CS 6643 - Computer Vision, Fall 2023 **Guido Gerig**

Project 3: Snake or Hough Transform

Out: Mon Nov. 13, 2023
Due: Thu Nov. 30, 2023 (deadline: 11:55PM)

Late submissions: Late submissions result in 10% deduction for each day. The assignment will no longer be accepted 3 days after the deadline.
Grading: 25% for implementations and 75% for experiments and written report including graphs, results, and critical discussion.

Office hours:

		Mon	Tue	Wed	Thur	Fri
Guido Gerig	gerig@nyu.edu				2-3pm ZOOM	
Pragnavi Ravuluri Sai	pr2370@nyu.edu	4-6pm				
Sai Rajeev Koppuravuri	rk4305@nyu.edu			4-5pm		

Please remember that we also use campuswire for communication on homeworks.

Please read the instructions carefully. Note: we will not be running code. Rather, we will check your code to make sure your implementation is your own, and it matches your results. Your grade is primarily based on your written report. This means going beyond just showing results, but also describing them. You should produce a standalone lab report, describing results in enough detail for someone else (outside of class) to follow. Please read every page in this assignment. **Please submit a single PDF/HTML with all code included as an appendix.**

Important: You can either choose **Problem 1 (Snake)** or **Problem 2 (Hough Transform)** for your solution. In the creative part Problem 3, you may then augment your Problem 1 or Problem 2 with some new extension and/or applications.

Project 3: Snake or Hough Transform

Problem 1: Snake Implementation

As a guideline, study the greedy snake implementation by Trucco&Verri which is uploaded as a separate pdf file (snake-Trucco-Verri.pdf).

This project requires several components:

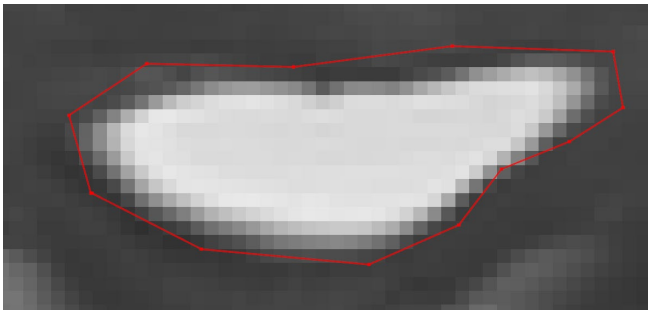
1. Computing a gradient magnitude image from your original image, a method which you have implemented in project 2.
2. A method to pick points in an image to create an initial polygon.
3. Overlay of the polygon on the original image and/or the gradient magnitude external force image.
4. **A core snake algorithm** to calculate snake propagation based on internal and external forces.
5. Same as in 3, overlay an updated polygon over the original image to mimic a dynamic propagation.

Whereas the core algorithm (4) has to be implemented by yourself, the other components to use cursor clicking, drawing of polygon lines, and overlay of the evolving snake can make use of existing library functions.

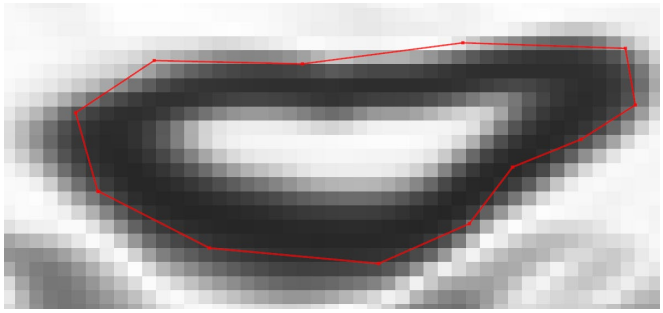
The following procedure is suggested:

- Choose a small image and simple object to develop and test the procedure.
- Start developments with a polygon with limited number of points.
- Consider a reasonable choice for the sigma of the Gaussian filter of the gradient magnitude external force. Would sigma be too small, a snake may not “feel” the external force to move towards the boundary, respectively the locations of the lowest external forces.
- The intensities of the gradient magnitude image need to be inverted to create dark minimum intensities at edges and larger intensities outside. It is suggested to change the [Min-Max] of the gradient magnitude to a range of [Max-Min] to stay in the positive intensity range.
- Reconsider the internal stretching force suggested by Trucco&Verri to keep the total length constant – this may impede the snake to contract. You may allow for some step-by-step shrinking.
- The parameters to combine the two internal with the external force are heuristic choices. It is suggested to write those forces into a table after each iteration, so that you may get a feeling on the magnitude of these energies accumulated along the contour. Also list the total energy after this cycle to come up with a decision when to stop.
- Consider two propagation strategies:
    - Calculate new pixel location via minimum energy calculation in a local neighborhood, move point to new location, and circle to the next point. Here, you may iteratively circle around each point until you reach a minimum energy.
    - Calculate a new location around each polygon point as before, but only move all polygon points together after each cycle.
    - It is not clear to me which strategy may be better or “more right”.

- Once all is working well and debugged, you may choose larger images with more polygon points.



Initial snake overlaid over the original image.



Initial snake overlaid over the inverted GradMag image

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.

You may use the test images provided with this project, the simulated scene containing edges and lines "edges-lines", the airport scene or the crosswalk, 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 (gradient magnitude) 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. IMPORTANT: Be sure that you only get a small number of edge pixels to save computation time.
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 if speed is an issue and if you can sacrifice precision.
c) After calculating the HT for your image points and accumulation of curves, detect the N largest maxima and write the pairs (ρ_j, θ_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. Here, you can make use of an existing library function.
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:
    a. Find maximum peaks as local maxima in the accumulator (i.e. cells that have larger votes than all neighboring cells).
    b. 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 (ρj, θj) parameter cells.
f) 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 or histogram stretching program), and the resulting lines. You can overlay the selected peaks to the accumulator, and the corresponding lines to the original image.
g) Apply the HT to two images you select from the test images or other image libraries.

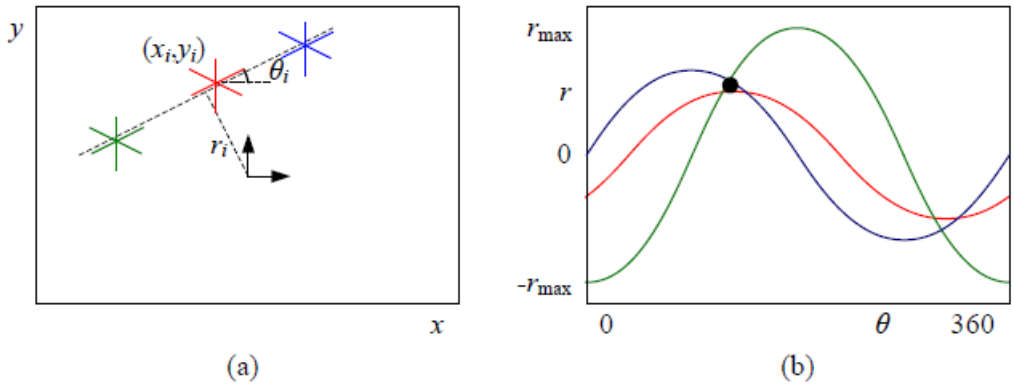
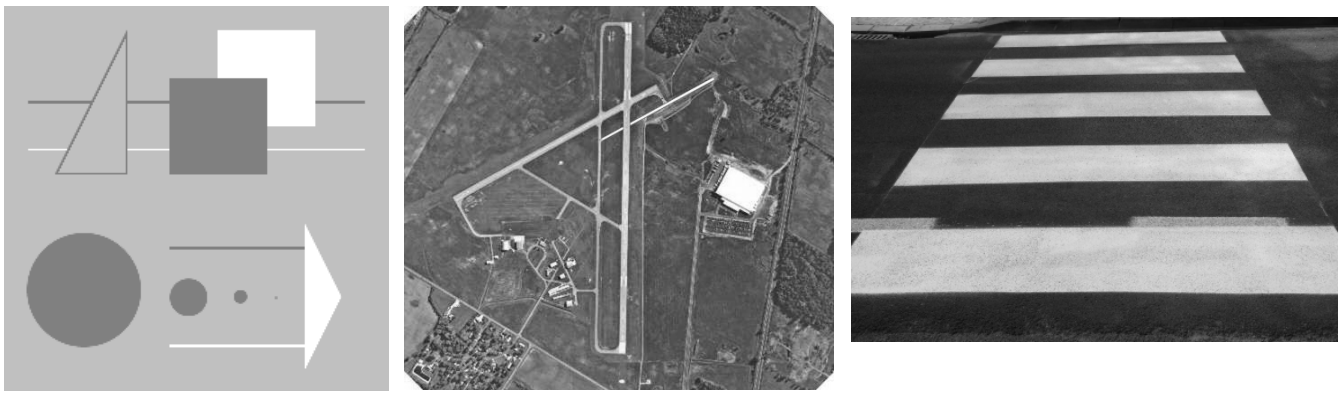


Figure 7.46 *Original Hough transform: (a) each point votes for a complete family of potential lines* $r_i(\theta) = x_i \cos\theta + y_i \sin\theta$*; (b) each pencil of lines sweeps out a sinusoid in* (r, θ)*; their intersection provides the desired line equation.*



Figures: Example of images presenting straight line features

3. Creative Part:

This is completely on your own to extend either the Snake or Hough Transformation based on your ideas. Possibilities could be:

Snakes:

- Use the gradient direction information to speed up the propagation. Remember that in addition to gradient magnitude, you can also calculate the gradient direction using the x- and y-derivatives. Consider this information to drive the snake "towards" the edge and to avoid a kind of blind search in each local neighborhood.
- You may apply your snake algorithm to a short video sequence, such as lip syncing or something else. Here, the minimum energy solution would be used as a new initialization on a consecutive frame and thus propagated through the sequence. You may choose very small images for testing due to long computation times.
- Others ….

Hough Transform:

- One possibility would be to augment/simplify your Hough Transform by using the edge orientation in addition to the position of edge points. This means you may now include the local edge orientation, estimated via edge detection and the x- and y-derivatives (see course notes and slides on edge detection and Canny filtering), for incrementing the Hough space. One problem you will encounter is the limited precision of the edge direction angle information. A possibility discussed in the literature would be to increment the accumulator by a small region rather than a single point. This means that you would calculate a (ρ_j, θ_j) location in the accumulator for incrementation, but would also increment its 3x3 or 5x5 neighborhood. See figure from course slides below.
- You may implement HT for circles of a specific size following discussions in the course notes.
- Others…

Project 3: Snake or Hough Transform

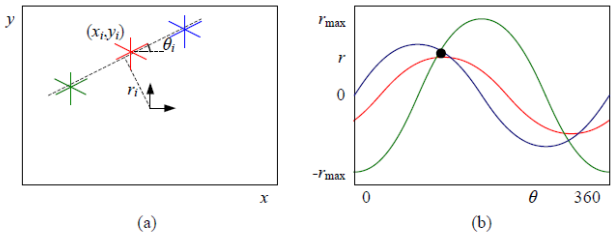


Figure 7.46 *Original Hough transform: (a) each point votes for a complete family of potential lines $r_i(\theta) = x_i \cos\theta + y_i \sin\theta$; (b) each pencil of lines sweeps out a sinusoid in (r, θ); their intersection provides the desired line equation.*

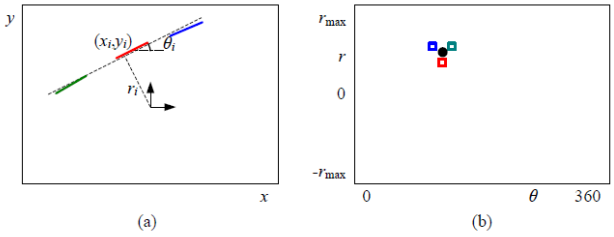


Figure 7.47 *Oriented Hough transform: (a) an edgel re-parameterized in polar (r, θ) coordinates, with $\hat{\mathbf{n}}_i = (\cos\theta_i, \sin\theta_i)$ and $r_i = \hat{\mathbf{n}}_i \cdot \mathbf{x}_i$; (b) (r, θ) accumulator array, showing the votes for the three edgels marked in red, green, and blue.*

Instructions, Requirements, and Restrictions

a) Your report should summarize your approach to solving 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 of what you did for each experiment and how you did it.
b) 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 of the most essential core information is sufficient, don't expand too much.
c) We do not allow to use ready-made toolbox functions or existing image processing libraries or solutions found elsewhere for implementation of core algorithms. We want you to experience the challenges of implementation and give all students the same conditions for code development.
d) 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 Brightspace.