Video Stitching README

By: Zachary Carey

Introduction

This document will act as a complete user manual for the Pritzker Marine Laboratory's Multiple Raspberry Pi Camera Video Stitching software. This software will take all the videos recorded by the raspberry pi's as an input, and create a panorama type output with all the videos in a single clip. Each video is reshaped/resized and placed in a specific location in the output video by using a pixel coordinate system defined more in the section "How does the Video Stitching Program decide Stitch Location". In the output video an animal of interest can be seen in video progressing through the entire tank with minimal video error occurring at the camera borders.

Install Python, Python IDE, and Python Packages

Python and Pip Install

In order to run and or modify a python based program, the control computer must have both python and pip installed. If these programs are already installed skip this step.

To install python follow Link 1 in the "Links" section. This link will take you to python's home page where you can select the version of python you want to install and what kind of computer it's being installed on (Windows, Apple, Linux, etc). For the purposes of this lab python version 3.9.5 was used because of its recent release at the time of building the program. For the most success with working with this program using at least a version of python 3.9 is advisable because some functions used in the program have changed since older versions. If it is desired to install python 3.9.5 on windows follow Link 2 and select "Windows installer (__-bit)" where the blank is either 64 or 32 and is specific to your computer. Select the one that matches your computer and complete the installation process.

Next we should install pip, which will be helpful later for installing the necessary python packages for the program to run. To install pip follow Link 3 which is a guide to installing pip based on specific situations (first install of pip, update pip, kind of computer being installed on, etc). Follow the guide and install pip before continuing.

Install Python IDE

Now that python and pip are installed, a python IDE (or Integrated Development Environment) must be installed in order to open and edit python software. Any python IDE is fine to use and should not affect the function of the python program. If there is already a python IDE installed skip this step.

The python IDE originally used to generate the program was the Geany IDE. Details about the Geany IDE and what makes it unique can be found in Link 4. Detailed instructions on how to install Geany can be found in Link 5. If using a Windows computer look at the row "Windows (64-bit)" under "Geany Releases" and click the link in the "File" column corresponding with that row. This should download the file and walk you through installing and opening Geany for the first time. Once able to open Geany continue on to installing necessary python packages.

Install Python Packages

There are three packages that need to be installed in order to run the picamera video stitching software. Because some readers will have some of these packages pre installed, each package is its own section for ease of access.

Install OpenCV or CV2

To install OpenCV or cv2 for python, pip is used in the command prompt (terminal). A step by step guide to pip install cv2 is outlined in Link 6. It is only necessary to follow steps 1-3 of the guide in Link 6. For step 3 follow the instructions from section 3a "Option 2", which will install the entire OpenCV package for use in python. After running the pip installation check to make sure that "cv2" can be imported into a python program with the line "import cv2" in any python program, run the program to confirm. If the program returns nothing with no errors cv2 is installed.

Install NumPy

NumPy, like cv2, is installed via pip installation. The step by step guide can be found at Link 7, where you follow the instructions in section “PIP”. The instructions in this guide give the line of code to enter into the command prompt, in order to install NumPy. This line of code is “pip install numpy” and after it runs NumPy package should be installed. Check this the same way as cv2 by trying the line “import numpy” in any python program, and run it.

Install Imutils

Imutils is also installed via pip installation. A complete breakdown of this package can be found at Link 8 but like the other pip installations all that is needed is the line of code to run in the terminal. To install imutils enter “pip install imutils” into the command prompt and run it. Imutils should now be installed but check its function in a python program with “import imutils”.

How to Install and Open Video Stitching Python Program

The “stitchVideo.py” program is available in the Github repository for this project found at Link 9. To download the entire Github repository containing the “stitchVideo.py” program follow the instructions from section “How to download a repository from GitHub” in Link 10. This should download a “.zip” folder of the entire Github repository, extract or unzip said folder to access its contents. Once unzipped, find the “stitchVideo.py” program inside and right click on it, select “Open with Geany”. The file should open a Geany window displaying the python code in the file as a tab in the window.

How to Run the Video Stitching Program

To run the program there must already be videos saved from multiple cameras with significant overlap between cameras. Move the “stitchVideo.py" program into the same folder as all the video files so that they can be used by the program. This system has ten picameras in one row and each camera saves the name of the video file as:

“cam#_output_year-month-day_hour-minute-seconds.h264”

Where # is the camera's number, year-month-day is the date of recording (in that order), and hour-minute-seconds is the exact time of recording (in that order). Using this naming convention to our advantage a function called “importVideos(n)” was created, where n is the total number of cameras. This function creates a loop from 1 to n where each iteration takes the iteration number as the camera number and reads in the video file associated with that camera and puts it in a list. Ultimately producing a list with all n videos. The date and time information for the group of videos should be identical, so replace the date and time information in line 14 with the correct date and time for the videos being imported. Do not alter anything else about line 14 other than the date and time information, or the loop that imports the videos may break.

The function “importVideos(n)” produces a list of 10 videos that the rest of the program will access and hopefully create a stitched panorama type output. Before discussing how video stitching occurs in this program it’s important to understand the relationship between these cameras. The cameras were not mounted in order of their camera numbers, so the true order of camera numbers from top to bottom (of the video on the screen) is Cam: 1, 2, 4, 5, 3, 7, 6, 8, 9, 10. The first part of the while loop in the function “stitch_vid()” both reorganizes these cam numbers into correctly ordered frame numbers as well as reading each individual frame from each video. If the cameras are organized in order edit lines 32-41 changing the number in “vid[#]” to correspond to the camera number. Remember that the # in “vid[#]” goes from 0-9 because its a list while cam # goes from 1-10 because they were named cameras.

One of the final steps before running the program is to scroll to the bottom of the program file to the function “main()”. In “main()” for the variable “n” set equal to the number of cameras (in this case 10). Once complete there is one more step before running the program, which is deciding how it will be displayed or saved. For output files being saved, look at line 26, the variable in single quotation marks is the name the file will be saved as, as well as the type it will be saved as (“.avi” in this case). Change this to the desired name of the output file to be saved in the same folder as the program. If a live stream of the stitching process is desired see line 84-85 and remove the starting “#” from both lines, a preview should now appear when running the program.

The program can be run in two ways. The first is that the file can be run from the terminal, which can be done by first changing directories by copying the directory from the file explorer and pasting it in the terminal like “cd *copied directory*”. Once in the correct directory use the line “py stitchVideo.py” into the terminal and hit enter. The program should run, and for another example walkthrough follow Link 11. The other way to run the program with the python IDE. If using Geany this is pretty simple, hit the “Save” button in the top row towards the left, and once saved hit the “Execute” button in the same row. The program should run after hitting the execute button. If a video appears in preview and or in the saved video file then this part was successful, but if no video appears follow the error messages to solve the issue and double check all the above steps are correct. If the videos appear misaligned from one another follow the next section to fix this error.

How to Edit the Stitch Location of the Program

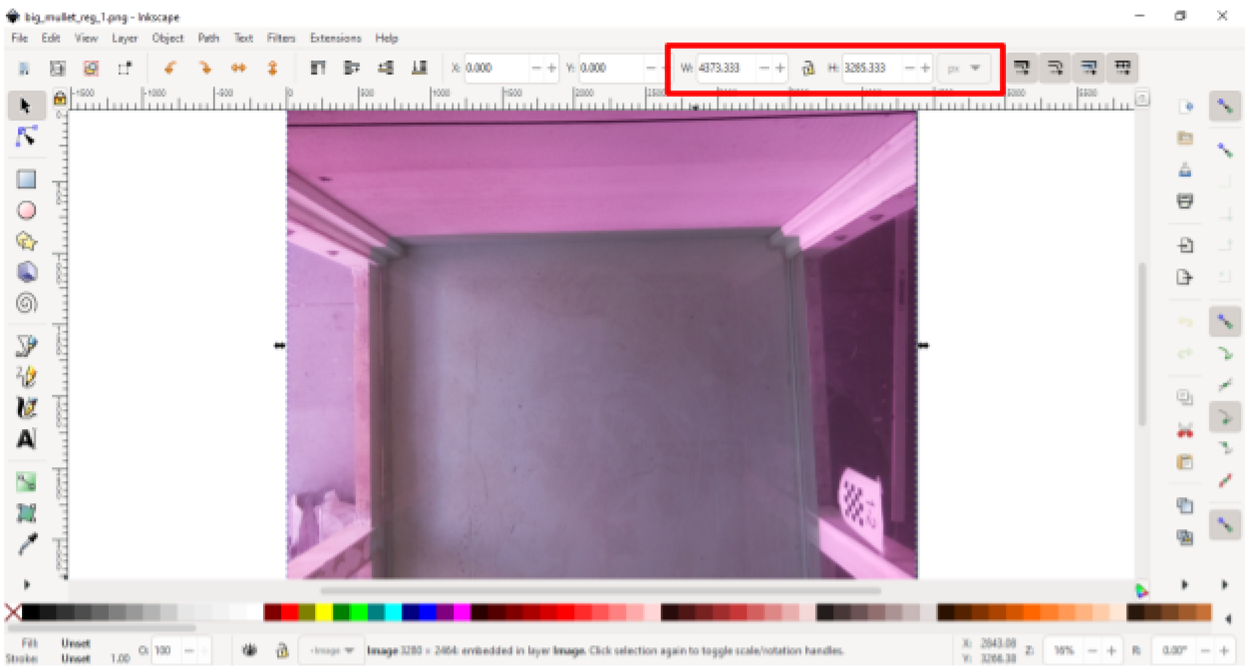
I have divided this into two sections. The first section discussing how to change the four corners of that will warp or crop the original video. The second section will discuss how to move videos around in the output file to get seamless stitching.

Change Corners the Input Video is Cropped to (pts1)

In order to manually fix stitching errors in the “stitchVideo.py” program, an image editing software can be used to help aid in visualizing these changes in pixel coordinates. This software is called inkscape and can be downloaded from Link 12. I followed the onscreen standard installation instructions and that's what I have used.

Open a photo (a saved still of any moment during the video, or a registration photo if collected) with Inkscape by right clicking on the image and selecting “open with Inkscape”. Should look like Figure 1 below. The raw still frame from the video file (Figure 1) was imported into Inkscape at a larger size than the video frame itself. To change this edit the dimensions in the red box of Figure 1 to the dimensions of the video frame. For the Pritzker system this is 640 width and 480 height in terms of pixels. The result should look like Figure 2.

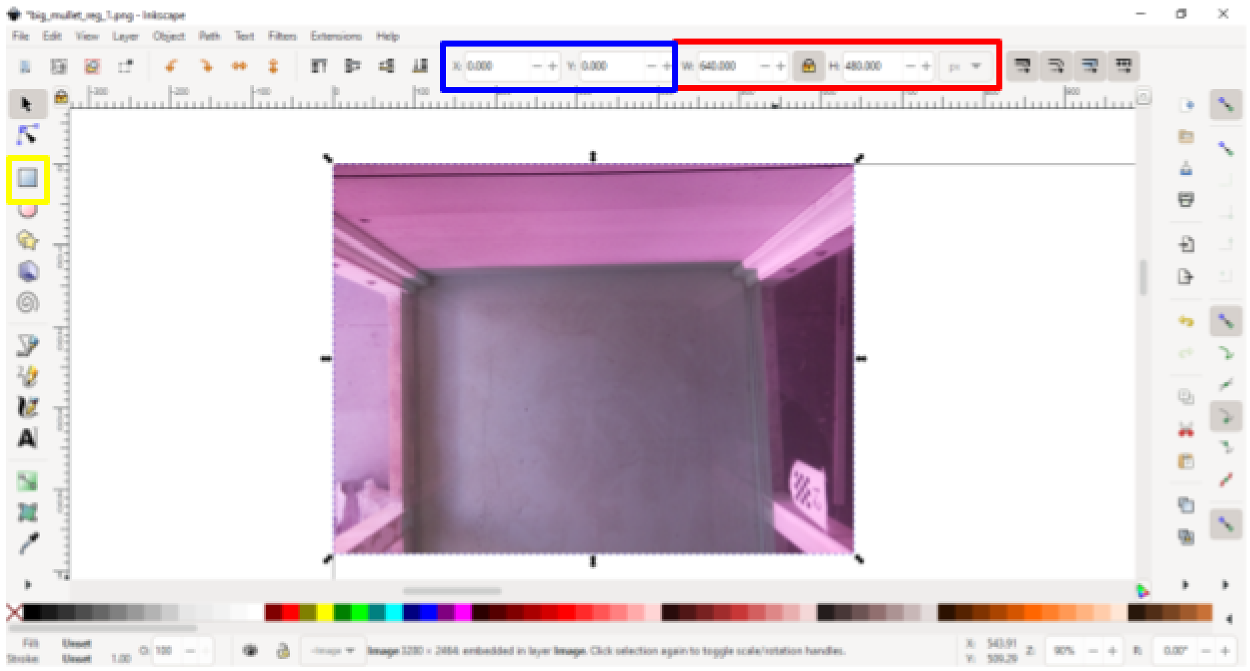
In order to visualize the desired area to be included in the input corner coordinates (pts1) a trial crop of this image can be made, attempting to keep the waterline as straight as possible. To do this click the icon in the yellow box of Figure 2 to draw a rectangle. Draw a rectangle over top of the image of the tank and angle it as needed to get a straight image. This is demonstrated in Figures 3 and 4. Use the mouse to get the coordinates of the four corners of the blue rectangle in order (top right, top left, bottom right, bottom left) and each one in the order (x,y).


big_mullet_reg_1.png - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help

X: 0.000 Y: 0.000 W: 4373.333 H: 3285.333 px

Image 3280 × 2464: embedded in layer Image. Click selection again to toggle scale/rotation handles.

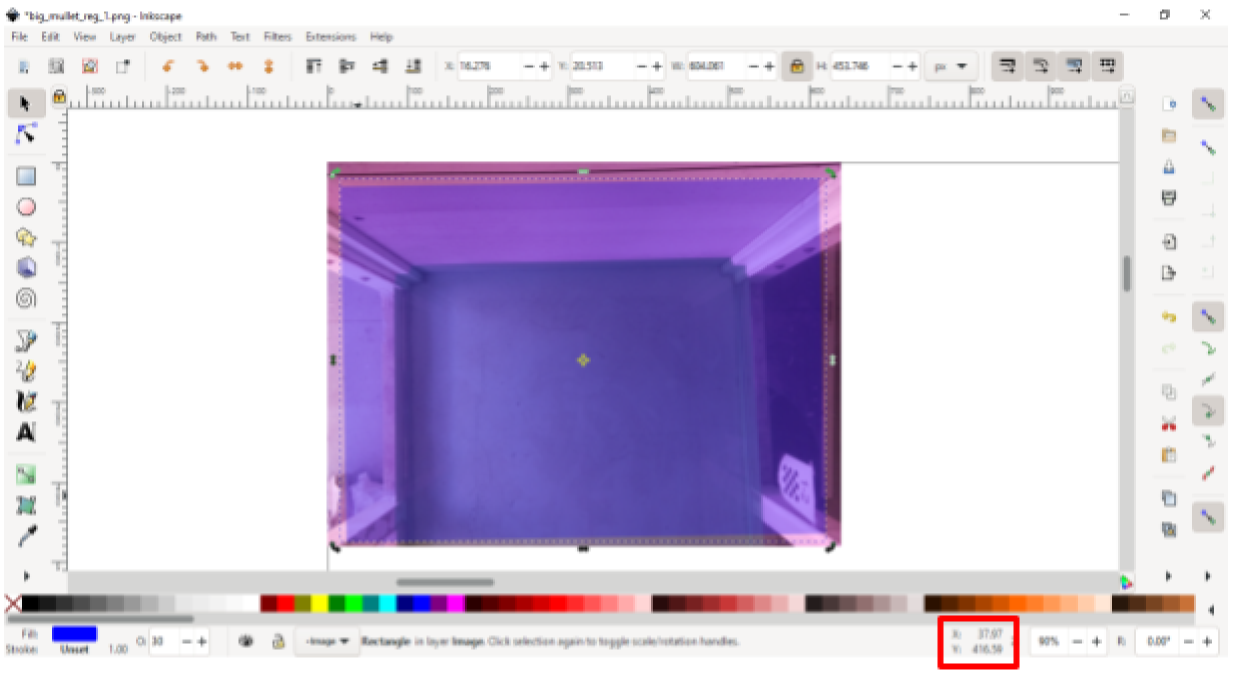
Figure 1: Shows what an image opened in Inkscape may look like. The image may be imported into Inkscape at a size larger than that of the video frame. Check the Image dimensions in Inkscape in the red box where H is height and W is width. Make sure that px for pixels is selected from the drop down.


*big_mullet_reg_1.png - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help

X: 0.000 Y: 0.000 W: 640.000 H: 480.000 px

Image 3280 × 2464: embedded in layer Image. Click selection again to toggle scale/rotation handles.

Figure 2: Shows the image from Figure 1 resized to the correct dimensions, highlighted by the red box again. The blue box shows the coordinates of the top left corner of the image. For ease make sure this is (0,0) (x,y). The logo highlighted by the yellow box allows for the drawing of rectangles in Inkscape used more shortly.


*big_mullet_reg_1.png - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help

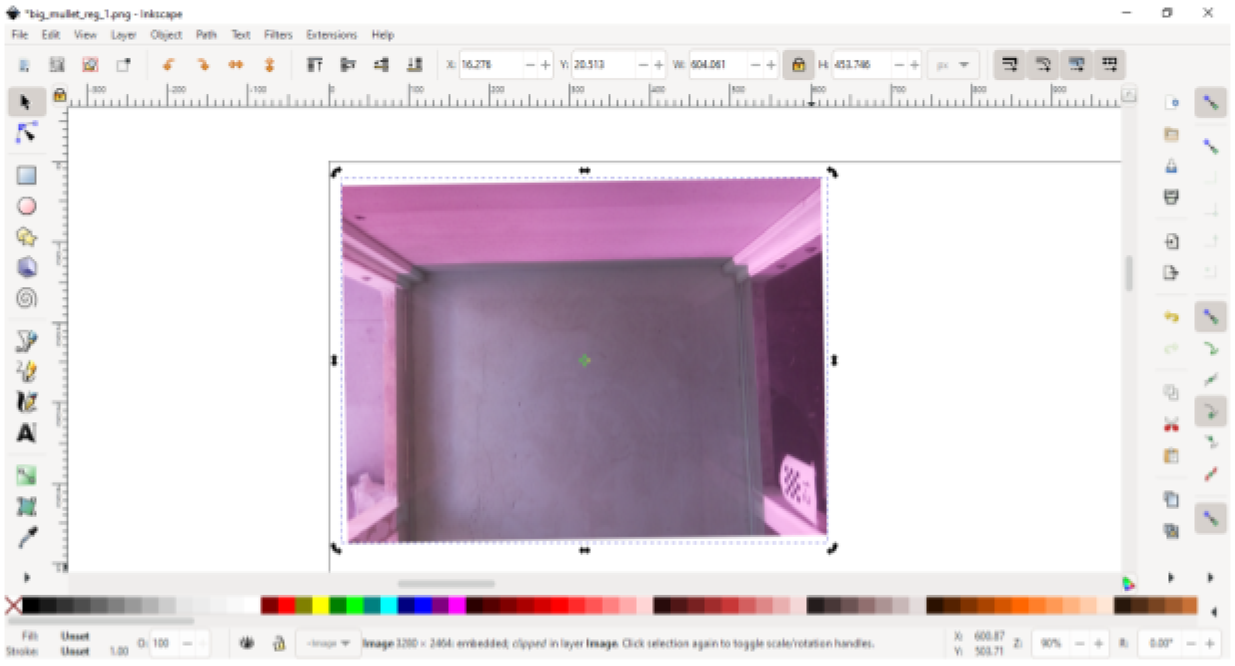
X: 16.276 Y: 20.513 W: 604.061 H: 453.746 px

Fill: Stroke: Unset 1.00 O: 30

Rectangle in layer Image. Click selection again to toggle scale/rotation handles.

X: 37.97 Y: 416.59 90% R: 0.00°

Figure 3: Shows a rectangle drawn overtop of the initial input image. If this is the desired cropping dimensions of the video frame (confirm with Figure 4) you can use the four corners of the blue rectangle as the four corners of the input coordinates. When the mouse is hovered over a location on the image the red box shows the coordinates of the mouse.


*big_mullet_reg_1.png - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help

X: 16.276 Y: 20.513 W: 604.061 H: 453.746 px

Fill: Unset Stroke: Unset 1.00 O: 100

Image 3280 × 2464: embedded; clipped in layer Image. Click selection again to toggle scale/rotation handles.

X: 600.87 Y: 503.71 Z: 90% R: 0.00°

Figure 4: Shows how you can check if the area of the image covered by the rectangle is a good location to crop and warp the output video. To do this click on the blue rectangle then hold the Shift key and click the image as well. Both should be highlighted. Right click on the two selected objects and click Set Clip from the menu. This will produce the cropped image. To undo this to get the cropping coordinates hit “Ctrl + Z”.

Using Figures 3 and 4 as guides you should be able to create a good well aligned input image based on the coordinates of the blue rectangle. An example of what the output image will look like based on the input coordinates is shown below in Figure 5.

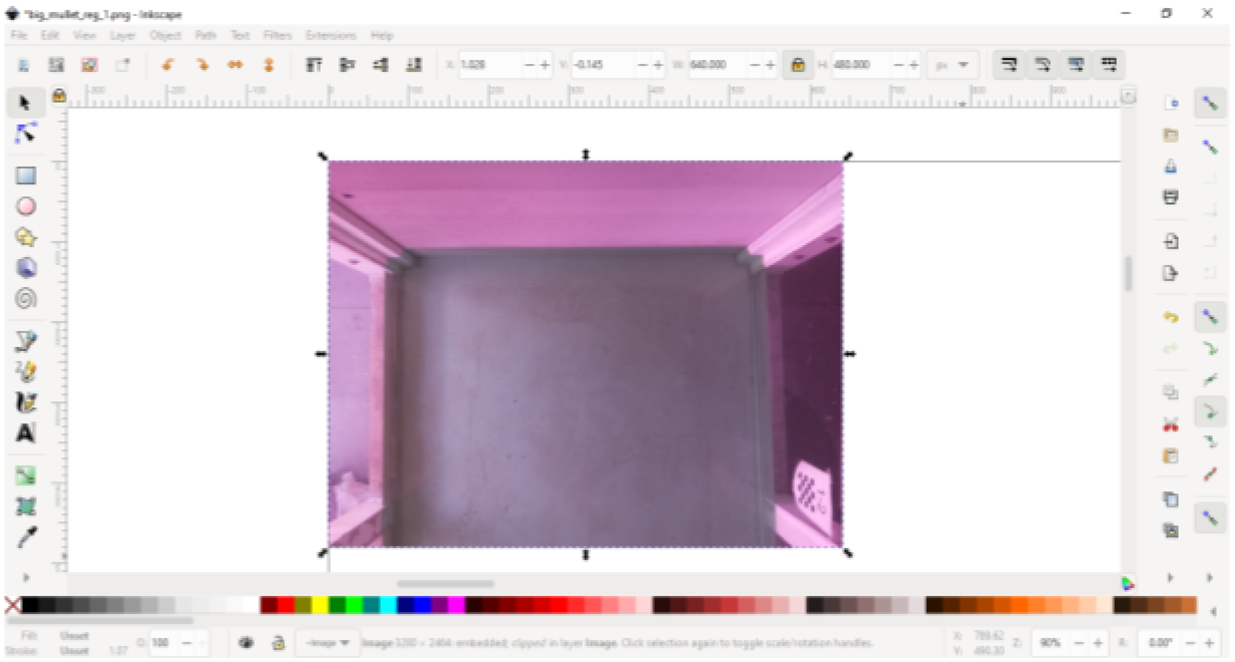

*big_mullet_reg_1.png - Inkscape
File Edit View Layer Object Path Text Filters Extensions Help

Figure 5: An example of the warped/cropped output of the video file is shown. This is what will be output to the destination coordinates of pts2

Change the Destination Coordinates (pts2) of the Input Video

The destination coordinates are less based on how each individual input video looks but more about how multiple of the input videos align in the final output example. The general location of each input video is estimated in the output video based on the camera's location in the series. But sometimes these estimated locations are not exactly perfect creating an imperfect stitching like seen in Figure 6. Don’t worry this is fixable but requires some guessing and checking to create a seamless stitched output.

In Figure 6 to fix the stitching error occurring in video 4 the output coordinates are edited to potentially better match the other videos. To do this focus on the direction each corner would need to move to better align the image. For example the top left corner (first coordinate in pts2) of video 4 needs to move to the right and potentially up. To move any corner of video 4 to the right increases the value of x (coordinates in order (x,y)). Increasing the x-coordinate moves the

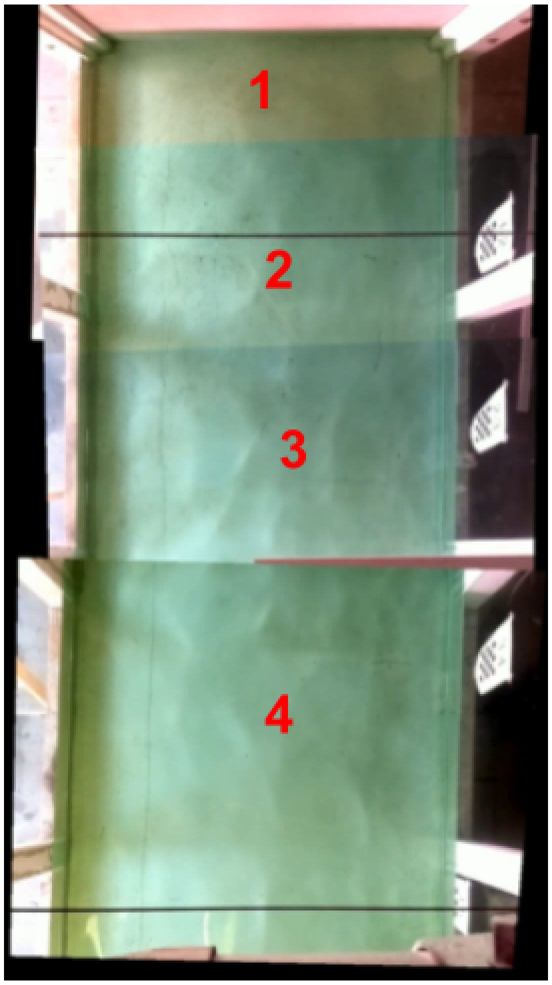


Figure 6: Shows an example of a stitching error occuring in Video 4. The red numbers are not actually apart of the output image but added for ease of interpretation. They represent the different cameras represented in this stitched output.

video to the right in the output and decreasing the x-coordinate would move the video to the left. To move any corner of video 4 up in the output video, decrease the y coordinate. Decreasing the y coordinate would move the corner up while increasing the y coordinate would push that corner down. This is because the coordinates of the videos uses the top left corner of the entire output as the origin (0,0).

Making small changes in the x and y coordinates of all the corners allows for the input video to be perfectly aligned in the output video. It will take multiple times to get the video in the perfect place just keeping modifying the coordinate locations using the defined relationships above.

Links

1. https://www.python.org/downloads/

2. https://www.python.org/downloads/release/python-395/
3. https://pip.pypa.io/en/stable/installation/
4. https://www.geany.org/
5. https://www.geany.org/download/releases/
6. https://pypi.org/project/opencv-python/
7. https://numpy.org/install/
8. https://github.com/PyImageSearch/imutils
9.
10. https://www.itpro.com/software/development/359246/how-to-download-from-github
11. https://pythonbasics.org/execute-python-scripts/
12. https://inkscape.org/