CSC478 - Robotic Perception

Assignment 2
Due: March 3, 2023, at 4 PM

1 Pen and Paper (40 Marks)

Submission: Please upload a PDF of solutions – either TeX or hand-written and scanned. Please ensure non-TeX solutions are legible or you may lose marks.

1.1 Greedy Matching (5 Marks)

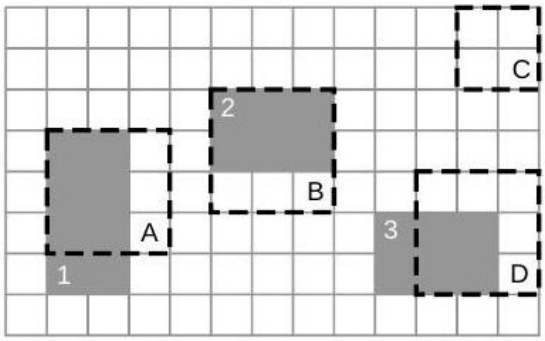
We are given the following matching scores for 5 detections from frame i to $i+1$. The value at j-th row, k-th column is the matching score of the j-th box in frame i to the k-th box in frame $i+1$. Use the greedy matching algorithm to match detections across frames i to $i+1$.

$i+1$	1	2	3	4	5
1	0.93	0.52	0.01	0.09	0.19
2	0.18	0.35	0.24	0.80	0.81
3	0.85	0.55	0.41	0.05	0.61
4	0.09	0.30	0.15	0.67	0.89
5	0.67	0.56	0.33	0.49	0.59

What is the matching assignment of the greedy solution?

1.2 Detection Performance (15 Marks)

Consider the following "image" where the three gray shaded boxes $(1, 2, 3)$ correspond to the true objects and the four dashed boxes (A,B,C,D) correspond to detections of an object detector:



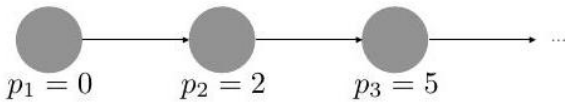
(a) Calculate the IoU (Intersection-over-Union) metric for the pairs $(A, 1), (B, 2)$ and $(D, 3)$. Assuming a detection threshold of $\tau = 0.5$, which of the three objects has been correctly detected?

(b) Calculate the number of true positives (TP), false positives (FP) and false negatives (FN) assuming the same detection threshold as in the previous question ($\tau = 0.5$).

(c) Calculate precision and recall at a detection threshold of $\tau = 0.6$.

1.3 Kalman Filtering (20 Marks)

We will now use a Kalman filter to estimate the state $\mathbf{x}_t = \begin{bmatrix} p_t \\ v_t \end{bmatrix}$, i.e. the position and velocity, of a moving ball. We will assume simple 1D linear motion as shown below,



where $y_t = [p_t]$ are the observations at each timestep t. As introduced in the lecture, Kalman filtering consists of two steps which are described using the following Kalman filter equations. The first step is the prediction step where we predict the mean $\bar{x}_t$ and covariance $\bar{P}_t$ of the state one timestep ahead, given x_{t-1} and P_{t-1},

$$\hat{\mathbf{x}}_t = \mathbf{F}\mathbf{x}_{t-1}$$
$$\hat{\mathbf{P}}_t = \mathbf{F}\mathbf{P}_{t-1}\mathbf{F}^{\mathrm{T}}$$

where $\mathbf{F}$ is the state transition matrix and $\mathbf{H}$ is the observation matrix. The second step is the correction step which corrects our estimate of the state x_t using the observations y_t, using the following equations,

$$\mathbf{K}_t = \hat{\mathbf{P}}_t\mathbf{H}^{\mathrm{T}}\left(\mathbf{H}\hat{\mathbf{P}}_t\mathbf{H}^{\mathrm{T}} + \mathbf{R}\right)^{-1}$$
$$\mathbf{x}_t = \hat{\mathbf{x}}_t + \mathbf{K}_t\left(\mathbf{y}_t - \mathbf{H}\hat{\mathbf{x}}_t\right)$$
$$\mathbf{P}_t = \left(\mathbf{I} - \mathbf{K}_t\mathbf{H}\right)\hat{\mathbf{P}}_t$$

where $\mathbf{K}_t$ is the Kalman gain and R is the sensor/observation noise.

(a) Determine $\mathbf{F}$ and $\mathbf{H}$ if we assume a simple constant velocity model for the motion of the ball.

(b) Use the Kalman filtering equations described above to determine $\mathbf{x}_t, \mathbf{P}_t$ for $t = \{2, 3\}$. Assume that the initial state $\mathbf{x}_1 = \begin{bmatrix} 0 \\ 2 \end{bmatrix}$ is given, $\mathbf{R} \approx 0$, and $\mathbf{P}_1 = \mathbf{I}$ (identity).

2 Coding Exercises (60 Marks)

The following exercises will be coding exercises. They are all contained in the colab notebook [link]. The notebook itself is self-contained but you can also use this document as guidance.

Again, please copy the notebook to your own Google Drive to work on instead of modifying the original one. There are hints in the notebook in case you get stuck.

Submission: Submit your code and results in an IPython notebook (.ipynb) with all the **images and video outputs embedded** in it. You will lose marks if your notebook works but doesn't contain outputs.

2.1 Iterative Closest Point with given correspondences (45 Marks)

The question requires several missing functions as follows:

(a) compute mean point of an array of points in function `compute_mean()`;

(b) compute cross covariance matrix **W** in `function compute_W()`;

(c) compute estimated rotation and translation via **SVD** in function `compute_R_t()`;

(d) apply the estimated rotation and translation and find the value of the squared error function to discover how good the estimated positions are in function `compute_error()`;

(e) complete the fuction `icp_known_corresp()`.

2.2 Iterative Closest Point without given correspondences (15 Marks)

You will need to iteratively find the point correspondences and using these perform the ICP updates. Make you algorithm stop after convergence.