CS6640 – Final Projects & Report
Assigned Nov. 17, 2010
Due Mon Dec. 13 (Just before midnight)
Instructor: Guido Gerig
TA: Miaomiao Zhang

Projects

The goal of the final project is not only to implement a basic algorithm as in the previous projects but to build extensions based on materials published in papers, to perform tests and validation and provide an indepth discussion of your own implementation, your solution strategy, and your results. Each project requires reading paper(s), come up with your own solution, and being creative with experiments to demonstrate strengths of the method but also remaining shortcomings and limitations.
You also need to write a report that summarizes your solution, tests, and provides a critical discussion of results.

You can choose among 3 different projects:

1. Hough Transform
2. Image Mosaicing
3. Active Contours and Snakes

Each question includes an "extra task", where you can spend additional development effort to refine the algorithm. The "extra task" will decide about a grading of "good" versus "excellent". A strong requirement is that **you have to work fully on your own** as this is **your final project**. Further, you need to implement your **own code**, it is not allowed to find or copy code from elsewhere – this would result in a fail of this final project.

1 Hough Transform

Required Readings: Generalizing the Hough Transform ..., D.H. Ballard 1981
Linking Image Space and Accumulator-Space, G. Gerig, ICCV'87
Notes to parameter space reduction via projection (from Gerig 1986)

1.1 Detection and localization of straight lines

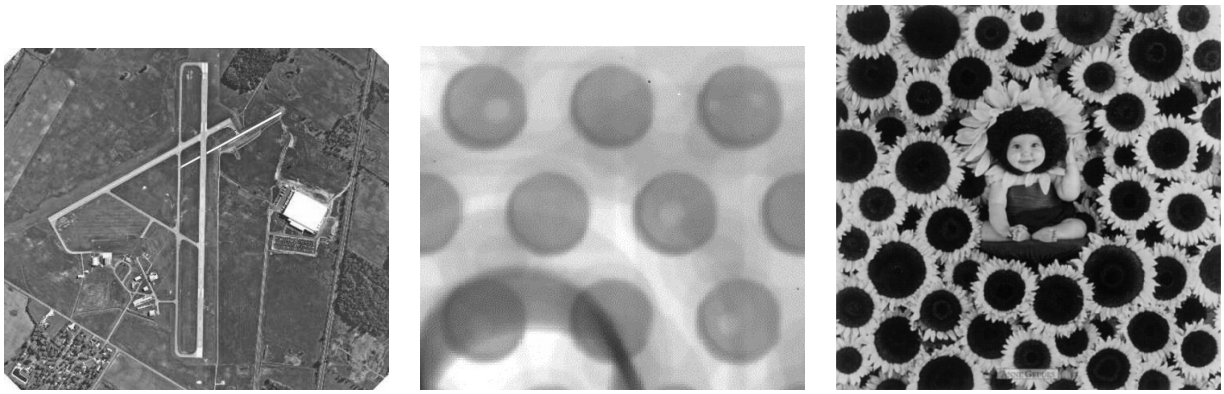

Figure 1: Example of airport scene containing straight lines and of x-ray image of surface-mount packaging (this is a quality control problem where chip manufacturers are trying to determine how well the contacts align for each mounted component). The right image represents circular structures of varying radii.

1. From Project 4: Apply edge or line filtering followed by nonmaximum suppression and thresholding to extract the set of strong edge pixels.
2. From Project 5: Implement the HT to detect straight lines (course notes, papers and discussions). Put the center into the middle of the image to allow for a more uniform parametrization.
3. After incrementing the accumulator space, implement a strategy to reliably find peaks. It is recommended to first slightly smooth the accumulator and then apply the "decrementation strategy" as described in the Gerig-ICCV87 paper and course notes.
4. The remaining peaks, after thresholding for a minimum number of votes (equals length of line) form the set of major lines. These lines can be overlayed on the original image.
5. The Gerig-ICCV87 paper outlines a strategy to additionally characterize the location and position of a line by linking the set of image points to each remaining maximum in the accumulator space. Implement such a strategy to detect the spatial extensions of line pieces, i.e. to calculate the set of contiguous straight line pieces that match the image structures.
6. Do all the steps with an accumulator built by not only using point location information (x,y) but additionally the information of the normal to the edge (gradient direction). Compare solutation with and without the use of edge orientation.

1.2 Detection of circular structures

1. Design an HT algorithm to detect circular contours of a fixed radius. Apply it to the X-ray manufacturing image by estimating the sought radius.

2. Similarly to line detection, you need to find a strategy to efficiently detect the "best" set of maxima. Again here, you might use the decrementation strategy as described in Gerig-ICCV87 to minimize false maxima.

Extra Task: GHT

*Hint: When detecting shapes at different scales and orientation, you might need a higher dimensional accumulator (e.g. 4-dim for (x,y,scale,rotation). In Gerig-ICCV87, there is a strategy that projects the best maxima to the 2-D x-y plane by maximum projection, but keeps information about the associated scale and rotation as additional attributes in a second buffer. This avoids a search for maxima in a high-dimensional accumulator.

1. Implement the GHT method (see Ballard paper) to detect arbitrarily shaped 2D contours. Be creative to choose a picture with repetitive patterns which can vary in size and orientation but have the same shape.

2. Given a specific shape to be detected in images, convert the shape contour into a list of vectors from a reference center to the boundary, with each vector attributed by the angle of the contour normal.

3. Create an R-table.

4. Remember that fixed radius circle detection above can be extended to search for circles with arbitrary radii using the GHT (i.e. in the GHT you make use of the edge gradient). This would include only a scale but not a rotation parameter. Your GHT method could be tested with circles first and then with shapes that vary in scale and rotations.

5. Detect shapes with different rotations and scale using the R-table method of Ballard.

2 Mosaicing

Mosaicing is used to generate large pictures from sets of images taken in different spatial directions. The basic algorithm for two images has been discussed as part of the course lectures on geometric transformations.

Required Readings: Course notes Geometric Transformations
Materials: Slides CSCE 641 Computer Graphics: Image Mosaicing, Jinxiang Chai
Some hints on intensity-based mosaicing (pdf document)

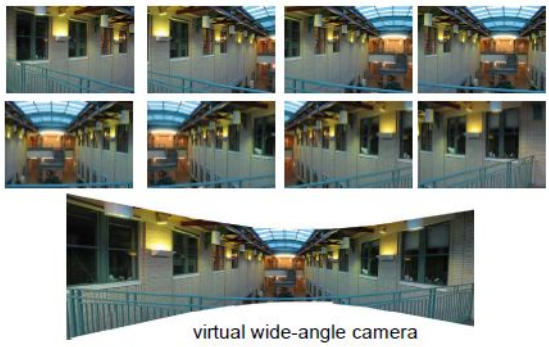

Figure 2: Example of for mosaicing a set of images into a virtual wide angle view (Jinxiang Chai, TAMU).

1. From Project 3: Needs method for applying a transformation to an image with bilinear interpolation.
2. Take a set of pictures from the same viewpoint by pointing your camera in different directions, but ensure that there is sufficient overlap between consecutive images.
3. As discussed in the course (see slides), a minimal number of 4 points define the 8 unknown to calculate the transformation between pairs of images.
4. Given a series of images, remember that you need to cascade the transformations (e.g. given 5 images with the middle the reference, the neighboring image can be transformed directly, whereas the outermost images have to go through a transformation relative to its neighbors and then additionally through the neighbor's transformation).
5. Experiment with different numbers of control points etc., and report your results, with pictures.

2.1 Questions (to be answered in your report):

- How many control points does it take to get a "good" transformation between images?
- How does the algorithm behave at the theoretical minimum of the number of control points?
- From your experiments how does the accuracy of the control points affect the results?

2.2 Details

To the extent that your algorithm handles these details in a good way, it will improve your grade.

Contrast: A good algorithm should automatically adjust for major intensity differences.

Feathering: A good algorithm should alleviate sharp transitions between images.

Image size: A good algorithm should should produce an output image that is a good size and shape to display all of the images.

Cascading: A typical set of images and control points might not include connections directly back to the target output image. Hopefully you can always get from any given image to the target by a series of projective transformations. This is a graph traversal problem (not a hard one). A good algorithm would automatically find a transformation back to the target (if one exists).

2.3 Hints

Inverse Transforms: Remember that the projective transformation need for image mosaicing is described by a 3x3 matrix A. The inverse of this matrix describes the inverse of the associated projective transformation.

Cascading: The net result of cascading two consecutive projective transformations is described by multiplying their associated matrices (in homogeneous coordinates), i.e. AB.

Sizing Outputs: In order to figure out the size and extent needed for the output, you can look at where the corners of an image map. The image maps to a quadrilateral defined by these four points.

Extra Task: Towards automatic mosaicing

- Cameras and imaging packages offer automatic alignment of images. You might design a strategy towards this goal based on some work as proposed below:

    1. The slides of J. Chai towards the end discuss an automatic stitching by recognizing sets of similar features, using RANSAC for a radom selection of testing possible pairings of features, and then choose the set that produces the best match in the overlapping region (e.g. measured by sum of squares differences SSD between overlapping pixel values).
    2. The website http://www.umiacs.umd.edu/ hismail/Mosaic/node2.html#SECTION00020000000000000000 describes an optimization strategy to vary the 8 parameters until the best overlap is found (using sum of squares differences).

- The first solution selects possible landmark pairings among a small set of landmarks, calculates the transformation matrix, calculates the SSD in the overlapping region, and then chooses the best combination. This requires the extract of a very small set of potential candiate points to do pairing, e.g. Sift features (e.g. http://www.vlfeat.org/~vedaldi/code/sift.html or http://www.cs.ubc.ca/~lowe/keypoints/) or somewhat simpler the Harris Corner Detector (search the internet).

- The second requires the solution of a nonlinear minimization problem as described in the additional document.

3 Active Contours and Snakes

The name "snakes" is used for a methodology that uses deformable models for object segmentation. A user-defined model curve is deformed to match the desired contour, using internal forces (deformable properties of model) and external attraction forces from image structures to guide the initial contour close to the object boundary.

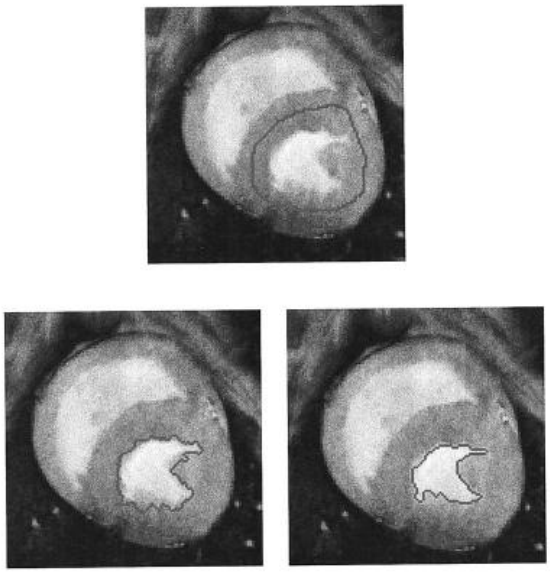

Fig. 3. Contour extraction from MRI heart image via snake.

Figure 3: Example of object segmentation by a snake (from Yezzi, TMI).

Required Readings: Greedy Implementation of Deformable Contour Segmentation (Trucco and Verri)
Snakes: Active Contour models: Kass, Witkin and Terzopoulos, 1988

1. From Projects 2 and 4: Needs method to calculate gradient magnitude image with Gaussian blurring.
2. From Project 3: Needs method to specify series of points or a contour (e.g. circle) into an image for initialization.
3. Implement a procedure to initialize a contour by clicking at a sequence of points close to the object to be segmented, the contour is then defined as the descrete set of consecutive points (analogue to the shape vector used in ASMs).
4. Implement a procedure to define the image force term. The most common procedure is an edge detection which consists of a Gaussian smoothing followed by the calculation of the local gradient and gradient magnitude. The gradient magnitude image represents locations of strong boundaries and its negative value can be directly used as a energy potential that serves as an edge attraction term. Would you use line images (e.g. line drawings), the image intensity itself can be used as the image potential, preferably after Gaussian smoothing.

5. Implement a "snake" algorithm following the Trucco and Verri instructions with a greedy optimization. Design a stopping criterion for the iterative process (e.g., stop when the number of points to be moved would be below a certain threshold). (Please note that the snake might oscillate between two states of similar energy, so that a second criterion based on maximum number of iteration steps might be helpful too).

6. Systematic testing of the algorithm, its parameters, and the influence of the initialization of the contour to the final result on a set of images of your choice:

   (a) Choose a few test images with objects of your choice. Since we use the gradient magnitude as the edge attraction term, objects should preferably differentiate from the background by local intensity contrast. You might also choose a challenging picture showing a contour illusion (similar to the Kass, Witkin and Terzopoulos paper), e.g. from google images with search for "contour illusion".

   (b) Choose different initializations of your starting contour, also with different number of control points, and observe the evolution of the deformable contour. Interesting questions are, for example, if the snake can follow a protrusion or intrusion (concavities) of a contour, etc.

   (c) Choose different image force fields by choosing different Gaussian smoothing as part of the image force term. How does the increased capture range for the image force term help to successfully guide the snake into the optimal solution?

   (d) Systematically explore the parameters for continuity and curvature to explain its effects on the resulting contour.

   (e) Note that the snake does not need to be a closed contour but can be an open curve, deforming under image forces. You might give it a try.

7. Best is to use a few snapshots of the snake evolution process to demonstrate the dynamic evolution of the deformable contour. You do not need to include ALL your experiments into the report, it is sufficient if you illustrate the major results of the various experiments.

Extra Task: Additional Exploratory Analysis

This section describes various ways to further explore "snakes". It is not expected that you would do all but just to give you hints on additional exploration give a successful implementation.

- Could we eventually implement a coarse-to-fine concept by capturing the snake with a coarse-scale image force field but as the snake gets closer to the boundary, decrease the image smoothing to attract the snake to a more detailed boundary? Please note that there are two locations where smoothness is defined, first in the gradient-magnitude calculation at a specific scale and second in the curvature energy term. You might do an experimental design to answer this question and to test feasibility.

- The original "Snakes" paper by Kass, Witkin and Terzopoulos introduces springs and volcanoes to attract and repell snakes based on user interaction. Try to implement such additional forces that allow a user to interact with the snake evolution.