CSCE 485
Bruno Lopes
LAB4 - Continuous Transformation for Scene Understanding

1)

The famous ground breaking paper from Duda and Hart picks up the research done by PVC Hough and apply Hough computationally efficient parametric algorithm for line detection in pictures. One of the machine vision issues at the time was to detect straight lines on digital images, Hough research using slope and intercept for 2D planes lead to the creation of the Hough Transform. The idea was simple, to find line relations among points detected on a (theta, r) plane, go thru each point and seek for significant sinusoidal line relations among its neighbors. This process of grouping points follows a criteria could be determine by a set threshold (>=35 used on sample) and the computational result is a linear value between number of points and values of theta.

The paper goes over two examples, demonstrating how significant lines were missing because there were not enough detected points (too few) and also how the theta was chosen to be higher than 20 degrees. A smaller theta (or finer angular quantization) would increase line relations per points found and give a better result but it would also significantly increase computation time (more scans per point). The ideal form for an accurate Hough transform is done by having bounded parameters for the parametric lines. This basic transform was proven to be one of the most computationally efficient for basic image analysis, based on any mathematical parametric shape its possible to measure and calculate any set of lines. This paper was of great contribution to computer vision and image analysis, even with the basic linear Hough transform we are also able to detect edges and represent 3D space on a 2D plane, a good example would be the depth from the 3D plane where the sample box was located, the lines found due to the transform relate object and space by the perspective on draw box and ground plane representation.

2)

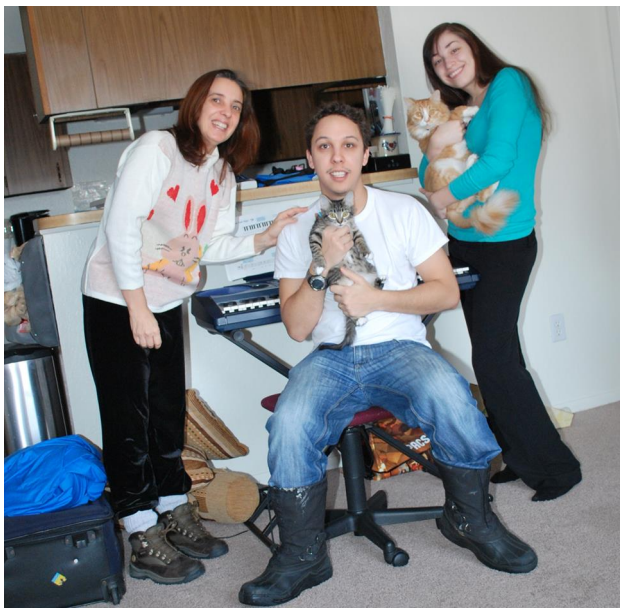
Building the hough_line example:

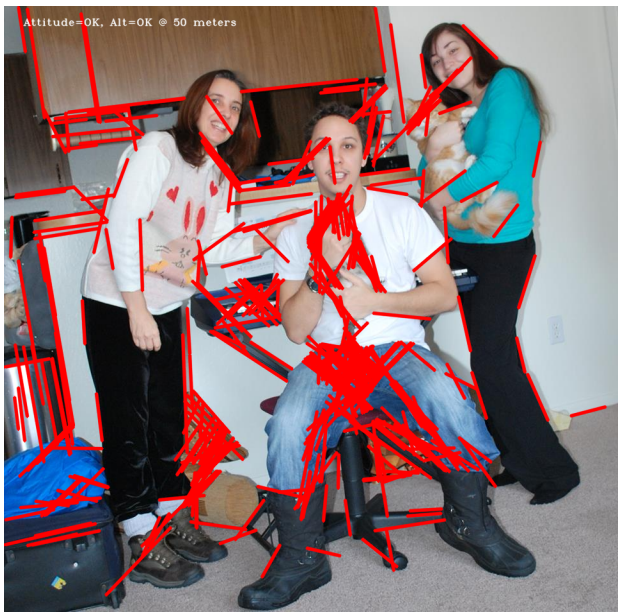
```
root@b:/home/b/Desktop/CSCE485/LABS/LAB4/2# make
g++ -O0 -g   -c captureskel.cpp
captureskel.cpp: In function ‘int main(int, char**)’:
captureskel.cpp:994:18: warning: deprecated conversion from string constant to ‘char*’ [-W
write-strings]
         dev_name = "/dev/video0";
                    ^
g++  -O0 -g   -o captureskel captureskel.o `pkg-config --libs opencv` -L/usr/lib -lopencv_
core -lopencv_flann -lopencv_video
g++ -O0 -g   -c capture.cpp
capture.cpp: In function ‘int main(int, char**)’:
capture.cpp:1000:18: warning: deprecated conversion from string constant to ‘char*’ [-Wwri
te-strings]
         dev_name = "/dev/video0";
```

Entering default picture for submission sample:

```
root@b:/home/b/Desktop/CSCE485/LABS/LAB4/2# ./
canny          captureskel   hough_line    sobel
capture        hough_circle  skeletal
root@b:/home/b/Desktop/CSCE485/LABS/LAB4/2# ./hough_line 1.jpg
init done
opengl support available
^C
root@b:/home/b/Desktop/CSCE485/LABS/LAB4/2#
```

Image before/after hough line transformation:





Note all many straight lines found on my striped tabby cat!

5)

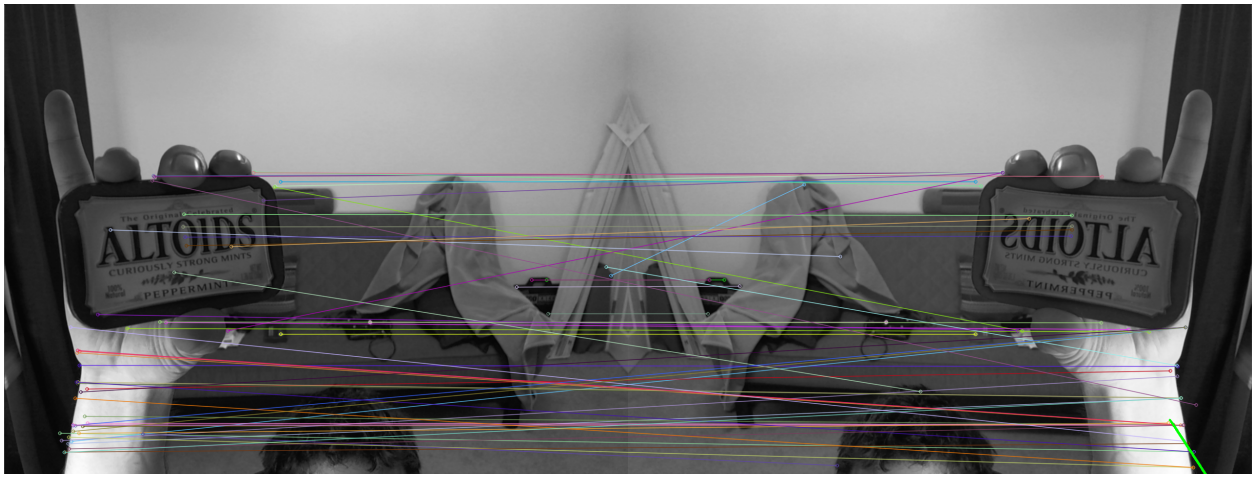
On the SIFT paper published by David Lowe as the title suggests refers to using distinctive image features from scale invariant key points for object recognition, by correlating image features found to a large database containing multiple images (and therefore multiple feature points). The object recognition matching is then accomplished by comparing individual features using a fast neighbor algorithm along with Hough transform on selected grouping features clusters. The SIFT method is proven to be efficient, although computationally expensive it is capable of achieving close to real time performance in non-controlled environments due to its flexibility and complexity. In general the image features are highly distinctive giving each feature matching greater accuracy when comparing to a database. To optimize the performance SIFT implement a top-down approach where simple fast matches are initially done at top and more computational intensive comparisons are done on a deeper lower level. This filtering based analysis is based on a matched detail feature finding where the major stages to detect this common features can be summarized: Scale-space extrema detection (search features through all scales), Keypoint localization (found features marked as matching points), Orientation assignment (deeper analysis to different orientation also based on gradient directions) and Keypoint descriptor (gradients measured and compared around keypoints).

The paper goes on explaining how SIFT works by querying the database and making consecutive comparisons for every cluster formed by 3 or more features found, each time discharging false positives improving the accuracy and consistency of the already matched parameters. The matching of interest points evolved from Maravec's corner detection to then the Harris's corners, using this image matching recognition algorithm on a correlation window allowed arbitrary orientation comparison among different images and giving better results. In summary different researches led to better improved findings for feature points, allowing the comparison to the scaled and rotated to better match data points from a database.

The process of detecting keypoints using cascade filtering on scale-space uses the convolution formed from a Gaussian function to identify candidate locations for further feature point's comparison. The local extrema (min/max) then compares its neighbors and neighbors from the scales matched found giving the frequency of sampling in scale, which maximizes the extrema stability by systematically predicting where each feature might appear (rotated, scaled, affined, stretched, etc..) and increasing the correct measurement and accuracy for each grouping of features matched. As the extrema can be arbitrarily close there is a trade-off between sampling and detection, so by pre-smoothing the image prior to the extrema detection we increase the number of stable keypoints found. In order to provide a more accurate keypoint localization recent contributions used a mathematical quadratic Taylor expansion for better matching and for stability an eigenvalue for a Hessian matrix generated from the edges given from a Gaussian function is computed, and used. Finally orientation and rotation is accomplished by taking a histogram from the sample points with possible gradient orientations, this matched peaks correspond to the directions of local gradients. Descriptors are formed from vector values containing the orientation information from the histogram, this are used for nearest-neighbor comparison methods as mentioned before, avoiding noise and improving the keypoints feature

match finds.
The application to object recognition is one of the most important topics of the paper, it relates the previous explained findings and research to the actual realm of object recognition, by matching found keypoints to the nearest neighbors from equivalent images stored on a database. This discrepancy among false positives is controlled by a Euclidian distance descriptor vector, acting as a threshold that eventually reduces false positive to a ratio of 90% false matches and no more than 5% of correct matches, increasing its effectiveness. In order optimize performance we should be able to identify objects using the fewest possible number of matches, by clustering with a Hough transform we can predict orientation based on the object location along with a bin boundary system to refine error bounds.
The paper ends with the conclusion on how SIFT is useful due its distinctiveness and its overall simple generalization and optimizes the find of correct key points from a database of key points. The algorithm is rich in complexity and performs well for a variety of object recognition problems, taking in consideration distortion, noise, illumination and proving to be a great solution specially on non-controlled environments.

6)



7)

At home I have only 1 UVC compatible camera, I plan to stop by office hours in order to borrow and use one of the lab cameras in order to properly run the sample stereo program

```
root@b:/home/b/Desktop/CSCE485/LABS/LAB4/7# make
g++ -O0 -g -I/usr/local/opencv/include  -c capture.cpp
g++  -O0 -g -I/usr/local/opencv/include  -I/usr/local/opencv/include -o capture capture.o
`pkg-config --libs opencv` -L/usr/local/opencv/lib -lopencv_core -lopencv_flann -lopencv_v
ideo
g++ -O0 -g -I/usr/local/opencv/include  -c stereo_match.cpp
g++  -O0 -g -I/usr/local/opencv/include  -I/usr/local/opencv/include -o stereo_match stere
o_match.o `pkg-config --libs opencv` -L/usr/local/opencv/lib -lopencv_core -lopencv_flann
-lopencv_video
g++ -O0 -g -I/usr/local/opencv/include  -c capture_stereo.cpp
g++  -O0 -g -I/usr/local/opencv/include  -I/usr/local/opencv/include -o capture_stereo cap
ture_stereo.o `pkg-config --libs opencv` -L/usr/local/opencv/lib -lopencv_core -lopencv_fl
ann -lopencv_video
root@b:/home/b/Desktop/CSCE485/LABS/LAB4/7#
```