frobenius norm of the fundamental matrix to 1:

$$\|\mathbf{f}\|^2 = 1$$

This ensures that the solution for $\mathbf{f}$ is unique up to this normalization.

Solving for the Fundamental Matrix

To solve for the fundamental matrix $\mathbf{F}$, we employ a least squares solution. Our goal is to find $\mathbf{F}$ such that the epipolar constraint is satisfied as closely as possible for all corresponding points. This can be expressed as finding $\mathbf{f}$ such that:

$$\|\mathbf{A}\mathbf{f}\|^2 \text{ is minimized, subject to } \|\mathbf{f}\|^2 = 1$$

This is a constrained linear least squares problem, similar to solving for the projection matrix during camera calibration or the homography matrix for image stitching.

Once we obtain the solution vector $\mathbf{f}$, we can reshape it into the 3×3 fundamental matrix $\mathbf{F}$.

Computing the Essential Matrix

With the fundamental matrix $\mathbf{F}$ determined, we can compute the essential matrix $\mathbf{E}$. The relationship between the fundamental matrix and the essential matrix is given by:

$$\mathbf{E} = \mathbf{K}_l^\top \mathbf{F} \mathbf{K}_r$$

where $\mathbf{K}_l$ and $\mathbf{K}_r$ are the intrinsic calibration matrices of the left and right cameras, respectively.

Decomposing the Essential Matrix

Once we have the essential matrix $\mathbf{E}$, we can decompose it using singular value decomposition (SVD):

$$\mathbf{E} = \mathbf{T} \times \mathbf{R}$$

where $\mathbf{T}$ represents translation and $\mathbf{R}$ represents rotation. This decomposition allows us to recover the relative pose (translation and rotation) between the two camera views. when we have done, then our uncalibrated stereo system is now fully calibrated

Finding Correspondences and Creating a Dense Depth Map

Once the stereo system is fully calibrated, the next step is to find correspondences between points in the left image and the right image, resulting in a dense depth map of the three-dimensional scene.

Epipolar Geometry and Correspondences

Given: - The fundamental matrix $\mathbf{F}$ - A point (U_l, V_l) in the left image

We aim to find the equation of the epipolar line in the right image.

Equation of the Epipolar Line in the Right Image

Starting from the epipolar constraint equation:

$$\begin{bmatrix} U_l & V_l & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix} = 0$$

Given (U_l, V_l) and $\mathbf{F}$, we solve for (U_r, V_r). Expanding the matrix equation:

$$(f_{11}U_l + f_{21}V_l + f_{31})U_r + (f_{12}U_l + f_{22}V_l + f_{32})V_r + (f_{13}U_l + f_{23}V_l + f_{33}) = 0$$

Simplifying, we get the equation for the epipolar line in the right image:

$$a_r U_r + b_r V_r + c_r = 0$$

where:

$$a_r = f_{11}U_l + f_{21}V_l + f_{31}$$

$$b_r = f_{12}U_l + f_{22}V_l + f_{32}$$

$$c_r = f_{13}U_l + f_{23}V_l + f_{33}$$

Epipolar Line in the Left Image

Similarly, we can calculate the epipolar line in the left image for a point in the right image. Given a point (U_r, V_r) in the right image, the equation of the epipolar line in the left image can be derived similarly using the fundamental matrix $\mathbf{F}$.

Computing Depth Using Triangulation

To compute the depth, we take corresponding pairs of points and use their image coordinates to estimate the three-dimensional coordinates of the points in the scene. This process is known as triangulation. Given the intrinsic parameters of the cameras, we can use the following imaging equations for the left and right cameras.

Imaging Equations

For the left camera, the imaging equation is:

$$\begin{bmatrix} U_l \\ V_l \\ 1 \end{bmatrix} = \begin{bmatrix} f_x^l & 0 & O_x^l & 0 \\ 0 & f_y^l & O_y^l & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{bmatrix}$$

For the right camera, the imaging equation is:

$$\begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix} = \begin{bmatrix} f_x^r & 0 & O_x^r & 0 \\ 0 & f_y^r & O_y^r & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix}$$

These matrices represent the intrinsic parameters of the left and right cameras.

Coordinate Transformation

To find X_l or X_r, we use the constraint of the relative position and orientation between the two cameras, which is given by:

$$\begin{bmatrix} X_l \\ Y_l \\ Z_l \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix}$$

This transformation matrix includes the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$, which describe the coordinate transformation from the right camera coordinate frame to the left camera coordinate frame.

Computing Depth Using Triangulation

To compute the depth, we take corresponding pairs of points and use their image coordinates to estimate the three-dimensional coordinates of the points in the scene. This process is known as triangulation. Given the intrinsic parameters of the cameras, we can use the following imaging equations for the left and right cameras.

Imaging Equations

For the left camera, substituting the coordinate transformation into the imaging equation, we get:

$$\begin{bmatrix} U_l \\ V_l \\ 1 \end{bmatrix} = \begin{bmatrix} f_x^l & 0 & O_x^l & 0 \\ 0 & f_y^l & O_y^l & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix}$$

Combining the matrices, we get:

$$\mathbf{P}_l = \begin{bmatrix} f_x^l r_{11} & f_x^l r_{12} & f_x^l r_{13} & f_x^l t_x + O_x^l \\ f_y^l r_{21} & f_y^l r_{22} & f_y^l r_{23} & f_y^l t_y + O_y^l \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix}$$

Thus, the left camera equation can be written as:

$$\tilde{U}_l = \mathbf{P}_l \tilde{X}_r$$

For the right camera, the imaging equation is:

$$\begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix} = \begin{bmatrix} f_x^r & 0 & O_x^r & 0 \\ 0 & f_y^r & O_y^r & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix}$$

Thus, the right camera equation can be written as:

$$\tilde{U}_r = \mathbf{M}_{\text{intr}} \tilde{X}_r$$

where $\mathbf{M}_{\text{intr}}$ is the intrinsic matrix of the right camera:

$$\mathbf{M}_{\text{intr}} = \begin{bmatrix} f_x^r & 0 & O_x^r & 0 \\ 0 & f_y^r & O_y^r & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

The imaging equation for the right camera is given by:

$$\tilde{U}_r = \mathbf{M}_r \tilde{X}_r$$

$$\begin{bmatrix} U_r \\ V_r \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix}$$

The imaging equation for the left camera is given by:

$$\tilde{U}_l = \mathbf{P}_l \tilde{X}_r$$

$$\begin{bmatrix} U_l \\ V_l \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{bmatrix}$$

Step 1: Multiply the First Matrix by M

Start with the original first matrix:

$$\begin{pmatrix} U_r \\ V_r \\ 1 \end{pmatrix} = \begin{pmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{pmatrix} \begin{pmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{pmatrix}$$

Multiply the first row of the left matrix by the third row of the right matrix:

$$\begin{pmatrix} U_r \cdot m_{31} & U_r \cdot m_{32} & U_r \cdot m_{33} & U_r \cdot m_{34} \\ V_r \cdot m_{31} & V_r \cdot m_{32} & V_r \cdot m_{33} & V_r \cdot m_{34} \end{pmatrix}$$

Then subtract the corresponding elements of the first row from the elements of the first row in the original matrix:

$$\begin{pmatrix} U_r m_{31} - m_{11} & U_r m_{32} - m_{12} & U_r m_{33} - m_{13} \end{pmatrix}$$

Do the same for the second row:

$$\begin{pmatrix} V_r m_{31} - m_{21} & V_r m_{32} - m_{22} & V_r m_{33} - m_{23} \end{pmatrix}$$

Step 2: Subtract Elements from the Second Matrix's Rows

Similarly, for the second matrix:

$$\begin{pmatrix} U_l \\ V_l \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X_r \\ Y_r \\ Z_r \\ 1 \end{pmatrix}$$

Multiply the first row of the left matrix by the third row of the right matrix:

$$\begin{pmatrix} U_l \cdot p_{31} & U_l \cdot p_{32} & U_l \cdot p_{33} & U_l \cdot p_{34} \\ V_l \cdot p_{31} & V_l \cdot p_{32} & V_l \cdot p_{33} & V_l \cdot p_{34} \end{pmatrix}$$

Then subtract the corresponding elements:

$$\begin{pmatrix} U_l P_{31} - P_{11} & U_l P_{32} - P_{12} & U_l P_{33} - P_{13} \end{pmatrix}$$

And similarly for the second row:

$$\begin{pmatrix} V_l P_{31} - P_{21} & V_l P_{32} - P_{22} & V_l P_{33} - P_{23} \end{pmatrix}$$

Step 3: Final Rearranged Matrices

Now, combine these into the final form:

$$\begin{pmatrix} U_r m_{31} - m_{11} & U_r m_{32} - m_{12} & U_r m_{33} - m_{13} \\ V_r m_{31} - m_{21} & V_r m_{32} - m_{22} & V_r m_{33} - m_{23} \\ U_l P_{31} - P_{11} & U_l P_{32} - P_{12} & U_l P_{33} - P_{13} \\ V_l P_{31} - P_{21} & V_l P_{32} - P_{22} & V_l P_{33} - P_{23} \end{pmatrix} \begin{pmatrix} X_r \\ Y_r \\ Z_r \end{pmatrix} = \begin{pmatrix} m_{14} - m_{34} \\ m_{24} - m_{34} \\ p_{14} - p_{34} \\ p_{24} - p_{34} \end{pmatrix} \tag{3.23}$$

Least Squares Solution Using Pseudo-Inverse

Given the overdetermined system of linear equations:

$$AX_r = b$$

where A is a 4×3 matrix and b is a 4×1 matrix. To find the least squares solution, we use the pseudo-inverse.

First, compute the normal equations:

$$A^\top A X_r = A^\top b$$

Then, solve for X_r using the pseudo-inverse:

$$X_r = (A^\top A)^{-1} A^\top b$$

This solution minimizes the squared error between AX_r and b. By repeating this process for each pair of corresponding points in the left and right images, we obtain a complete three-dimensional depth map of the scene.

3.4.5 Calibration

The goal of the calibration is to obtain the intrinsic matrices of the cameras (i.e. K_L and K_R) as well as determining the transformation matrix $[R|T]$ that will transform the line equation from the reference frame of the right camera to the left one. This is done by using the python library CV2 from OpenCV. The calibration method we are using involves calibrating each camera separately and obtaining the intrinsic matrices first, then we are stereo-calibrating the cameras together to obtain the $[R|T]$ matrix. In order find the relative distance between the cameras and their intrinsic properties, we need a third reference frame in which we know where specific points are. For that we use a checkerboard pattern as shown below.

This is a 9×6 board as we are only interested in the inner corners. we assign to each corner a coordinate such that the top left corner is $(0, 0, 0)$. Recall, the X-axis points towards the right, the Y-axis points down and the Z-axis points towards the object from the camera. Therefore, the corner next to the top left one is assigned the value $(a, 0, 0)$ and so on. **NOTE**: a is the length of the square in our real world system. Now we have the (U, V) coordinates from the picture and the (X, Y, Z) coordinates in the real world reference frame. We can use the equation derived in the previous section.

$$X_{image} = K X_{world}$$

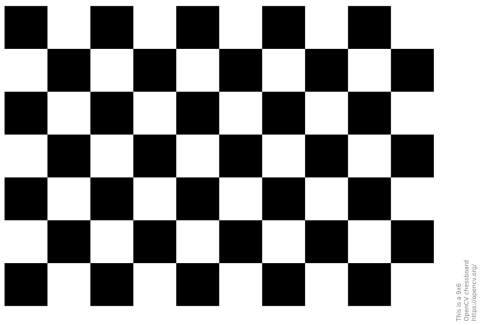


Figure 3.7: Checkerboard pattern used for calibration

To find the intrinsic Matrix of each camera. Having the intrinsic matrices we can find the coordinates of the points on the checkerboard in each camera's frame which will lead us to finding the distance to the between the board and the camera as well as the rotation between each camera and the board. Knowing the position and rotation of the board w.r.t. each camera, we can easily find the Translation vector and the rotation matrix of one camera w.r.t. the other.

NOTE: the Checkerboard corners will be auto-detected by the code by scanning for the edges where we have white on one diagonal and black on the other which indicates that it's an edge between two black squares and two white squares.

These are the steps you need to calibrate the cameras:

- Take 25 to 30 pictures of the checkerboard from each camera. The pictures need to be simultaneous and the board needs to be at various distances and angles. we suggest to start close to the cameras and take a picture of the boards facing the cameras, then rotate the board to the left, right, up, down and any combination of these angles. The pictures need to be in focus and the board needs to be apparent in the pictures of both cameras.
- After capturing the pictures, make sure that you can distinguish between those from camera 1 and camera 2. We do that by keeping them in two different folders or by including in the picture name "c1" and "c2". The first option is recommended because the code we are are using uses two folders as an input.
- If you are using the library written by our team. You need to use the function called "Calibrate_cameras". It requires two parameters: Primary folder and Secondary folder. The primary folder should contain the pictures of the camera you intend to be the frame of reference.
- After running the code you will be shown he pictures with the edges of the checkerboard highlighted and then the calibration calculation will run. At the end the code

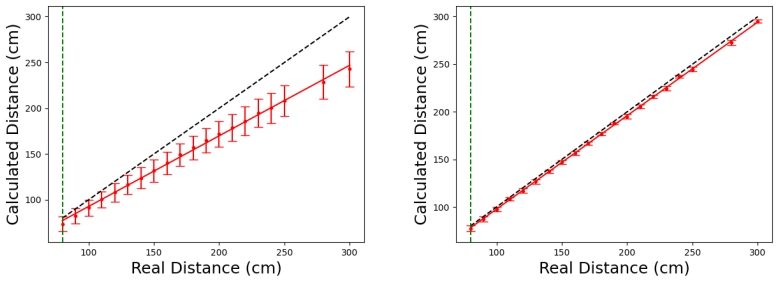
will print the Intrinsic matrices, the Rotation matrix and the Translation vector. **NOTE:** the matrices' components are in pixels while the translation vector is in cm (the units chosen for the size of a square, is you use a board where the square is not 2.1cm or the inner corners are not 9×6 change that from the library code the library is not general it is made for the experiment we are conducting. If we end up releasing it the size of the board and the squares would be included as a parameter). The calibration data will be saved in an XML file in the same directory where you ran your code.

- To import the data and use it in another code use the function "import_Calibration(p)" where p is the pixel size of your camera sensor which can be easily found by checking the camera manual. This function will return 4 objects is that specific order: Kl, Kr, R, T. All in pixels ready for the coordinate calculation.

The library we are talking about is included in the extra files folder in the overleaf document. uploaded on September 19Th 2024.

3.4.6 Testing This Method

A program was written to test this method. check the next section for more details. The data like before was taken by measuring a distance between the cameras and the desired object. Then the pictures were taken at this distance noting it down. After that we ran the data through the old code where we used the equations (3.10) with no calibration and once through the new code we got by using the new method. The results are as follows:



(a) Without calibration (b) With calibration

Figure 3.8

We can clearly see how the Uncalibrated results have a really high error the further away we get, while the results from the calibrated coordinates show very little error all throughout.

3.4.7 Finally

Now that we can achieve stereoscopic imaging using none parallel cameras, we can combine it with the box designed for fast imaging and the high-speed cameras to achieve

fast stereoscopic imaging without modifying the overall size of the box. This will be discussed in more detail in the next chapter.

3.5 Software

We wrote two programs, one to calibrate the cameras using a Checkerboard pattern and one to use the calibration to calculate the coordinates of a point of our choice. The calibration is done through the cv2 library. About 30 pictures are taken of the board at different angles and distances then the pictures are inputted in the code where the edges between the squares are auto-detected and assigned coordinates w.r.t. the board itself. Now we have the real world coordinates for the edges and the coordinates on the image. We use this to find the intrinsic matrices of the cameras. The more calibration pictures we have the more accurate the results will be. After finding the intrinsic matrix we need to find the $[R|T]$ matrix that transforms the right camera coordinates to the left camera's. To do so we need at least 8 common points chosen on the picture(s). Here the points of the checkerboard pattern come in handy again. we use the pictures again to determine the R matrix and T vector. This is done by comparing the points and finding the matrix that transforms the ones from the right camera to the ones from the left one. The matrices and Vector are then exported to an xml file where we store them for future use. The second code starts by importing the data from the xml. A function is to take 2 points we select on the pictures and use eq(3.23) to calculate the the coordinates using the least square method mentioned above. The function then returns the coordinates that we can use for anything we like. in the code we are using, we analyze the coordinates obtained and plot the results as a function of the real world measurements taken in the lab during the experiment in order to determine the accuracy of the method.

Chapter 4

Fast-Stereoscopic Imaging

4.1 The final goal

As stated before the end goal of this project is to achieve fast-stereoscopic imaging in order to be able to capture clear images of the UFOs in a tokamak's plasma flow (Fast Imaging) and determine its 3D position (Stereoscopic Imaging). In this section we are going to finally combine both techniques. The final test is to use this to image Tungsten-laser interaction in the PLD and calculate the velocity $\vec{V}$ and from its components (V_X, V_Y, V_Z). Which will be similar to plasma wall interactions in a tokamak as the walls are made out of tungsten. In our case we need a frame-rate that is achievable by the Chronos cameras without using the fast imaging technique. In cases where we need much higher frame rates we can simply use the fast imaging setup in place of each camera.

4.2 High speed imaging and focal length

For imaging tungsten-laser interaction we need somewhere around 10K fps which is easily achievable by the Chronos 1.4. However, in order to achieve this fps we must lower our picture resolution. By lowering the resolution we are decreasing the intersection between the two pictures which is directly related to the stereoscopic imaging as we need the object to appear in both pictures taken by the two cameras in order to determine its position. The fix to this problem is to use a lens with a smaller focal length. With a smaller focal length we get a wider picture at higher resolutions. At lower resolutions the lower focal length will give the same field of view as the higher resolution and bigger focal length. The only downside to using this method is that our picture will be more "grainy" as it is at lower resolutions.

4.3 Camera projection

In the first chapter the target was to orient the cameras such that both of them have the exact same field of view, The calculation of that scenario was done easily in the first

chapter. However, now we need to start moving the camera that captures the reflected light and we need a way of calculating its relative position w.r.t. the other camera. For a beam splitter at a 45° angle the movement in 1 to 1 in the opposite direction, i.e. moving the camera capturing the reflection 5 cm to the left in similar to moving its projection 5 cm to the right in the other camera's frame. This is the simplest scenario. We need to derive a general equation that depends on the angle of the beam splitter, so that we are not bound to a 45° angle and to mathematically prove the statement above.

4.3.1 Proving the projection transformation geometrically

We start by finding the transformation we need to find the projection. Consider a mirror, or a beam splitter in our case but we are only interested in the reflection part, at an arbitrary angle. A beam of light incident to that mirror at point A gets reflected as shown in Fig:4.1.

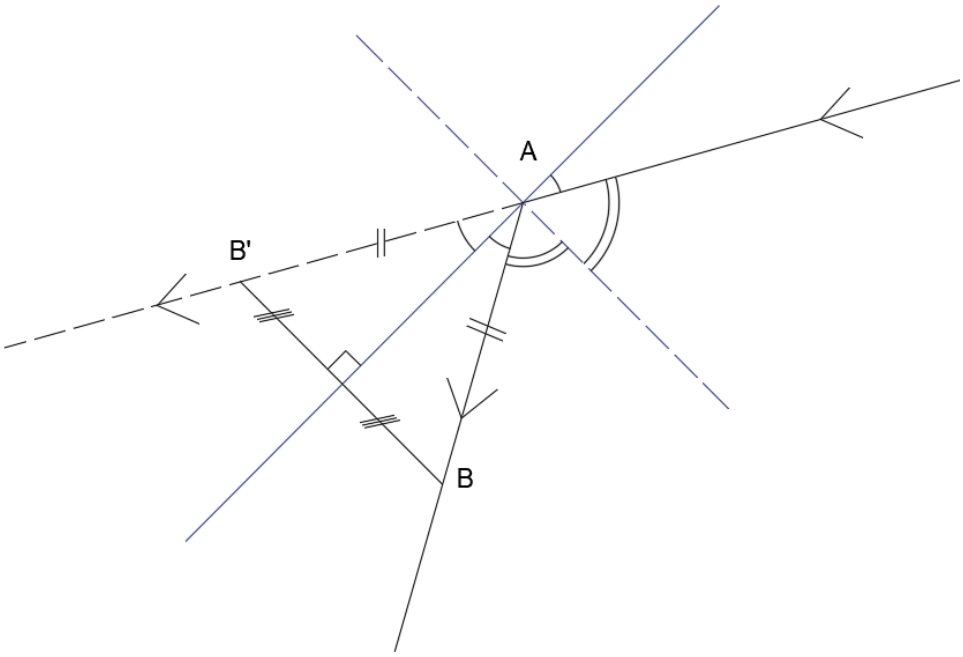


Figure 4.1: Projection Geometry

The blue line in the figure represents the mirror. The incident beam gets reflected obeying the reflection law. The angle, i, that the incident beam makes with the normal to the mirror at point A is equal to the angle between the reflected ray (AB) and the normal.

NOTE: the normal is represented by the dashed blue line.

The angles that the light beams make with the mirror are equal they are equal to $\alpha = 90° - i$. If the incident beam were to continue its path uninterrupted by the mirror, we would get the dashed black line (AB'), Which forms the angle α with the mirror as it is the opposite angle of that formed between the incident ray and the mirror.

Assume an object B that intersects the reflected beam at a distance AB it's projection B' should exist on the hypothetical line and have the same distance from A.

$$AB' = AB$$

Therefore, the triangle ABB' is isosceles at A. In addition, we know that the mirror is the first bisector of the angle $\widehat{BAB'}$. Hence, the mirror is the mediator of the line BB', i.e.

$$d(B, mirror) = d(B', mirror)$$

$$\& \quad Mirror \perp [BB']$$

Which is the definition of: B and B' are symmetrical w.r.t. the mirror.

4.3.2 The algebraic expression of the projection

Now that we know that the projection is simply a symmetry w.r.t. the mirror we can find the matrix that transforms a point into its projection. We also need to express the coordinates at the end in the coordinate system of the other camera. First things first, we define our reference frames as per Fig:4.2. The reference frames O and O" are taken at the edges of the boxes and the reference frame O' is taken at the edge of the beam splitter such that the x'-axis of that frame is co-linear with the beam splitter. The goal here is to take the coordinates in the O frame and express their projections in the O" frame.

The first part is to project our point symmetrically w.r.t. to the beam splitter. The method we are going to use here is to switch to the O' frame where the symmetry becomes w.r.t. the x'-axis which is simply changing the sign of the z' coordinate. To transform from frame O to frame O' we need two operations: translate the frame and then align the axes. As we are using translation we are going to do the math in the Homogeneous Coordinates mentioned before. The translation matrix T1 is related to the vector $\vec{V_1}$ pointing from O' to O via the following matrix:

$$T_1 = \begin{bmatrix} I_2 & V_1 \\ 0^T & 1 \end{bmatrix} \tag{4.1}$$

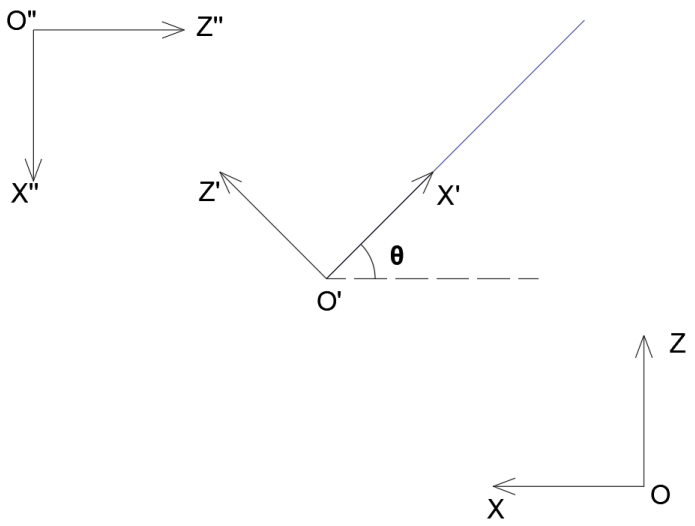


Figure 4.2: Reference frames of the box

Where I_2 is the 2×2 Identity matrix

Let us define second matrix that aligns the axes as R_1 such that:

$$R_1 = \begin{bmatrix} -cos(\theta) & sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

Where θ is the angle that the beam splitter makes with the horizontal.

Finally, the transformation matrix from the frame O to O' is:

$$Tr_1 \equiv R_1 \, T_1 \tag{4.3}$$

Now, in the O' frame we can perform the projection, a mirroring w.r.t. the x-axis, by simply switching the sign of the Z coordinate. In homogeneous coordinates the matrix is:

$$M_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.4}$$

Therefore, the projection from coordinates in the frame O expressed in the frame O becomes:

$$P_{O/O} = (Tr_1)^{-1} \, M_x \, Tr_1 \tag{4.5}$$

In order to express the projection in the O" frame we need to perform a transformation from the frame O to O". The translation is done similarly to T_1. A vector $\vec{V_2}$ pointing from O" to O' gives:

$$T_2 = \begin{bmatrix} I_2 & V_1 \\ 0^T & 1 \end{bmatrix} \tag{4.6}$$

In case aligning the axes is simpler, the Matrix to do that can be expressed by:

$$R_2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.7}$$

$$\implies Tr_2 \equiv R_2 \, T_2 \tag{4.8}$$

Finally, the projection from coordinates in the frame O expressed in the frame O" is:

$$P_{O"/O} = Tr_2 \, (Tr_1)^{-1} \, M_x \, Tr_1 = Tr_2 \, P_{O/O} \tag{4.9}$$

It is worth noting that the projection matrix we got is a 3×3 homogeneous matrix that acts on a homogeneous vector $[x, z, 1]$ and outputs the homogeneous vector $[x', z', 1]$. The P matrix is of the form:

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.10}$$

Where the last row is the homogeneous row. We can remove the last row which results P becoming:

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \end{bmatrix} \tag{4.11}$$

Here P acts on a homogeneous vector $[x, z, 1]$ and outputs the Cartesian vector $[x', z']$. However, now that we removed the homogeneous row we can not combine it with other matrices like before.

4.4 CAD