CS6643 Computer Vision S2018 – Project 2 Image Transformations

Assigned April 3, 2018

Due Tue April 17, 2018 (Just before midnight)

Instructor: Guido Gerig

Goals

The purpose of this assignment is to learn about linear and nonlinear image transformations (warping), and also about detection of linear structures from images. Please read following instructions carefully.

1 Problem 1: Affine Image Transformation

1.1 Resampling

Implement a program that transforms an image given an affine transformation (6 parameters), which includes rotation, translation, scaling and shear. Please note that the transformation is generally applied from the target image backwards to the source image, i.e. you step through each pixel of the new image, determine the position in the source image, and take/calculate the intensity at this pixel to be used for the target image. Two types of interpolations need to be implemented:

1. Nearest neighbor (NN): Take the intensity from the pixel which is closest to the non-grid position of the transformation. Please note that this can be easiest achieved by rounding a non-grid (x,y)-ccordinate to the next integer coordinates.
2. Implement bilinear interpolation from the 4 neighbors of the non-grid coordinate, following the discussion in the course.

Take an input image of your choice and apply via backward transform (target to source):

1. Separate translation, rotation, scaling, and shear transformations, and apply NN and bilinear interpolation.*
2. An affine transformation with 6 parameters (you can cascade some transformations in the previous item).
3. For the affine transformation, demonstrate the differences between nn and bilinear, best by large zooming of an image subregion). Provide a short description on what you did and what you see.

*Hint: By transforming an image, the transformation may move parts outside of the original image size. It is suggested to keep the target image the same size, and to paint pixels that have no partner pixel in the source image as black (i.e. start with an empty target image).

1.2 Calculation of affine transform from landmarks

As discussed in the course, a transformation can be determined based on a set of corresponding landmarks in a source and a target image. A minimum of 3 points with (x,y) coordinates is required, but more landmarks result in a more stable solution by solving an overconstrained linear equation system. Please note that you can use Matlab help and Mathworks web-based help for hints on solutions.

- Use or implement a module that gives pixel positions by clicking locations with the mouse.

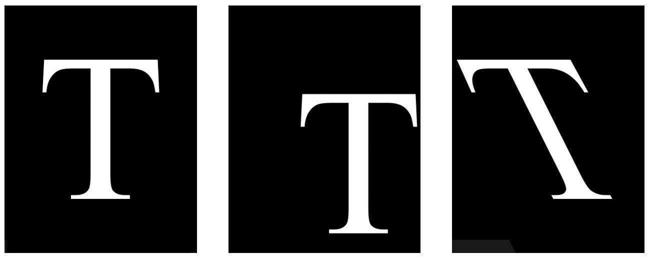

Figure 1: Example of image transforms (original, translation, shear).

- Use this module to create sets of corresponding pixel positions in a source and a target image.
- Set up the linear equation system and implement a solution to solve for the affine transformation between the source and target image(*).
- Apply the transformation and check for the correctness of the result by displaying source, target and resulting images side by side. You can even create another result by blending the result and source together (e.g. adding the images) to have some visual check of geometry differences.

As test images, you can use own pictures (e.g. of frontal view of faces of humans or animals or whatever you like). Would such pictures not be available, you can search the web (e.g. http://www.face-rec.org/databases/). You can be creative about the choice of images, applications as shown in the course slides are for example lip reading in video sequences or normative atlas building in medicine.

(*) see additional materials in regard to solving a linear equation system and Matlab help for solving overconstrained systems. E.g, for solving the system *A.x=b* in Matlab, there are the choices of *x=A/b (fastest), or x=pinv(A)*b, or x=inv(A'*A)*A'*b).*

Problem 2: Hough Transform for straight lines without edge orientation

Write a program that calculates the Hough Transform for straight lines using the parametric form $\rho = x\cos\theta + y\sin\theta$*. With (x,y) being the image coordinates, the origin for the HT is put into the left corner of the image.*

Each point in image space generates a cos*-curve which is accumulated in parameter space. The increment can be chosen as* 1 *for each point, but could also be proportional to the edge-strength which is the brightness attribute for each contour point (see contour images described below), so that the accumulation reflects the* importance of edge points*.*

You may use the test images provided with this project, the simulated scene containing edges and lines "edges-lines" and the airport scene, but feel free to choose images of your choice that contain straight edges to be detected.

a) Images need to be preprocessed with edge detection and thresholding to get a small set of edge candidate pixels to be processed. Use your edge-detection method as implemented in the previous project.

b) Implement the Hough-Transform and create an accumulator that corresponds to a 2-D image array with axes θ *and* ρ*. Choose the cell size* $\Delta\rho$ *and* $\Delta\theta$ *as a parameter in your program for doing your own tests. Defaults could be 1 deg and 1 pixel steps, but you may also choose a coarser accumulator grid of speed is an issue and if you can sacrifice precision.*

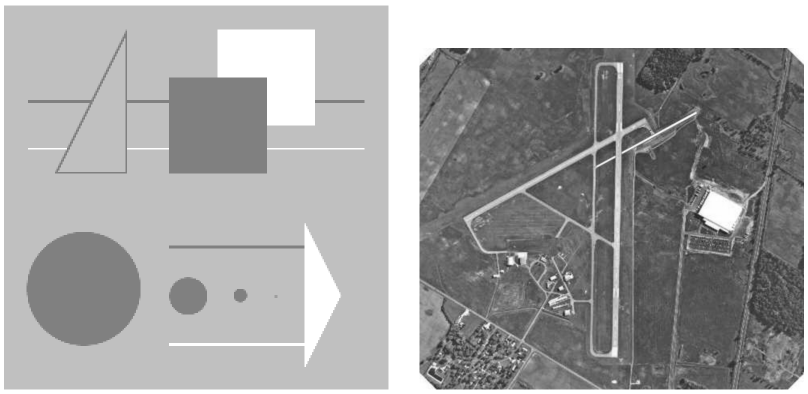

Figure 2: Example of images presenting straight line features.

c) *After calculating the HT for your image points and accumulation of curves, detect the N largest maxima and write the pairs* $(\rho_j, \theta_j, nvotes_j)$ *to a file for checking (nvotes are the accumulated votes per maxima).*

d) *Calculate the corresponding N straight lines and draw these lines back into the image space for comparison with the original image. Nice would be to overlay the lines in color onto the original gray-level image. Examples of overlays in Matlab can be found in the Matlab instruction files from the first week.*

e *Hint on peak detection: There may be several peaks nearby due to discretization of the accumulator. You may implement one of the following options:*

   – *Find maximum peaks as local maxima in the accumulator (i.e. cells that have larger votes than all neighboring cells).*
   – *Gaussian smoothing of the accumulator with a small kernel to reduce noise. Then apply the procedure above for finding peaks. Select the Gaussian width parameter based on the notion that the "pixels" are* (ρ_j, θ_j) *parameter cells.*

- *Display the original image, the accumulator space (please make sure that you enhance the image so that it is not all black, just use some image editing program), and the resulting lines. You can overlay the selected peaks to the accumulator, and the corresponding lines to the original image.*
- *Apply the HT to two images you select from the test images or other image libraries.*

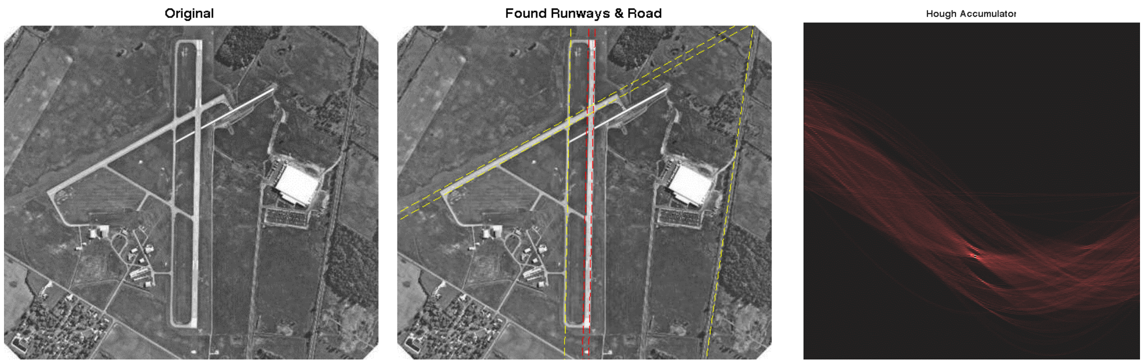

Figure 3: Result on running HT on the runway image.

Bonus Question: Hough Transform for straight lines with edge orientation

As discussed in the course, the Hough transform can be using the edge orientation in addition to the position of edge points. Repeat the procedure of the previous question but now include the local edge orientation, estimated via edge detection via x- and y-derivatives after smoothing (see course notes and slides on edge detection and Canny filtering), for incrementing the Hough space.

- *Implement the Hough transform for line detecting using edge point location and additionally edge orientation.*
- *Apply the procedure to the same images as used before.*
- *Compare results from this modified procedure to the results previously obtained. Discuss eventual improvements, gain in computing time, or new challenges related to peak detection, for example.*

2 Instructions, Requirements, and Restrictions

1. *Please use your name* **"NAME"** *in all reports and submitted materials to uniquely identify your projects.*
2. *Your report should summarize your approach to solve the problems, show graphs, images of the results and include a critical evaluation/discussion of the results. Please do not copy the project description into your report, just the title is sufficient, but provide a short description what you did for each experiment and how you did it. The report can be written with any text program of choice, and the handin needs to be in pdf format.*
3. *You should have in your report a short description of each algorithm you used and documentation on how your code is organized. A short summary on most essential core information is sufficient, don't expand too much.*
4. *We* **do not allow to use ready-made toolbox functions** *(e.g. Imaging Toolbox) or other existing image processing libraries or solutions found elsewhere. We want you to experience challenges of implementation and give all students the same conditions for code development.*
5. *Your project report will be in the form of a pdf file, which together with a zipped archive of code and all necessary image files is submitted to NYUclasses.*
6. **Please remember to look-up the honor code and requirement to provide your own solution as discussed in the syllabus.**
7. **Please look up the late policy as defined in the syllabus**