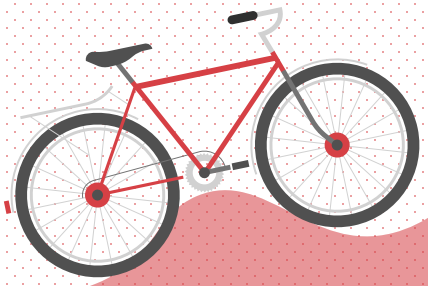
PHYSICS 301

Advanced Signal and Image Processing

ACTIVITY

05 VIDEO PROCESSING



Rene L. Principe Jr.
2015-04622

Dr. Maricor N. Soriano

objectives

Acquire live images from webcam to facilitate optical flow and face detection

Apply color segmentation on camera feed to track colored objects

key take-aways

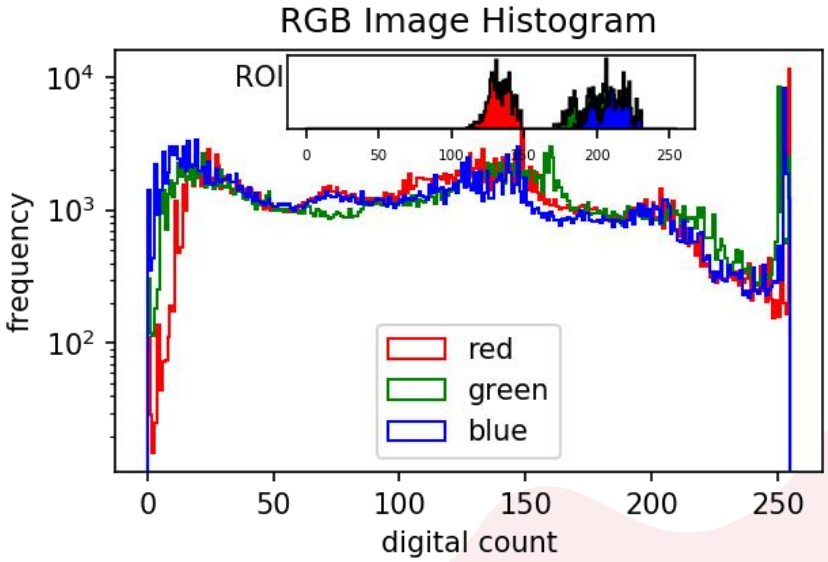
- Image processing techniques can be applied on videos which are essentially a series of still images.
- Motion has three spatial dimensions and the fourth one is time. Non-coplanar motion and low sampling rate (fps) would affect the quality tracking.
- Tracking based on color is easy but unreliable especially with the auto white-balance built in latest camera models.

SOURCE CODE

- Physics-301/Activity 05 - Basic Video Processing.ipynb at main · reneprincipejr/Physics-301 (github.com)
- https://drive.google.com/file/d/1S1juRZtkc_vtyp8Ffvfp62fFIlmtMSSb/view?usp=sharing

Video Capture (OpenCV)

In this activity, the camera feed is loaded using Open-CV Python [1]. The main principle of video processing is that **videos are a collection of sequential images**, hence, the **image processing** techniques can still be **applied frame by frame**. Shown below is a snapshot from the webcam and the corresponding histograms of the image and the cyan region of interest (ROI). The entire image has a continuous distribution across all channels while the ROI have a distinct and narrow distributions. The **ROI** shows high pixel intensities in its **green** and **blue** **histograms**; the two primary color channels constituting the **cyan color**.



Blob detection

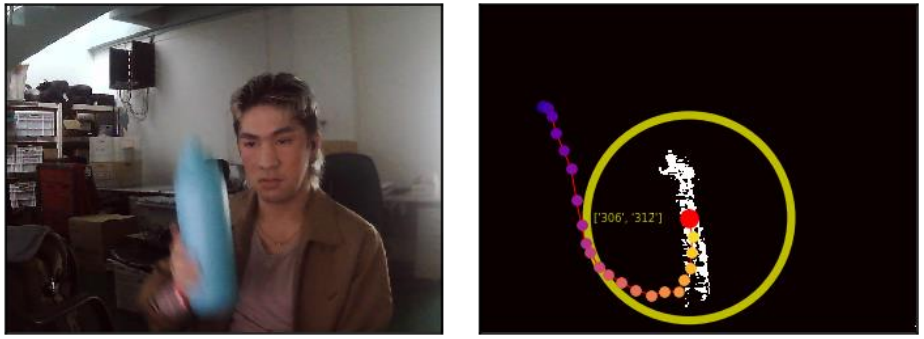
Non-parametric segmentation uses the **histogram back-projection**, a **non-computational** technique which treats histograms as look-up tables [2][3]. Recall that in Activity 1, the non-parametric method was generalized to be **robust to intensity variation and can sometimes reveal extra details**. We exploit this method's advantage because in this activity, we're dealing with scenes in very non-ideal conditions. Shown below is sample image segmentation of the **cyan aquaflask** as the object of interest. To track the object's movement, we employed **blob detection** using the image moments to **get the centroid coordinates** [4].

Tracking

From the webcam live feed, we first took 100 frames, segmented each image, and then stored the centroid coordinates. Shown below are some snapshots at different time frames while I move the object sinusoidally.

FRAME 22

FRAME 44

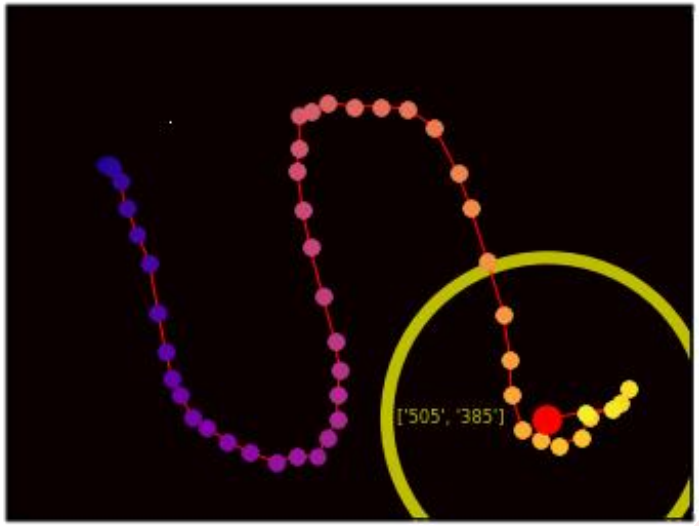


FRAME 66

FRAME 88

Discussion

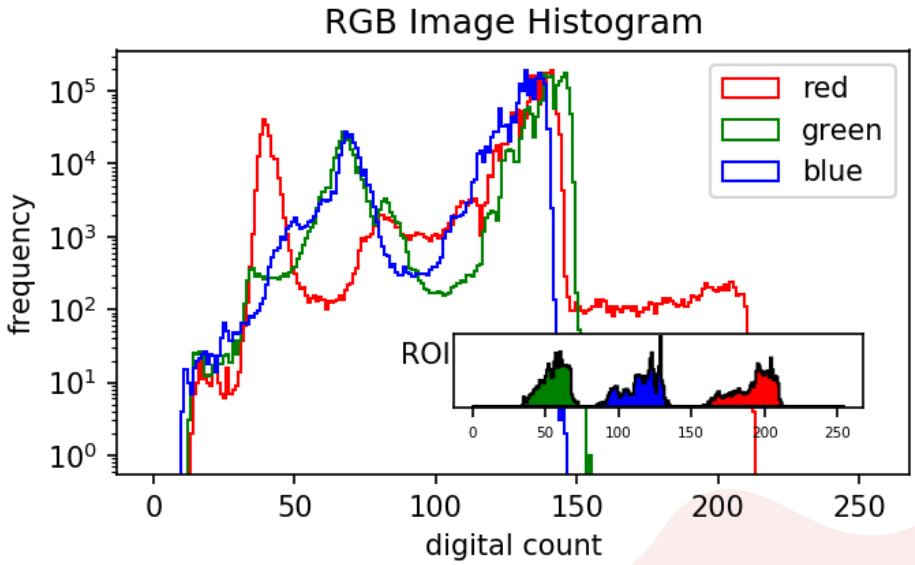
In summary, we were able to acquire live images from webcam and applied color segmentation to track objects. **Fundamental problems** include (1) built-in **auto white-balancing** in the webcam used which continuously change the colors in the scene and (2) **low resolution/frame rate** which results to poor capture of the motion. The effects are evident on the latter half of the tracking where the segmentation evidently struggles, consequently affecting the centroid detection and thus, the tracking.



View GIF here: https://giphy.com/gifs/ZpCk0dFM9jun27ONXp/fullscreen

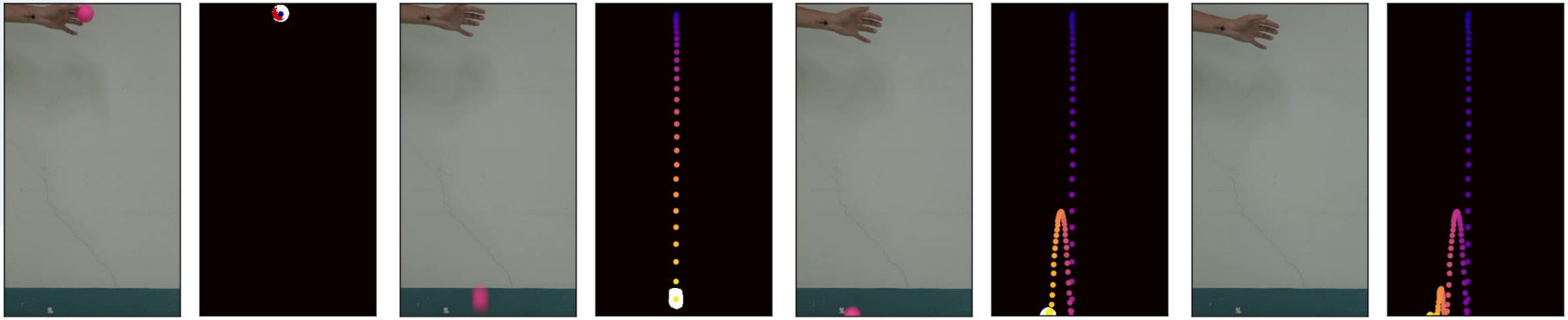
Applications

Since we've been limited by the resolution, framerate, and auto white-balancing of a webcam, here we try to **apply tracking on a video recorded using a digital camera**. To demonstrate a sample practical application, which I also did in my App Physics 186 class, we attempt to **track free fall motion and experimentally determine the acceleration due to gravity** [5]. The video used is at 60 frames-per-second, color-consistent, and has a high resolution. Shown below is a snapshot of the first image frame and the comparison of its histogram vs the ROI's (pink rubber ball).



Free Fall

Overall, the segmentation and centroid detection went smooth, not to mention that there were no artifacts since we have a uniform background.

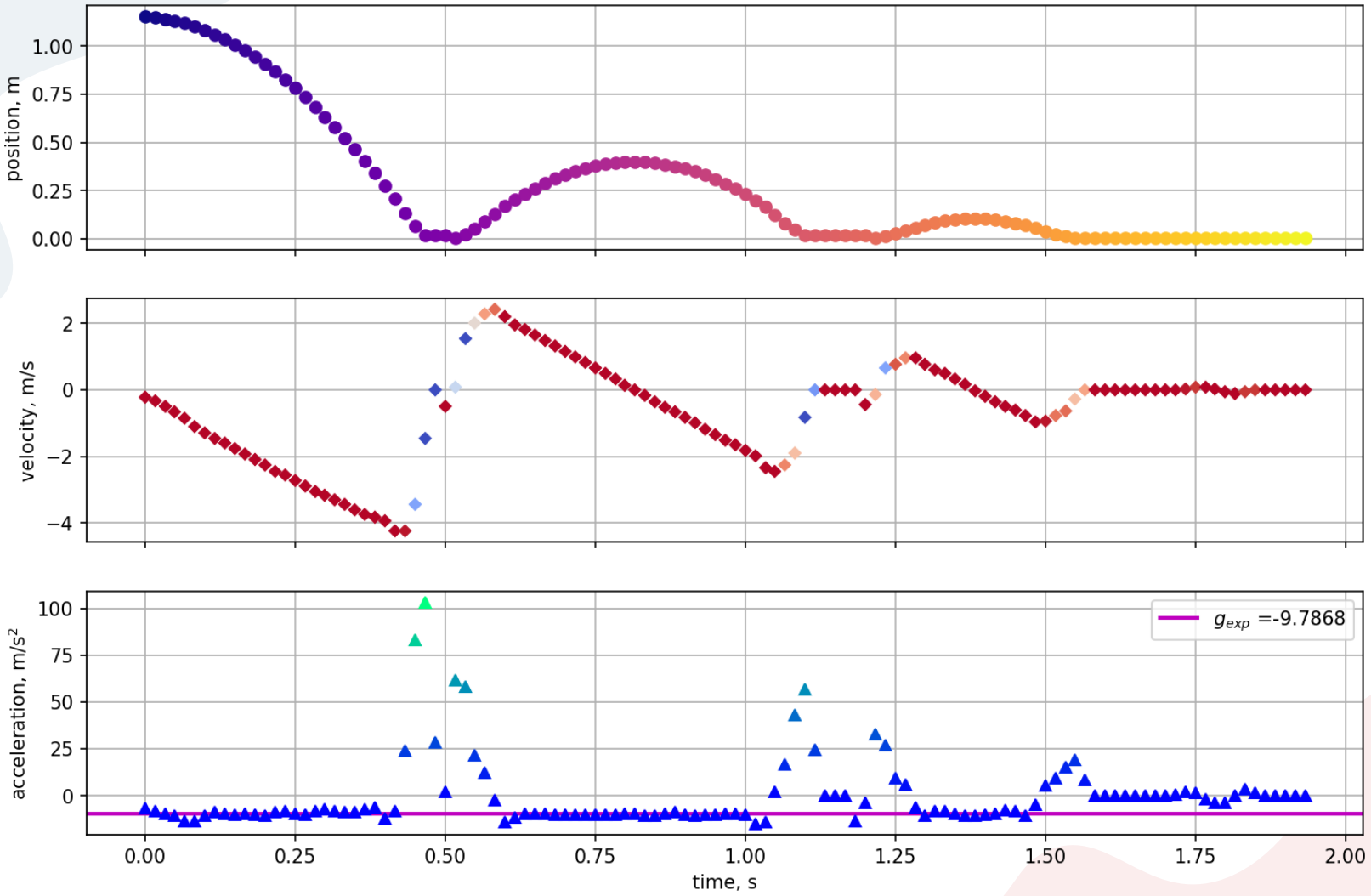


View GIF here: https://giphy.com/gifs/python-tracking-opencv-D4xFw0VusmBMC0UlKf/fullscreen

Using the **pixel-to-meter ratio** obtained by relating the balls diameter in pixels and in meters, we were able to **convert pixel trajectory into actual height values**. To extract g_{exp}, we take the first and second derivative of the trajectory which represents the velocity and acceleration functions of the free-falling body, respectively. As shown in the plots (next page), we got **g_{exp} = -9.79 m/s^2**, which deviates by **0.2% from the theoretical value**.

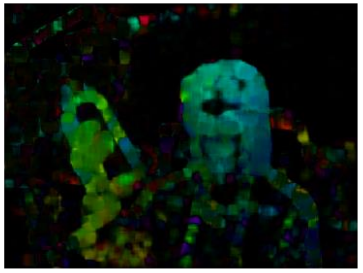
In conclusion, we were able to successfully perform a kinematics experiment through video processing.

Kinematic Graphs

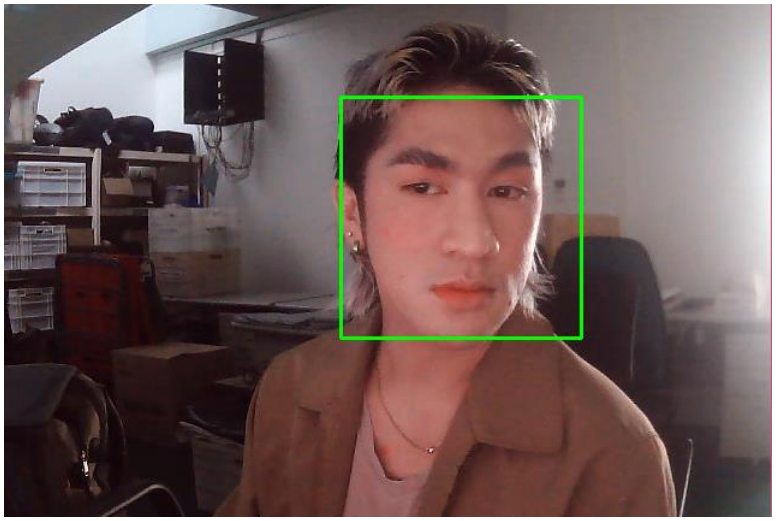
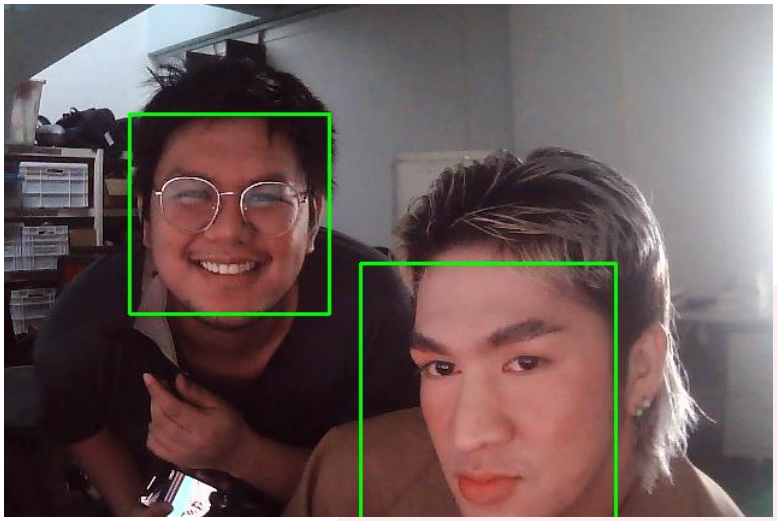


Optical Flow



Face Detection

reflection

This activity was quite challenging since it deals with non-static data. I took me a while to realize the webcam that I used webcam had auto-white balancing but eventually, it was satisfying to see my code run smoothly and live-track a moving object. In addition, I was able to perform a staple kinematics experiment without having to use the usual motion sensor that's only available in the school premises. Lastly, we we're able to apply the optical flow [6] and face detection [7] counterparts on python using Open-CV which opens an endless number of potential applications. I'd like to give credit to Kenneth Domingo for the pre-recorded video of the free-falling ball which I again used in this report. I think I gave out decent visualizations of the tracking process and discussed the nuances in the observations.

With that said, I'd give myself a score of **98/100.**

references

- [1] OpenCV: Getting Started with Videos
- [2] M. Soriano, Applied Physics 186 - Image Segmentation, (2019).
- [3] M. Soriano, Physics 301 - Color Segmentation, (2022).
- [4] Find Center of a Blob (Centroid) Using OpenCV (C++/Python) | LearnOpenCV
- [5] Basic Video Processing (rlprincipe.wixsite.com)
- [6] OpenCV: Optical Flow
- [7] Face Detection in 2 Minutes using OpenCV & Python | by Adarsh Menon | Towards Data Science