Assignment 1

CHANGE DETECTION USING FITTING MODELS

Saikiran Ambati

PSID: 1524311

Abstract:

The Goal of this assignment is to find the best fitting model to detect the changes in a pair of images, the fitting models used for this purpose are line fitting with robust estimators and Gaussian fitting.

Included Open CV 3.0.2 libraries in C++ environment to accomplish this task.

Reading the original images and displaying:

```
imgOriginal = cv::imread("image1.jpg");   //reading an image
cv::namedWindow("imgOriginal", CV_WINDOW_AUTOSIZE); //Creating a window to hold image
cv::imshow("imgOriginal", imgOriginal); // diaplaying an image
```

Calling ‘imread’ function which is defined in open CV library will read the image form the specified file and returns a NULL if image not read properly.

‘namedWindow’ function in open CV lib calls to create a window to hold images.

‘imshow’ displays the image in a window which has been auto sized to fit to the image resolution.

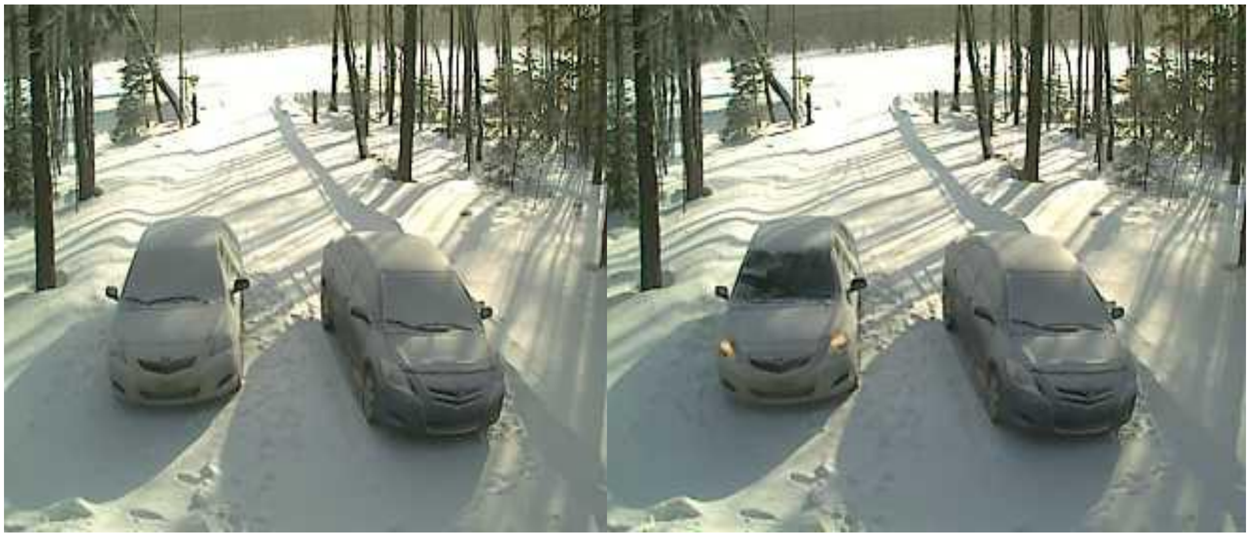


Fig 1: Original RGB images

Writing to Images:

Images read can be saved by calling function ‘imwrite’

Bool imwrite (const string& **filename**, InputArray **img**, const vector<int>& **params**=vector<int> ())

```
imwrite("save1.png", imgOriginal); // saving an image
```

Converting RGB to Gray Scale Images:

```
cv::cvtColor(imgOriginal, imgGrayscale, CV_BGR2GRAY);   // convert to grayscale
```

Converting to Gray scale is essential to capture the pixel information as it ranges in between 0 to 255 ranging in intensity of pixel.

‘cvtColor’, coverts the image form one color space to another, in this case RGB to gray scale.

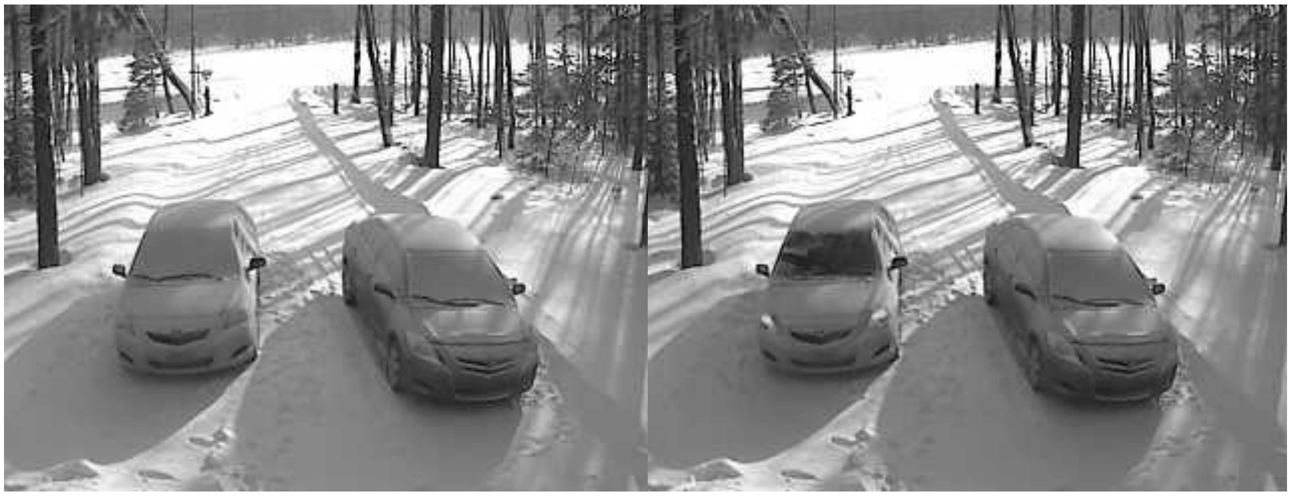


Fig 2: Converted Color Space to Gray Scale

Loading image directly as Gray Scale:

We can directly call image as an intensity one by including

“CV_LOAD_IMAGE_GRAYSCALE”.

```
cv::imread("image1.jpg",CV_LOAD_IMAGE_GRAYSCALE);
//loads the image as an intensity one
```

Considering I_1 & I_2 are the two input images on which we need to find the change in scene. The size of these selected images is 240 x 320. Taking n=240 and m= 320. We define a new matrix which has intensity of pixels of two images as $D = \{d_{11}, d_{12} \ldots, d_{nm}\}$.

such that $d_{ij} = \{I_1(i, j), I_2(i, j)\}$ be a set of data points, where $I_1(i, j)$ is the intensity of the pixel located at (i, j) in image I_1 and I_2

$I_1 and\ I_2$ were grey scale images, $0 \leq I_1(i, j), I_2(i, j) \leq 255\ \forall\ (i, j)$

Where ‘i’ value ranging from 1 to 240, ‘j’ value ranging from 1 to 320

A) FITTING

a) Plotting Data Points on the images:

The two images read as Gray Scale images and stored in a Mat variable, as described earlier we define a matrix D which have the intensity values both the images which can be accessed by calling rows and columns. By running loop for accessing each data pixel in images and later storing in a variable to push back to plot the data points

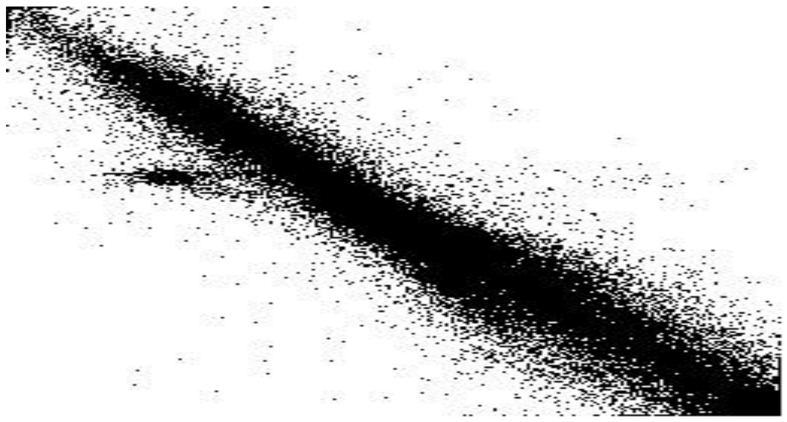


Fig 3: Plotted intensity values of the image.

b) Line fitting for the data points:

Total least squares is a type of errors-in-variables regression, a least squares data modeling technique in which observational errors on both dependent and independent variables are taken into account.

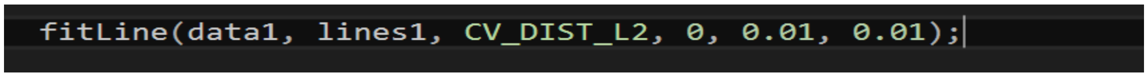
Here the perpendicular distance between each data point and line is calculated and later computing sum of squared distance and then minimizing the line parameters each time and there by repeating the loop until we get the best fit line.

$E = \sum_{i=1}^{n}(ax_i + by_i - d)^2$ minimizing this equation will be our goal to achieve the optimal parameters.

In Open CV, we have a function to obtain this best fit which can compute total least squares to obtain the optimal parameters.

fitLine (InputArray points**, OutputArray** line**, int** distType**, double** param**, double** reps**, double** aeps**)**

param – Numerical parameter (C) for some types of distances. If it is 0, an optimal value is chosen
reps – Sufficient accuracy for the radius (distance between the coordinate origin and the line)
aeps – Sufficient accuracy for the angle. 0.01 would be a good default value for reps and aeps.



```
fitLine(data1, lines1, CV_DIST_L2, 0, 0.01, 0.01);
```

The output line has following parameters (v_x, v_y, x0, y0), where (v_x, v_y) is a normalized vector collinear to the line and (x_0, y_0) is a point on the line. The parameters obtained in this case are (v_x, v_y) the normalized vector**: (0.7014, 0.7127),** point on line (x_0, y_0) is **(142.27, 136.16).**

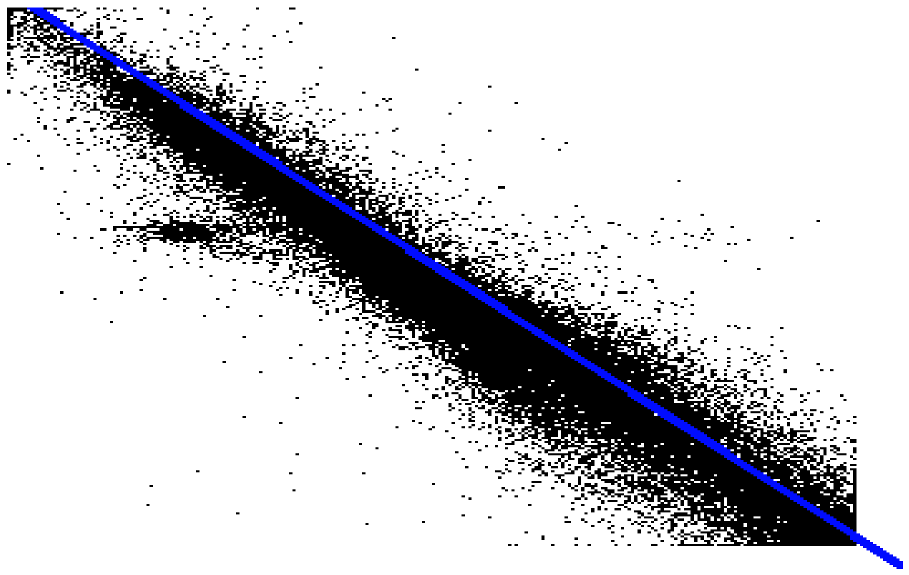


Fig 4: Line fitted to Data Points

c) Fitting a line using Robust Estimators:

Robust estimator is a nonlinear optimization problem that must be solved iteratively to get the best fit model by not considering the data points which are far than a threshold limit. The general approach is to minimize

$$\sum_i \rho(r_i(x_i, \theta); \sigma)$$

r_i (x_i, θ) – residual of ith point w.r.t. model parameters θ
ρ – robust function with scale parameter σ

$$\rho(u;\sigma) = \frac{u^2}{\sigma^2 + u^2}$$

The robust function ρ behaves like squared distance for small values of the residual u but saturates for larger values of u. The effect of outlier is eliminated in robust model i.e. it contributes very less to the least squares. The Least squares solution can be used here as initialization.

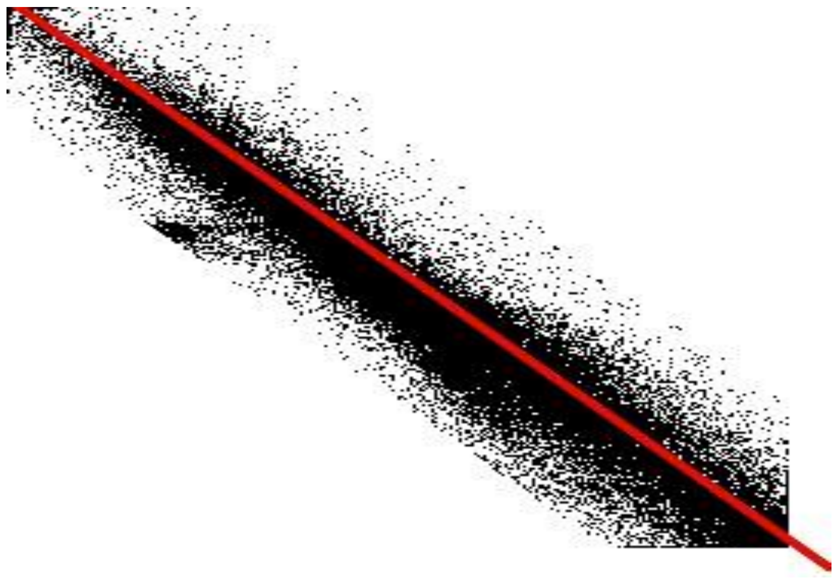


Fig 5: line fitting with robust estimators.

We can clearly see the outlier effect on the least square is completely neglected by setting some threshold. Thus, in robust estimator's method there will be more probability that a given point is an inlier. The parameters obtained in this case are (v_x, v_y) the normalized vector**: (0.7064, 0.7067),** point on line (x_0, y_0) is **(140.24, 139.93).**

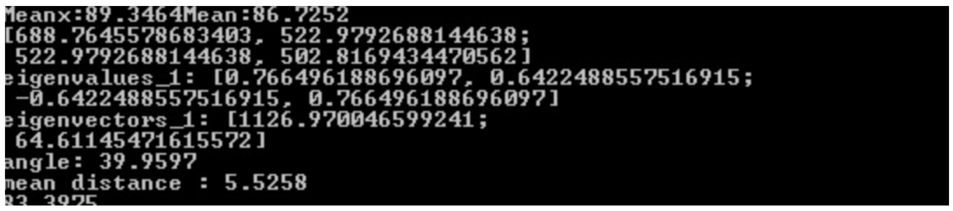
Open CV provides certain parameters for which we can get a best fit line few of them with scales are **HUBER (1.345), Fair (1.3998), Welsch (2.9846**)

d) Fitting data points to a Gaussian Model:

Gaussian functions arise by composing the exponential function with a concave quadratic function. In 2-Dimension, the power to which the exponential is raised in the Gaussian function is any negative – definite quadratic form. Consequently, the levels set of the Gaussian will always be ellipse. In a Gaussian process, every point in some continuous input space is associated with a normally distributed random variable.

In Open CV, to implement this Gaussian model to fit the data points we find the contours of the data points and find the mean of the points to find the density distribution and fit the ellipse to cover maximum of points as inliers and the points that over fit the model are outliers and can be easily found by contour detection.

We will then try to fit ellipse in the model. The parameters of calculated mean and covariance matrix.



```
Meanx:89.3464Mean:86.7252
[688.7645578683403, 522.9792688144638;
 522.9792688144638, 502.8169434470562]
eigenvalues_1: [0.766496188696097, 0.6422488557516915;
 -0.6422488557516915, 0.766496188696097]
eigenvectors_1: [1126.970046599241;
 64.61145471615572]
angle: 39.9597
mean distance : 5.5258
83.3975
```

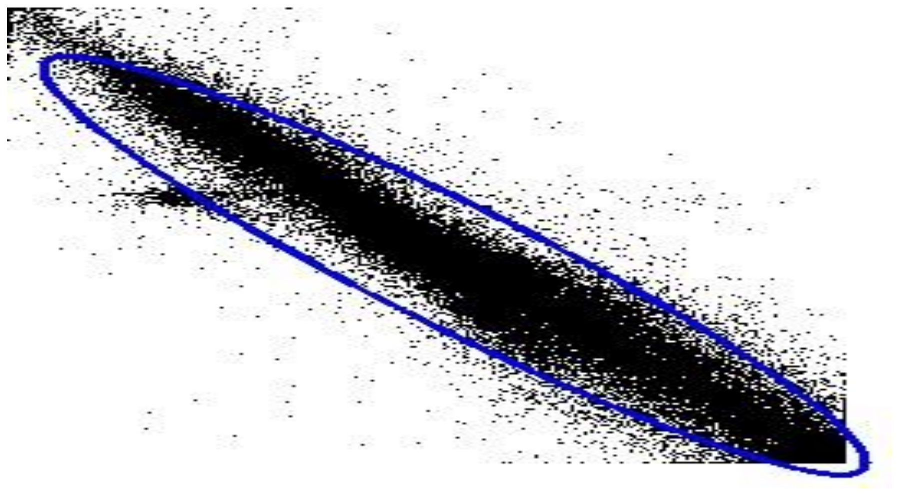


Fig 6: Gaussian ellipse fit to the data

B) CHANGED PIXEL DETECTION

a) Line Model:

In the above Section, we plotted the data points and fitted models for line, robust and Gaussian. Now, we detect the pixels that are far from the line by fixing the threshold.

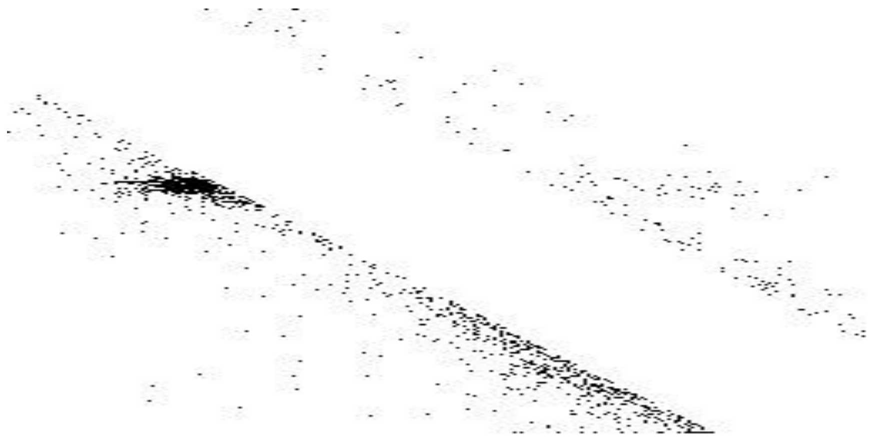


Fig 7: Gaussian ellipse fit to the data

b) Gaussian Model:

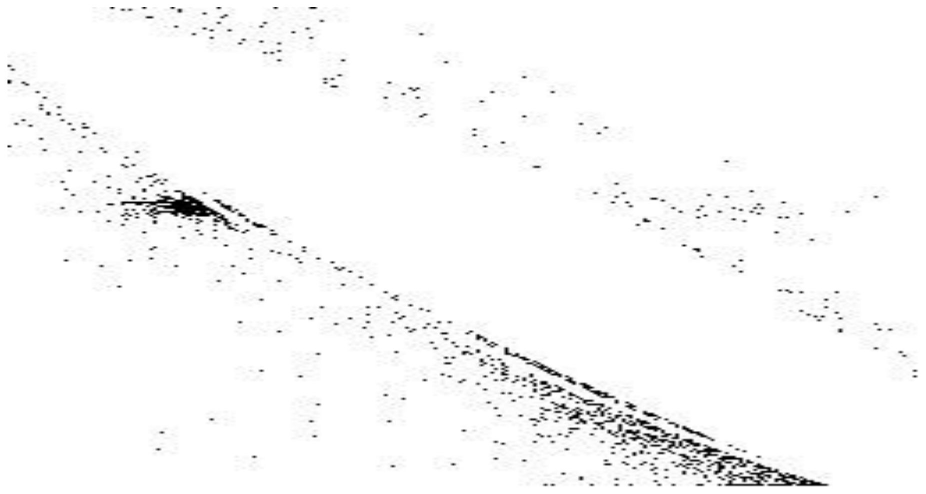


Fig 8: Gaussian ellipse fit to the data

C) SEGMENTATION

a) Segmentation for Line fitting

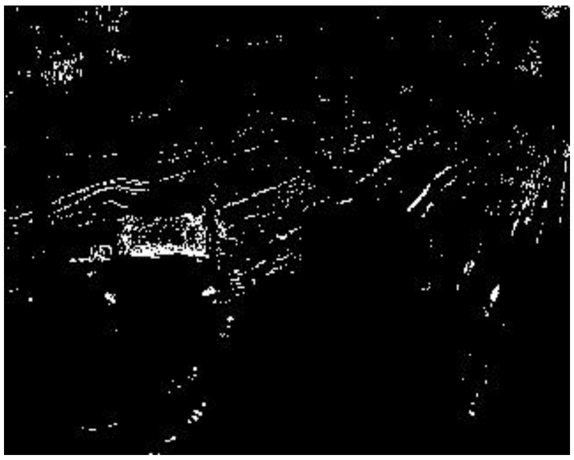


Fig 10 a): Changed pixel from line fitting

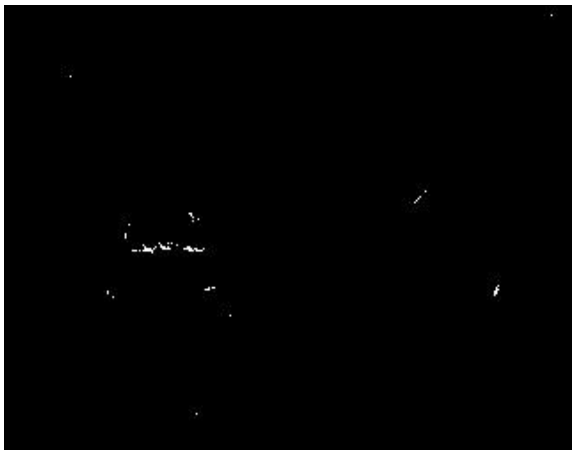


Fig 10 b): noise removal through erosion

b) Segmentation for Guassian fitting

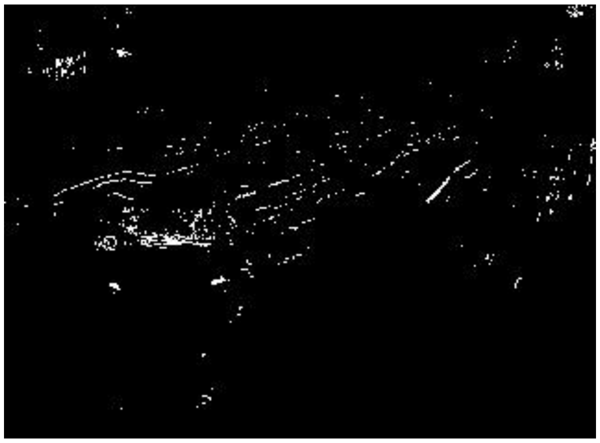


Fig 11 a): Changed pixel from Gaussian fitting

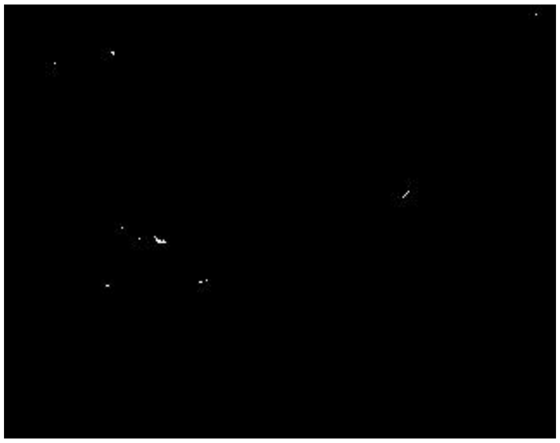


Fig 11 b): noise removal through erosion

Conclusion:

Hence performed the change detection by fitting algorithms, using line parameters and enhanced it by using robust estimators, also performed Gaussian model fit and plotted the changed pixels on image and removed noise by erosion.