Proposal for Summer of Code 2k17 under FOSSEE

Project : Image Processing for Medical Healthcare Research

Name : SAQIB AZIM

Mentor : Hina

Department : Electrical Engineering(B. Tech)

Current Study Year : 2nd

Slack Nickname : dd1707

Github : https://github.com/saqib1707

Email : azimsaqib96@gmail.com

Telephone/Mobile : +91-8828290924

Project Abstract : This project deals with processing of images obtained from medical devices for the detection of cervical cancer in patients.

Image Stitching and Quality Assessment : Given multiple images, stitch them together to create a panorama and analyze the panorama image quality.

Image Stitching Algorithm Overview -

1. Detect key Points (locally distinct points using SIFT or SURF Algorithm) and extract local invariant descriptors from the two images (two images that are to be stitched).
2. Match the descriptors in the two images.
3. Estimate Homography matrix with four matched keypoints using RANSAC Algorithm.
4. Apply a warping transformation using the homography matrix obtained.

SIFT (Scale Invariant Feature Transform) Algorithm for detecting interest points (locally distinct points) and descriptors in an image and matching those descriptors with another image.

This algorithm is scale-invariant that is it can detect and match interest points of an object in two images even if the scale of the object in the two images are different or the images are taken from two different viewpoints. One most important feature of SIFT is that it is invariant to rotation that is if I rotate the image it’s going to give almost the same local descriptors in the original image and the rotated image. So the task is to look for locally distinct points. The overview steps for detecting interest points and their descriptors in an image are as follows:

1. Detecting key (interest) points in the image.
2. Detect the outlier points (points that are not interest points) using the difference of Gaussian filtered images and remove them.
3. Attach descriptors with each interest point.

Here I am describing each of the aforementioned points in detail:

1. Detecting interest points

First we will smooth the image using Gaussian filters having different standard deviation values as well as with differently scaled images. Then we will take the differences of those adjacent filtered images which basically gives the local variations in the image. Next we will find the extrema(minima/maxima) among those difference images by considering a 3x3 matrix around each point in the difference image and its adjacent images. So it's like checking whether the central pixel is extrema among 27 pixel values (9 in same image + 2x9 in adjacent difference of gaussian filtered images) and if it is an extrema then we consider it as an interest point.
This image gives a diagramatic view about the above process.

The gaussian distribution in two dimension is defined as follows:

$$G(x,y,\sigma) = (1/(2\pi\sigma^2))\ e^{-(x2+y2)/(2\sigma2)}$$

First we need to approximate the Laplacian of Gaussian by the difference of Gaussians using this relation:

Partial differentiation of $G(x,y,\sigma)$ w.r.t $\sigma = \sigma(\triangle^2 G)$

$\Rightarrow G(x,y,k\sigma) - G(x,y,\sigma) = (k-1)(\sigma^2)\triangle^2 G$

Typical values of $\sigma = 1.6$ and $k = \sqrt{2}$ that have been found in experiments.

Next build a scale space.

1. In the first octave, convolve the training image with gaussian matrix with values of the standard deviation parameter being σ, $k\sigma$, $k^2\sigma$, $k^3\sigma$, $k^4\sigma$ and so on.
2. According to this paper by David Lowe, the maximum repeatability (maximum chances of matching of interest points in two partially overlapping images) occurs when no of scale per octave = 3. Hence we will convolve the training image with Gaussian filter with deviation parameter being σ, $k\sigma$, $k^2\sigma$, $k^3\sigma$, $k^4\sigma$ only.

3. Then we will take the difference of the two adjacent gaussian filtered images to get Difference of Gaussian(DOG) image which basically will show the locally distinct points.
4. Next we need to do this for the next octave where the resolution of the image will change.

** Here octaves are set of images with same resolution but smoothened by different gaussian filters. Similarly next octave can be considered to be taking every alternative row and column so that resolution becomes one-fourth.**

Now we need to find the extrema among these DOG. So for that we will take a 3x3 matrix surrounding each pixel and if the value of the central pixel at the middle image is an extrema among the 27 pixels then that pixel corresponds to an interest point.

2. Detect the outlier points (points that are not interest points) using the difference of Gaussian and remove them

Among the interest points which we got, there can be many that are outlier points that are selected because of noise or incorrect data or poorly localized points along an edge.

For removing them these are the steps that need to be followed:

1. **Initial Outlier Rejection** : Use Taylor series expansion of the difference of the Gaussian D(x) that we got while detecting the interest points. Differentiating the taylor series to get the maxima or minima points. Now if the value of the D at these extrema points are less than a threshold set then those points need to be rejected.

2. **Further Outlier rejection Method or Extrema Suppression** : The DOG finds blob-like or corner like structures but also finds key points along edges which are bad for matching(since the points along the same edge are not locally distinct). For this we need to eliminate the key points along the edges. So, we need to compute the Hessian of D(x) as explained in the image link below.

This image explains the above outlier rejection process.

3. Orientation Assignment

Now it is the time to attach descriptor to each interest point after rejection of outliers.

a. To achieve rotation invariance.
b. Will be used for SIFT descriptor.

Compute central derivatives, gradient magnitude and direction of L(smooth image) at the scale of key point (x,y).

$m(x,y) = \mathrm{sqrt}((L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x, y-1))^2)$
$\theta(x,y) = \tan^{-1} ((L(x,y+1) - L(x, y-1)) / (L(x+1,y) - L(x-1,y)))$

Now around each interest point, we are going to create a weighted histogram using a 4x4 pixel surrounding area and will categorize the magnitude and direction into 8 bins with the direction vectors pointing between 0-45 degrees in first bin, 45-90 in second bin and so on upto 360. The peak among all bins will be taken as the direction of the interest point.

4. Local Image Descriptor at interest Points

A descriptor describes not only that interest point but a local neighbourhood around it and by making comparisons between descriptors of the interest points, we can make correspondence between the two images.

1. Use of gradient orientation histogram.
2. Here we will compute relative orientation and magnitude in a 16x16 neighbourhood of an interest point.
3. The 16x16 matrix can be divided into groups of 4x4 matrix with 16 of these. Each 4x4 represents a vector of 8 bins as illustrated in the 3rd point. Hence for each local descriptor for a key point we need a 8x16 dimensional vector.

So a SIFT feature is given by a vector computed at a local extreme point in the scale space - < p, s, r, f >
Where p = location of the key point in the image
s = scale
r = orientation
f = 128 dimensional feature vector

5. Interest Point Matching in two images

** Assuming we know the interest points and their descriptors in the two images **
We need to do this because in order to join any two images into a bigger image we must obtain as to what are the overlapping points. These overlapping points will give us an idea of the orientation of the second image w.r.t to the other one. This is done by matching the local descriptors of interest points in the two images. So we take one key point in 1st image and match it with descriptors of all the key points in 2nd image and select the best match i.e the descriptor with minimum euclidean distance. But there may be situations where a 2nd best match is the actual match for the key point and not the best match or the corresponding matching is not correct. Hence we need to get rid of the outliers matching. For that there is a criteria for matching the points. If the ratio of

the euclidean distance with the best match to the 2nd best match is less than 0.8(say) then the best match is considered a match while if it's greater than 0.8 then this key point is not taken into account.
** Euclidean distance = sqrt of the sum of squares of difference of 128 dim. feature vectors.

6. Compute Homography Matrix

Homography matrix is a 3x3 transformation matrix that efficiently maps points that are projected in both the images. It transforms image plane P1 to another image plane P2.

$$I_1 = H \times I_2 \qquad \text{-------------------------------------} \quad \text{eq . 1}$$

I_1 = 1st image interest points
I_2 = 2nd image corresponding interest point (matching with first one)
H = Homography matrix

For finding the homography matrix using RANSAC Algorithm we need to know these matching points.

Random Sample Consensus (RANSAC Algorithm)

Ransac is an iterative model to estimate parameters of a mathematical model. In this learning technique algorithm, first we take a sample of 4 random matching interest points from our dataset that will be sufficient for fitting eq . 1 and compute homography matrix from them. In python opencv the function getPerspectiveTransform takes 4 points as input and returns the homography matrix.

M = cv2.getPerspectiveTransform(pts1,pts2)

These sampled points are called maybeinliers. Then I will fit all the remaining points to this model and if the error margin is less than some threshold then these points will be added to alsoinliers. After It will check if the number of points in alsoinliers are greater than the previous max. value. Then it will calculate the total error by fitting all the points in maybeinliers and alsoinliers to eq. 1 and if this error is less than the previous minimum error calculated then this model is chosen as the best model till now. This process iterates a number of times and selects the homography matrix which satisfies the maximum number of matching points within a threshold margin. More the number of iterations, much better will be the estimated homography matrix.

Ransac Algorithm Pseudo code

```
best error = some large value
While iterations < k:
    maybeInliers = n randomly selected points out of the observed dataset
```

```
        alsoinliers = empty set
        maybemodel = a proposed model fitted to the maybeinliers
        For all points in dataset and not in maybeinliers :
                If point satisfies the maybemodel with error less than p percent:
                        Add point to alsoinliers

        If number of elements in alsoinliers > some threshold:
                bettermodel = model fitted to all points in maybeinliers and alsoinliers
                presenterror = a measure of how well the model fits these points
                If presenterror < besterror:
                        best error = present error
                        best fit model = better model
        Iterations ++

Final estimated model = best fit model
```

7. Warping and Stitching

Once we have estimated the homography matrix i.e, we know how the second image will look from the 1st image perspective, we need to transform it into a new space. This process is called warping. So to warp, essentially to change the field of view, we apply the homography matrix to the image.
The warpPerspective function takes a 3x3 transformation(here it is homography) matrix
warped_image = cv2.warpPerspective (image, homography_matrix, dimension_of_warped_image)

Once we have obtained a warped image, we need to simply add the warped image with the second image. Thus we get a panorama of two images.

Stitching Multiple Images into a panorama

For stitching multiple images, we just need to provide all images in left to right orientation (in order) and it will stitch taking two images at a time.
Pseudo code :

```
  images [] <-- Input images
  Assuming, that the center image is no_of_images/2
  let centerIdx = length(images)/2

  for each images[] at positions 0 -> centerIdx :
      perform leftward stitching
```

```
for each images[] at positions centerIdx -> length(images):
    perform rightward stitching
```

Image Quality Qualitative Analysis

While the final arbiter of image quality is the human viewer, efforts have been made to create objective measures of quality(some mathematical model). Many objective measures of quality require the existence of a distortion-free copy of an image, called the reference image, that can be used for comparison with the image whose quality is to be measured. The image formation process can be described by the ideal pin-hole model where only the light rays from the depicted scene that passes through the aperture can fall on the image plane and hence image quality can be measured as the deviation of the image with respect to the original distortionless image. Imaging systems may introduce some amount of distortion or artifacts in the image. So In a digital image, the image quality depends on the storage, compression, transmission of image. Unless we have a lossless compression system, the output image will be different from the original image.
In a digital image, the image quality depends on three factors - one is how much the image formation process of the camera deviates from the ideal camera pin-hole model. and also in what encoding the image is stored whether - it is png or jpg or some other format. Another thing is when we transmit the image, then the receiver side image may get distorted as compared to the sender side image.

Image quality can be measured in two ways-
- ***Full reference method*** in which the image quality is measured with respect to a reference image that is assumed to have perfect quality. For example the comparing of a jpeg converted image with its original image is a full reference.
- ***No reference method*** in which the image quality is not measured with respect to a reference.

Image quality depends on various factors:

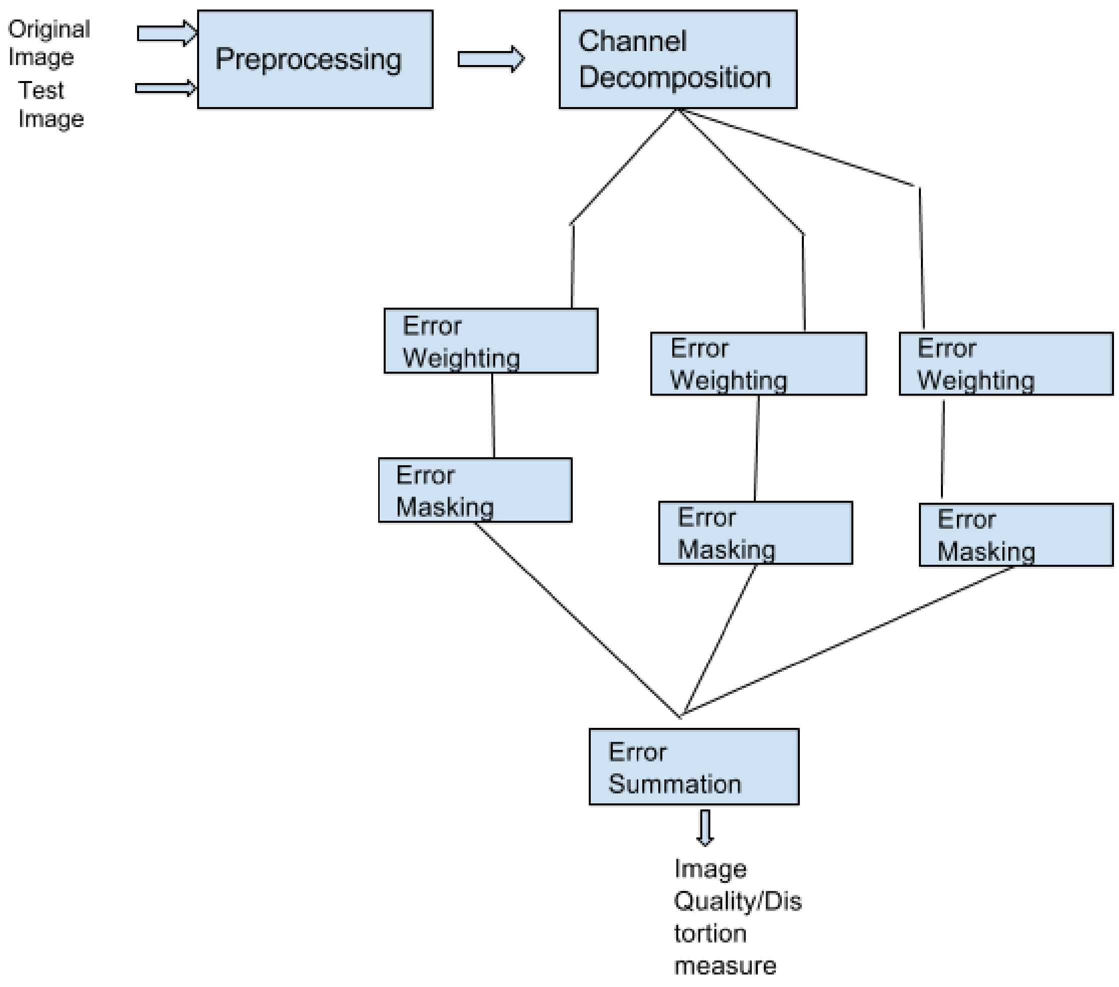
1. Sharpness
    a. Determines the amount of detail an image can convey.
2. Brightness
3. Contrast
4. Resolution
    a. The actual resolution is determined by the sensor , pixel size and actual distance to the object.
5. Dynamic Range

    a. How well a camera captures both bright and dark regions in the same image is described by its dynamic range.
6. Noise and SNR
    a. Noise corresponds to the grain in the image.
7. Sensor and Pixel Size
    a. The bigger the sensor the higher the image quality.
    b. The size of a pixel is important w.r.t the quantity of light it can collect.
8. Quantum Efficiency
    a. The number of electrons produced per number of photon is the quantum efficiency.
    b. Higher the quantum efficiency , more information will be available in the image.

Image Quality Quantification

Error Sensitivity Based Approach

First the original and test images are preprocessed (possibly alignment, color transformation etc). The output is preprocessed original and test signals. The channel decomposition method is applied to the preprocessed signals, resulting in two sets of transformed signals for different channels. The decomposed signal is treated differently in different channels according to human visual sensitivities measured in the specific channel. The error between the two signals in each channel is calculated and weighted by a contrast sensitivity function(CSF). The weighted error signals are adjusted by a visual masking effect model which reflects the reduced visibility of errors presented in the background signal. Finally an error summation method is employed to supply a single quality value of the whole image being tested. This model has its weaknesses that’s why I am explaining the next model.



Structural Distortion Based Image Quality Measurement

The image quality metrics should be decided as follows : ***The main function of human eyes is to extract structural information from the viewing field. Hence a measurement of structural distortion should be a good approximation of perceived image distortion. The key point of this method is to switch from error measurement to structural distortions measurement.***

A simple but effective quality indexing algorithm. Let x = {xi | i = 1,2,3,...N} and y = {yi | i = 1,2,3… N} be the original and test images respectively. The proposed quality index is defined as

$$Q = 4\sigma_{xy}\, x_m\, y_m /(\sigma_x^2 + \sigma_y^2)[(x_m)^2 + (y_m)^2]$$

where $x_m = 1/N \sum x_i$ and $y_m = 1/N \sum y_i$ $\forall$ i = 1,2,3… N

$\sigma_x^2 = 1/(N-1) \sum(x_i - x_m)^2$ and $\sigma_y^2 = 1/(N-1) \sum(y_i - y_m)^2$

$\sigma_{xy} = 1/(N-1) \sum(x_i - x_m)(y_i - y_m)$

The dynamic range of Q is [-1, 1].The best value of 1 is achieved when xi = yi for all i=1,2,...N. This quality index models any distortion as a combination of three different factors - loss of correlation, mean distortion and variance distortion. We can rewrite the definition of Q as a product of three terms :

$$Q = (\sigma_{xy}/\sigma_x\sigma_y) * (2x_m y_m/[(x_m)^2 + (y_m)^2]) * (2\sigma_x\sigma_y/[\sigma_x^2 + \sigma_y^2])$$

The first component is the linear correlation coefficient between x and y whose dynamic range is [-1, 1]. The second component with a value range of [0,1] measures how close the mean values are between x and y. The third component measures how similar the variances of the signals are. Its range of values are also [0,1] where the best value of 1 is achieved only when $\sigma_x = \sigma_y$.
The quality index is applied to natural images using a 8x8 sliding window approach. The quality indices are calculated within the sliding window, leading to a quality map of the image. The overall quality index value is the average of the quality map.

Other Approaches :
Apart from this the image processing toolbox provides several function that can be used to measure image quality.

a. **psnr** - The peak signal-to-noise ratio measure of quality works by first calculating the mean squared error (MSE) and then dividing the maximum range of the data type by the MSE. This measure is simple to calculate but sometimes doesn't align well with perceived quality by humans.
b. **ssim** — The Structural Similarity (SSIM) Index measure of quality works by measuring the structural similarity that compares local patterns of pixel intensities that have been normalized for luminance and contrast. This quality metric is based on the principle that the human visual system is good for extracting information based on structure.

Parallel Computing : Python provides **multiprocessing** library which allows one to leverage multiple processors on a given machine.
I have written more about parallel computing in this github file since the proposal was getting a little lengthy.
An example code on multithreading .
An example code on multiprocessing.

Extract Region of Images : This is a simple task in python opencv where once loading the image, one can sample any region of the image something like this :

```
img = cv2.imread(‘path to the image’)
extracted_image = img[ initial_row : final_row, initial_col : final_col ]
```

Academic Experience

So I'm a sophomore undergraduate student of Electrical Engineering at the Indian Institute of Technology Bombay.I wouldn't say electrical engineering was my passion when I chose to pursue it. I have always loved to tinker with old electronic items in my house but I never thought I would pursue it as a career. I have always loved science and technology, especially physics and mathematics, and electrical engineering seemed like a good bet as it would introduce me to a wide spectrum of engineering. I have been blessed to study in a good institute and I've fallen in love with my subject. Though I program a lot, I try my level best to stay focussed on my course work and I hope to pursue a career that has an apt combination of the two.

My Past Projects

Apart from that I have done quite a number of **self-projects**, some of them I have listed below-

Local Positioning System : We (with my friends) have created a system that can localize an object in a local environment (something that GPS cannot achieve) with accuracy in cms.
A simple Calculator : It is a simple calculator which uses PyQt for it’s GUI. I built it as a part of Summer of Code conducted last year under Competitive Programming.
RattleSnake : This is a GUI-based application for coaching classes to maintain the database of students as well as teachers and allow them to perform various functions.
Memory Puzzle Game : A simple memory puzzle game made using PyGame.

More about my past projects can be found here.

On the last day of every week, I **intend to work on the documentations** adding any significant changes I've made. I plan on using Sphinx for compiling this documentation, and so I will write the comments and docstrings while coding, and will also provide links to other references and blogs which I will read during that period.

These are references list which I read as well as found important - **References List**