Programming assignment 2
CM2000, Sports and Health Instrumentation

Preparation

Before the lab, you should install python and vscode. You can choose another IDE, but then I do not guarantee that I will be able to help as much. Install also python and python notebook extensions for vscode. Make sure you can run python both from vscode and your terminal.

It is recommended that you set up a virtual environment before you start to install packages on system python. The list of packages necessary to run this lab is on requirements.txt. You can install them by opening the terminal on your project folder, activating your virtual environment, and running pip install -r requirements.txt.

Please note that opencv will work on assignments A and B but not on assignment C. For assignment 3 you will need to install opencv-contrib-python. You can do this by creating a new virtual environment and installing opencv-contrib-python instead of opencv-python (recommended). You can also uninstall opencv-python and install opencv-python in the same environment. Note that having both packaged installed will not work.

If you are not familiar with python, the following list of resources can help:

Instructions on how to properly manage python virtual envs:

- unbuntu
- mac/windows
- using conda

Easily switch between environments on vscode: link

If needed, you can also use google colab for this assignment: link

General instructions (all parts)

To pass the assignment, you need to complete all the mandatory tasks of parts A, B and C. To get a higher grade (VG), you need to successfully complete at least two of the bonus assignments, necessarily from different parts.

You can implement your code using modules, but you are required to deliver a report in a Jupyter Notebook file for each part, using the structure provided. You are free to use and change all the code, including the additional modules tools.py and kalmanFilter.py - but make sure to explain your changes in the written assignment text or in comments.

Report requirements:

- Python notebooks, free report structure (creativity is appreciated; not chaos).
- You are allowed to place code outside of the .pynb file, but make sure to upload it.
- One should be able to re-run your notebooks.
- Important changes to the code should be mentioned in the text or explained in comments.

Part A: Object detection and object tracking

In this part we will track an object in a video. To track an object, one approach would be to use pre-trained machine learning models to detect the object in every frame. We could also fine-tune those models to identify objects of our interest. However, we don't always know ahead of time what we would like to track - and maybe we will not have enough data to train a machine learning image detector. To our luck, there is a different approach which uses image processing techniques (such as blob, edge and contour detection) to follow any given region of interest in a video.

In this lab, we will use OpenCV's library to pick any given region of interest in the first frame of a video and follow it throughout the video. Those algorithms can be implemented to work directly in your web or mobile app, allowing access to real-time feedback. For the sake of didactics, we will perform the analysis in python with pre-recorded videos. Feel free to play around further.

- OpenCV tutorials: https://learnopencv.com/how-to-select-a-bounding-box-roi-in-opencv-cpp-python/
- OpenCV in real time: https://docs.opencv.org/3.4/dc/de6/tutorial_js_nodejs.html
- Kalman filter for object tracking: https://towardsdatascience.com/what-i-was-missing-while-using-the-kalman-filter-for-object-tracking-8e4c29f6b795
- Kalman filter: https://medium.com/@jaems33/understanding-kalman-filters-with-python-2310e87b8f48
- Kalman filter theory and examples: https://www.kalmanfilter.net/multiExamples.html

Code provided:

- (python) Defining and tracking a region of interest in a video.

Tasks

Below you will find a summarized description of the tasks, note that details and tips can be found on the jupyter notebook.

1. Object detection for a video file
   1.1. Code assignment: Run object detection tracking for a .mp4 file. Use snatch.mp4 as an example and track the barbell path by defining the correct region of interest (ROI).
   1.2. Written assignment: Change the definition of the mask and see what happens. Write a paragraph about your observations. Propose a way of making the definition of the mask more generalizable (you don't need to implement it, just describe what you would do)
2. Draw centroid
   2.1. Code assignment: Draw a circle of radius 2 and colour red marking the centroid of the region of interest in every frame. Implement the code to retrieve the coordinates of the centroid of the ROI. Then save a .gif with the drawing of the ROI and centroid.
3. Centroid path
   3.1. Code assignment: Plot x and y components of the centroid path.
   3.2. Written assignment: Write a short paragraph analysing and discussing the result of the centroid tracking. Then, write a paragraph discussing possible use cases of this algorithm in this or other sports.

4. Draw contours
    4.1. Code assignment: Implement the code to find and draw contours within the region of interest. Look up cv2.findContour and try different thresholds. Save a .gif with the drawing of the ROI and contours.
    4.2. Written assignment: Analyse and compare the results that you get from centroid and contour tracking. Which one was more robust? Which one would you use to track the barbell? Write a short paragraph.

Bonus tasks

Below you will find a summarized description of the tasks, note that details and tips can be found on the jupyter notebook.

1. (For you who like sports and programming): Use kalmanFilter.py (or implement your own Kalman filter) in the centroid tracking. Change the initialization parameters. Explain why you chose certain parameters and how they change the results. Draw on the video both the original centroid (red circle) and the kalman filtered prediction (green circle). You don't need to draw the barbell path. Save a .gif file to ./object_tracking/snatch_kalman.gif' with the result. Plot a chart with the x and y the coordinates of the centroid with and without the filter, analyse and discuss your results.
2. (For you who like sports and biomechanics): Draw the barbell path of the centroid on the video. Save a .gif file to './object_tracking/snatch_path.gif' showing the barbell paths. Then, compute the vertical and horizontal velocities of the barbell in m/s. Plot the x and y velocities. Find and plot the frame where the velocity was maximum when the athlete is pulling the barbell up. If you did the previous bonus task, compare the velocity computed with and without the Kalman filter.
3. (For you who like sports): Record YOUR OWN sports-related video and perform object tracking. Analyse the results and discuss how they would be used in real life.

Part B: Pose estimation and skeleton tracking

In this part we will follow the tutorial provided by tensorflow in python. The tutorial uses MoveNet, which is one of the best performing pose estimation models currently available for multiple platforms. There are several versions of MoveNet which can be implemented directly in your web or mobile app, allowing access to real-time feedback. For the sake of didactics, we will perform the analysis in python with pre-recorded videos. Feel free to play around further.

- Instructions on how to install tensorflow: https://www.tensorflow.org/install
- Link to the original tutorial: https://www.tensorflow.org/hub/tutorials/movenet
- More on pose estimation: https://learnopencv.com/yolov7-pose-vs-mediapipe-in-human-pose-estimation/

Code provided:

• (python) Skeleton tracking for an image file and for an image sequence.

Tasks

Below you will find a summarized description of the tasks, note that details and tips can be found on the jupyter notebook.

1. Video (image sequence) skeleton tracking
   1.1. Code assignment: Run skeleton tracking for a .mp4 file. Use running_2.mp4 as an example.
2. Joint coordinates
   2.1. Code assignment: Change the code to retrieve the joint coordinates and confidence scores. Plot joint coordinates of hip, knee and ankle of right and left sides together with confidence scores of the predictions.
   2.2. Written assignment: Analyse the correctness of the predictions and the confidence scores. Try to establish a threshold. Write down your reflections and conclusions.
3. Knee angles
   3.1. Code assignment: Compute and plot the knee angles (right and left) on 2D (sagittal) plane.
   3.2. Written assignment: Analyze and discuss the results, limitations and sources of error. Then, write about how it could be used to improve technique in this or other sports.

Bonus tasks

Below you will find a summarized description of the tasks, note that details and tips can be found on the jupyter notebook.

1. (For you who like sports): Choose another sports-related video, perform the skeleton tracking and compute important angles or other biomechanics variables. Analyse the results and discuss how they would be used in real life.
2. (For you who like sports and programming): Think of a method to identify the predictions where the model did a bad job. Is the confidence score enough? Can you identify where right and left sides swap? Think of a method to improve those predictions and implement it. Describe your method and compare the results with and without your method. (Tips: There are multiple possible approaches. You have access to the confidence score of every prediction. You have access to a Kalman Filter implementation. Ask ChatGPT. Etc.)

Part C: Homography

In this part we will project a 2D image of a sports event recorded from a camera into a 2D image of a sports field. In order to do that transformation, we will use a technique called homography.

In this lab, we will use OpenCV's library to compute the homography matrix and project the video frame to the field image. We will also use OpenCV tracking algorithms based on detection – therefore different from Part A. Please note that opencv-python package will work on assignments A and B but not on assignment C. For assignment 3 you will need to install opencv-contrib-python. You can do this by creating a new virtual environment and installing opencv-contrib-python instead of opencv-python (recommended). You can also uninstall opencv-python and install opencv-python in the same environment. Note that having both packaged installed will not work.

- Some theory on homography: https://docs.opencv.org/4.x/d9/dab/tutorial_homography.html
- More theory on homography: https://towardsdatascience.com/understanding-homography-a-k-a-perspective-transformation-cacaed5ca17
- Examples: https://learnopencv.com/homography-examples-using-opencv-python-c/

Code provided:

- (python) Homography for an image and object tracking for an image sequence.

Tasks

Below you will find a summarized description of the tasks, note that details and tips can be found on the jupyter notebook.

1. Homography of an image
   1.1. Code assignment: Run homography for a frame of a hockey and american football play. Save your results as './homography/homography_hockey.jpg' and './homography/homography_football.jpg'. Then, choose another sport where you think the same technique could work, find a picture of the field and a frame of a play, and perform the homography. Save your results as './homography/homography_sport.jpg'
   1.2. Written assignment: Try using different points in the field to perform the homography. Identify the limitations and write a paragraph with your reflections.
2. Homography of a video
   2.1. Code assignment: Run the code to use the homography matrix computed for the first frame of the video and perform the homography for the whole video. Do it for both hockey and football. Save your results as './homography/homography_hockey_static.gif' and './homography/homography_football_static.gif'.
   2.2. Written assignment: Analyse carefully the results for both videos. Identify the most important problem with using the first frame homography matrix for the entire video (we will fix it next!). Write a paragraph with your reflections.
3. Homography of a video with moving camera
   3.1. Code assignment: To account for the moving camera, we perform object tracking of the points used in the homography and recompute the matrix for every frame. Run the code that does point tracking and homography computation. Do it for both hockey and football. Save your results as './homography/homography_hockey_dynamic.gif' and './homography/homography_football_dynamic.gif'.

3.2. Code assignment: Choose another sport where you think the same technique could work, find a picture of the field and a short video of a play, and perform the homography. Save your results as './homography/homography_sport_dynamic.gif

3.3. Written assignment: Think about how this technology could be applied in the sport that you chose on task 3.2. Write a paragraph with your reflections.

Bonus tasks

Below you will find a summarized description of the tasks, note that details and tips can be found on the jupyter notebook.

1. Now you are already familiar with multiple tracking algorithms, choose one of the videos provided (hockey or american football) and perform the homography while tracking the position of one or more players. Plot the position(s) of the player(s) that you track on the image of the field as dots of different colors. Save a gif displaying the dots moving on the field as './homography/bonus_task.gif'.