Lab 3: Template Matching

Due Sep 14, 2020 by 4pm **Points** 10
Submitting a text entry box, a website url, a media recording, or a file upload
Available after Sep 11, 2020 at 3pm

Overview

In this lab, you'll read in a movie and use template matching to track targets as they move through the movie.

Prologue: Reading the location of mouse clicks

In this lab it will help to have the user click on an image, and have your program read the location of the mouse click. You can do this in OpenCV by assigning a "callback" function to a particular window, using "cv2.setMouseCallback()". You pass in the name of the window, the name of your callback function, and an optional parameter (see the OpenCV documentation for this function).

When the user clicks in that window, your callback function is called. To return the location of the click, pass in an empty list as the optional parameter to the callback function, and have your callback function append the (x,y) location of the mouse click to the list:

```
# Mouse callback function. Appends the x,y location of mouse click to a list.
def get_xy(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDOWN:
        print(x, y)
        param.append((x, y))
```

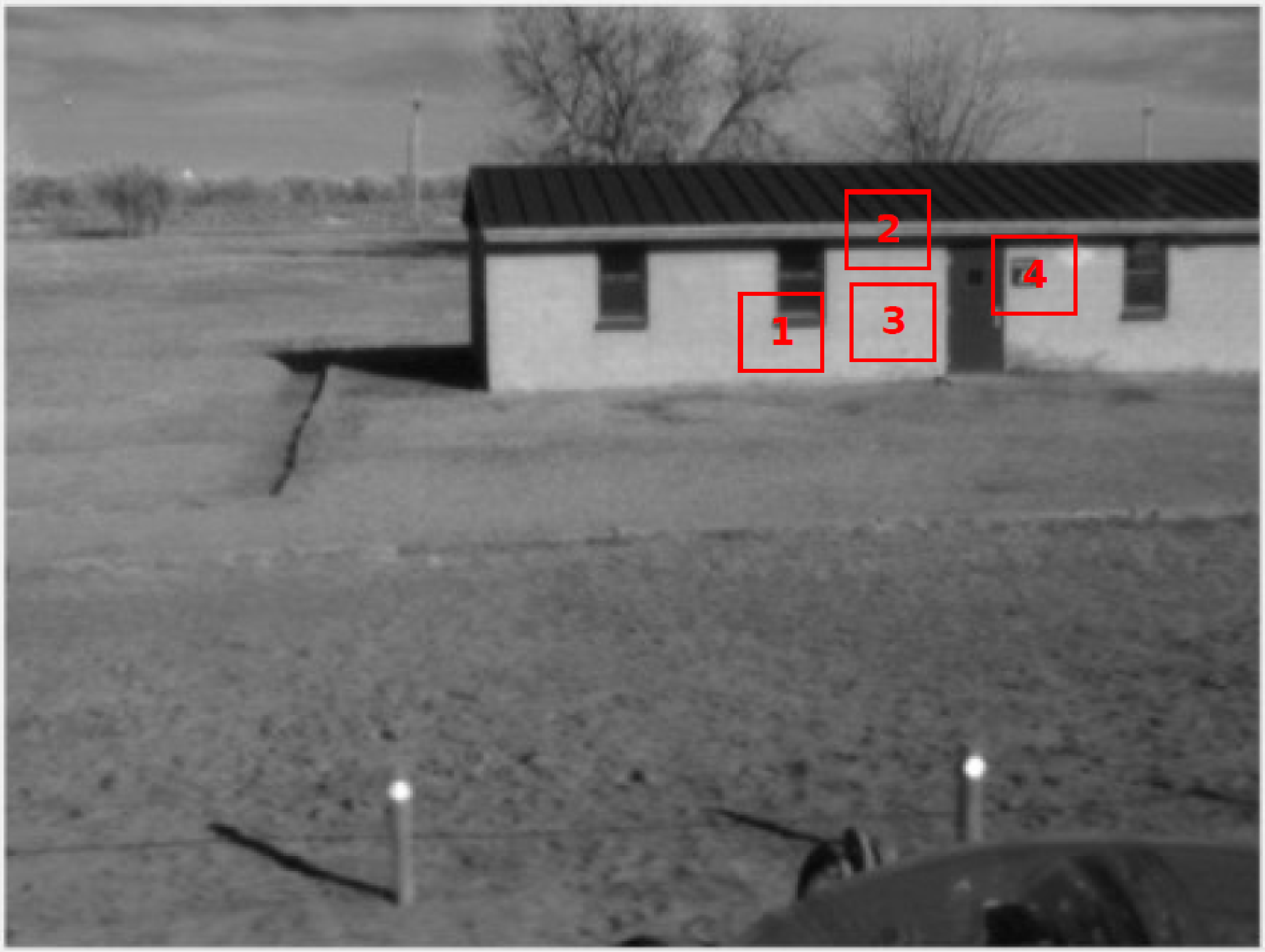
The Actual Lab

Write a program to track a template sub-image, through each frame of the movie file **building.avi** (https://elearning.mines.edu/courses/25410/files/1886667/download?download_frd=1)

.

Namely, read the first image of the movie and let the user pick a template to track. Then use the method of “normalized cross correlation” to track that template to each subsequent image in the movie.

1. Draw a rectangle indicating the location of the matched template on each image. Create an output movie file showing the locations of the matched template on each image.
2. Experiment with the choice of template that you extract from the first image. Try the corner of the window (point 1), the edge of the roof (point 2), the middle of the wall (point 3), and the sign next to the door (point 4). Which point seems to allow the best tracking, and why? (Note - the match may not be correct in every single image, no matter which point you pick.)



3. When the template is matched correctly, approximately what correlation scores do you see?

Post your video, and the answers to questions 2 and 3 on Canvas, along with the Python code you used to compute the answers. Each person on the team should post the answers and code.