University of Colorado Boulder

EXERCISE 4

BY
Amreeta Sengupta

07/19/2019

Question 2,3,4 have been performed using Jetson Nano and 6,7 have been performed on virtual machine

QUESTION 1

- Detection of straight lines is an integral part of Computer vision which is required for various image processing applications like self-driving car, object detection etc. Hough employed the slope-intercept parameters for detection of straight lines. Since slope is unbounded and could go on till infinity, it made this technique complex.

- Richard O. Duda and Peter E. Hart proposed the concept that every point on the edge of an image has a range of lines which have a unique perpendicular distance 'ρ' from the origin and a unique angle 'θ' off the x-axis. Basically, the point-to-curve transformation can occur by considering the rotation of a line around that point at a distance ρ form the origin. In the ρ-θ space, points correspond to a sinusoidal curve and intersection of these sinusoids can generate a line.

- An example for inspection of the features of the transform used a 120 X 120 picture which was converted into a bit map by thresholding. The potential lines of intersection were rotated by 360 degrees at 20-degree increments. The number of sinusoidal intersections was tracked and recorded and if it corresponded to a value greater than the decided threshold (i.e. 35), then it was considered to be a significant edge line of the image.

Thus, it is a computationally efficient technique of analysis of linear elements of an image by considering an array of pixels and converting it into a binary bit map and transforming it into the ρ-θ space by considering the distance from any point in an image to a potential line rotated by θ. Infact, this concept can be further applied to different families of curves including circle, ellipse etc.

However, some of the limitations of the proposed method include erroneous results due to detection of meaningless set of collinear points, false line detection due to incorrect threshold value selection and it can become computationally expensive if finer quantization is applied.

QUESTION 2

Build and Run

```
amreeta@desktop:~/Downloads/capture-transformer$ make hough_line
g++ -O0 -g   -c hough_line.cpp
g++  -O0 -g   -o hough_line hough_line.o `pkg-config --libs opencv` -L/usr/lib -
lopencv_core -lopencv_flann -lopencv_video -lpthread
amreeta@desktop:~/Downloads/capture-transformer$ ./hough_line Cows.ppm
Gtk-Message: 10:24:31.444: Failed to load module "canberra-gtk-module"
```

Code Explanation

This code detects straight line in an image by using the Hough-line transform. It includes the following steps:

- The image on which the transform is performed is taken as an argument from the user during runtime. The imread function is used to load the image.
- Next, the canny function is used to detect the edges present in the image.
- The cvtColor function is used to display the image in BGR format.

Standard Hough Line Transform

- This approach keeps track of the intersection points between the curves present in an image and considers it to be a line if the number of intersections is above a certain threshold value.
- A variable called lines which is of vector type is used to store 2 float values for the detected lines.
- The HoughLines function is used next to return the values of the points of the detected lines.
- The lines are hence drawn on the image using the line function.

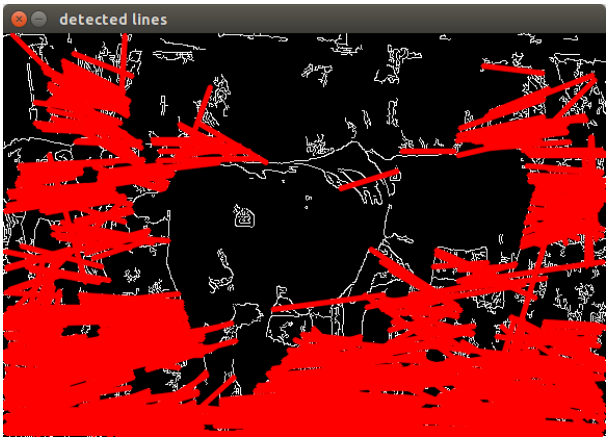
Probabilistic Hough Line Transform

- This approach detects the endpoints of a line present in an image.
- A variable called lines which is of vector type is used to store 4 integer values for the detected lines.
- The HoughLinesP function is used next to return the values of the points of the detected lines.
- The lines are hence drawn on the image using the line function.
- The imshow function is used to display the original as well as the resultant image.

Output

Standard Hough Line Transform

Probabilistic Hough Line Transform



Applications of Hough Line Transform

- Hough Transform, is an efficient and fairly unaffected by noise algorithm which, can be used to detect lines in an image.
- It can be employed for object identification or inspection in various industries.
- Its use can also be extended to detect the orientation or position of certain region of interest in an image.

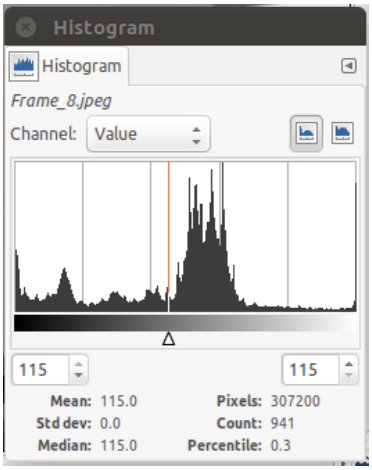
QUESTION 3

Build and Run

```
amreeta@desktop:~/Downloads/ex4/Q3$ make
g++ -O0 -g   -c skeletal.cpp
g++  -O0 -g   -o skeletal skeletal.o `pkg-config --libs opencv` -L/usr/lib -lope
ncv_core -lopencv_flann -lopencv_video
amreeta@desktop:~/Downloads/ex4/Q3$ ./skeletal
Gtk-Message: 21:02:38.122: Failed to load module "canberra-gtk-module"
Max Time = 1.004861 seconds
amreeta@desktop:~/Downloads/ex4/Q3$
```

Code Explanation

- This code is used to perform skeletal transform on all the frames that are continuously captured from the camera.
- Each frame captured from the camera is first converted into its greyscale by using the cvtColor function.
- The binary bit map of each image frame is computed by using the threshold function and the threshold value is determined using histogram analysis as shown below.



- Each pixel is inverted so that we get white pixels for the region of interest.
- The medianblur function can be used to smoothen the image by replacing central element with the median of its all the pixels.
- A structuring element is chosen to perform the morphological operations with a cross-shaped structure and 3X3 kernel size.
- Erosion function basically uses the structuring element to discard the pixels near the boundaries of the object and it helps in removal of white noise.
- Dilation function uses the structuring element to basically increase the size of the foreground object.
- Erosion while removing the noise, also shrinks the object, hence dilation is used to increase the object size.
- Next, the original image and the resultant image post erosion and dilation is subtracted.
- Bitwise or is used for the empty image and this subtracted image.
- Done variable is used to check whether the count of non-zero elements of the image (i.e. the white pixels) is zero or not so that the transform is performed on all pixels of interest.
- Iterations variable is used to exit the loop after 100 iterations.
- The frames are saved in the file and are used to re-encode the video.

Ffmpeg command to re-encode the video

```
amreeta@desktop:~/Downloads/ex4/Q3$ ffmpeg -r 30 -f image2 -i Frame_%d.jpeg q3.mpeg
ffmpeg version 3.4.6-0ubuntu0.18.04.1 Copyright (c) 2000-2019 the FFmpeg developers
  built with gcc 7 (Ubuntu/Linaro 7.3.0-16ubuntu3)
  configuration: --prefix=/usr --extra-version=0ubuntu0.18.04.1 --toolchain=hardened --lib
dir=/usr/lib/aarch64-linux-gnu --incdir=/usr/include/aarch64-linux-gnu --enable-gpl --disa
ble-stripping --enable-avresample --enable-avisynth --enable-gnutls --enable-ladspa --enab
le-libass --enable-libbluray --enable-libbs2b --enable-libcaca --enable-libcdio --enable-l
ibflite --enable-libfontconfig --enable-libfreetype --enable-libfribidi --enable-libgme --
enable-libgsm --enable-libmp3lame --enable-libmysofa --enable-libopenjpeg --enable-libopen
mpt --enable-libopus --enable-libpulse --enable-librubberband --enable-librsvg --enable-li
```

In the video, the hand seems to be moving quite fast due to slow processing speed of Jetson. Even if there is a slight movement in the hand, the Jetson is still processing the previous frame and therefore it

doesn't catch the slight movements in the hand and directly jumps to the frame where the position of the hand has changed. Hence, the hand appears to be moving quite fast.

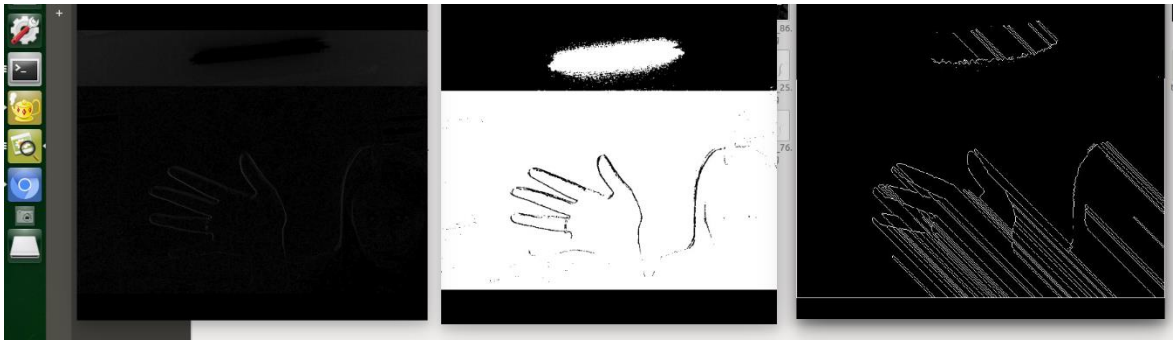
QUESTION 4

Build and Run

```
amreeta@desktop:~/Downloads/ex4/Q4$ make
g++ -O0 -g   -c q4.cpp
g++  -O0 -g   -o q4 q4.o `pkg-config --libs opencv` -L/usr/lib -lopencv_core -lo
pencv_flann -lopencv_video
amreeta@desktop:~/Downloads/ex4/Q4$ ./q4
Gtk-Message: 21:35:36.364: Failed to load module "canberra-gtk-module"
Max Time = 0.467799 seconds
amreeta@desktop:~/Downloads/ex4/Q4$
```

Code Explanation

- This code is used to perform skeletal transform on all the frames that are continuously captured from the camera.
- Each frame captured from the camera is first converted into its greyscale by using the cvtColor function.
- For background elimination in order to preserve the region of interest, each present frame can be subtracted from the previous frame by using the absdiff function after the frames are converted to their grey scale value. However, I have not used this function as due to frame differentiation, the hand was getting eliminated and the more edges of the fingers were getting detected in the skeletal transform which was not accurate.



- The binary bit map of each image frame is computed by using the threshold function and the threshold value is determined using histogram analysis.
- The medianblur function can be used to smoothen the image by replacing central element with the median of its all the pixels.
- A thinning algorithm is applied which is uses crossing number (chi) that represents the 0-to-1 and 1-to-0 pixel transitions and this chi value determines the points that will be removed for thinning the image. The points at the boundary of the object are removed when the chi value is 2 and they are retained if chi is greater than 2 or 0. Sigma represents the sum of the eight pixel values around

the pixel of interest the points are only removed if sigma is not equal to 1 in order to preserve the line ends so that it properly represents the object.

- The frames are saved in the file and are used to re-encode the video.

Ffmpeg command to re-encode the video

```
amreeta@desktop:~/Downloads/ex4/Q4$ ffmpeg -r 30 -f image2 -i Frame_%d.jpeg q4.mpeg
ffmpeg version 3.4.6-0ubuntu0.18.04.1 Copyright (c) 2000-2019 the FFmpeg developers
  built with gcc 7 (Ubuntu/Linaro 7.3.0-16ubuntu3)
  configuration: --prefix=/usr --extra-version=0ubuntu0.18.04.1 --toolchain=hardened --libdir=/usr/
lib/aarch64-linux-gnu --incdir=/usr/include/aarch64-linux-gnu --enable-gpl --disable-stripping --en
able-avresample --enable-avisynth --enable-gnutls --enable-ladspa --enable-libass --enable-libblura
y --enable-libbs2b --enable-libcaca --enable-libcdio --enable-libflite --enable-libfontconfig --ena
ble-libfreetype --enable-libfribidi --enable-libgme --enable-libgsm --enable-libmp3lame --enable-li
bmysofa --enable-libopenjpeg --enable-libopenmpt --enable-libopus --enable-libpulse --enable-librub
berband --enable-librsvg --enable-libshine --enable-libsnappy --enable-libsoxr --enable-libspeex --
enable-libssh --enable-libtheora --enable-libtwolame --enable-libvorbis --enable-libvpx --enable-li
bwavpack --enable-libwebp --enable-libx265 --enable-libxml2 --enable-libxvid --enable-libzmq --enab
le-libzvbi --enable-omx --enable-openal --enable-opengl --enable-sdl2 --enable-libdc1394 --enable-l
ibdrm --enable-libiec61883 --enable-chromaprint --enable-frei0r --enable-libopencv --enable-libx264
```

Comparison of Bottom-up and Top-down approach

- The maximum execution time for each frame is calculated for both approaches as shown in the Build and Run images for both. From that we find that the maximum frame execution time for top-down approach is 1.004861 seconds while the maximum frame execution time for bottom-up approach is 0.467799 seconds. Hence, we can conclude that bottom-up approach requires less execution time as compared to top-down approach.
- From the video of each approach, we can see that top-down approach displays the skeletal transform more clearly with better quality as compared to bottom-up approach. Hence, we can conclude that bottom-up approach reduces the transform quality as compared to the top-down approach.

QUESTION 5

- Scale Invariant Feature Transform is a robust algorithm for image perception which extracts distinct features from an image and is capable of performing reliable matching between different views or angles of an object in an image. The features have the quality of being invariant to image scale, rotation, distortion, change in 3D viewpoint, noise and change in illumination while also being highly distinctive. These extracted features can additionally be used for object recognition by comparing the individual feature with a database of features and further verification steps which finally lead to the identification of the object of interest. Thus, thus applications of this algorithm are manifold including Sorting and tracking systems, Optical navigation etc.
  The principal steps for generation of these features include:
  - Scale-space extrema detection which is used to determine the best interest points in an image which are required to form the keypoint feature vectors.
  - Keypoint localization which is used to filter out the keypoints based on their stability in order to eliminate the noisy feature vectors.

- Orientation assignment which involves assignment of the one or more orientations to each keypoint location to ensure orientation invariance.
- Keypoint descriptor includes measurement of the local image gradients at the selected scale for each keypoint.

The features thus extracted are stored in a database and for image matching and recognition, the features of the image of interest are matched with the features present in the database. The perfect match is determined based on the Euclidean distance of the feature vectors.

- The various detailed steps encapsulating the process of derivation of distinctive invariant keypoints are as follows:
  - To detect the keypoints, first the images are made scale invariant by generation of a number of octaves whose size keep becoming half. Within each octave, Gaussian blur operator is applied to progressively blur the images which creates the scale space of image.
  - After blurring the image, to locate the edges and the corners of the image, the difference between the consecutive scales is computed which gives a scale invariant Laplacian of gaussian approximations.
  - Next, in order to detect the Local maxima and minima, each sample point is compared to its eight neighbors in the current image and nine neighbors each in the adjacent scales and is selected as a keypoint only if it is larger or smaller than all of these neighbors.
  - Next, in order to determine the positional accuracy of each feature, the image undergoes several transformations including rotation, scaling, affine stretch etc. The image is also expanded to create more sample points using linear interpolation which increases the number of stable keypoints.
  - For accurate keypoint localization, Taylor expansion of the image around the approximated keypoint is performed which gives the subpixel keypoint location.
  - In order to extract only the useful features, the low-contrast features are eliminated. This is done by rejecting the intensity magnitudes of the subpixels which are below a certain level.
  - To provide orientation and rotation invariance, the keypoint descriptor are represented relative to the orientation of the keypoint. This can be determined by computing the highest peak in the orientation histogram and any local peak within 80% of the highest peak which will create a keypoint with that orientation. Finally, a parabola is fit to the 3 histogram values closest to each peak to interpolate the peak position for improved accuracy.
  - To compute a descriptor for the local image region, image gradient magnitudes and orientations are sampled around the keypoint location. A Gaussian weighting function is used to assign a weight to the magnitude of each sample point. Interpolation is used to generate orientation and magnitude data in between pixels. The descriptor is formed from a vector containing the values of all the orientation histogram.

- These derived distinctive invariant keypoints can be used for object matching and recognition by performing the following steps:

- The first step is Keypoint matching where the best candidate match for each keypoint is determined, by identifying its nearest neighbor i.e. the keypoint with minimum Euclidean distance, in the database of keypoints from training images. For efficient nearest neighbor indexing, Best-Bin-First (BBF) algorithm is used which searches the bins in the order of their closest distance from the query location.
- Some of the matches may be incorrect due to a cluttered background and hence to eliminate this, clusters of at least 3 features are first identified using the Hough transform which identifies clusters of features with a consistent interpretation by using each feature to vote for all object poses that are consistent with the feature which increases the probability of a correct match.
- Then, each cluster is checked by performing a detailed geometric fit to the model, and the result is used to accept or reject the interpretation.

Thus, Scale Invariant Feature Transform is an important algorithm for extracting the interesting points of an object in order to provide the feature description of the object. It is a robust technique which even allows identification of various objects in a cluttered environment. The algorithm is efficient enough to extract several thousand keypoints from an image in real time. The features thus extracted are not only invariant to image scaling, rotation, change in illumination, noise etc. but are also capable of determining accurate object pose and location. These features can also be used for object recognition by matching individual features to a database of features from known objects using a fast nearest-neighbor algorithm, followed by a Hough transform to identify clusters belonging to a single object, and finally performing verification through least-squares solution for consistent pose parameters. It can be utilized in many real-time applications including 3D scene modeling, recognition and tracking, Robot localization and mapping, image panorama assembly, epipolar calibration etc. It can basically be used in any system implementations which require identification of matching locations between images. In long term, this can have further improvements in terms of systematic testing on data sets with full 3D viewpoint and illumination changes, learning specific features suited for recognition of particular object categories in order to cover a broad range of possible appearances of objects etc.

QUESTION 6

Build and Run

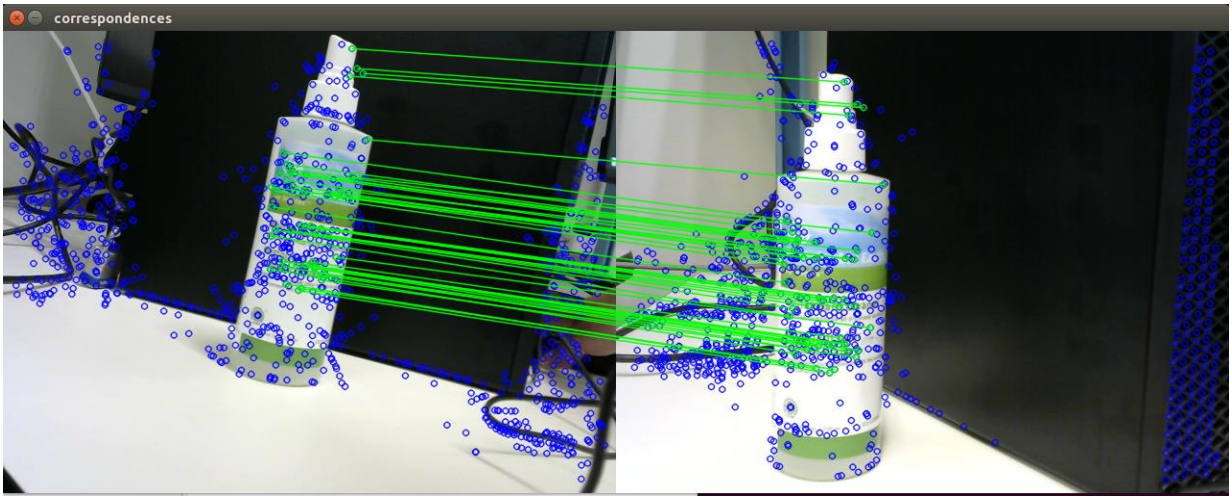
```
desktop@desktop-VirtualBox:~/Downloads/sift$ make
g++ -O0 -g   -c descriptor_extractor_matcher.cpp
g++  -O0 -g   -o descriptor_extractor_matcher descriptor_extractor_matcher.o `pkg-config --libs opencv` -L/usr/lib -lopencv_core -lopencv_flann -lopencv_video
desktop@desktop-VirtualBox:~/Downloads/sift$ ./descriptor_extractor_matcher SURF SURF BruteForce CrossCheckFilter left.jpg right.jpg 3
< Creating detector, descriptor extractor and descriptor matcher ...
>
< Reading the images...
>

< Extracting keypoints from first image...
1041 points
>
< Computing descriptors for keypoints from first image...
>

< Extracting keypoints from second image...
965 points
>
< Computing descriptors for keypoints from second image...
>
< Matching descriptors...
>
< Computing homography (RANSAC)...
>
```

Code Explanation

- This code is used to match the keypoint features between the left and the right image.
- It takes either 7 (for one image) or 8 (for two images) runtime arguments.
- The first run-time argument represents the detector type which is selected as SURF which is used as the Feature Detector type for efficient extraction of the optimum features.
- The second run-time argument represents the descriptor type which is also selected as SURF.
- The third run-time argument represents the matcher type which is selected as Brute Force matcher which uses the descriptor of a feature in one image and matches it with all the features in the second image and uses distance calculation returning the closest one.
- The run-time fourth argument represents the match filter type which is selected as Crosscheck filter which returns the best match after matching the features of the two images. Simple matching can also be used which uses the two image descriptors to match the image. This is a simpler but less efficient approach.
- Two images are then taken as next two run-time arguments from the user and read by using the imread function.
- The last argument is the ransacReprojThreshold value for the homography matrix.
- Using the above arguments, the keypoints from each image is detected and the descriptor for each image keypoint is computed.
- The descriptors are matched and the homography transformation between the two sets of points are evaluated which using RANSAC which provides an efficient matching technique and this function returns the inlier and outlier points which are then matched in the two images.
- The drawMatches function is used to draw these points on the image and the result is displayed in the window named correspondences.

Left Image and Right Image

Output



QUESTION 7

Capture_stereo

Build and Run

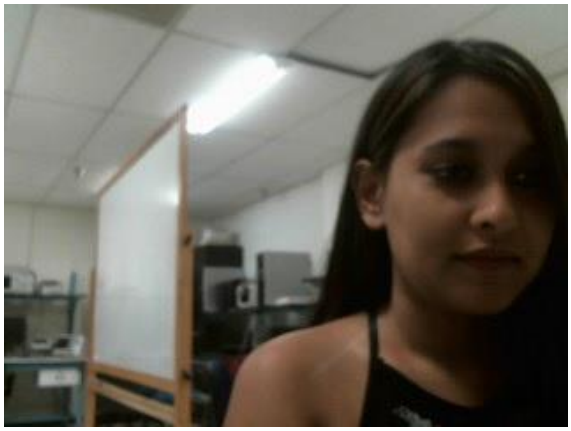
```
desktop@desktop-VirtualBox:~/Downloads/example-stereo$ make capture_stereo
g++ -O0 -g -I/usr/local/opencv/include  -c capture_stereo.cpp
g++  -O0 -g -I/usr/local/opencv/include  -I/usr/local/opencv/include -o capture_stereo capture_stereo.o `pkg-config --libs opencv` -L/usr/local/opencv/lib -lopencv_core -lopencv_flann -lopencv_video
desktop@desktop-VirtualBox:~/Downloads/example-stereo$ ./capture_stereo 0 1
argv[1]=0, argv[2]=1
Will open DUAL video devices 0 and 1
LEFT dt=1563497076291.083496 msec, RIGHT dt=1563497076291.084229
LEFT dt=82.239502 msec, RIGHT dt=82.239258
LEFT dt=64.242188 msec, RIGHT dt=64.242188
LEFT dt=69.394531 msec, RIGHT dt=69.394531
LEFT dt=68.055420 msec, RIGHT dt=68.055176
LEFT dt=70.564453 msec, RIGHT dt=70.564453
LEFT dt=69.195557 msec, RIGHT dt=69.195801
LEFT dt=63.631104 msec, RIGHT dt=63.631104
LEFT dt=65.799805 msec, RIGHT dt=65.799805
LEFT dt=65.871094 msec, RIGHT dt=65.871094
LEFT dt=63.662354 msec, RIGHT dt=63.662842
LEFT dt=67.697998 msec, RIGHT dt=67.697754
LEFT dt=67.997559 msec, RIGHT dt=67.997559
LEFT dt=66.263672 msec, RIGHT dt=66.263428
LEFT dt=66.507324 msec, RIGHT dt=66.507324
LEFT dt=64.372314 msec, RIGHT dt=64.372314
LEFT dt=66.797852 msec, RIGHT dt=66.800293
LEFT dt=66.801025 msec, RIGHT dt=66.800049
LEFT dt=55.982666 msec, RIGHT dt=55.981445
LEFT dt=64.919678 msec, RIGHT dt=64.919434
LEFT dt=72.321533 msec, RIGHT dt=72.321777
LEFT dt=66.892578 msec, RIGHT dt=66.892334
```

Code Explanation

- The depth of the images in a system can be defined as inversely proportional to the difference between the distance of the two image points and the camera center. This code is used to find the disparity map of images and the depth can be estimated from this.
- This code takes the camera device numbers from the user as a runtime argument as shown above.
- It can either use one camera or two cameras as per the run-time argument specified by the user.
- cvSetCaptureProperty function is used the set the resolution for both the cameras.
- A myStereoVar object is used to set the parameters for disparity which include the number if pyramid layers, minimum and maximum disparity values, smoothness parameter etc.
- The myStereoVar function is used to compute the disparity using variational matching algorithm.

- The disp argument in the myStereoVar function holds the correspondences for each pixel for both the images captured from the two cameras.
- The clock_gettime function is used to display the frame time of each frame in the left and right images.
- 'ESC' key can be pressed to save the left and right images.

Left and Right Image



Output

Disparity correspondence analysis for left and right eye image

- Basicall, disparity is computed by the difference in the distance of each point in the left and right images which can give us information about the depth of the images.
- If an object distance from camera is small, then the point movement is assumed to be larger as they share an inverse relation. So, all the white parts displayed in the image are closer while the black ones are further away.

- The depth of the objects, which can be defined as inversely proportional to the difference between the distance of the two image points and the camera center, can hence be estimated from the disparity map.
- Unlike stereo_match, this code does not set the intrinsic and extrinsic parameters of the camera nor does it perform undistortion and rectification on the image which will affect the quality of the disparity calculation. If these factors are taken care of, we can get an improved and efficient disparity map.

Stereo_match

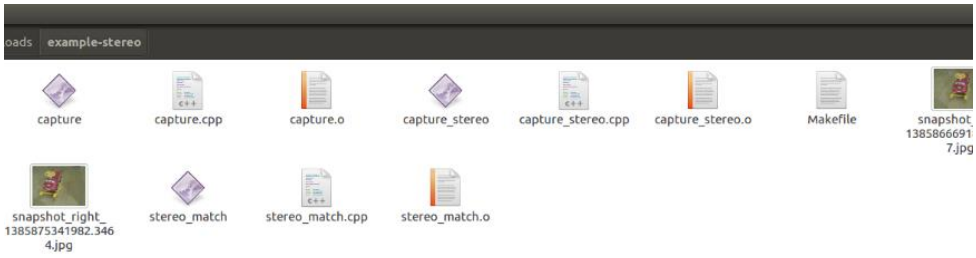
Build and Run

```
desktop@desktop-VirtualBox:~/Downloads/example-stereo$ make
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 stereo_match.o `pkg-config --libs opencv` -L/usr/local/opencv/lib -lopencv_core -lopencv_flann -lopencv_video
desktop@desktop-VirtualBox:~/Downloads/example-stereo$ ./stereo_match snapshot_left_1385866691881.1577.jpg snapshot_right_1385866691881.1580.jpg
Time elapsed: 36.392976ms
press any key to continue...
```

Code Explanation

- This code is used to find the disparity map of images.
- It takes two images from the user as a runtime argument as shown above to compute the disparity.
- After reading the images, the images are scaled to a proper size. The intrinsic and extrinsic parameter are adjusted and stereoRectify function is used to compute the rotation matrix.
- The initUndistortRectifyMap is used to compute the undistortion and rectification transformation map.
- The Stereo SGBM parameters are set to produce the optimum disparity map using the Stereo SGBM algorithm.
- The code also displays the time elapsed.

Output



REFERENCES

- http://delivery.acm.org/10.1145/370000/361242/p11-duda.pdf?ip=128.138.65.133&id=361242&acc=ACTIVE%20SERVICE&key=B63ACEF81C6334F5%2E1FC7ACB276C876CF%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1562957675_9ded7ae6bd890b770920bc02bb0b8acd
- http://ecee.colorado.edu/~siewerts/extra/ecen5763/ecen5763_doc/Lectures/
- RTES Exercise 4 by Amreeta Sengupta
- ER-Davies-Chap-9-Skeletal-Transform.pdf
- EMVIA Exercise 3 by Amreeta Sengupta
- https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf
- https://en.wikipedia.org/wiki/Scale-invariant_feature_transform
- http://aishack.in/tutorials/sift-scale-invariant-feature-transform-scale-space/
- https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_calib3d/py_depthmap/py_depthmap.html
- http://www.cse.uaa.alaska.edu/~ssiewert/a485_code/
- https://docs.opencv.org