TTK4255: Robotic Vision

Homework 2: Feature detection

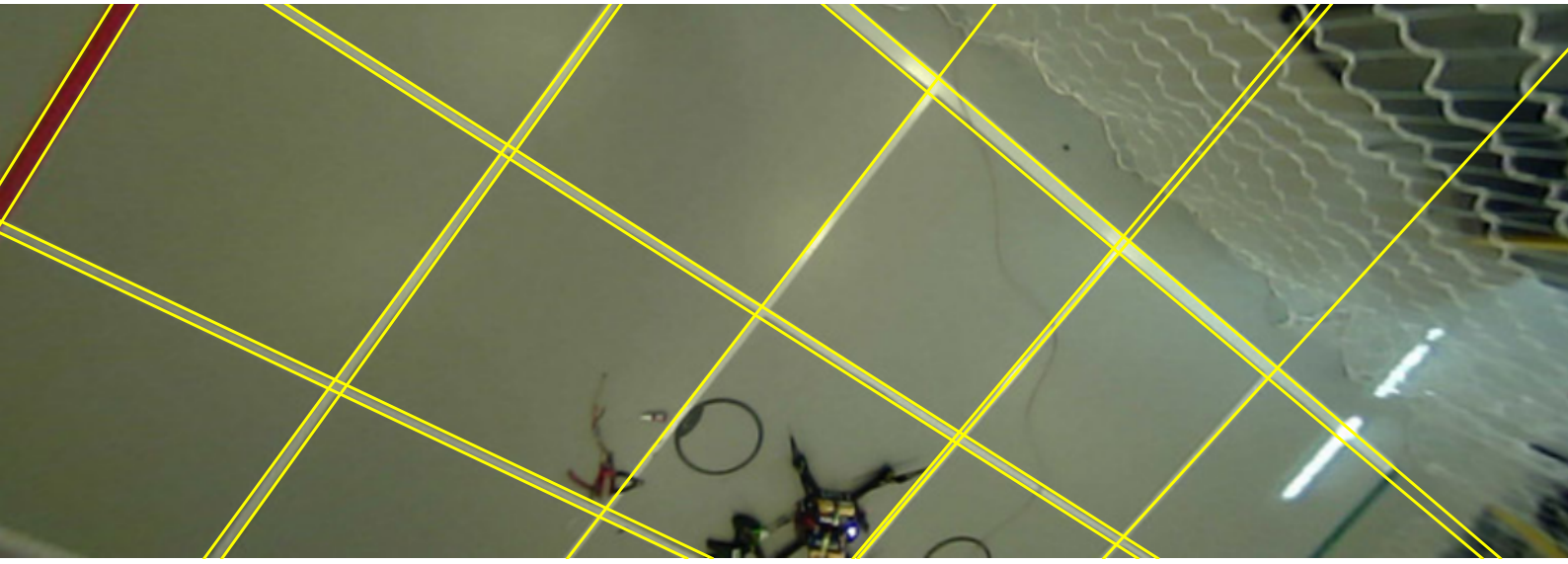

Figure 1: Sample output from a line detection algorithm you will implement.

Instructions

For more information about the course work and how to get help, see the `Course work/about.pdf` document on BB. To get your assignment approved, you need to complete **35%** of the tasks (*21/01: reduced from 60%*). Upload the requested answers and figures as a single PDF. You don't need to submit your code. You may collaborate with other students and submit the same report, but you still need to upload it individually on Blackboard. Please write your collaborators' names on your report.

About the assignment

In this assignment you will learn how to detect some simple image features. Features play an important role in establishing correspondences, which is a core operation in pose estimation and 3D reconstruction. Detecting simple features, such as lines or corners, is also often part of algorithms for detecting more complex objects, such as calibration checkerboards or fiducial markers.

An important class of features are point features. These are specific, well-localized points where the image attains a maximum in some measure of interest. Among the many measures proposed in the literature, a well-known one is the Harris-Stephens measure used in the "Harris corner" detector, which you will implement here.

Unlike point features, which are identified by their local neighborhood, higher-level features tend to require aggregating information from the whole image. The Hough transform (pronounced "huff") is a general technique that performs this aggregation by voting. Here you will use the Hough transform to detect lines, but it can also be used to detect circles, ellipses and other analytical shapes.

Relevant reading

The Harris corner detector and related keypoint detectors is derived in Szeliski 4.1.1. Note that the terms corner, interest point and keypoint are often used interchangeably. To add further confusion, the term corner does not necessarily refer to projections of 3D corners. Instead, it only refers to points with neighborontaining strong evidence of variation along two directions. Such points can be present on various types of structures besides corners, such as highly textured surfaces.

The Hough transform you will implement is described in 4.3.2. Note that the Hough transform described in 4.3.2 modifies the original algorithm (which is presented in Lecture 2) to use edge directions, such that each involved point only votes for a single line.

Feature description, matching and tracking (4.1.2-4.1.4) are topics that we will return to later in the course, and are not required reading for this assignment.

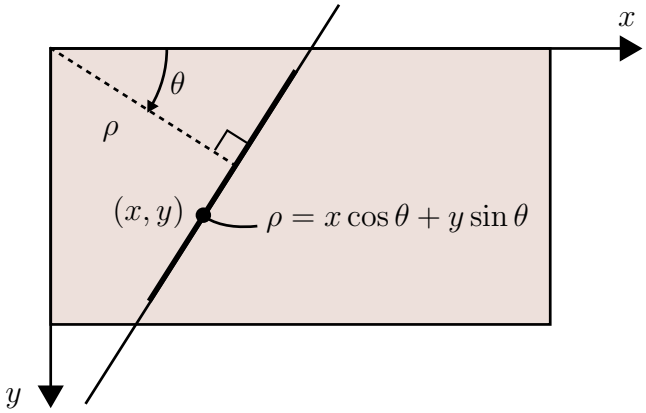

Figure 2: In normal form a line is represented by its angle θ and distance to the origin ρ. Any point (x, y) on the line satisfies the equation $\rho = x\cos\theta + y\sin\theta$. Note that the y-axis here is pointing downward to match the axes convention of images, so positive θ indicates clockwise rotation.

Part 1 Theory questions (30%)

Task 1.1: (6%) Lines can be represented in different ways. In the standard form $y = ax + b$, a line is represented by its slope and intercept (a, b). However, for the Hough transform we prefer to use the normal (polar) form (ρ, θ) where $\rho = x\cos\theta + y\sin\theta$. Why might the standard form be problematic?

Task 1.2: (8%) Consider a square image of size L, with $(x, y) \in [0, L] \times [0, L]$, and let lines be represented in normal form (ρ, θ). Keeping the angle fixed at each value below, what is the range of possible ρ values attainable by a *visible* line (a line that intersects the image) with that angle?

(a) $\theta = 0°$ (b) $\theta = 180°$ (c) $\theta = 45°$ (d) $\theta = -45°$

Task 1.3: (8%) Several keypoint detectors are based on the auto-correlation matrix, which is defined per-pixel as the outer product of the image derivatives I_x, I_y, integrated over a weighted neighborhood:

$$\mathbf{A} = \begin{bmatrix} w * (I_x^2) & w * (I_x I_y) \\ w * (I_x I_y) & w * (I_y^2) \end{bmatrix}, \tag{1}$$

where w is a weighting function. Szeliski 4.1.1 describes several scalar "corner strength" measures based on the auto-correlation matrix. The measure proposed by Harris and Stephens is

$$\lambda_0\lambda_1 - \alpha(\lambda_0 + \lambda_1)^2 = \det(\mathbf{A}) - \alpha\,\mathrm{trace}(\mathbf{A})^2 \tag{2}$$

where λ_0, λ_1 are the eigenvalues of $\mathbf{A}$. Suppose w is radially symmetric and explain why the Harris-Stephens measure is invariant to intensity shifts ($I(\mathbf{x}) \to I(\mathbf{x}) + c$) and image rotation. Suppose w is not radially symmetric (e.g. a box blur kernel), is the measure still invariant to these transformations?

Task 1.4: (8%) The Harris-Stephens measure is not invariant to contrast changes ($I(\mathbf{x}) \to cI(\mathbf{x})$). However, does this affect which points are detected as corners if the acceptance threshold is specified *relatively* as a fraction of the highest occurring corner strength (as in Task 3)?

Part 2 Hough transform (40%)

Here you will implement the line detector in Szeliski 4.3.2 based on the Hough transform. The basic idea is to accumulate votes for the presence of specific lines (ρ, θ) in an accumulator array H. Specifically, an edge with location (x_i, y_i) and orientation θ_i votes for the line (ρ_i, θ_i), where $\rho_i = x_i \cos\theta_i + y_i \sin\theta_i$. The vote is added by incrementing the corresponding cell in the accumulator array. Because arrays can only store values at a finite set of indices, the $\rho\theta$-space must first be quantized into $N_\rho \times N_\theta$ bins, with a pair of ranges $[\rho_{\min}, \rho_{\max}] \times [\theta_{\min}, \theta_{\max}]$ mapping the continuous parameters to indices in H:

$$\text{row} = \text{floor}(N_\rho \cdot \frac{\rho - \rho_{\min}}{\rho_{\max} - \rho_{\min}}) \in [0, N_\rho], \quad \text{column} = \text{floor}(N_\theta \cdot \frac{\theta - \theta_{\min}}{\theta_{\max} - \theta_{\min}}) \in [0, N_\theta]. \tag{3}$$

After accumulating votes from all edges, those elements of H with more votes than their immediate neighbors, and a minimum number of votes, are extracted and interpreted as dominant lines.

The `task2` script provides code to generate the requested figures on a sample image. The basic edge detector from Homework 1 is also provided, which handles the first step of extracting edges.

Task 2.1: (5%) Determine appropriate ranges for θ and ρ based on the input image resolution. Your ranges should ensure that all potentially visible and distinct lines can be represented in the array.

Note: As discussed in Szeliski, using edge directions in the voting gives a line two distinct orientations, depending on the edge direction (whether the image went from dark to bright or vice versa along the line normal). Therefore, the minimal range for θ here is of size 2π. Note also that the ranges provided in the book chapter assume normalized pixel coordinates, but you should use ordinary pixel coordinates.

Task 2.2: (25%) Compute the accumulator array as described above and include a figure showing the array for the sample image. Let the horizontal axis of your array represent the angle θ and the vertical axis represent ρ; otherwise the provided code for the figure will be wrong. For now, use a small resolution ($N_\rho = N_\theta = 200$). You should be able to see several bright spots, corresponding to the parameters of the dominant lines in the sample image.

Task 2.3: (10%) Use the provided function `extract_local_maxima` to extract the dominant lines. The function takes a minimum acceptance threshold, specified as a fraction between 0 and 1 of the maximum array value. Set this to 0.2. Convert the row and column indices back to continuous (ρ, θ) quantities and include figures showing the location of local maxima in the accumulator array and the lines drawn back onto the input image. For better results, you can try to increase the accumulator array resolution and possibly adjust the acceptance threshold.

Task 2.4: **(Optional self-study task - 0%)** The results shown in Fig. 1 were achieved by two further improvements. The first was to implement the optional edge detector improvements described in Homework 1. The second was to modify `extract_local_maxima` to use a larger neighborhood, in order to suppress smaller nearby maxima. Are you able to get similar results?

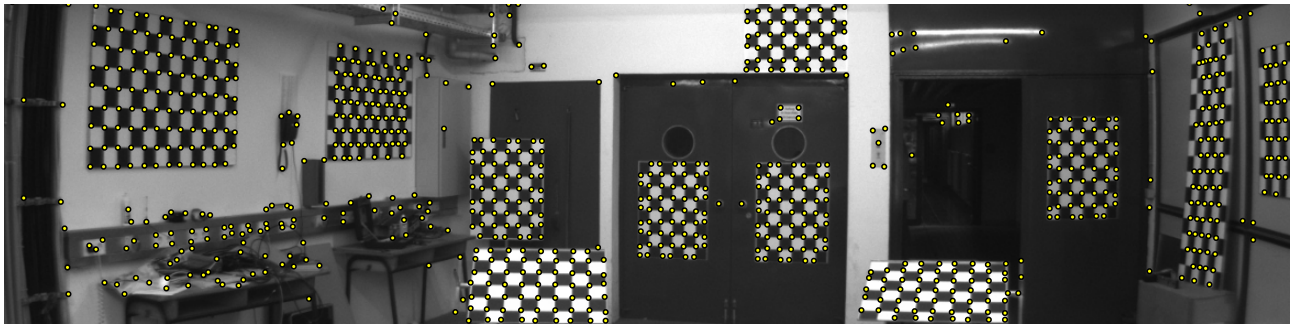

Figure 3: Detected Harris corners in a sample image from (Geiger *et al.*, 2012). The checkerboards are used for camera calibration. Some well-known calibration packages localize the interior vertices of the checkerboard patterns using the Harris corner detector. (Today there are more robust methods).

Part 3 Harris detector (30%)

Here you will implement the Harris corner detector to gain a deeper understanding of its usage and limitations. As described in Szeliski 4.1.1, its basis is the auto-correlation matrix

$$\mathbf{A} = \begin{bmatrix} w * (I_x^2) & w * (I_x I_y) \\ w * (I_x I_y) & w * (I_y^2) \end{bmatrix} \tag{4}$$

where I_x and I_y are the horizontal and vertical image derivatives. Conceptually, the quantities I_x^2, I_y^2, and $I_x I_y$ are formed at each pixel and locally integrated over a neighborhood weighted by w. A corner strength image can then be computed using one of the measures in Szeliski 4.1.1 and analyzed for local maxima. Using the Harris-Stephens measure mentioned earlier gives the well-known Harris detector.

The `task3` script provides code to generate the figures requested in Task 3.1 and 3.4.

Task 3.1: (15%) Compute the Harris-Stephens measure for the sample image `calibration.jpg`. Include a figure showing the resulting corner strength as a grayscale image.

Image derivatives should be computed by convolving with the partial derivatives of a 2-D Gaussian with standard deviation σ_D (the differentiation scale). The weighting function w should be a 2-D Gaussian with standard deviation σ_I (the integration scale). The hand-out code includes a function `derivative_of_gaussian` to compute the image derivatives. Use $\sigma_D = 1, \sigma_I = 3$ and $\alpha = 0.06$.

Task 3.2: (5%) What do negative corner strength values indicate?

Task 3.3: (5%) Why do some of the checkerboards have a weaker response than others?

Task 3.4: (5%) Use the provided function `extract_local_maxima` to extract strong corners. Set the acceptance threshold to 0.001 of the maximum corner strength. Include a figure showing the extracted corners as a scatter-plot over the sample image.

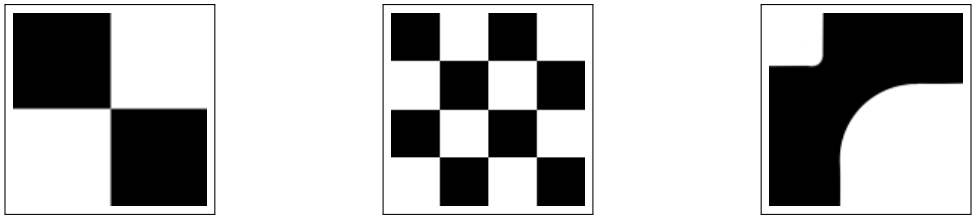

Figure 4: Test images. Left-to-right: `checker1.png`, `checker2.png`, `arcs.png`.

Task 3.5: (Optional self-study task - 0%) The Harris detector is not scale invariant. Thus, given two images taken at different distances from a scene, the detected corners may not be attached to the same world points in both images. When specifying the parameters σ_D and σ_I, we can control what characteristic scale of a corner we are looking for. For example, in the `arcs` image, should the lower-right structure be considered as a corner or a circular edge?

The answer depends on the application and σ_D and σ_I must be chosen appropriately. The differentiation scale σ_D is used to stabilize gradients in the presence of image noise, and to simulate the smoothing effects of capturing the image at a greater distance. The integration scale σ_I is chosen larger than σ_D, and is used to control the characteristic scale.

Try your Harris detector on the three test images shown above (included in the zip), using different values for σ_D and σ_I, and try to answer the following questions:

(a) Keeping σ_D fixed (e.g. equal to 1) and increasing σ_I (e.g. from 1 to 100), do the detected corners converge to any particular point?

(b) Are some types of corners more stable in scale than others?