Università della Svizzera italiana

Faculty of Informatics

Computer Vision & Pattern Recognition

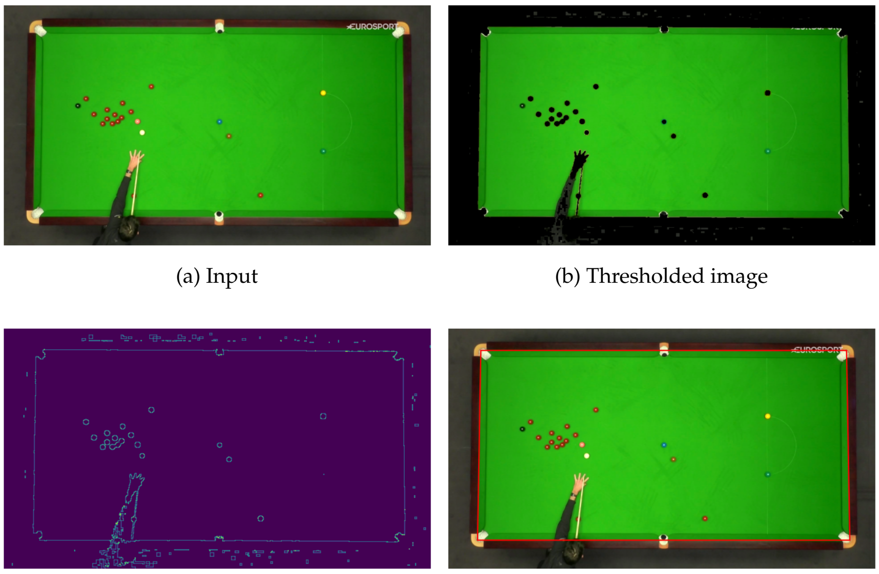
Spring 2022

Assignment 4

March 28, 2022

Exercise 1 [15 points]

Write a program that sequentially reads frames from an overhead video of a snooker match (available on iCorsi). Here, we consider the boundary of the snooker table to be where the green felt meets the wooden paneling. Your program should automatically locate the boundary of the snooker table. Additionally, you are to locate the centres of each of the red balls that are on the table.
A simple approach would be to exploit the table cloth's vibrant green colour for thresholding the image. One can then employ the Hough line transform to locate the lines formed in the thresholded image. It is then possible to compute a closed boundary for the table. Once you have implemented this, the problem of locating the balls should be relatively trivial.



(a) Input (b) Thresholded image

(c) Detected edges (d) Boundary

Figure 1: Exemplar boundary detection pipeline.

To speed up your implementation, it is recommended that you resize each frame using OpenCV's build-in function (`cv2.resize(A,0.5)`) and skip odd frames.

Exercise 2 [3 points]

The Stephen-Harris corner detector relies upon a measurement of the *sum of squared differences* between a given pixel's neighbourhood $A(s, t)$ and the shifted neighbourhood window $A(s + \Delta x, t + \Delta y)$ given after shifting by the quantities $(\Delta x, \Delta y)$ as,

$$C(\Delta x, \Delta y) = \sum_{s=-1}^{1} \sum_{t=-1}^{1} \left[A(s+\Delta x, t+\Delta y) - A(s,t)\right]^2 .$$

N.B. as in the lecture, to simplify the presentation of the formulae, the global image coordinate (x, y) is discarded, and just the relevant window shifts $(\Delta x, \Delta y)$ and window indexing shifts (s, t) are given. If it helps, you can think of this as short-hand for $A(x+s, y+t)$ and $A(x+s+\Delta x, y+t+\Delta y)$, and later on $G_x(x+s, y+t)$ and $G_y(x+s, y+t)$.
In practice, we only need to compute $C(1, 0)$ and $C(0, 1)$. Since Δx and Δy are small, the shifted window may be approximated as,

$$A(s+\Delta x, t+\Delta y) \approx A(s,t) + \Delta x G_x(s,t) + \Delta y G_y(s,t).$$

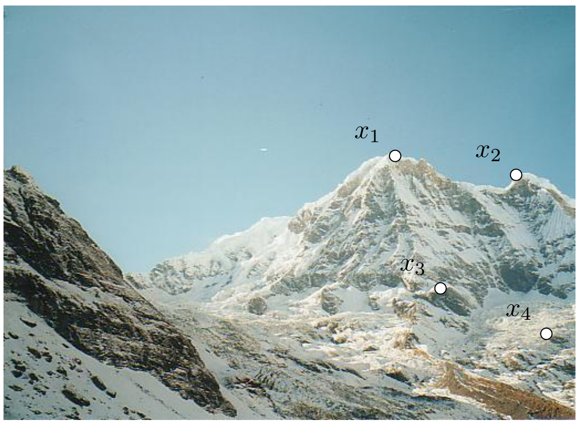
When incorporated into the sum of squared differences equation, we obtain the following:

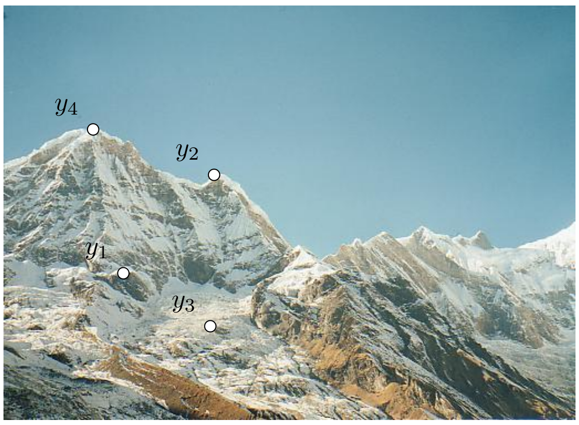
$$\sum_{s=-1}^{1} \sum_{t=-1}^{1} \left[A(s+\Delta x, t+\Delta y) - A(s,t)\right]^2 \approx \begin{bmatrix}\Delta x & \Delta y\end{bmatrix} \left(\sum_{s=-1}^{1} \sum_{t=-1}^{1} \begin{bmatrix} G_x(s,t)^2 & G_x(s,t)G_y(s,t) \\ G_x(s,t)G_y(s,t) & G_y(s,t)^2 \end{bmatrix} \right) \begin{bmatrix}\Delta x \\ \Delta y\end{bmatrix}$$

Show how we arrive at this approximation of the SSD. Please explain and justify why, in practice, this is sufficient for the detector.

Exercise 3 [2 points]

We have developed our own local feature detector/descriptor for establishing correspondences between two images. In the two images below, four interest points were detected (x_1, x_2, x_3, x_4 and y_1, y_2, y_3, y_4); however, we do not yet know which pairs match.





For the set of detected interest points we have used our descriptor to compute the following feature vectors in the first image (where $X_i = \text{Desc}(x_i)$). Similarly, for a second image we have computed ($Y_j = \text{Desc}(y_j)$), which are given in the table below.

X_1	X_2	X_3	X_4
1	6	9	9
7	5	1	7
4	7	1	8
5	5	2	3
7	4	9	4
1	0	8	7
5	7	9	8

Y_1	Y_2	Y_3	Y_4
7	9	6	2
2	1	7	1
2	3	8	3
0	5	0	6
2	3	3	8
4	4	9	8
1	0	0	2

Construct a cost matrix C where $C_{i,j}$ measures the similarity between X_i and Y_j (e.g., $\|X_i - Y_j\|$). Then, apply the following procedures to match pairs of descriptors between the two images.

1. First, for each feature vector in the first image X_i, find the most similar feature vector in the second image Y_j, i.e., the lowest cost entry in C in row i.
2. Treating this as an assignment problem, find the optimal permutation of the feature vectors (e.g., using the Hungarian algorithm).

For convenience the tabulated data is given in a Python array below. Although, you may solve this problem either by hand or using a computer.

```
X = [[1,7,4,5,7,1,5],[6,5,7,5,4,0,7],[9,1,1,2,9,8,9],[9,7,8,3,4,7,8]]
Y = [[7,2,2,0,2,4,1],[9,1,3,5,3,4,0],[6,7,8,0,3,9,0],[2,1,3,6,8,8,2]]
```

Solutions must be returned on April 11, 2022 via iCorsi