# [Fall 2023] ROB-GY 6203 Robot Perception Homework 1

Ifedayo Olusanya

Submission Deadline (No late submission): NYC Time 11:00 AM, October ~~04~~ 11, 2023
Submission URL (must use your NYU account): https://forms.gle/TNrJy7r3smx2t1ed8

1. Please submit the **.pdf** generated by this LaTex file. This .pdf file will be the main document for us to grade your homework. If you wrote any code, please zip all the **code** together and **submit a single .zip file**. Name the code scripts clearly or/and make explicit reference in your written answers. Do NOT submit very large data files along with your code!

2. You don't have to use AprilTag for this homework. You can use OpenCV's Aruco tag if you are more familiar with them.

3. You don't have to physically print out a tag. Put them on some screen like your phone or iPad would work most of the time. Make sure the background of the tag is white. In my experience a tag on a black background is harder to detect.

4. Please typeset your report in LaTex/Overleaf. Learn how to use LaTex/Overleaf before HW deadline, it is easy because we have created this template for you! **Do NOT submit a hand-written report!** If you do, it will be rejected from grading.

5. Do not forget to update the variables "yourName" and "yourNetID".

## Contents

**Task 1 Sherlock's Message (2pt)** **2**
a) (1pt) . . . 2
b) (1pt) . . . 3

**Task 2. Low Dimensional Projection (5pt)** **4**

**Task 3 Camera Calibration (3pt)** **5**

**Task 4 Tag-based Augmented Reality (5pt)** **6**

# Task 1 Sherlock's Message (2pt)

Detective Sherlock left a message for his assistant Dr. Watson while tracking his arch-enemy Professor Moriarty. Could you help Dr. Watson decode this message? The original image itself can be found in the data folder of the overleaf project (https://www.overleaf.com/read/vqxqpvbftyjf), named `for_watson.png`

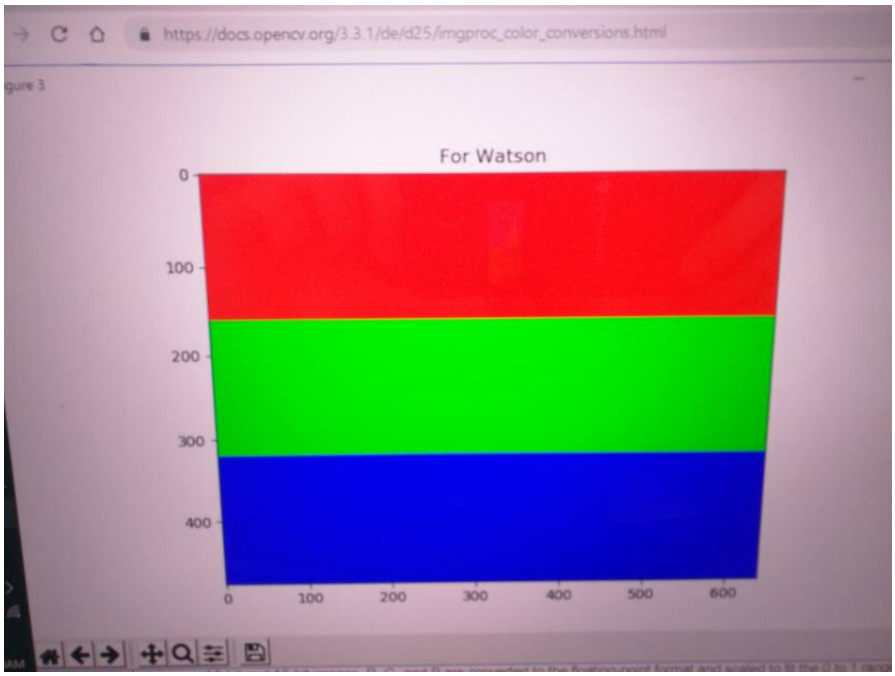


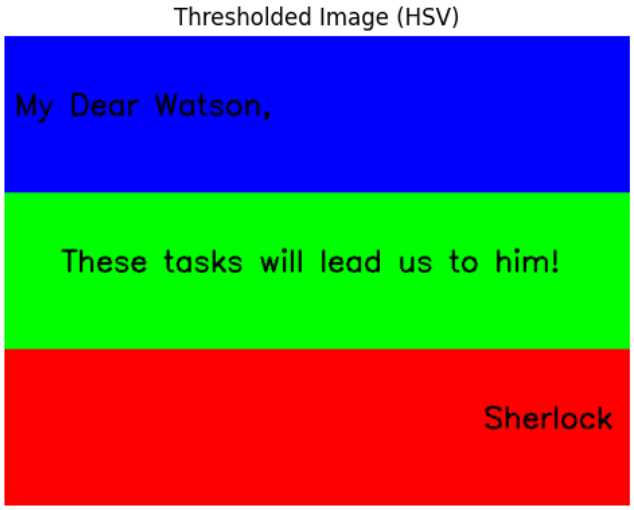

Figure 1: The Secret Message Left by Detective Sherlock

## a) (1pt)

Please submit the image(s) after decoding. The image(s) should have the secret message on it(them). Screenshots or images saved by OpenCV is fine.

**Answers:**

You can use this code snippet to include a picture





Name: Ifedayo Olusanya, netID: iio2004

## b) (1pt)

Please describe what you did with the image with words, and tell us where to find the code you wrote for this question.

**Answers:**
Type your answer here

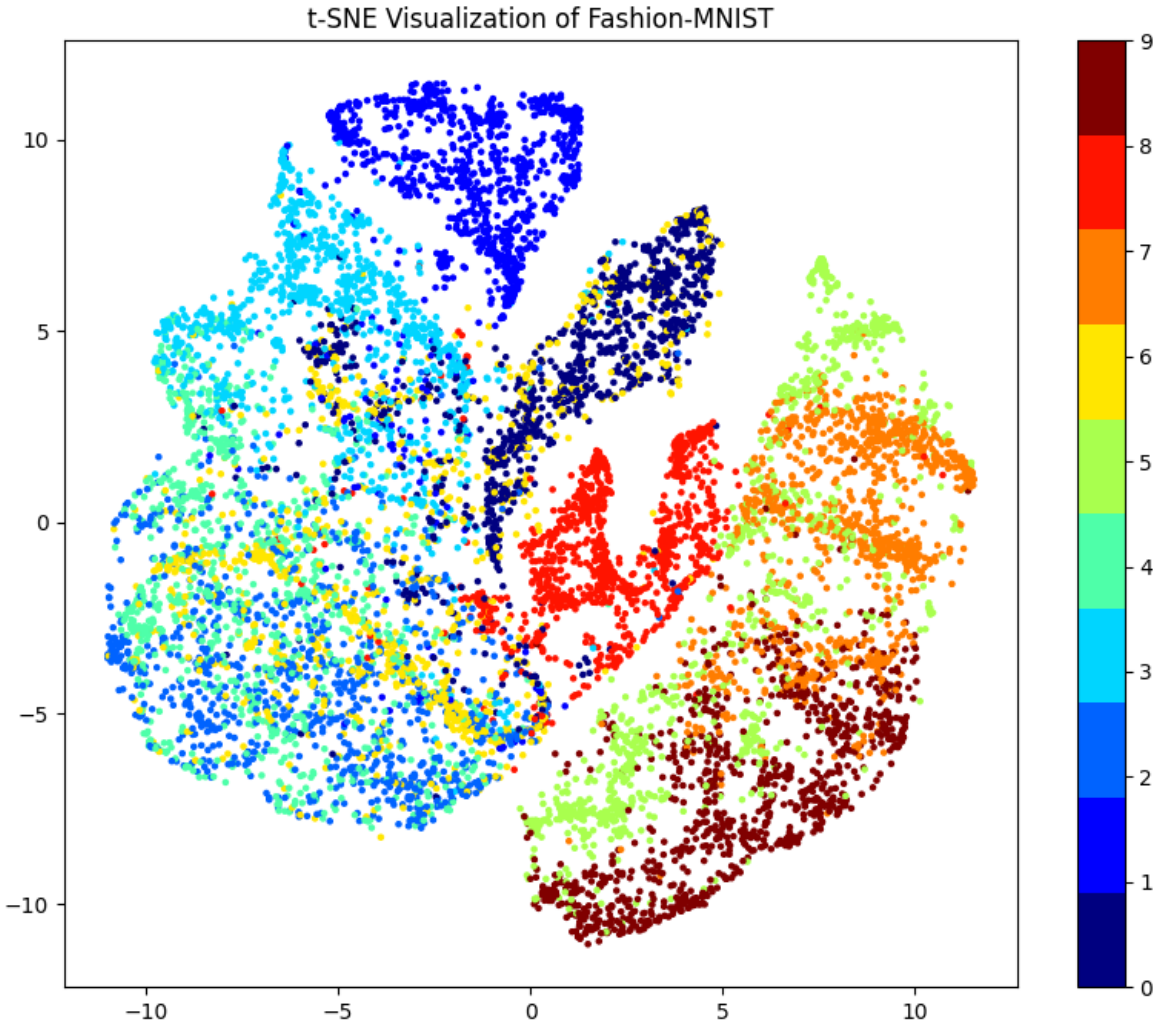
To figure out the puzzle, I converted the image from RGB2HSV then I set my HSV upper and lower threshold values using a custom built GUI.I did a bitwise and masking and stoped when I found suitable threshold values that displayed the messages from sherlock

code can be found here: https://github.com/IJAMUL1/ROB-GY-6203-Computer-Vision/blob/main/HW1/robot_percetion_HW1_Q1.ipynb

# Task 2. Low Dimensional Projection (5pt)

Given the Fasion-MNIST dataset, **train** an unsupervised learning neural network that gives you a lower-dimensional representation of the images, after which you could easily use tSNE from `Scikit-Learn` to bring the dimension down to 2. **Visualize** the results of all 10000 images in one single visualization.

**Answers:**
Type your answer here In addition to the autoencoder's denoising and image reconstruction capabilities, this code




further extends its utility by incorporating dimensionality reduction using t-Distributed Stochastic Neighbor Embedding (t-SNE). After encoding the test images with the trained autoencoder, the resulting encoded data is subjected to t-SNE transformation. By applying t-SNE, a powerful technique for reducing high-dimensional data into a two-dimensional space, it becomes possible to visualize the distribution of encoded data in a more compact form. In the provided code, t-SNE is configured to reduce the data into two dimensions, and the resulting two-dimensional points are plotted in a scatter plot. Each point represents an encoded data instance, and the color of the points corresponds to their original labels in the Fashion MNIST dataset. This visualization provides insights into how the autoencoder has learned to group similar data points together in a reduced space, potentially revealing underlying patterns or clusters within the data. It serves as an illustrative example of how autoencoders can be coupled with dimensionality reduction techniques to gain a deeper understanding of complex datasets, such as Fashion MNIST.

code can be found here: https://github.com/IJAMUL1/ROB-GY-6203-Computer-Vision/blob/main/HW1/robot_perception_HW1_Q2.ipynb

# Task 3 Camera Calibration (3pt)

Use the pyAprilTag package provided in the class, or other free packages (e.g., OpenCV's camera calibration toolkit) that you may be aware of, to **calibrate** your camera and provide the full K matrix, with the top two distortion parameters k1 and k2.

**Answers:**
Type your answer here

the top two distortion parameters k1 and k2 are [ -0.07590859, 0.03753864 ] respectively.

the full camera Matrix is

```
[[589.71911352   0.         308.98611708]
 [  0.         589.52255378 270.53585099]
 [  0.           0.           1.        ]]
```

code can be found here: https://github.com/IJAMUL1/ROB-GY-6203-Computer-Vision/blob/main/HW1/robot_perception_HW1_Q3_Camera_callibration.ipynb

# Task 4 Tag-based Augmented Reality (5pt)

Use the pyAprilTag package to detect an AprilTag in an image (or use OpenCV for an Aruco Tag), for which you should take a photo of a tag. Use the K matrix you obtained above, to draw a 3D cube of the same size of the tag on the image, as if this virtual cube really is on top of the tag. **Document** the methods you use, and **show** your AR results from at least two different perspectives.

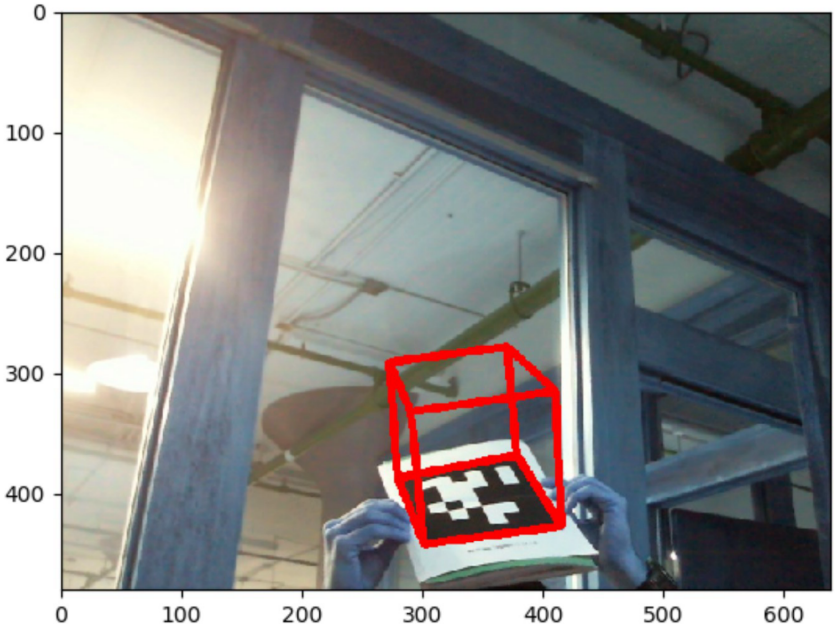



Figure 2: Caption

**Tips**: There are many ways to do this, but you may find OpenCV's `projectPoints`, `drawContours`, `addWeighted` and `line` methods useful. You don't have to use all these methods.

**Answers:**
Type your answer here

code can be found here: https://github.com/IJAMUL1/ROB-GY-6203-Computer-Vision/blob/main/HW1/robot_perception_HW1_Q4_Augmented_Reality.ipynb

The code I developed implements an Augmented Reality (AR) application using opencv and the AprilTags library. It begins by loading essential camera calibration data, crucial for precise 3D-to-2D transformations. The AprilTag detector is initialized, enabling the identification of AprilTags in real-time webcam frames. When detected, the code estimates the pose of these tags in the camera's coordinate system. It then overlays 3D cubes on the tags, projecting them into the 2D image plane. These visualizations are not only displayed in real-time but also saved as images for later review.

see augmented reality output below