Exercise 3
Author: Daniel Lindberg

Step 1
Summarize
Digital video has quickly become the new standard for cinema, mobile digital media, and other video formats. The reasons for this is that digital video can be on demand, tied to social networks, used on big data analytics and it's cost-effective and secure to edit, and send across to consumers. Digital media has also looked for ways to become interactive. Digital media can be incorporated into augmented reality (AR) where you can edit a view of the world. Sometimes the embedded systems that can produce digital media such as mobile phones may require additional tools to help create an immersive user experience. Android for example has some applications that may require it's data to be uploaded to a cloud for much faster processing than most android phones can develop, digital video encoding can help videos compress and transport this content at faster rates, graphical rendering can help interact with camera data to produce content through much more complex developer environments and finally advanced user interaction which may include facial expression recognition. There are a number of examples online to develop this interactive media, tools such as Intel's OpenCV API to help image process, tools like Video For Linux 2 to help process camera data, the ability to transport codes by encoding with formats such as MPEG4. Sometimes the technology of transforming frames may require advanced compression ratios, used in MPEG4. These transformations generally will change the look of individual frames to help understand and recognize the scene that it is capturing. Finally incorporating these scenes into certain software can help render 3D animation. This is helpful for when the user wants to step through a shot frame by frame and look at topics such as lighting and point of view.

Question: Do you think researchers have a valid concern to think that real video would be indistinguishable from generated (rendered) video?
I think it is a valid concern that researchers believed that it would be hard to distinguish real video from generated video. Some of the tools that have been developed to render these scenes such as Blender are able to render video that looks incredibly lifelike. Thinking about modern movies today such as most comic book movies today , they have to render these complex animations and objects into a scene that can produce very life like objects. Now it also depends on the researchers, so the assumption that real video would mean that it is a life like video frame can be sort of misleading. This rendered video that is produced today have a number of added effects to help the user understand the scene more. For instance a person working on rendered video may add more contrast or apply a tensor transformation to sharpen the image frames, this could help the viewer understand the scene a little more; while in "real video" would probably by default have less contrast and not as sharp images.

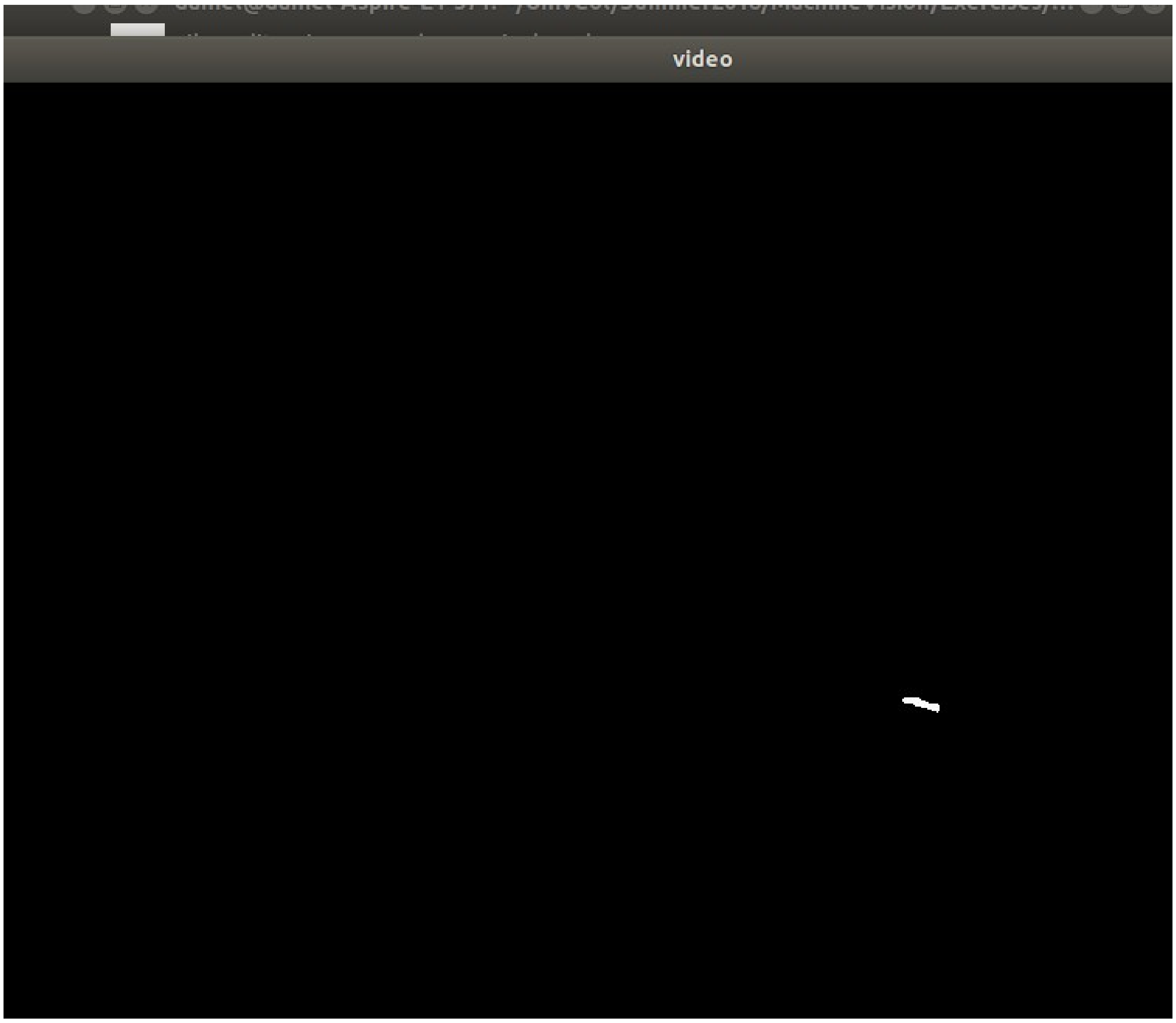
Step 2
In the Step 2 folder of the zipped file for Exercise 3 , you can find my code listed as MedianFilter.py.
To run it simple type "python MedianFilter.py".
You can look at the initial image as it is also in the Step 2 folder labeled as image.png
You can find the median filter image in the Step 2 folder labeled as processed_image.png.

Question: Did this help enhance the laser spot boundary at all?
I believe that this median filter did enhance the laser spot boundary as it reduced some of the noise with these videos. I believe it helped preserve some of the edges that the laser tracker had. Since the median filter is to run through the frame, replacing each entry with the median of the neighboring entries. I believe since the laser was able to produce part of the red signal far beyond it's initial dot it lands on you can see some of the reflecting light after the median filter.

Step 3

In the Step 3 folder of the zipped file for Exercise 3, you can find my code listed as FrameDifferencing.py. To run it simply type “python FrameDifferencing.py”. **Note:** My code is very slow on my embedded system, it may take a bit longer for some systems. Give it some time to display.

My algorithm goes and takes the video, frame by frame, then puts it into gray scale, I apply the medianBlur of the previous step in this lab and then I iterate through that image. I iterate pixel by pixel of the image and check to see if the value is below or above my threshold. If it is below my threshold value I set the pixel value to 0.0, so therefore it effectively reduces the background clutter of the bookshelf. However since in each frame, you go pixel by pixel and change the majority of the pixels it is pretty computationally intensive. It also has reduced the noise of the video , while in Step 2 I felt as though the median filter had the reflection of the red light being displayed , this code will reduce the reflection and follow a single dot. There are still some components of the bookshelf that were a bit hard to eliminate.
Screenshot below:



Step 4

I wrote code to extract the video by frames and then save it as a pgm file. You can find this code in the Step4 folder and it is titled: GrayScalePgm.py. To run it simply type “python GrayScalePgm.py”. A warning though it will convert all of the videos into frames and fill up the directory folder. I have them deleted in the submission since I didn’t want my submission to be humongous. Then once that code is complete to reassemble it as a mpeg4 video you run the following command:

```
ffmpeg -r 1/1 -i Frame%04d.pgm -codec:v mpeg4 -vf fps=25 -pix_fmt yuv420p ReEncode.mp4
```

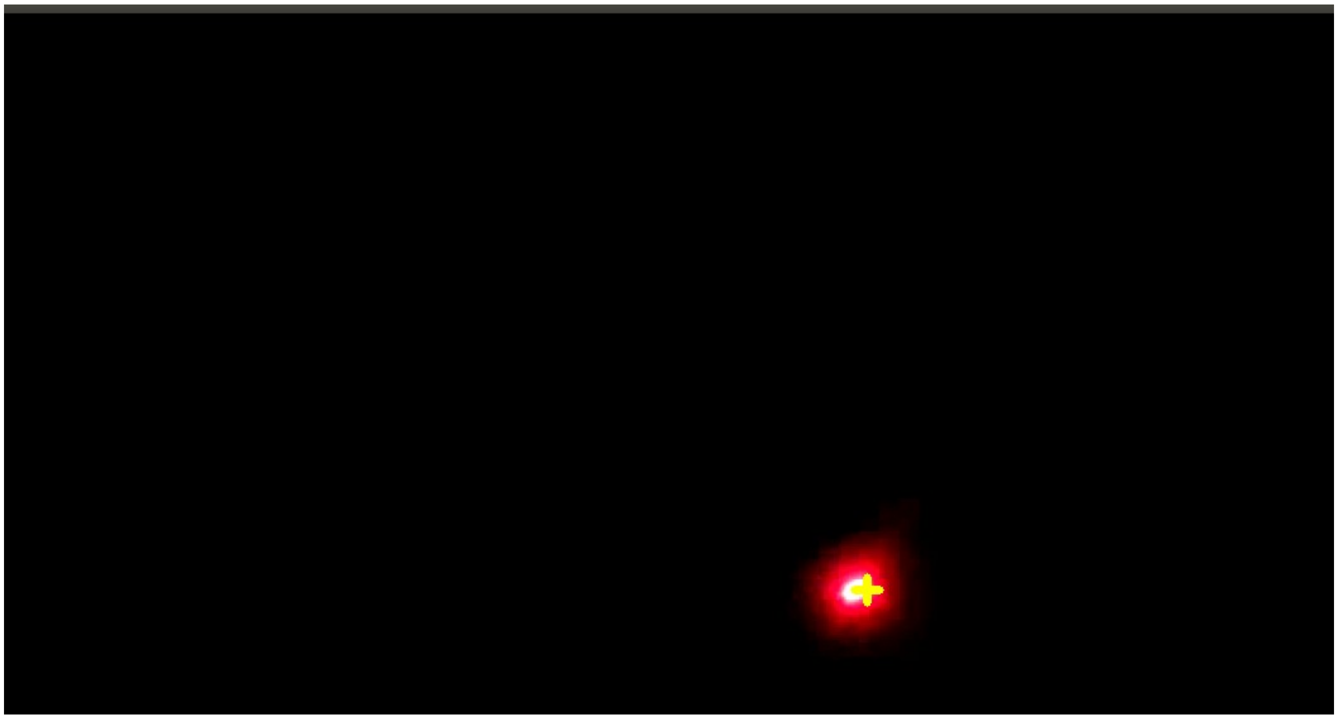
This will generate a video using mpeg4.

Step 5

I wrote code to track the laser pointer and place crosshairs on the object. This involved applying a threshold filter to the frames, finding the contours, finding the largest contour, getting the center object then drawing a yellow crosshair on the original frame that we received.

You can find my code in the Step 5 folder, to run it simply type “python TrackOverlay.py”

Screenshot below:



Step 6

I wrote the code to track the laser with the clutter. I adopted my code from Step 3, and added in the COM from Step 5. You can find my code in the Step 6 folder, to run it simply type “python ClutterTrackOverlay.py”

Screenshot below: