POLITECNICO DI MILANO
Facolta di Ingegneria dell'Informazione

POLO REGIONALE DI COMO
Corso di Laurea Specialistica in Ingegneria Informatica

AN EVALUATION OF FEATURE MATCHING STRATEGIES FOR VISUAL TRACKING

Supervisor: Prof. Matteo MATTEUCCI

Master Thesis of: Arash MANSOURI

Matriculation: 735043

Academic Year 2010/2011

Contents

1 Abstract 4
2 Introduction 5
3 Feature Detection and Detectors 6
3.1 Introduction 6
3.2 Definition of a Feature 6
3.2.1 Image Feature Types 7
3.2.2 Feature detector categorization base on their application 8
3.2.3 Properties of a Ideal Local Feature 9
3.3 Harris Detector 12
3.4 Hessian Detector 14
3.5 Kanade-Lucas-Tomasi Detector 16
4 Descriptors 20
4.1 Introduction 20
4.2 SIFT: Scale-invariant feature transform 20
4.3 SURF: Speed Up Robust Features 28
5 Evaluation and Statistics Definitions 29
5.1 Epipolar Geometry and the Fundamental Matrix 29
5.2 Planar Homographies 32
5.3 Generalized Extreme Value Distributions 36
5.4 Goodness of Fit Test 38
6 Proposed Methods 41
6.1 Tracking Methods 41
6.2 Tests Structure 47
6.3 Result Evaluation Structure 50
7 Numerical Test Results 53
7.1 Set A 53
7.1.1 1st Test – Total Point to Point 53
7.1.2 2nd Test –Total Point to Point with Euclidean distance comparison 58
7.1.3 3rd Test - Region of interest (ROI-Points) 61
7.1.4 4th Test - ROI with Scale comparison 66

7.1.5 5th Test - ROI group points 68
7.2 Set B 70
7.2.1 6th Test – Total Point to Point-2frames 71
7.2.2 7th Test - ROI Points 2frames 76
7.3 Set C 80
8 Discussion, Conclusion and Future Development 81
9 Bibilography 82

1 Abstract

2 Introduction

3 Feature Detection and Detectors

3.1 Introduction

In this chapter, the definition and algorithm of detectors which are used in this work will be explained. First an introduction to the definition of feature is given and then "Kanade-Lucas-Tomasi" detector, "Harris detector" and "Fast-Hessian" detector which are used in this work will be simply presented.

3.2 Definition of a Feature

Let's say that there is not a certain and exact definition for what constitutes a feature. This definition is strongly related to the problem and type of the image application. But generally it could be defined that a feature illustrates "interesting" part of an image and they are used as a starting point for many computer vision algorithms. Feature detection is a low-level image processing operation and they are used as the starting point and main primitives for subsequent algorithms so it shows how much this starting part could be essential for overall algorithm. Many computer vision algorithms use feature detection as the initial step, so as a result, a very large number of feature detectors have been developed. These vary widely in the kinds of feature detected, the computational complexity and the repeatability.

A local feature is an image pattern which differs from its immediate neighborhood. It is usually associated with a change of an image property or several properties simultaneously, although it is not necessarily localized exactly on this change. The image properties commonly considered are intensity, color, and texture. Figure below shows some examples of local features in a contour image (left) as well as in a gray value image (right). Local features can be points, but also edgels or small image patches. Typically, some measurements are taken from a region centered on a local feature and converted into descriptors. The descriptors can then be used for various applications.

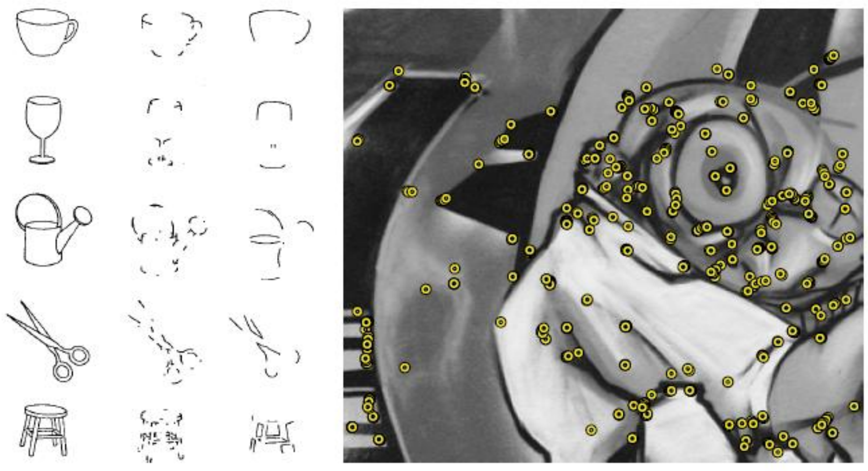


Figure. Importance of corners and junctions in visual recognition and an image example with interest points provided by a corner detector

3.2.1 Image Feature Types

We could sort some type of image feature as following:

Edges

Edges are points where there is a boundary (or an edge) between two image regions. In general, an edge can be of almost arbitrary shape, and may include junctions. In practice, edges are usually defined as sets of points in the image which have a strong gradient magnitude.

Corners

The terms corners and interest points are used somewhat interchangeably and refer to point-like features in an image, which have a local two dimensional structure. The name "Corner" arose since early algorithms first performed edge detection, and then analyzed the edges to find rapid changes in direction (corners). These algorithms were then developed so that explicit edge detection was no longer required, for instance by looking for high levels of curvature in the image gradient. It was then noticed that the so-called corners were also being detected on parts of the image which were not corners in the traditional sense (for instance a small bright spot on a dark background may be detected). These points are frequently known as interest points, but the term "corner" is used by tradition.

Blobs / regions of interest or interest points

Blobs provide a complementary description of image structures in terms of regions, as opposed to corners that are more point-like. Nevertheless, blob descriptors often contain a preferred point (a local maximum of an operator response or a center of gravity) which means that many blob detectors may also be regarded as interest point operators. Blob detectors can detect areas in an image which are too smooth to be detected by a corner detector.

3.2.2 Feature detector categorization base on their application

In the following, we distinguish three broad categories of feature detectors based on their possible usage. It is not exhaustive or the only way of categorizing the detectors but it emphasizes different properties required by the usage scenarios:

1. One might be interested in a specific type of local features, as they may have a specific semantic interpretation in the limited context of a certain application. For instance, edges detected in aerial images often correspond to roads; blob detection can be used to identify impurities in some inspection task; etc. These were the first applications for which local feature detectors have been proposed.
2. One might be interested in local features since they provide a limited set of well localized and individually identifiable anchor points. What the features actually represent is not really relevant, as long as their location can be determined accurately and in a stable manner over time. This is for instance the situation in most matching or tracking applications, and especially for camera calibration or 3D reconstruction.

3. A set of local features can be used as a robust image representation that allows recognizing objects or scenes without the need for segmentation. Here again, it does not really matter what the features actually represent. They do not even have to be localized precisely, since the goal is not to match them on an individual basis, but rather to analyze their statistics.

So as it was mentioned, each of the above three categories imposes its own constraints, and a good feature for one application may be useless in the context of a different problem.

Notice: Interest Point and Local Feature

In a way, the ideal local feature would be a point as defined in geometry: having a location in space but no spatial extent. In practice however, images are discrete with the smallest spatial unit being a pixel and discretization effects playing an important role. To localize features in images, a local neighborhood of pixels needs to be analyzed, giving all local features some implicit spatial extent. For some applications (e.g., camera calibration or 3D reconstruction) this spatial extent is completely ignored in further processing, and only the location derived from the feature extraction process is used (with the location sometimes determined up to sub-pixel accuracy). In those cases, one typically uses the term interest point.

3.2.3 Properties of a Ideal Local Feature

Followings are some properties of a Local Feature:

- Repeatability: Given two images of the same object or scene, taken under different viewing conditions, a high percentage of the features detected on the scene part visible in both images should be found in both images.
- Distinctiveness/in formativeness: The intensity patterns underlying the detected features should show a lot of variation, such that features can be distinguished and matched.

- Locality: The features should be local, so as to reduce the probability of occlusion and to allow simple model approximations of the geometric and photometric deformations between two images taken under different viewing conditions (e.g., based on a local planarity assumption).
- Quantity: The number of detected features should be sufficiently large, such that a reasonable number of features are detected even on small objects. However, the optimal number of features depends on the application. Ideally, the number of detected features should be adaptable over a large range by a simple and intuitive threshold. The density of features should reflect the information content of the image to provide a compact image representation.
- Accuracy: The detected features should be accurately localized, both in image location, as with respect to scale and possibly shape.
- Efficiency: Preferably, the detection of features in a new image should allow for time-critical applications.

Repeatability, arguably the most important property of all, can be achieved in two different ways: either by invariance or by robustness.

- Invariance: When large deformations are to be expected, the preferred approach is to model these mathematically if possible, and then develop methods for feature detection that are unaffected by these mathematical transformations.
- Robustness: In case of relatively small deformations, it often suffices to make feature detection methods less sensitive to such deformations, i.e., the accuracy of the detection may decrease, but not drastically so. Typical deformations that are tackled using robustness are image noise, discretization effects, compression artifacts, blur, etc. Also geometric and photometric deviations from the mathematical model used to obtain invariance are often overcome by including more robustness.

Clearly, the importance of these different properties depends on the actual application and settings, and compromises need to be made.

Repeatability is required in all application scenarios and it directly depends on the other properties like invariance, robustness, quantity etc. Depending on the application increasing or decreasing them may result in higher repeatability.

Distinctiveness and locality are competing properties and cannot be fulfilled simultaneously: the more local a feature, the less information is available in the underlying intensity pattern and the harder it becomes to match it correctly, especially in database applications where there are many candidate features to match to. On the other hand, in case of planar objects and/or purely rotating cameras, images are related by a global homography, and there are no problems with occlusions or depth discontinuities. Under these conditions, the size of the local features can be increased without problems, resulting in a higher distinctiveness.

Similarly, an increased level of *invariance* typically leads to a reduced distinctiveness, as some of the image measurements are used to lift the degrees of freedom of the transformation. A similar rule holds for *robustness* versus distinctiveness, as typically some information is disregarded (considered as noise) in order to achieve robustness. As a result, it is important to have a clear idea on the required level of *invariance* or robustness for a given application. It is hard to achieve high invariance and robustness at the same time and invariance, which is not adapted to the application, may have a negative impact on the results.

Accuracy is especially important in wide baseline matching, registration, and structure from motion applications, where precise correspondences are needed to, e.g., estimate the epipolar geometry or to calibrate the camera setup.

Quantity is particularly useful in some class-level object or scene recognition methods, where it is vital to densely cover the object of interest. On the other hand, a high number of features have in most cases a negative impact on the computation time and it should be kept within limits. Also robustness is essential for object class recognition, as it is impossible to model the intra-class variations mathematically, so full invariance is impossible. For these applications, an accurate localization is less important. The effect of inaccurate localization of a feature detector can be countered, up to some point, by having an extra robust descriptor, which yields a feature vector that is not affected by small localization errors.

3.3 Harris Detector

The Harris detector, proposed by Harris and Stephens, is based on the second moment matrix, also called the auto-correlation matrix, which is often used for feature detection and for describing local image structures. This matrix describes the gradient distribution in a local neighborhood of a point:

$$\boldsymbol{M} = \boldsymbol{б}_{\boldsymbol{D}}^{2}\boldsymbol{g}(\boldsymbol{б}_{\boldsymbol{1}}) * \begin{bmatrix} \boldsymbol{I}_{\boldsymbol{x}}^{2}(\boldsymbol{x},\boldsymbol{б}_{\boldsymbol{D}}) & \boldsymbol{I}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{б}_{\boldsymbol{D}})\boldsymbol{I}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{б}_{\boldsymbol{D}}) \\ \boldsymbol{I}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{б}_{\boldsymbol{D}})\boldsymbol{I}_{\boldsymbol{y}}(\boldsymbol{x},\boldsymbol{б}_{\boldsymbol{D}}) & \boldsymbol{I}_{\boldsymbol{y}}^{2}(\boldsymbol{x},\boldsymbol{б}_{\boldsymbol{D}}) \end{bmatrix}$$

With

$$\boldsymbol{I}_{\boldsymbol{x}}(\boldsymbol{x},\boldsymbol{б}_{\boldsymbol{D}}) = \frac{\boldsymbol{\partial}}{\boldsymbol{\partial}_{\boldsymbol{x}}}\, \boldsymbol{g}\,(\boldsymbol{б}_{\boldsymbol{D}}) * \boldsymbol{I}\,(\boldsymbol{x})$$

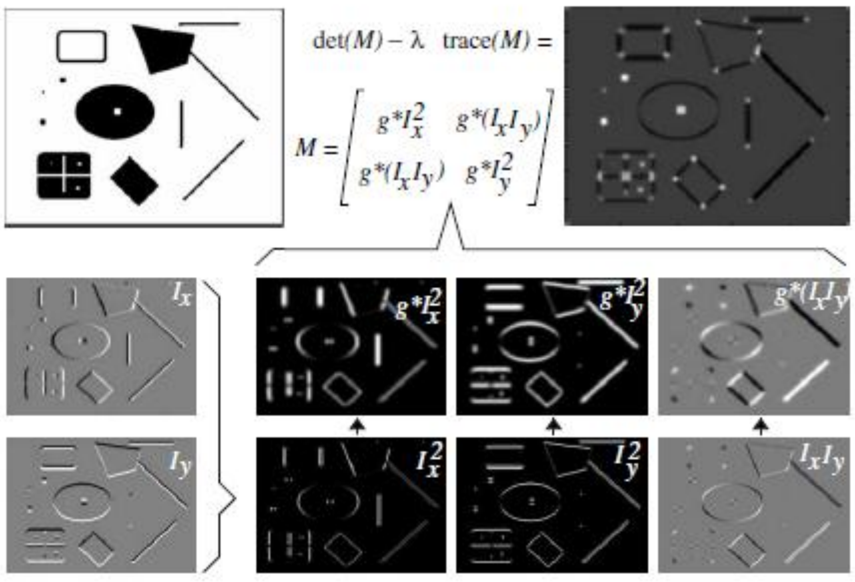
$$\mathrm{g}(б) = \frac{1}{2\pi б^{2}}\, \mathrm{e}^{-\frac{x^{2}+y^{2}}{2б^{2}}}$$

The local image derivatives are computed with Gaussian kernels of scale $\boldsymbol{б}_{\boldsymbol{D}}$ (the differentiation scale). The derivatives are then averaged in the neighborhood of the point by smoothing with a Gaussian window of scale $\boldsymbol{б}_{\boldsymbol{I}}$ (the integration scale). The eigen values of this matrix represent the principal signal changes in two orthogonal directions in a neighborhood around the point defined by $\boldsymbol{б}_{\boldsymbol{I}}$. Based on this property, corners can be found as locations in the image for which the image signal varies significantly in both directions, or in other words, for which both eigen values are large. In practice, Harris proposed to use the following measure for cornerness, which combines the two eigen values in a single measure and is computationally less expensive:

$$\boldsymbol{cornerness} = \boldsymbol{det}\,(\boldsymbol{M}) - \boldsymbol{\lambda}\, \boldsymbol{trace}(\boldsymbol{M})$$

with $det(M)$ the determinant and $trace(M)$ the trace of the matrix M. A typical value for λ is 0.04. Since the determinant of a matrix is equal to the product of its eigen values and the trace corresponds to the sum, it is clear that high values of the *cornerness* measure correspond to both eigenvalues being large. Adding the second term with the trace reduces the response of the operator on strong straight contours.

Subsequent stages of the corner extraction process are illustrated in Figure :



$\det(M) - \lambda \; \mathrm{trace}(M) =$
$M = \begin{bmatrix} g*I_x^2 & g*(I_xI_y) \\ g*(I_xI_y) & g*I_y^2 \end{bmatrix}$
I_x
I_y
$g*I_x^2$
$g*I_y^2$
$g*(I_xI_y)$
I_x^2
I_y^2
I_xI_y

Fig. Illustration of the components of the second moment matrix and Harris cornerness measure.

Figure ?? shows the corners detected with this measure for two example images related by a rotation. Note that the features found correspond to locations in the image showing two dimensional variations in the intensity pattern. These may correspond to real "corners", but the detector also fires on other structures, such as T-junctions, points with high curvature and etc.

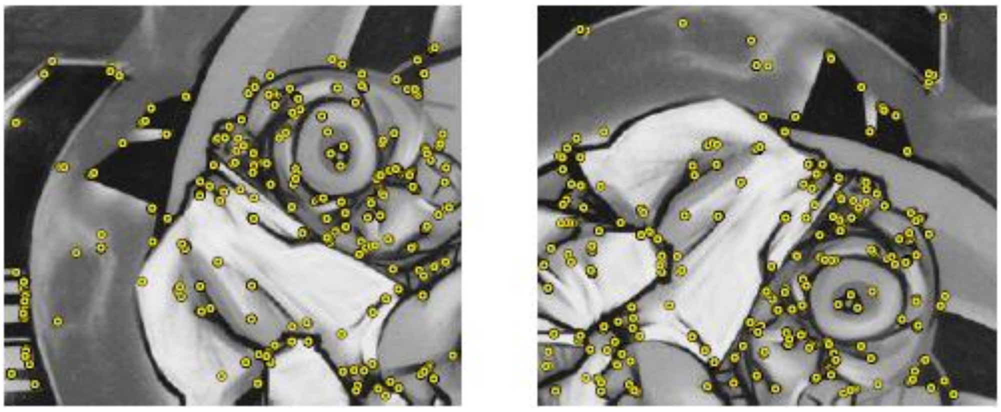


Fig. Harris corners detected on rotated image examples.

As can be seen in the figure, many but not all of the features detected in the original image (left) have also been found in the rotated version (right). In other words, the repeatability of the Harris detector under rotations is high. Additionally, features are typically found at locations which are informative, i.e., with a high variability in the intensity pattern. This makes them more discriminative and easier to bring into correspondence.

3.4 Hessian Detector

The second 2×2 matrix issued from the Taylor expansion of the image intensity function $I(x)$ is the Hessian matrix:

$$H = \begin{bmatrix} I_{xx}(x, \boldsymbol{\sigma_D}) & I_{xy}(x, \boldsymbol{\sigma_D}) \\ I_{xy}(x, \boldsymbol{\sigma_D}) & I_{yy}(x, \boldsymbol{\sigma_D}) \end{bmatrix}$$

with I_{xx} etc. second-order Gaussian smoothed image derivatives. These encode the shape information by describing how the normal to an isosurface changes. As such, they capture important properties of local image structure. Particularly interesting are the filters based on the determinant and the trace of this matrix. The latter is often referred to as the Laplacian. Local maxima of both measures can be used to detect blob-like structures in an image The Laplacian is a separable linear filter and can be approximated efficiently with a Difference of Gaussians (DoG) filter. The Laplacian filters have one

major drawback in the context of blob extraction though. Local maxima are often found near contours or straight edges, where the signal change is only in one direction.
These maxima are less stable because their localization is more sensitive to noise or small changes in neighboring texture. This is mostly an issue in the context of finding correspondences for recovering image transformations. A more sophisticated approach, solving this problem, is to select a location and scale for which the trace and the determinant of the Hessian matrix simultaneously assume a local extremum.

This gives rise to points, for which the second order derivatives detect signal changes in two orthogonal directions. A similar idea is explored in the Harris detector, albeit for first-order derivatives only.
The feature detection process based on the Hessian matrix is illustrated in Figure ????. Given the original image (upper left), one first computes the second-order Gaussian smoothed image derivatives (lower part), which are then combined into the determinant of the Hessian (upper right).

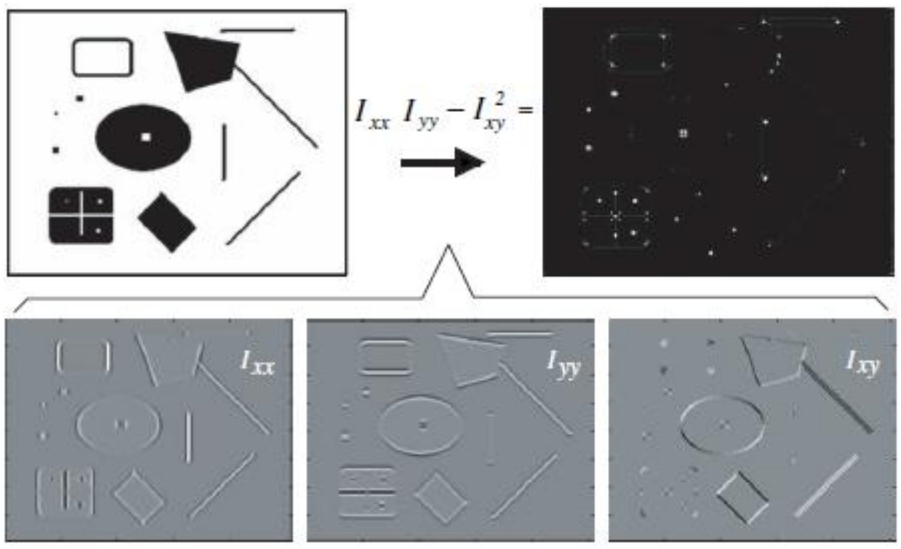


Fig. Illustration of the components of the Hessian matrix and Hessian determinant.

The interest points detected with the determinant of the Hessian for an example image pair are displayed in Figure ????. The second-order derivatives are symmetric filters, thus they give weak responses exactly in the point where the signal change is most significant. Therefore, the maxima are localized at ridges and blobs for which the size of the Gaussian kernel σ_D matches by the size of the blob structure.

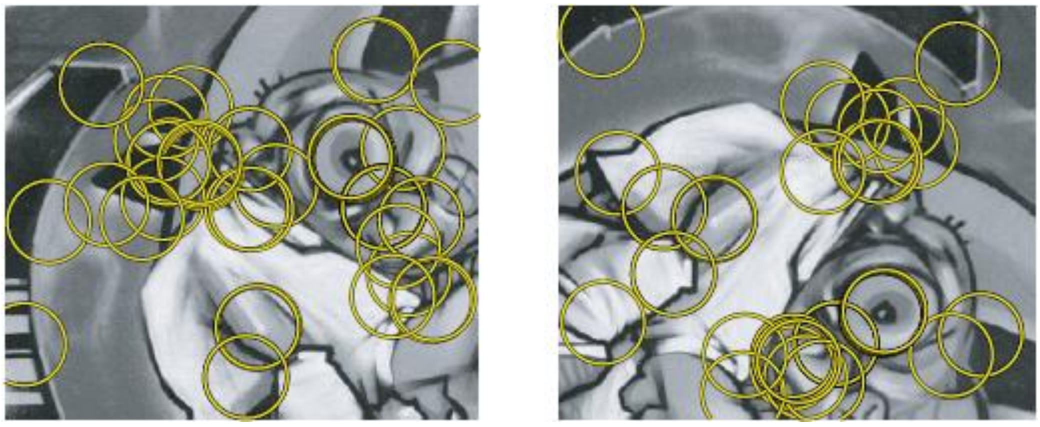


Fig. Output of the Hessian detector applied at a given scale to example images with rotation (subset).

3.5 Kanade-Lucas-Tomasi Detector

The KLT algorithm is one of the selected detectors which are used in this work. In this section the definition of KLT feature detector is explained in detail but briefly, good features are located by examining the minimum eigenvalue of each 2 by 2 gradient matrix. In this method, it was shown how to monitor the quality of image features by using a measure of feature dissimilarity that quantifies the change of appearance of a feature between the first and the current frame. The idea is straightforward: dissimilarity is the feature’s rms residue between the first and the current frame and when dissimilarity grows too large the feature should be abandoned.

Basic requirements for KLT

As the camera moves, the patterns of image intensities change in a complex way. These changes can often be described as image motion:

$$I(x, y, t + \tau) = I\left(x - \xi(x, y, t, \tau), y - \eta(x, y, t, \tau)\right)$$

Thus,a later image taken at time $t + \tau$ can be obtained by moving every point in the current image, taken at time t, by a suitable amount. The amount of motion $\delta = (\xi, \eta)$ is called the displacement of the point at $= (x, y)$.
An affine motion field is a better representation:

$$\delta = DX + d$$

Where

$$D = \begin{bmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{bmatrix}$$

is a deformation matrix, and d is the translation of the feature window's center.The image coordinates x are measured with respect to the window's center. Then,a point x in the first image I moves to point $Ax + d$ in the second image J,where $A = 1 + D$ and 1 is the 2×2 identity matrix:

$$\mathrm{J(Ax + d) = I\,(x)}$$

Because of image noise and as the affine motion model is not perfect, above equation $\mathrm{J(Ax + d) = I\,(x)}$ is in general not satisfied exactly. The problem of determining the motion parameters is then that of finding the A and d that minimize the dissimilarity:

$$\varepsilon = \iint_W [J(Ax + d) - I(x)]^2 \, w(x) dx$$

Where W is the given feature window and w(x) is a weighting function. To minimize the residual of above equation we should differentiate it with respect to the unknown entries of the deformation matrix D and the displacement vector d and set the result to zero. We then linearize the resulting system by the truncated Taylor expansion:

$$J(Ax + d) = J(x) + g^T(u)$$

This yields the following linear 6×6 system:

$$Tz = a$$

Where $z^T = [d_{xx} \quad d_{yx} \quad d_{xy} \quad d_{yy} \quad d_x \quad d_y]$ collects the entries of the deformation D and displacement d.

T can be written as :

$$T = \iint_w \begin{bmatrix} U & V \\ V^T & Z \end{bmatrix} wdx$$

Z would be vital for our next calculation in good feature detection of KLT. For further information on U and V please refer to the reference paper.

$$\mathrm{Z} = \begin{bmatrix} \mathrm{g}_\mathrm{x}^2 & \mathrm{g}_\mathrm{x}\mathrm{g}_\mathrm{y} \\ \mathrm{g}_\mathrm{x}\mathrm{g}_\mathrm{y} & \mathrm{g}_\mathrm{y}^2 \end{bmatrix}$$

In the KLT algorithm, they propose a more principled definition of feature quality. For selecting the reliable feature the symmetric matrix Z of the system must be both above the image noise level and well-conditioned.

The noise requirement implies that both eigen values of Z must be large, while the conditioning requirement means that they cannot differ by several orders of magnitude. Two small eigen values mean a roughly constant intensity profile within a window. A large and a small eigen value correspond to a unidirectional texture pattern. Two large eigen values can represent corners, salt-and-pepper textures, or any other pattern that can be tracked reliably.

In practice, when the smaller eigen value is sufficiently large to meet the noise criterion, the matrix Z is usually also well conditioned. In fact, the intensity variations in a window are bounded by the maximum allowable pixel value. so that the greater eigenvalue cannot be arbitrarily large. In conclusion, if the two eigenvalues of Z are $\lambda_1$and λ_2,we accept a window if

$$\min(\lambda_1, \lambda_2) > \lambda$$

where λ is a predefined threshold.

It would be nice if we notice that a feature with a high texture content, as defined in the above equation, can still be a bad feature to track.For instance,in an image of a tree, a horizontal twig in the foreground can intersect a vertical twig in the background. This intersection occurs only in the image- not in the world- since the two twigs are at different depths. Any selection criterion would pick the intersection as a good feature to track, and yet there is no real world feature there to speak of. The measure of dissimilarity defined in equation $\varepsilon = \iint_{w} [J(Ax + d) - I(x)]^2 \, w(x) dx$ can often indicate that something is going wrong. Because of the potentially large number of frames through which a given feature can be tracked, the dissimilarity measure would not work well with a pure translation model.

4 Descriptors

4.1 Introduction

Interesting points on the object can be extracted to provide a "feature description" of the object. This description, extracted from a training image, can then be used to identify the object when attempting to locate the object in a test image containing many other objects. Image matching is a fundamental aspect of many problems in computer vision, including object or scene recognition, solving for 3D structure from multiple images, stereo correspondence, and motion tracking. A “Descriptor” describes image features that have many properties that make them suitable for matching differing images of an object or scene. That features could be invariant to image scaling and rotation, and partially invariant to change in illumination and 3D camera viewpoint. To perform reliable recognition, it is important that the features extracted from the training image are detectable even under changes in image scale, noise and illumination. Such points usually lie on high-contrast regions of the image, such as object edges. More than above, relative positions between them in the original scene shouldn't change from one image to another. For example, if only the four corners of a door were used as features, they would work regardless of the door's position; but if points in the frame were also used, the recognition would fail if the door is opened or closed. Similarly, features located in articulated or flexible objects would typically not work if any change in their internal geometry happens between two images in the set being processed.

The following section will introduce two powerful descriptors - Scale-invariant feature transform and speed up robust features- in detail. The speed up robust features (SURF) is one which is implemented and used in this work as a descriptor.

4.2 SIFT: Scale-invariant feature transform

According to David Lowe, Following are the major stages of computation used to generate the set of image features:

1. **Scale-space extrema detection:** The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-

of-Gaussian function to identify potential interest points that are invariant to scale and orientation.

2. **Keypoint localization:** At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability.
3. **Orientation assignment:** One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.
4. **Keypoint descriptor:** The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination.

This approach has been named the Scale Invariant Feature Transform (SIFT), as it transforms image data into scale-invariant coordinates relative to local features.

Detection of scale-space extrema

The first stage of keypoint detection is to identify locations and scales that can be assigned in a repeatable manner under differing views of the same object. The scale space of an image is defined as a function, $L(x, y, \sigma)$, that is produced from the convolution of a variable-scale Gaussian, $G(x, y, \sigma)$, with an input image, $I(x, y)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

where * is the convolution operation in x and y, and

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

Using scale-space extrema in the difference-of-Gaussian function convolved with the image, $D(x, y, \sigma)$:

$$D(x, y, \sigma) = \left(G(x, y, k\sigma) - G(x, y, \sigma)\right) * I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

From above equation we could reach to following approximation (the proof is described in the reference [Distinctive Image Features from Scale-Invariant Keypoints,by david Lowe]) :

$$G(x,y,k\sigma) - G(x,y,\sigma) \approx (k-1)\sigma^2\nabla^2 G$$

Where $\sigma^2\nabla^2 G$ is scale-normalized Laplacian of Gaussian. This shows that when the difference-of-Gaussian function has scales differing by a constant factor it already incorporates the σ^2 scale normalization required for the scale-invariant Laplacian. The factor $(k - 1)$ in the equation is a constant over all scales and therefore does not influence extrema location. An efficient approach to construction of $D(x,y,\sigma)$ is shown in Figure ???. The initial image is incrementally convolved with Gaussians to produce images separated by a constant factor k in scale space, shown stacked in the left column.

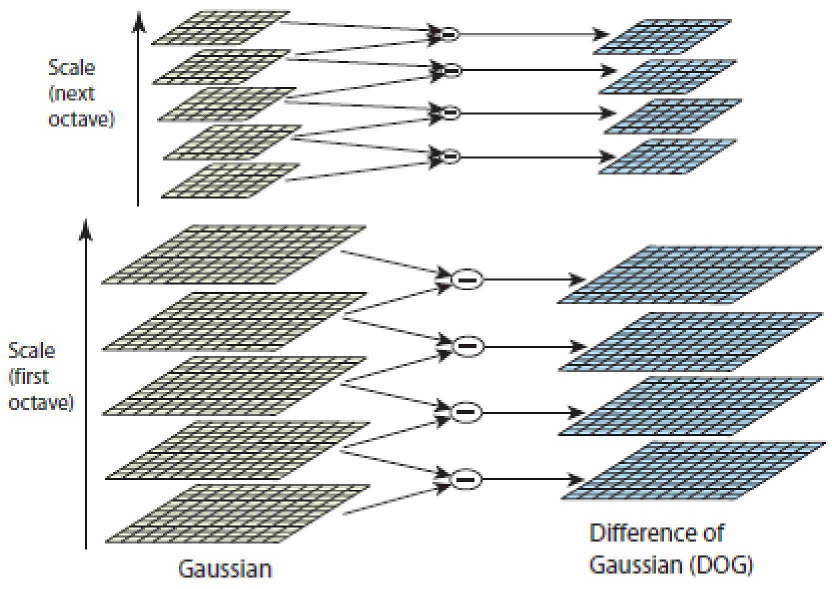


Figure ????: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

Local extrema detection

In order to detect the local maxima and minima of $D(x, y, \sigma)$, each sample point is compared to its eight neighbors in the current image and nine neighbors in the scale above and below. It is selected only if it is larger than all of these neighbors or smaller than all of them. The cost of this check is reasonably low due to the fact that most sample points will be eliminated following the first few checks. (see figure 2???)

An important issue is to determine the frequency of sampling in the image and scale a domain that is needed to reliably detect the extrema. Unfortunately, it turns out that there is no minimum spacing of samples that will detect all extrema, as the extrema can be arbitrarily close together. Therefore, we must settle for a solution that trades off efficiency with completeness.

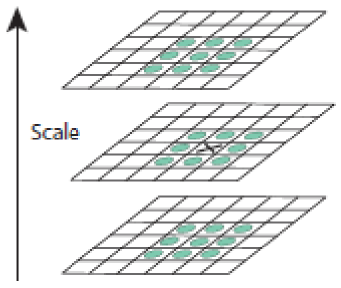


Figure 2: Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3x3 regions at the current and adjacent scales (marked with circles).

Accurate keypoint localization

Once a keypoint candidate has been found by comparing a pixel to its neighbors, the next step is to perform a detailed fit to the nearby data for location, scale, and ratio of principal curvatures. This information allows points to be rejected that have low contrast (and are therefore sensitive to noise) or are poorly localized along an edge.

Implementing of the above approach leads to

$$D(x) = D + \frac{\partial D^T}{\partial x} x + \frac{1}{2} x^T \frac{\partial^2 D}{\partial x^2} x$$

where D and its derivatives are evaluated at the sample point and $x = (x, y, \sigma)^T$ is the offset from this point. The location of the extremum, $\hat{x}$, is determined by taking the derivative of this function with respect to x and setting it to zero, giving

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \frac{\partial D}{\partial D}$$

The function value at the extremum, $D(\hat{x})$, is useful for rejecting unstable extrema with low contrast. This can be obtained by substituting two above, giving

$$D(\hat{x}) = D + \frac{1}{2} \frac{\partial D^T}{\partial x} \hat{x}$$

Eliminating edge responses

For stability, it is not sufficient to reject keypoints with low contrast. The difference-of-Gaussian function will have a strong response along edges, even if the location along the edge is poorly determined and therefore unstable to small amounts of noise.

The first step here, is computing The principal curvatures from a 2x2 Hessian matrix, H, computed at the location and scale of the keypoint:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{yx} & D_{yy} \end{bmatrix}$$

The derivatives are estimated by taking differences of neighboring sample points. Now, Let α be the eigenvalue with the largest magnitude and β be the smaller one. Then, we can compute the sum of the eigenvalues from the trace of **H** and their product from the determinant:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$

$$Det(H) = D_{xx} D_{yy} - \left(D_{xy}\right)^2 = \alpha\beta$$

Now, Let r be the ratio between the largest magnitude eigenvalue and the smaller one, so that $\alpha = r\beta$. Then by some calculation we could reach to:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}$$

So it shows that this quantity is at a minimum when the two eigenvalues are equal and it increases with r.

Orientation assignment

By assigning a consistent orientation to each keypoint based on local image properties, the keypoint descriptor can be represented relative to this orientation and therefore achieve invariance to image rotation. This approach contrasts with the orientation invariant descriptors of Schmid andMohr (1997), in which each image property is based on a rotationally invariant measure. The disadvantage of that approach is that it limits the descriptors that can be used and discards image information by not requiring all measures to be based on a consistent rotation.

Following experimentation with a number of approaches to assigning a local orientation, the following approach was found to give the most stable results. The scale of the keypoint is used to select the Gaussian smoothed image, L, with the closest scale, so that all computations are performed in a scale-invariant manner. For each image sample, $L(x, y)$, at this scale, the gradient magnitude, $m(x, y)$, and orientation, $\theta(x, y)$, is pre-computed using pixel differences:

$$m(x,y) = \sqrt{\left(L(x+1,y) - L(x-1,y)\right)^2 + \left(L(x,y+1) - L(x,y-1)\right)^2}$$

$$\theta(x,y) = \tan^{-1}\left(\left(L(x+1,y) - L(x-1,y)\right)^2 + \left(L(x,y+1) - L(x,y-1)\right)^2\right)$$

The local image descriptor

The previous operations have assigned an image location, scale, and orientation to each keypoint. These parameters impose a repeatable local 2D coordinate system in which to describe the local image region, and therefore provide invariance to these parameters. The next step is to compute a descriptor for the local image region that is highly distinctive yet is as invariant as possible to remaining variations, such as change in illumination or 3D viewpoint. One obvious approach would be to sample the local image intensities around the keypoint at the appropriate scale, and to match these using a normalized correlation measure. However, simple correlation of image patches is highly sensitive to changes that cause misregistration of samples, such as affine or 3D viewpoint change or non-rigid deformations. A better approach has been demonstrated by Edelman, Intrator, and Poggio (1997). Their proposed representation was based upon a model of biological vision, in particular of complex neurons in primary visual cortex. These complex neurons respond to a gradient at a particular orientation and spatial frequency, but the location of the gradient on the retina is allowed to shift over a small receptive field rather than being precisely localized. Edelman *et al.* hypothesized that the function of these complex neurons was to allow for matching and recognition of 3D objects from a range of viewpoints. They have performed detailed experiments using 3D computer models of object and animal shapes which show that matching gradients while allowing for shifts in their position results in much better classification under 3D rotation.

Descriptor representation

First the image gradient magnitudes and orientations are sampled around the keypoint location, using the scale of the keypoint to select the level of Gaussian blur for the image. In order to achieve orientation invariance, the coordinates of the descriptor and the gradient orientations are rotated relative to the keypoint orientation. A Gaussian weighting function with α equal to one half the width of the descriptor window is used to assign a weight to the magnitude of each sample point. The purpose of this Gaussian window is to avoid sudden changes in the descriptor with small changes in the position of the window, and to give less emphasis to gradients that are far from the center of the descriptor, as these are most affected by misregistration errors.

The keypoint descriptor allows for significant shift in gradient positions by creating orientation histograms over 4x4 sample regions. It is important to avoid all boundary affects in which the descriptor abruptly changes as a sample shifts smoothly from being within one histogram to another or from one orientation to another. Therefore, trilinear interpolation is used to distribute the value of each gradient sample into adjacent histogram bins. In other words, each entry into a bin is multiplied by a weight of $1 - d$ for each dimension, where d is the distance of the sample from the central value of the bin as measured in units of the histogram bin spacing.

The descriptor is formed from a vector containing the values of all the orientation histogram entries, corresponding to the lengths of the arrows.

Finally, the feature vector is modified to reduce the effects of illumination change. First, the vector is normalized to unit length. A change in image contrast in which each pixel

value is multiplied by a constant will multiply gradients by the same constant, so this contrast change will be canceled by vector normalization. A brightness change in which a constant is added to each image pixel will not affect the gradient values, as they are computed from pixel differences. Therefore, the descriptor is invariant to affine changes in illumination. However, non-linear illumination changes can also occur due to camera saturation or due to illumination changes that affect 3D surfaces with differing orientations by different amounts. These effects can cause a large change in relative magnitudes for some gradients, but are less likely to affect the gradient orientations.
You can find all the above explanation in figure ???:

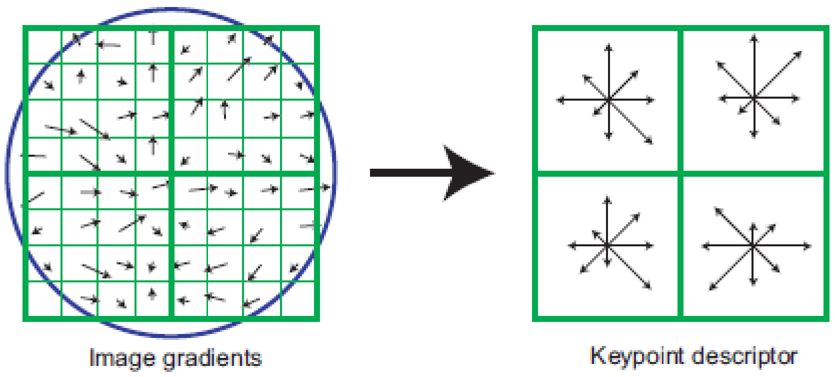


Figure ???: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradientmagnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in this paper use 4x4 descriptors computed from a 16x16 sample array.

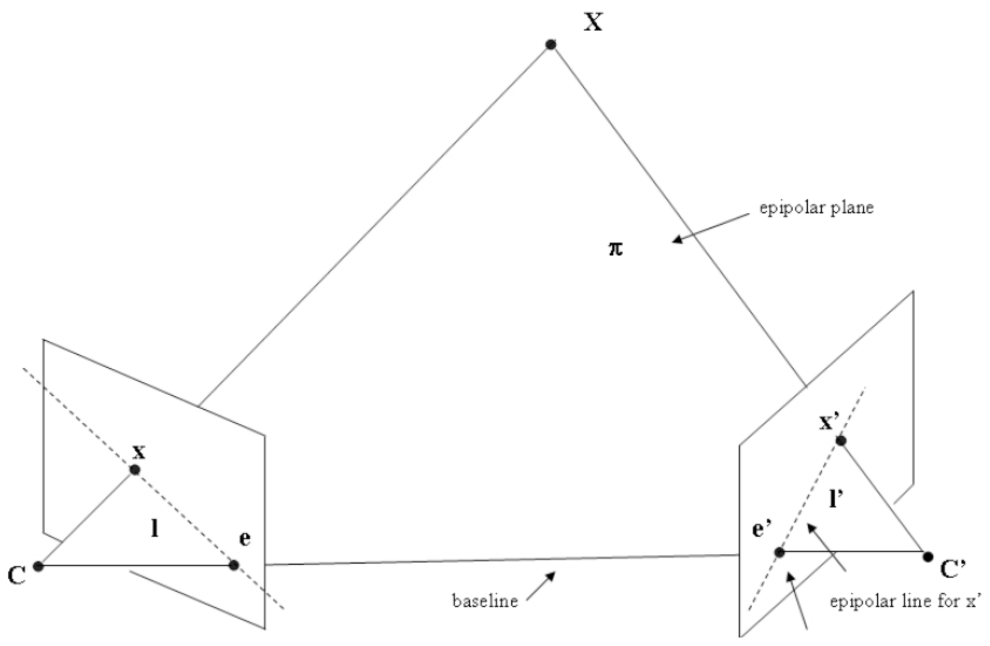
4.3 SURF: Speed Up Robust Features

5 Evaluation and Statistics Definitions

5.1 Epipolar Geometry and the Fundamental Matrix

According to Richard Hartley in "Multiple view geometry in computer vision" chapter 9, the epipolar geometry is the intrinsic projective geometry between two views.

It is independent of scene structure, and only depends on the cameras internal parameters and relative pose. Suppose a point X in 3-space is imaged in two views, at x in the first, and x′ in the second. As shown in figure ???? the image points x and x', space point X, and camera centers C are coplanar and lie on a plane π, called the epipolar plane. Moreover we may define the baseline as the line joining the two camera centers and the epipole as the point of intersection of the line joining the camera centers (the baseline) with the image plane. Equivalently, the epipole is the image in one view of the camera center of the other view.



Figure???: The two cameras are indicated by their centers C and C′ and image planes. The camera centers, 3-space point$X$, and its images x and x' lie in a common plane π. An image point x back-projects to a ray in 3-space defined by the first camera center, C, and x. This ray is imaged as a line l' in the second view. The 3-space point X which projects to x must lie on this ray, so the image of X in the second view must lie onl'. The camera baseline intersects each image plane at the epipoles e and e'.

Supposing now that we know only x, we may ask how the corresponding point x' in the second view is constrained. See figure ????above: the epipolar plane is determined by the baseline and the ray defined by x. From above we know that the ray corresponding to the (unknown) point x' lies in π, hence the point x' lies on the line of intersection l' of π with the second image plane.

This line l′ is the image in the second view of the ray back-projected from x. It is the epipolar line corresponding to x. Thus there is a mapping x to l′ that can be represented in a 3x3 matrix F such that:

$$l^{'} = Fx$$

As we are dealing with geometric entities that are expressed in homogeneous coordinates. This representation permits to write a simple equation to determine if a point x lies on a line l, that is

$$x^T l = 0$$

From above we have that the point x in the first view correspond to x' in the second view, and x' must lie on the epipolar line l'. So we can write:

$$x'^T l' = 0$$

Using $l^{'} = Fx$ in the last equation leads to the most basic properties of the fundamental matrix F that is:

$$x'^T F\, x = 0$$

This is true, because if points x and x' correspond, then x' lies on the epipolar line $l' = Fx$ corresponding to the point x. In other words $0 = x'^T\, l = x'^T Fx$. Conversely, if image points satisfy the relation $x'^T F\, x = 0$ then the rays defined by these points are coplanar. This is a necessary condition for points to correspond. Note that, geometrically, F represents a mapping from the 2-dimensional projective plane of the first image to the pencil of epipolar lines through the epipole e' on the second image. Thus it represents a mapping from a 2-dimensional onto a 1-dimensional projective space, and hence must have rank 2. Moreover, we emphasize the fact that the fundamental matrix is an homogeneous quantity, so it is defined up to a scale factor.

Now, Given sufficiently point correspondences $x \sim x'$ (more than 7 points), this equation can be used to compute the unknown matrix F. To explain this, we write $x = (x, y, 1)^T$, $x' = (x', y', 1)^T$ and

$$\mathrm{F} = \begin{bmatrix} \mathrm{f}_{11} & \mathrm{f}_{12} & \mathrm{f}_{13} \\ \mathrm{f}_{21} & \mathrm{f}_{22} & \mathrm{f}_{23} \\ \mathrm{f}_{31} & \mathrm{f}_{32} & \mathrm{f}_{33} \end{bmatrix}$$

If we explicit the product in $x'^T F\, x = 0$ for a point correspondence we end up with this equation:

$$x'x\mathrm{f}_{11} + x'y\mathrm{f}_{12} + x'\mathrm{f}_{13} + y'x\mathrm{f}_{21} + y'y\mathrm{f}_{22} + y'\mathrm{f}_{23} + x\mathrm{f}_{31} + y\mathrm{f}_{32} + \mathrm{f}_{33} = 0$$

If we denote by f the 9-vector made up of the entries of F in row-major order then above equation can be expressed as a scalar product:

$$(x'x, x'y, x', y'x, y'y, y', x, y, 1)f = 0$$

From a set of N points correspondences, we obtain a set of linear equations of the form:

$$Af = \begin{pmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny & y'_n & x_n & y_n & 1 \end{pmatrix} f = 0$$

This is a homogeneous set of equations and f can only be determined up to scale. Now, for a solution to exist, matrix A must have rank at most 8, and if the rank is exactly 8, then the solution is unique (up to scale). Please note that a solution for homogeneous linear system of the form $Af = 0$ is a vector in the so called nullspace of A. The rank-nullity theorem states that the dimensions of the rank and the nullspace add up to the number of columns of A. Matrix A in above matrix has 9 columns so when its rank is 8 the nullspace is composed by only one vector f , that is the unique solution.

Obviously the data correspondences $x \sim x'$will not be exact because of noise and the rank of A may be greater or lower than 8. In the former case the nullspace is empty, so the only solution one can find is a least-squares solution. A method to obtain this

particular solution for$f$, is to select the singular vector corresponding to the smallest singular value of$A$.
The solution vector f is then the last column of V in the $SVD\ A = UDV^T$.
As one can see, the matrix A depends only on the points correspondences $x \sim x'$. Thus, its rank also will be conditioned by the point matches. There are cases in which these correspondences can lead to a matrix A with rank less than 8, thus avoiding the existence of a unique solution (for instance when all the probing sources lie on a common plane, or a common line). In this case infact the nullspace of A has dimension greater than 1, so a solution for above matrix is a linear combination of all the vectors in the nullspace For instance, when all the probing sources lie on a common plane, or a common line. Hence, it is important to choose the positions of the probing signals as random as possible within the field of view of the two acoustic cameras, in order to well condition the problem and avoid low rank problems.

5.2 Planar Homographies

Any two images of the same planar surface in space are related by a homography (assuming a pinhole camera model). This has many practical applications, such as image rectification, image registration, or computation of camera motion—rotation and translation—between two images. Once camera rotation and translation have been extracted from an estimated homography matrix, this information may be used for navigation, or to insert models of 3D objects into an image or video, so that they are rendered with the correct perspective and appear to have been part of the original scene.

If the camera motion between two images is pure rotation, with no translation, then the two images are related by a homography (assuming a pinhole camera model).

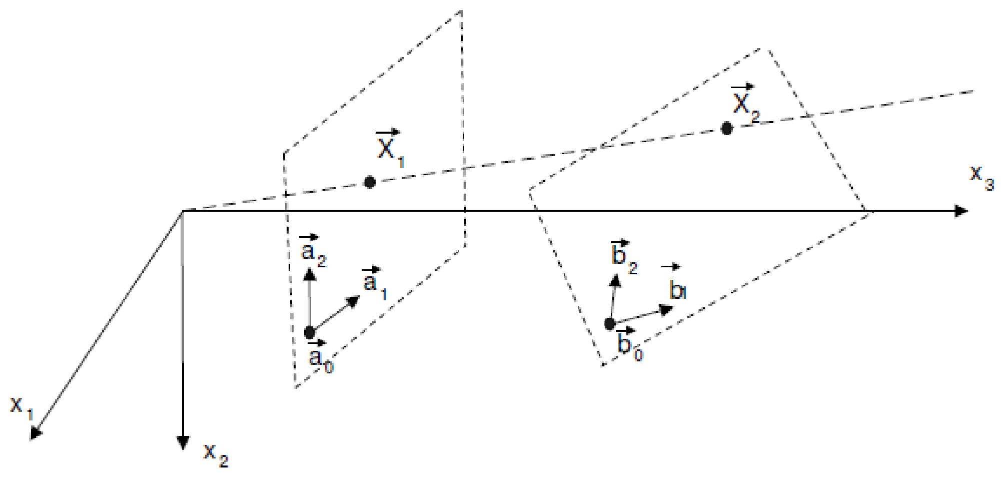


Fig.

Consider a 3D coordinate frame and two arbitrary planes, The first plane is defined by the point $\overrightarrow{b_0}$and two linearly independent vectors $\overrightarrow{b_1}$, $\overrightarrow{b_2}$ contained in the plane. Now, consider a point $\overrightarrow{X_2}$ in this plane. Since the vectors $\overrightarrow{b_1}$and $\overrightarrow{b_2}$ form a basis in this plane, we can express $\overrightarrow{X_2}$ as:

$$\overrightarrow{X_2} = q_1\overrightarrow{b_1} + q_2\overrightarrow{b_2} + \overrightarrow{b_0} = \left(\overrightarrow{b_1}, \overrightarrow{b_2}, \overrightarrow{b_0}\right)\begin{pmatrix} q_1 \\ q_2 \\ 1 \end{pmatrix} = B\vec{q}$$

Where $B = \left(\overrightarrow{b_1}, \overrightarrow{b_2}, \overrightarrow{b_0}\right) \in \Re^{3\times3}$ defines the plane and $\vec{q} = (q_1, q_2, 1)^T$defines the 2D coordinates of $\overrightarrow{X_2}$ with respect to the basis $\left(\overrightarrow{b_1}, \overrightarrow{b_2}\right)$

The similar identification is valid for the second plane

$$\overrightarrow{X_1} = A\vec{p}$$

Where $A = (\overrightarrow{a_1}, \overrightarrow{a_2}, \overrightarrow{a_0}) \in \Re^{3\times3}$ defines the plane while $\vec{q} = (p_1, p_2, 1)^T$ defines the 2D coordinates of $\overrightarrow{X_2}$ with respect to the basis $(\overrightarrow{p_1}, \overrightarrow{p_2})$

By imposing the constraint that point $\overrightarrow{X_1}$ maps to point $\overrightarrow{X_2}$under perspective projection, centered at the origin $\vec{X} = \vec{0}$, so

$$\overrightarrow{X_1} = \alpha\,(\vec{q})\overrightarrow{X_2}$$

Where $\alpha\,(\vec{q})$ is a scalar that depends on $\overrightarrow{X_2}$, and consequently on $\vec{q}$. Combining the equation above with the constraint that each of the two points must be situated in its corresponding plane, one obtains the relationship between the 2D coordinates of these points:

$$\vec{p} = \alpha(\vec{q}) A^{-1} B \vec{q}$$

Note that the matrix A is invertible because$\vec{a_1}, \vec{a_2}, \vec{a_0}$ are linearly independent and nonzero (the two planes do not pass through origin). Also note that the two vectors $\vec{p}$ and $\vec{q}$above have a unit third coordinate. Hence the role of $\alpha(\vec{q})$ is simply to scale the term $\alpha(\vec{q})A^{-1}B\vec{q}$ such that its third coordinate is 1. But, we can get rid of this nonlinearity by moving to homogeneous coordinates:

$$p^{\vec{h}} = H\, q^{\vec{h}}$$

Where $p^{\vec{h}}$, $q^{\vec{h}}$ are homogeneous 3D vectors. $H \in \Re^{3\times3}$ is called a homography matrix and has 8 degrees of freedom, because it is defined up to a scaling factor ($H = cA^{-1}B$ where c is any arbitrary scalar).

Applications of Homographies

Here are some computer vision and graphics applications that employ homographies:

- mosaics (image processing):
  Involves computing homographies between pairs of input images
  Employs image-image mappings
- removing perspective distortion (computer vision)
  Requires computing homographies between an image and scene surfaces
  Employs image-scene mappings
- rendering textures (computer graphics)
  Requires applying homographies between a planar scene surface and the image plane, having the camera as the center of projection
  Employs scene-image mappings
- computing planar shadows (computer graphics)
  Requires applying homographies between two surfaces inside a 3D scene, having the light source as the center of projection
  Employs scene-scene mappings

Now, under homography, we can write the transformation of points in 3D from two cameras (camera 1 to camera 2) as:

$$X_2 = HX_1 \quad X_{1,}X_2 \in \Re^3$$

In the image planes, using homogeneous coordinates, we have

$$\lambda_1 x_1 = X_1 \quad and \quad \lambda_2 x_2 = X_2\ ,therfore\ \ \lambda_2 x_2 = H\lambda_1 x_1$$

This means that x_2 is equal to Hx_1 up to a scale (due to universal scale ambiguity). Note that $x_2 \sim Hx_1$is a direct mapping between points in the image planes. If it is known that some points all lie in a plane in the first image, the image can be rectified directly without needing to recover and manipulate 3D coordinates.

Homography Estimation

To estimate H, we start from the equation $x_2 \sim Hx_1$. Written element by element, in homogenous coordinates we get the following constraint:

$$\begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} H_{11} & H_{12} & H_{13} \\ H_{21} & H_{22} & H_{23} \\ H_{31} & H_{32} & H_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix} \leftrightarrow \ x_2 \sim Hx_1$$

In inhomogeneous coordinates ($x'_2 = \frac{x_2}{z_2}, and\ \ y'_2 = \frac{y_2}{z_2}$):

$$x'_2 = \frac{H_{11}x_1 + H_{12}y_1 + H_{13}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1}$$

$$y'_2 = \frac{H_{21}x_1 + H_{22}y_1 + H_{23}z_1}{H_{31}x_1 + H_{32}y_1 + H_{33}z_1}$$

In this work, we set the $\mathrm{z}_1 = 1$,actually without loss of generality:

$$x'_2 = \frac{H_{11}x_1 + H_{12}y_1 + H_{13}}{H_{31}x_1 + H_{32}y_1 + H_{33}}$$

$$y'_2 = \frac{H_{21}x_1 + H_{22}y_1 + H_{23}}{H_{31}x_1 + H_{32}y_1 + H_{33}}$$

5.3 Generalized Extreme Value Distributions

Extreme value theory is a separate branch of statistics that deals with extreme events. This theory is based on the *extremal types theorem*, also called the *three types theorem*, stating that there are only three types of distributions that are needed to model the maximum or minimum of the collection of random observations from the same distribution.

According to Samuel Kotz and S. Nadarajah, Extreme value distributions are usually considered to comprise the following three families:

Type 1, (Gumbel-type distribution):

$$\Pr[\,X \le x] = \exp[-e^{(x-\mu)/\sigma}]$$

Type 2, (F'rCchet-type distribution):

$$\Pr[\,X \le x] = \begin{cases} 0 & x < \mu \\ \exp\left\{-\left(\frac{x-\mu}{\sigma}\right)^{-\varepsilon}\right\} & x \ge \mu \end{cases}$$

Type 3, (Weibull-type distribution):

$$\Pr[\,X \le x] = \begin{cases} \exp\left\{-\left(\frac{x-\mu}{\sigma}\right)^{\varepsilon}\right\} & x \le \mu \\ 0 & x > \mu \end{cases}$$

The generalized extreme value (GEV) distribution was first introduced by Jenkinson (1955).The cumulative distribution function of the generalized extreme value distributions is given by

$$F_x(x) = \begin{cases} \exp\left\{-(1+\varepsilon\left(\frac{x-\mu}{\sigma}\right)^{\frac{-1}{\varepsilon}})\right\} & \text{for } \varepsilon \neq 0 \\ \exp\{-e^{-\left(\frac{x-\mu}{\sigma}\right)}\} & \text{for } \varepsilon = 0 \end{cases}$$

It has also the following PDF:

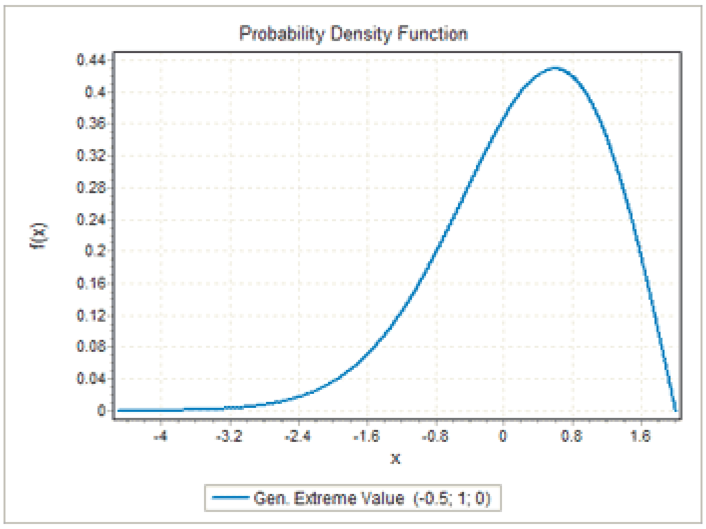
$$f(x) = \begin{cases} \frac{1}{\sigma}\exp\left(-(1+kz)^{\frac{1}{k}}\right)(1+kz)^{-1-1/k} & k \neq 0 \\ \frac{1}{\sigma}\exp(-z-\exp(-z)) & k = 0 \end{cases}$$

where $z = (x-\mu)/\sigma$, and k, σ, μ are the shape, scale, and location parameters respectively. The scale must be positive ($sigma > 0$), the shape and location can take on any real value. The range of definition of the GEV distribution depends on k:

$$1 + k\frac{(x-\mu)}{\sigma} > 0 \qquad k \neq 0$$

$$-\infty < x < +\infty \qquad k = 0$$

Various values of the shape parameter yield the extreme value type I, II, and III distributions. Specifically, the three cases k=0, k>0, and k<0 correspond to the Gumbel, Fréchet, and "reversed" Weibull distributions. The reversed Weibull distribution is a quite rarely used model bounded on the upper side. For example, for k=−0.5, the GEV PDF graph has the form:



When fitting the GEV distribution to sample data, the sign of the shape parameter k will usually indicate which one of the three models best describes the random process we are dealing with.

5.4 Goodness of Fit Test

The goodness of fit (GOF) tests measures the compatibility of a random sample with a theoretical probability distribution function. In other words, these tests show how well the selected distribution fits to the data and our test condition and hypothesis will be described.

Anderson-Darling test

The Anderson-Darling procedure is a general test to compare the fit of an observed cumulative distribution function to an expected cumulative distribution function. This test gives more weight to the tails. In this section, the definition of this test is explained in detail.

Anderson and Darling (1952, 1954) introduced the goodness-of-fit statistic

$$A_m^2 = m \int_{-\infty}^{\infty} \frac{\{F_m(x) - F_0(x)\}^2}{F_0(x)\{1 - F_0(x)\}}\, dF_0(x)$$

to test the hypothesis that a random sample $\mathrm{X}_1, \ldots, \mathrm{X}_\mathrm{m}$, with empirical distribution $F_m(x)$, comes from a continuous population with completely specified distribution function $F_0(x)$.Here $F_m(x)$ is defined as the proportion of the sample $XI, \ldots, Xm$ that is not greater than x. The corresponding two-sample version

$$\mathrm{A}_{\mathrm{mn}}^2 = \frac{\mathrm{mn}}{\mathrm{N}} \int_{-\infty}^{\infty} \frac{\{\mathrm{F_m(x)} - \mathrm{G_n(x)}\}^2}{\mathrm{H_N(x)}\{1 - \mathrm{H_N(x)}\}}\, \mathrm{dH_N(x)}$$

was proposed by Darling (1957) and studied in detail by Pettitt (1976). Here $\mathrm{G_n(x)}$ is the empirical distribution function of the second (independent) sample $\mathrm{Y}_1, \ldots, \mathrm{Y}_\mathrm{N}$ obtained from a continuous population with distribution function $\mathrm{G(x)}$, and $\mathrm{H_N(x)} = \{\mathrm{mF_m(x)} - \mathrm{nG_n(x)}\}/\mathrm{N}$, with $\mathrm{N} = \mathrm{m} + \mathrm{n}$, is the empirical distribution function of the pooled sample. The above integrand is appropriately defined to be zero whenever $H_N(x) = 1$.

The k-sample Anderson-Darling test is a rank test and thus makes no restrictive parametric model assumptions. The need for a k-sample version is twofold. It can either be used to establish differences in several sampled populations with particular sensitivity toward the tails of the pooled sample or it may be used to judge whether several samples are sufficiently similar so that they may be pooled for further analysis.

According to F. W. Scholz and M. A. Stephens (Sep.1987); Let X_{ij} be the jth observation in the ith sample $(j = 1, \ldots, n_i ;\ i = 1, \ldots, k)$. All observations are independent. Suppose that the ith sample has continuous distribution function F_i. We wish to test the hypothesis

$$H_0 \colon F_1 = \cdots = F_k$$

without specifying the common distribution F. Denote the empirical distribution function of the ith sample by $F_{ini}(x)$ and that of the pooled sample of all $N = n_1 + \ldots + n_k$ observations by $H_N(x)$. The k-sample Anderson-Darling test statistic is then defined as:

$$A_{kN}^2 = \sum_{i=1}^{k} n_i \int_{B_N} \frac{\{F_{ini}(x) - H_N(x)\}^2}{H_N(x)\{1 - H_N(x)\}}\, dH_N(x)$$

where $B_N = \{x \mathrm{E} R \colon H_N(x) < 1\}$. Under the continuity assumption on the F_i the probability of ties is zero. Hence the pooled ordered sample is $Z_1 < \cdots < Z_N$, and a straightforward evaluation of above equation yields the following computational formula for A_{kN}^2:

$$A_{kN}^2 = \frac{1}{N} \sum_{i=1}^{k} \frac{1}{n_i} \sum_{j=1}^{N-1} \frac{(NM_{ij} - jn_i)^2}{j(N-j)}$$

where M_{ij} is the number of observations in the ith sample that are not greater than Z_j.

In our test in this work, The Anderson-Darling statistic (A^2) is simply defined as:

$$A^2 = -n - \frac{1}{n} \sum_{n=1}^{n} (2i - 1) . [\ln F(X_i) + \ln(1 - F(X_{n-i+1}))]$$

Hypothesis Testing

The null and the alternative hypotheses are:

- H_0: the data follow the specified distribution;
- H_A: the data do not follow the specified distribution.

The hypothesis regarding the distributional form is rejected at the chosen significance level (α) if the test statistic, A^2, is greater than the critical. The fixed values of (0.01, 0.05 etc.) are generally used to evaluate the null hypothesis (H_0) at various significance levels. In general, critical values of the Anderson-Darling test statistic depend on the specific distribution being tested but the value of 0.05 is used for most tests.

The Anderson-Darling test implemented in this work uses the same critical values for all distributions. These values are calculated using the approximation formula, and depend on the sample size only. This kind of test (compared to the "original" A-D test) is less likely to reject the good fit, and can be successfully used to compare the goodness of fit of several fitted distributions.

6 Proposed Methods

6.1 Tracking Methods

Total Comparison Point to Point Method

In this method, we start with the first frame at time (t_0). The detection algorithm is run on the whole frame and all the features are detected in the first frame (f_0) . Then all these extracted features are passed to the descriptor with the needed parameters and so the descriptor is ready to describe each extracted point based on information on detector. It means that the features of frame f_0 are now completely detected and described in all related parameters such as scale, distance, laplacian parameters and descriptor array.

After the previous stage, the same procedure is done for the second frame (f_1) and all the features in that frame is extract and described for frame f_1 .

For matching step, in this method, each interest point in the frame f_1 will be compare with whole interest points from previous frame f_0"Point to Point". The method is shown in the figure ???.

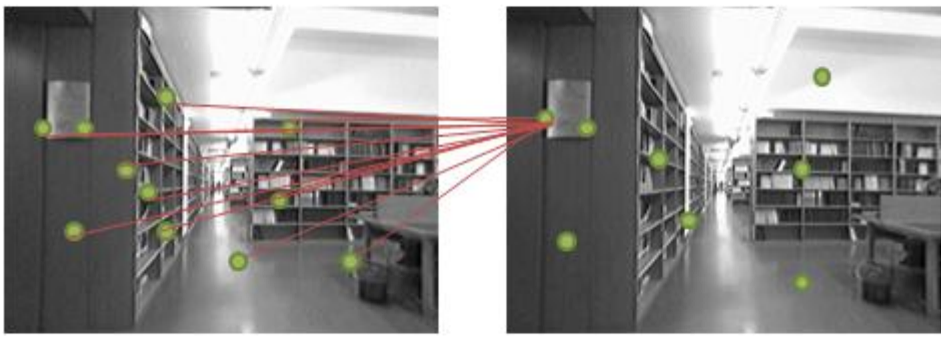


Fig.the left image is the frame f_0 and the green dots shows the interest points (note that in real algorithm these interest points are much more than above image). The right image is related to frame f_1 and the red line shows the comparison method

In this method it is expected that we face to higher match points with higher calculation time which leads to lower frame per second value (fps).

Region of Interest (ROI) Point to Point Method

In this method, the procedure of loading and detecting the interest points are same. It means that the detecting algorithm is run on the whole starting frame f_0 and so the interest features are detected. Then the whole algorithm will be run on the second frame f_1 and every interest points in this frame (current frame) will be described as well.

On this method -as it arises from the name – a region will be defined around each interest point in f_1 . This region could be in different pixel size such as 10×10 , 20×20 , 40×40 or even 80×80. The region will be mapped accordingly from frame f_1 to f_0 and the size of the region will be strictly maintained. Then each interest points in frame f_1 will be compared with the corresponded features in the defined ROI on frame f_0. This comparison will be point to point in the selected ROI. The figure ??? shows this method.

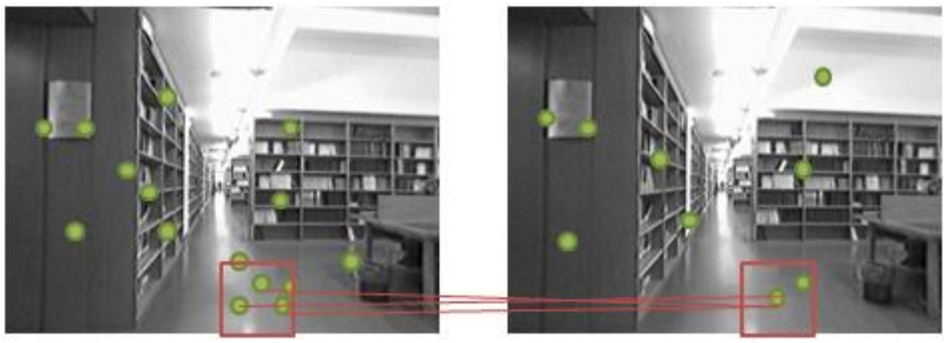


Fig. the left image shows the IPs in frame $f_0$while the red square is mapped ROI from frame f_1. The green dots are the IPs in both frames.

In this method it is expected that we face to lower match points in compare with previous method with lower calculation time which leads to higher frame per second value (fps).

Region of Interest (ROI) Group Centroid Comparison Method

As previous, the detection algorithm will be run on the whole frame in f_0 and all the features are detected in that frames, then the same algorithm will be run on the second frame f_1 and the interest points are described as well on that frame. Till this step it would be exactly same as the two previous methods.

Then the region of interest is defined around each IP in frame f_1 with the arbitrary size of 10×10 , 20×20 , 40×40 or 80×80. For comparison method, the selected ROI will be mapped accordingly in the frame f_0 with same size. In this step, we have a selected ROI with n interest points inside the region on frame f_0. In this method the centroid of these n interest points are calculated from their descriptor parameters by simply making the average from each parameter. Then an virtual interest point will be created as well with new parameters from all IPs in selected ROI. The comparison will be done between the interest point in frame $f_1$and the Centroid in framef_0. So the comparison will be just done between two points instead of n. It means that in this procedure we will have 1×1 comparison instead of $1 \times n$. the figure ??? shows this method.

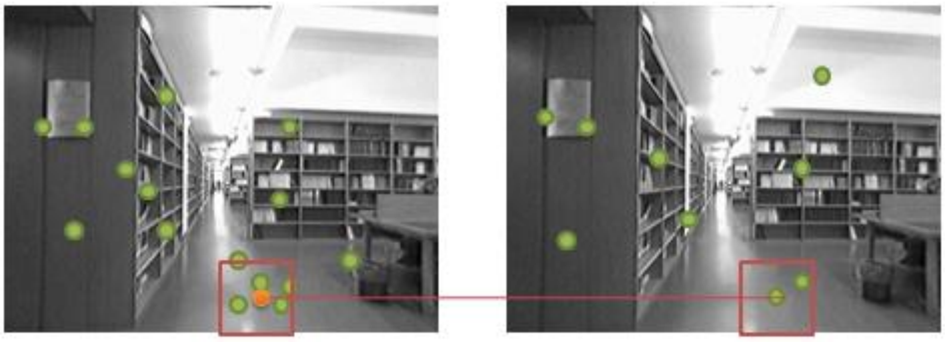


Fig. the left image shows the mapped ROI and its IPs. The orange dot shows the centroid of IPs in selected ROI.

In this method it is expected that we face to more match points in compare with previous method with more or less the same calculation time. So the matched points will be increased in this method.

Total Comparison Point to Point Method with Two Frames

In this method the whole algorithm is run on the frame $f_0$and the interest points are detected and described. Then the whole algorithm for detecting and describing is run on the second frame f_1 as well but unlike the three previous methods, this is not the point of comparison. The detecting and describing algorithm is run for third frame f_2 and all the interest points are extracted and describe. This would be the step of comparison between interest points. The procedure is as following; at first each interest points in frame f_2 (which is called as current frame) will be compared to all described interest points in previous framef_1, if the algorithm find any matching between the interest point of current frame and the previous frame so the point will be flagged as matched, otherwise, the current interest point in frame f_2 will be evaluated against all interest points in two-previous frame f_0 searching for any matches. If the point finds its match in whole interest points in two-previous frame it will be flagged as match as well. The interest point is called as not match if during these two frame it could not find any point that satisfy the threshold condition.

The figure ???? shows this algorithm. the left frame is frame f_0 , the middle and right frames are frames $f_1$and f_2 respectively. The two detected IPs in frame f_2 illustrate the algorithm of matching in previous frame and two-previous frame. The continues black lines show matching in previous frame while the dash red lines shows not matching in previous frame and so the algorithm referral in two-previos frame f_0 for finding matches.

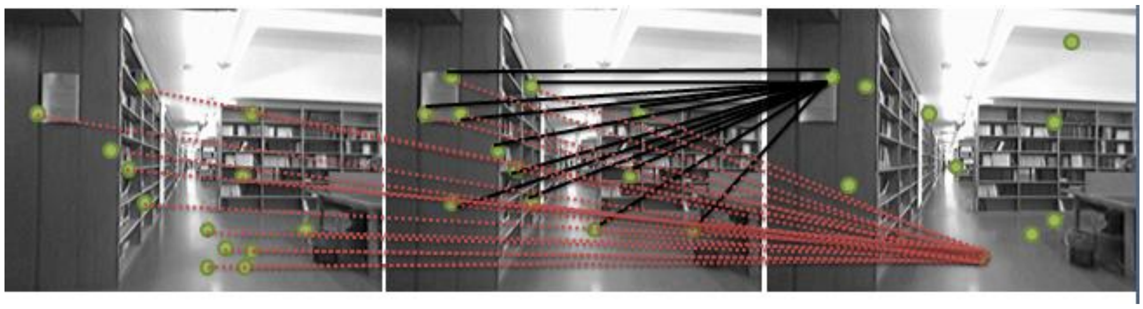


fig. Total Comparison Point to Point Method with Two Frames

Region of Interest (ROI) Point to Point Method with Two Frames

In this algorithm, the procedure of running detection algorithm is same as previous method, it means the detection algorithm is run on the whole first frame f_0 and all the IPs are detected. The same procedure of detection IPs is run for the second and third frames as well (f_1 & f_2). Now the comparison stage is start. In this step, a region of interest (ROI) is drawn around each interest point in current frame f_2. The size of this ROI could be selected from 10×10 , 20×20 , 40×40 or 80×80. After detecting this window, the algorithm maps exact size ROI on the two previous frames f_0 & f_1 as well and the potential match points will be search in these ROI. The detail is as following; at first the algorithm search to find the match interest point in previous frame (f_1) ROI. This matching search will be Point to Point in the selected ROI. It means the current frame IP will be evaluated against all IPs inside the previous ROI frame (f_1) in terms of descriptors. If any IP in the ROI passed from matching threshold , the IP in current frame is flagged as match, otherwise, the two-previous frame will be evaluated in exactly mapped ROI. If the current frame IP is matched with any IP in frame f_0 , the current IP will be flagged as match and if there are not any interest points passed the matching threshold condition even in two-frame previous, then the current IP in frame f_2 will be flagged as not matched.

The figure ??? shows the Region of Interest (ROI) Point to Point Method with Two Frames algorithm. The left frame is frame f_0 (the first frame), the middle and right frames are second and third frames (f_1 & f_2) respectively. The current two IPs in frame f_2(right frame) will matched with previous and two-previous frames in black lines and red dash lines respectively.

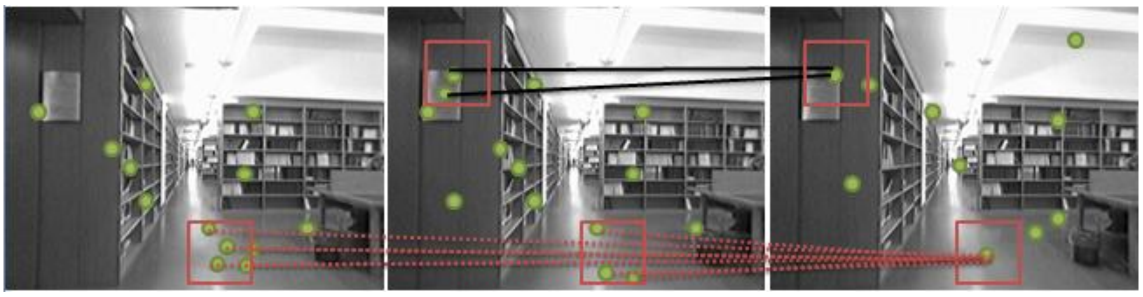


Fig. Region of Interest (ROI) Point to Point Method with Two Frames algorithm

Region of Interest (ROI) Group Centroid Comparison Method with Two Frames

In this method, the detection algorithm is run for first three frames separately and all the features are detected in each frame respectively. Note that this procedure is the same as above method in first step and it is not run simultaneously on three frames but frame by frame. After detecting IPs in each frame, the ROI will be selected around each feature in current frame which is f_2 in our example. Then the correspondence ROI will be mapped in previous frame and two-previous frame (framesf_1 & f_0 respectively).

The difference of this method in compares with "Region of Interest (ROI) Point to Point Method with Two Frames "is based on their comparison method. In this method, the average of descriptor parameter will be calculated for all the IPs in mapped ROI in frames f_1 & f_0, and the virtual Centroid of these IPs will be compare with the current frame IP. It means that we will have 1×1 comparison instead of $1 \times n$. If the Centroid of group IP in selected ROI is matched with the current frame IP, so its flag will be changed to "Matched" otherwise, the current frame IP must be evaluated with the Centroid of correspondence ROI in two-previous frame f_0. If the comparison between this group Centroid in frame $f_0$and the current frame IP in f_2 passed the threshold condition then the current frame IP flag will be change to matched and if non of the above condition were satisfied , then the Current frame IP will not be matched with any previous frames IP.

The figure??? Shows the algorithm. The left, middle and right frames are f_0, f_1& f_2 respectively. As it is shown, the orange points are the Centroid of group IP in selected ROI. If the Centroid satisfies the threshold condition in middle frame (the black line), the IP in frame f_2 will be flagged as matched otherwise the same situation will be evaluated for the Centroid in two-previous frame (red dash lines).

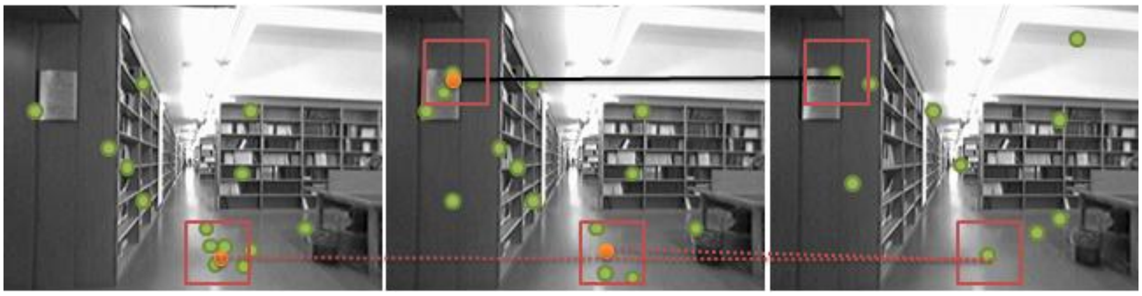


Fig. Region of Interest (ROI) Group Centroid Comparison Method with Two Frames

6.2 Tests Structure

In this work, the tests structures are grouped in three sets (A, B and C).

Set A

In this set, five tests are embedded. All of these tests are based on the first three above methods. It means we used Total Comparison Point to Point Method, Region of Interest (ROI) Point to Point Method, and Region of Interest (ROI) Group Centroid Comparison Method. The properties of these tests in some parameters are same and they made us to group them as a set A. all of the first five tests will be common in tracking frame sequence which means they just evaluate the current frame with the previous frame finding their matching points. In other words, the matching algorithm in these three tests, just compare frame f_1 as the current frame with the f_0 as the previous frame. There will not be any comparison between two-previous frames and current frame f_1.

The other common test parameter that made them to be grouped as set A is based on their detection algorithm. All the first three tests (Set A) are used the Fast Hessian as their feature detector. The detail of this detector was already explained in "feature detection" chapter on this book.

There is another common test condition available. All of these tests are used speed up robust features (SURF) as their descriptor. It means all the interest points which are detected by Fast Hessian algorithm will be delivered to the SURF algorithm for further describing.

All the tests in Set A with their result will be explained in "Test Result" chapter on this book.

Set B

The next tests set are based on the last three methods which are explained in above section in this chapter in detail. Same as previous set (Set A), three tests are embedded in this Set as well. It means that the Set B tests follow the structure of Total Comparison Point to Point Method with Two Frames, Region of Interest (ROI) Point to Point Method with Two Frames and Region of Interest (ROI) Group Centroid Comparison Method with Two Frames. These tests would be common in their tracking algorithm which based on two-previous frames comparison. In other words, the tracking and matching algorithm will compare frame f_2 as the current frame with frame $f_1 \ and \ f_0$ as two previous frames. The detail of this comparison was already explained in above section of this chapter in detail.

The other common test condition that made them to be grouped as set B is based on their detection algorithm. All the first three tests (Set B) are used the Fast Hessian as their feature detector exactly same as the previous set (Set A). The detail of this detector was already explained in "feature detection" chapter on this book.

In their descriptor algorithm, there is another common test condition available. All of these tests are used speed up robust features (SURF) as their descriptor. This condition is maintained from previous set tests (Set A). It means all the interest points which are detected by Fast Hessian algorithm will be delivered to the SURF algorithm for further describing.

Set C

In this last set, similar to previous set, three tests are embedded. The entire tests are based on the last three methods which are explained in above section of this chapter in detail. It means the tests follow the structure of Total Comparison Point to Point Method with Two Frames, Region of Interest (ROI) Point to Point Method with Two Frames and Region of Interest (ROI) Group Centroid Comparison Method with Two Frames. These tests would be common in their tracking algorithm which based on two-previous frames comparison. In other words, the tracking and matching algorithm will compare frame f_2 as the current frame with frame f_1 *and* f_0 as two previous frames. The detail of this comparison was already explained in above section of this chapter in detail.

In the both previous sets, Set A and Set B, we have used the Fast Hessian detector for interest point detection with Surf descriptor as point describing algorithm but after measuring the implementation time on each algorithm (Fast Hessian and Surf), it was learned that the describing algorithm will be much more time consuming part in compare with detecting algorithm. Actually the statistics showed that describing algorithm is 3.4 times more than detecting algorithm.

On the other hand, there were not too many choices for reducing this time unless the new descriptor is created which was not the aim of this research and it was decided to maintain the SURF descriptor. More than that, it was assumed that the new descriptor may not make huge reduction as the time proportion between these detecting and describing procedures is more than 3 times.

The other idea is to reduce the number of detected interest points which are extracted by detector and send to descriptor calculation. The idea is that, choosing robust and good feature will be more useful in compare to extracting too many features. For testing this idea, the KLT algorithm is chosen for feature detection as it could possibly select robust and good feature from all features. The algorithm of the KLT good feature detector is explained in detail on “Feature Detection” chapter. It is expected by sending these good features instead of “too many” features, we could reach to some reduction in descriptor time calculation.

6.3 Result Evaluation Structure

For evaluating the result, the total interest points of all frames is calculated beside frame interest points and frame match points. Then the proportion of frame interest points and matched interest points is calculated in each frame and draw in graph.

On the other hand, the FPS (frame per second) in whole test will be computed and drawn in a graph. It might be useful to have evaluation for matched IP (interest points) per second which could be the result of multiplication of matched IP and FPS.

For reducing the effect of outlier in output, the database was divided into eight sections and for each section the standard deviation is computed separately. The FPS and matched IP are computed in all section separately as well and their graphs are drawn.

For calculating Error, two concepts are defined. The first concept is based on the subtraction of frame interest points and frame matched points:

$$\Delta Error \ = \ Frame\ IP - Matched\ IP$$

Then the average of this error is computed as well. The whole Error graph based on this concept is drawn in results' report. The other diagram on this Error would be the PDF graph on whole error based on Anderson-Darling distribution fitting test.

The second concept on Error evaluation is based on the Fundamental matrix and Homography matrix which was explained in detail on "Evaluation and Statistic Definitions" chapter of this book.

In this Error type calculation, we try to compute True Positive (TP), False Positive (FP), True Negative (TN) and false Negative (FN) of each frame in matching procedure. From the frame match points we have the summation of TP and FP:

$$Frame\ Matched\ IP = TP + FP$$

So if we want to compute each TP and FP separately , we could use the Fundamental Matrix in which The epipolar geometry is described by the following equation:

$$[p_{2\,;}\,1]^T F\,[p_{1\,;}\,1] = 0$$

where F is fundamental matrix, p_1 and p_2 are corresponding points in the first and the second images, respectively. The fundamental matrix shows us the true positive (TP) value between two frames. By having the TP , the value of FP will be easily raised from the following subtraction:

$$FP = Frame\ Matched\ IP - TP$$

For calculating the TN, the Homography matrix is used. Actually from the $\Delta Error$ function we could reach to the summation of TN and FN as:

$$\Delta Error = TN + FN$$

For Homography matrix, we have to find the perspective transformation H between the source and the destination planes:

$$s_i \begin{bmatrix} x'_i \\ y'_i \\ 1 \end{bmatrix} \sim H \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

So that the back-projection error :

$$\sum_i \left(x'_i - \frac{h_{11}x_i + h_{12}y_i + h_{13}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2 + \left(y'_i - \frac{h_{21}x_i + h_{22}y_i + h_{23}}{h_{31}x_i + h_{32}y_i + h_{33}}\right)^2$$

is minimized. However, as not all of the point pairs ($srcPoints, dstPoints$) fit the rigid perspective transformation (i.e. there are some outliers), this initial estimate will be poor. Solving this problem we could use, RANSAC and LMeDS methods, which try many different random subsets of the corresponding point pairs, estimate the homography matrix using this subset and a simple least-square algorithm and then compute the quality/goodness of the computed homography (which is the number of inliers for RANSAC or the median re-projection error for LMeDs). The best subset is then used to produce the initial estimate of the homography matrix and the mask of inliers/outliers. The method RANSAC can handle practically any ratio of outliers, but it needs the threshold to distinguish inliers from outliers. The method LMeDSdoes not needs any threshold, but it works correctly only when there are more than 50% of inliers.

The next step after finding Homography matrix is based on projection approximation for all the points in the $TN + FN$ plate. So for each point in this domain we check the following condition :

$$P' \cong HP$$

Where $P'$the destination point in current is frame and P is the source point in previous frame. If the above condition satisfied, then we flagged this point as FN. Which means it was actually rejected in our algorithm as a matched point but it must be matched.

Please notice that the homography matrix could not count the exact FN as it uses the estimation in its projection algorithm but it could give us an acceptable view on total FN.

After computing the FN , we could easily calculate TN from:

$$TN = \Delta Error - FN$$

In this research, comparing total standard deviation on matched point distribution for each test, the Anderson-Darling test is used to find the best fitting distribution. Most of the tests are fitted with Generalized Extreme Value which was described in the “Evaluation and statistic definitions” chapter. And finally, the whole distribution PDF curve was drawn for each test with their computed parameters.

7 Numerical Test Results

In this chapter, the complete tests and their result will be sown. First, each test will be explained and then its results will be followed as well. Please notice if you just interest in the comparison between tests you could skip this chapter and refer to the next chapter on “Discussion on Results and Conclusion”.

7.1 Set A

7.1.1 1st Test – Total Point to Point

In the first test procedure the total numbers of extracted features (interest points) in current frame compare with the entire extracted feature in previous frame “Point to Point”.

The interest points (IP) in each frame $(t\ and\ t-1)$ are detected by Fast Hessian detector and will prepare for descriptor component. Then the SURF descriptor’s array is weighted against each other and the error will be extracted as well “Point to Point”.

Results:

Total extracted IP in 766 frames dataset = 128,930

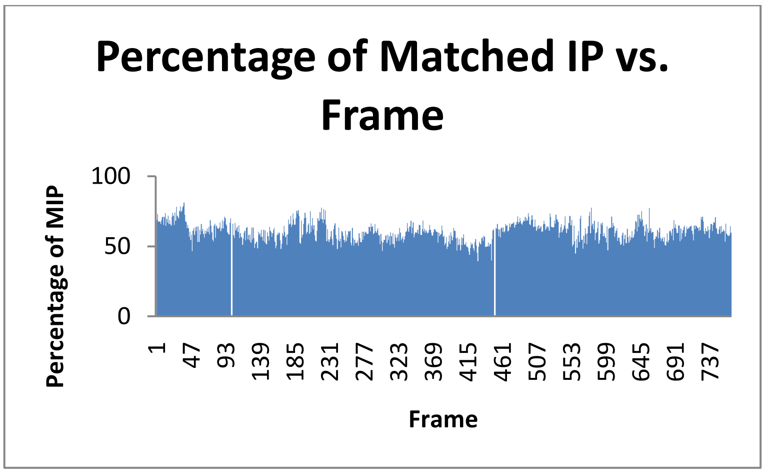
Average frames’ Interest Points (FIP) = 168.315

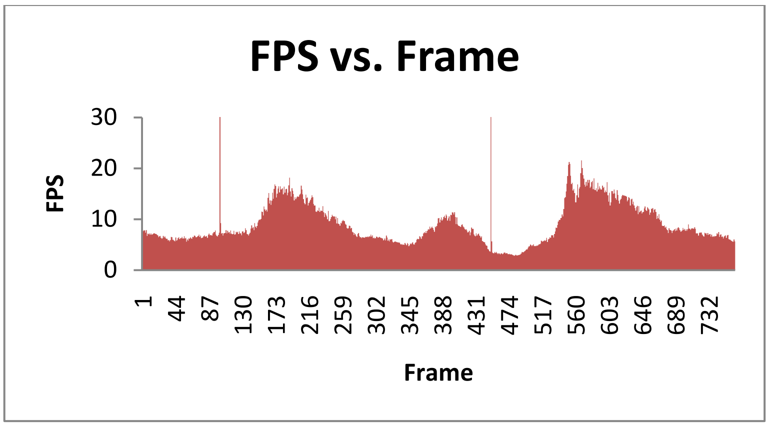
Average frames’ Match Points (MIP) = 102.1788512

Average frame per second (FPS) = 9.021562846

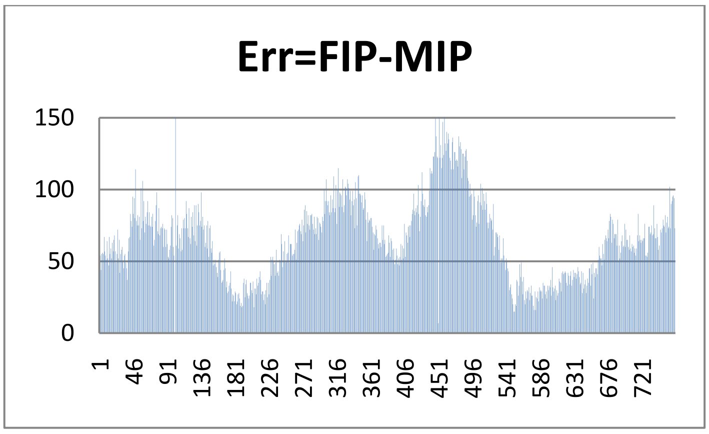
Output Graphs

Following you can find the graph for frame interest points (FIP), matched interest points (MIP) and frame per second (FPS).





Error Graph



$$\Delta Error = Frame\ IP - Matched\ IP$$

$$Average\ \Delta Error\ =\ 66.10588235$$

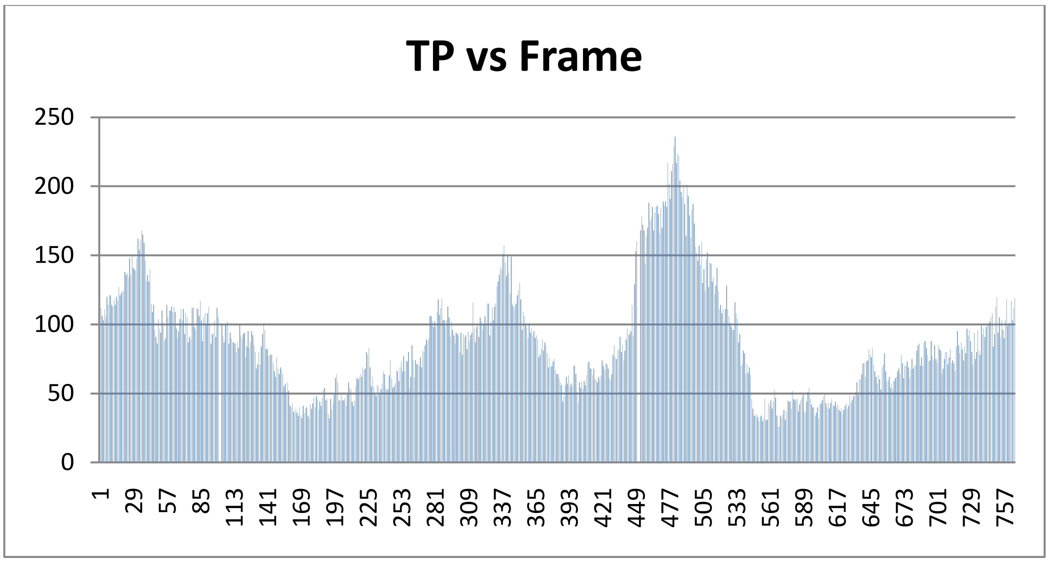
Calculating Error with Fundamental and Homography Matrices (TP, FP, TN and FN)

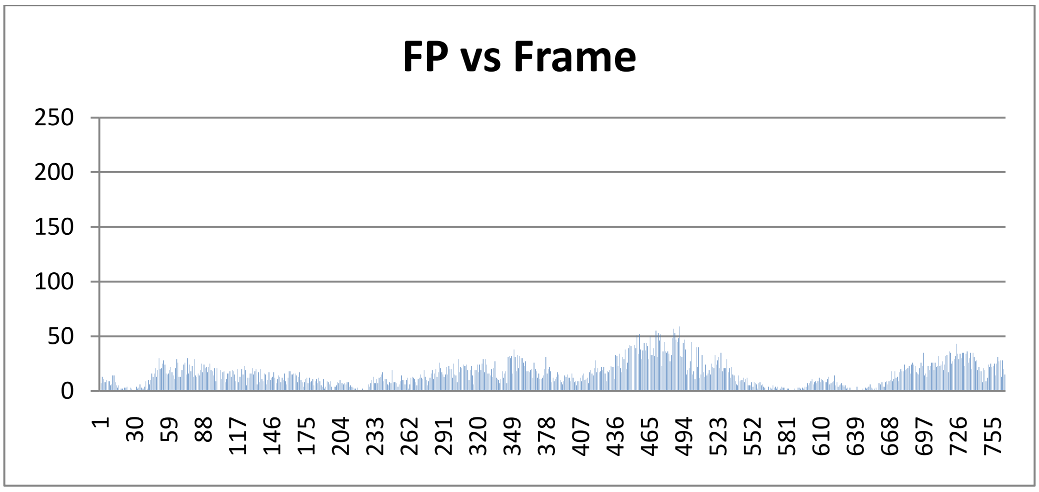
The average for TP: 51.59451697 %

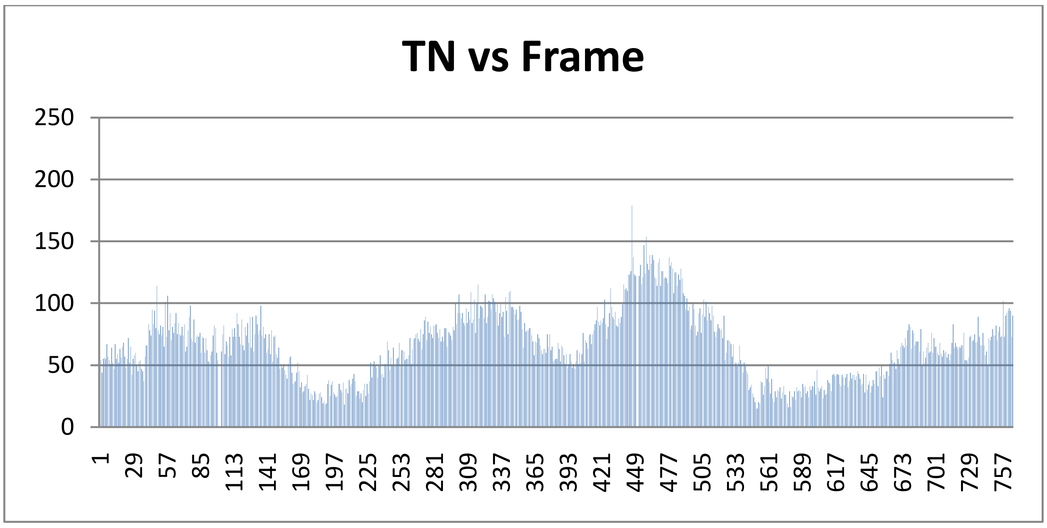
The average for FP: 8.7343342 %

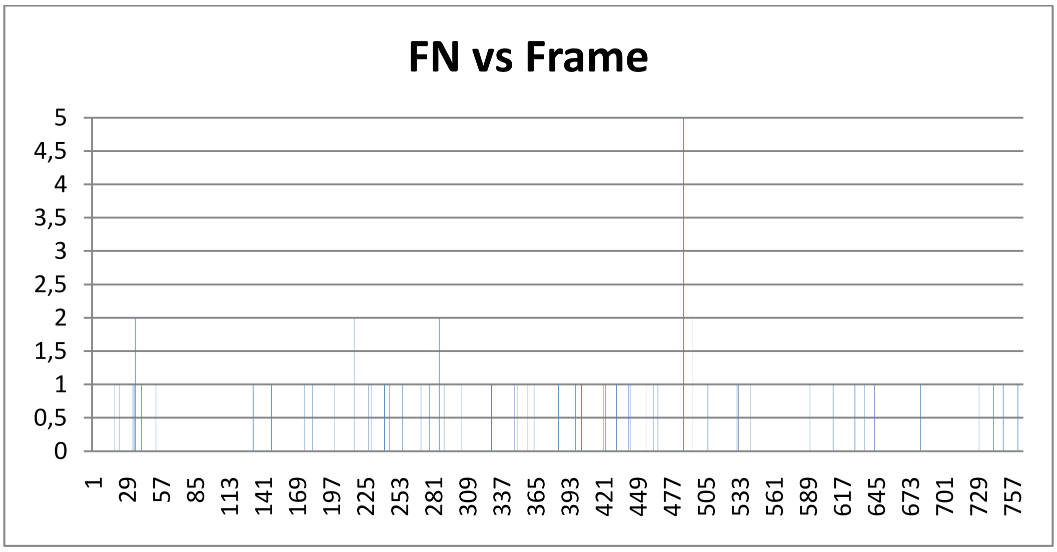
The average for TN: 38.96362924 %

The average for FN: 0.052467363 %

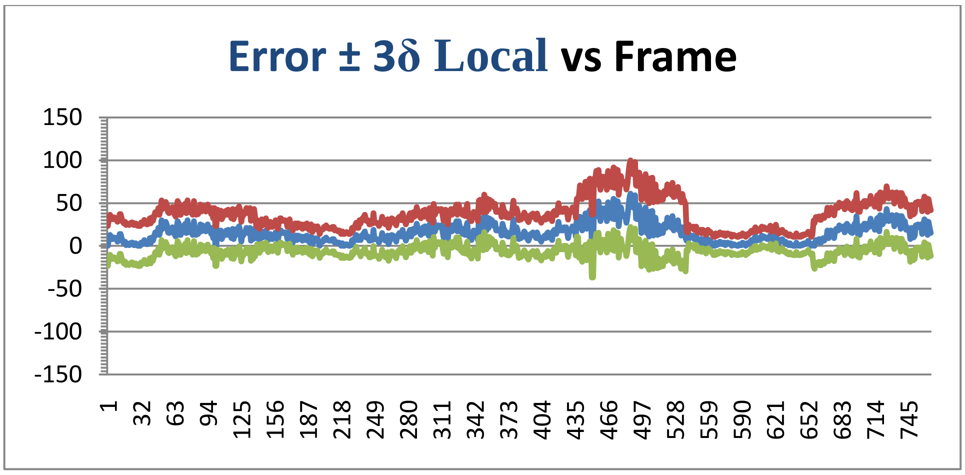




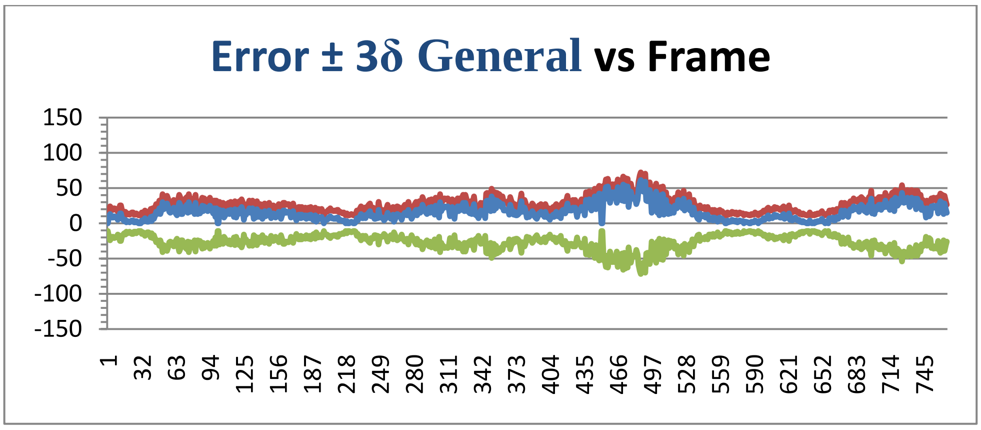




Calculating err function with local standard deviation of each distribution



Calculating err function with Overall standard deviation of each distribution

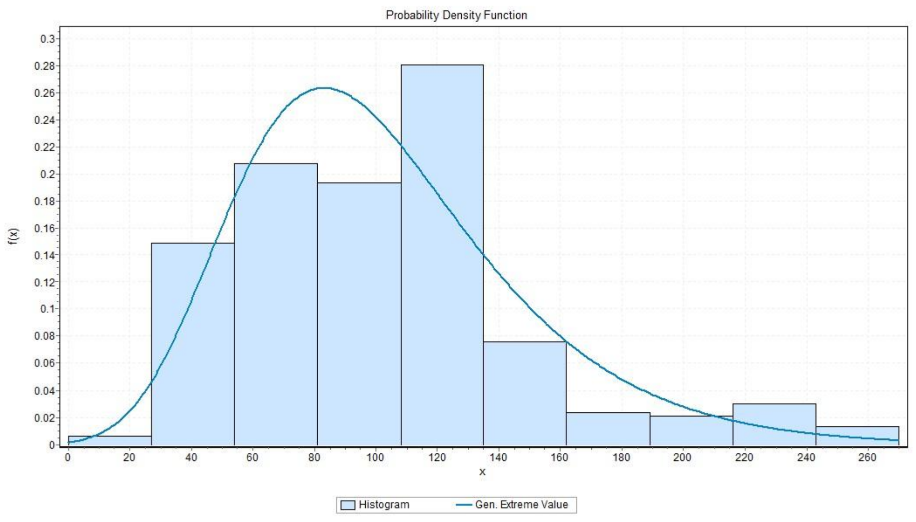


Static Calculation and Graphs

The Matched IP output data are fitted to distribution with Anderson Darling and fitting tests.

Assuming Continues dataset:

The best fitting distribution is for "Gen. Extreme Value" distribution and the parameters are:



$$k = -0.03647 \quad \sigma = 37.762 \quad \mu = 81.7$$

7.1.2 2nd Test –Total Point to Point with Euclidean distance comparison

In this test the previous condition is held, so we have the Surf descriptor array comparison, the only thing that will change our situation is the Euclidean distance comparison with two points which will be added in this test.

It means we consider two parameters for making two points match. First the SURF descriptor array as previous test and second the Euclidean coordinates distance limitation (x, y) between these two selected points between frame t and frame$t - 1$.

Results

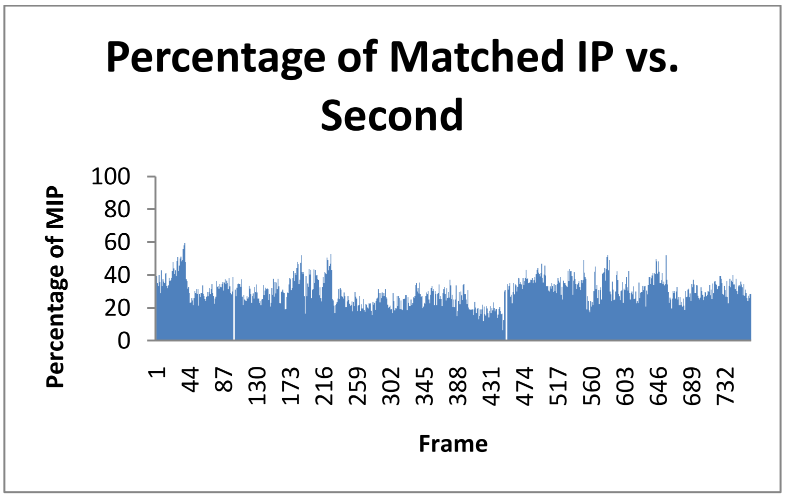
Total extracted IP in 766 frames database = 128,925

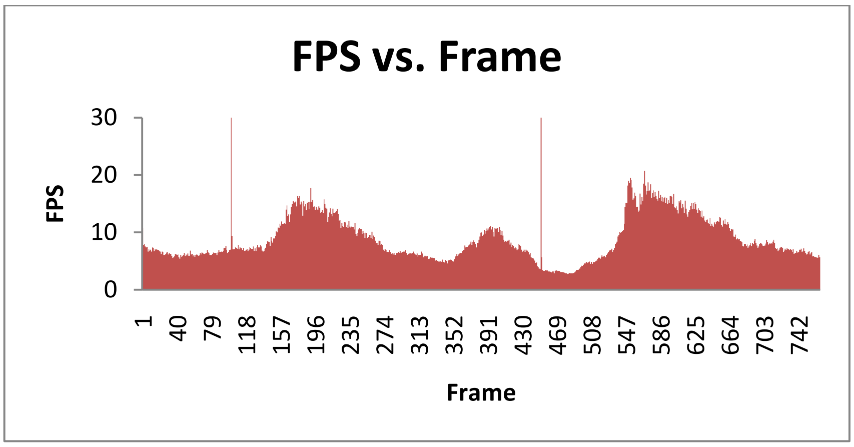
Average frames' Interest Points = 168.3159269

Average frames' Match Points = 50.7689295

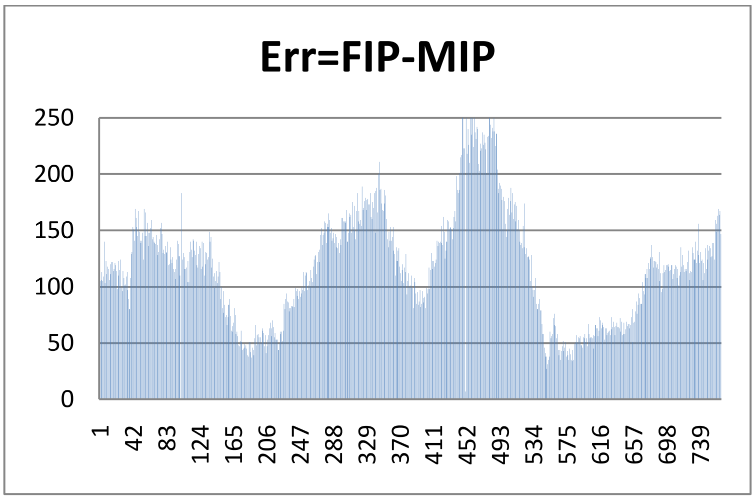
Average FPS = 8.852107076

Output Graphs





Error Graph



$$\Delta Error \;=\; Frame\ IP - Matched\ IP$$

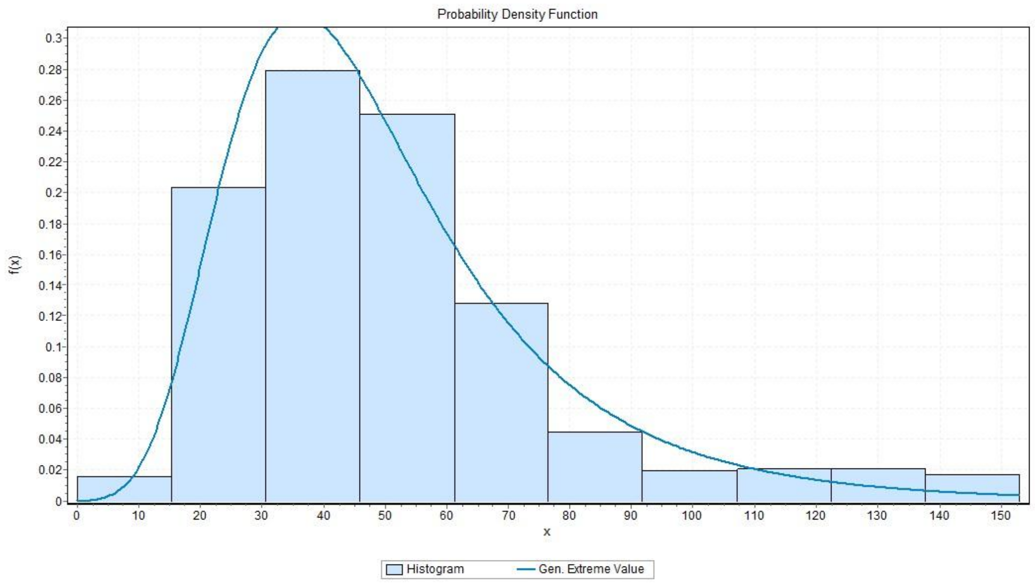
$$Average\ \Delta Error \;=\; 117.4915033$$

Static Calculation and Graphs

The Matched IP output data are fitted to distribution with Anderson Darling and fitting tests.

Assuming Continues dataset:

The best fitting distribution is for Gen. Extreme Value distribution and the parameters are:



$$k = 0.11374 \ \ \sigma = 17.978 \ \ \mu = 38.129$$

7.1.3 3rd Test - Region of interest (ROI-Points)

In this test, a region of interest ROI is defined with the size of 20*20 around each extracted interest point. Then we just compare the IP of current frame with their correspond extracted ROI interest points in previous frame and chose the best matched IP point to point in the ROI. In this comparison we just use the Surf Descriptor array for basic comparison. Note that the detector is Fast Hessian same as previous tests.

Results

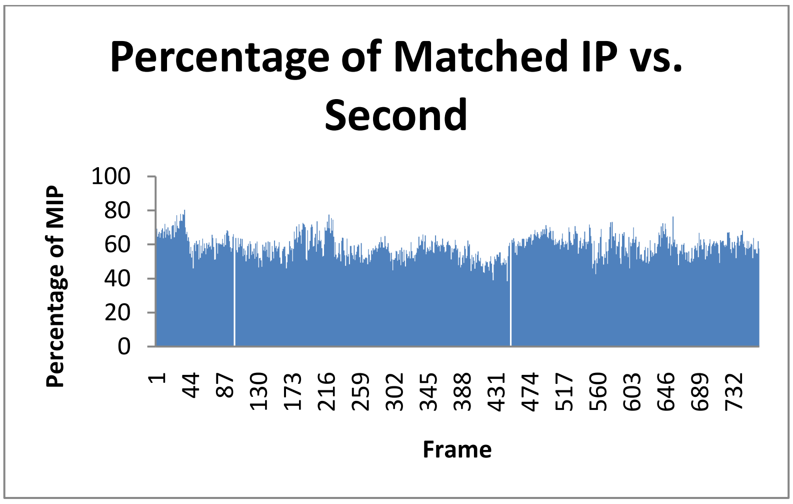
Total extracted IP in 766 frames database = 128,773

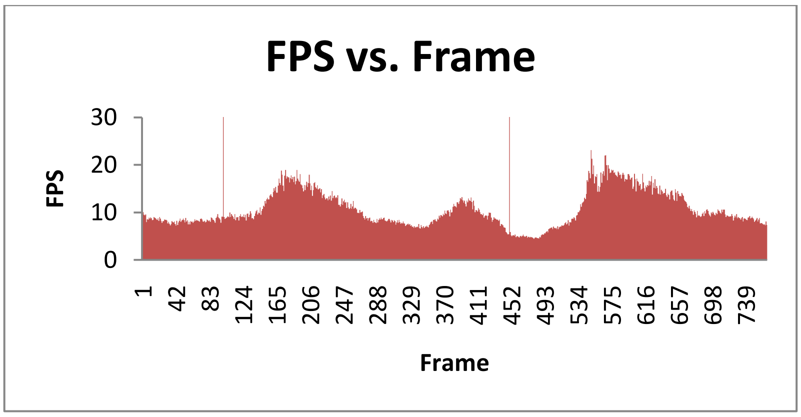
Average frames' Interest Points = 168.3307

Average frames' Match Points = 99.2092745

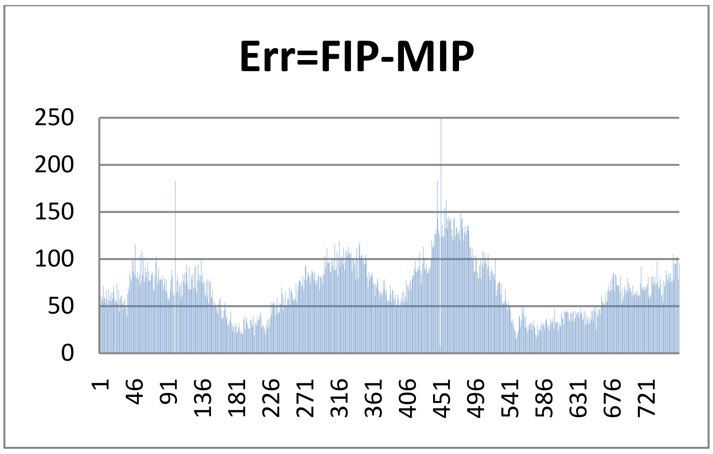
Average FPS = 10.74704119

Output Graphs:





Error Graph



$$\Delta Error \;=\; Frame\ IP - Matched\ IP$$

$$Average\ \Delta Error \;=\; 69.12156863$$

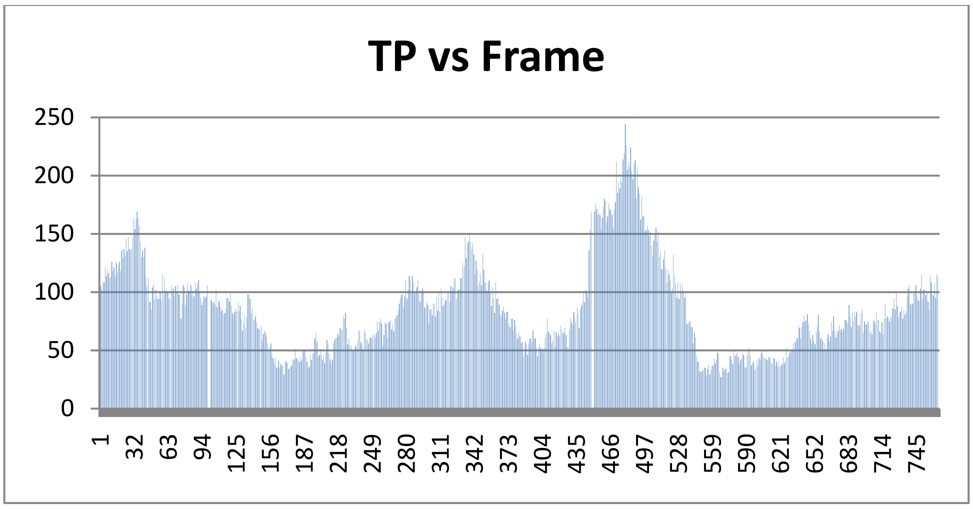
Calculating Error with Fundamental and Homography Matrices (TP, FP, TN and FN)

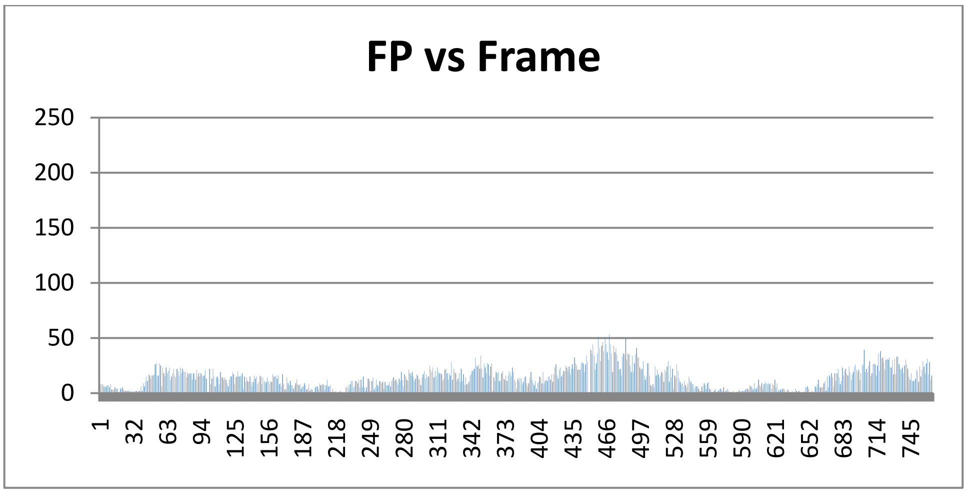
The average for TP: 50.83745098 %

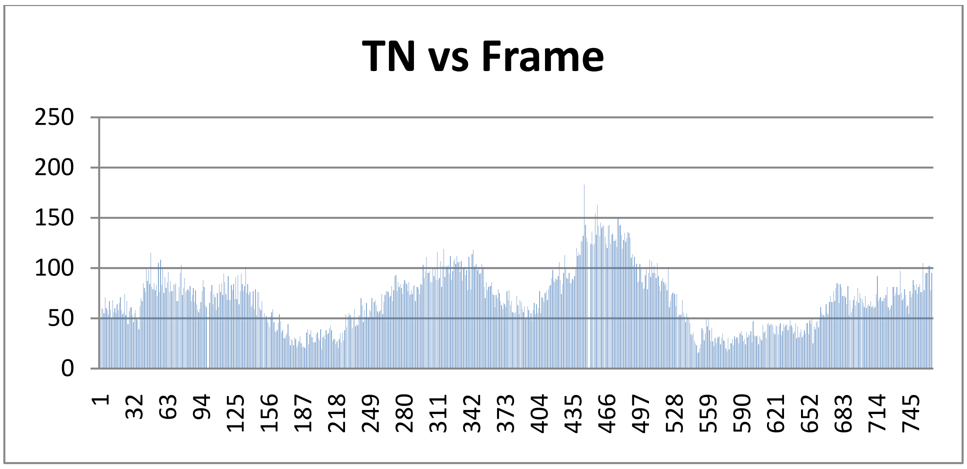
The average for FP: 7.81169935 %

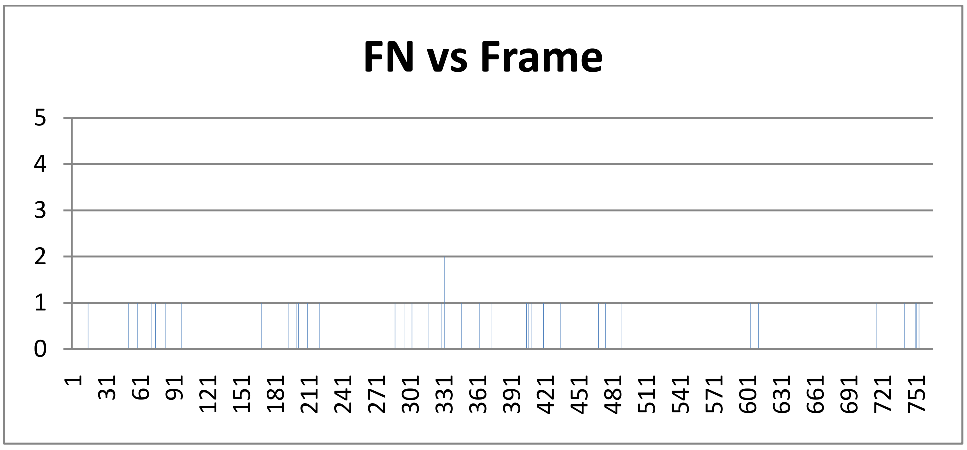
The average for TN: 40.79738562 %

The average for FN: 0.033287582 %

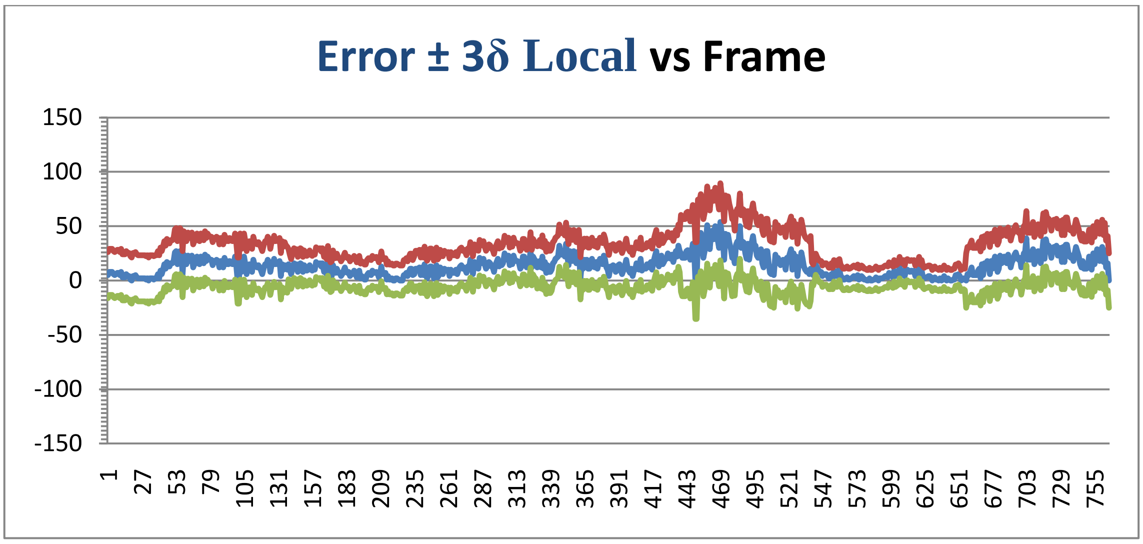



FP vs Frame
250
200
150
100
50
0
1
32
63
94
125
156
187
218
249
280
311
342
373
404
435
466
497
528
559
590
621
652
683
714
745

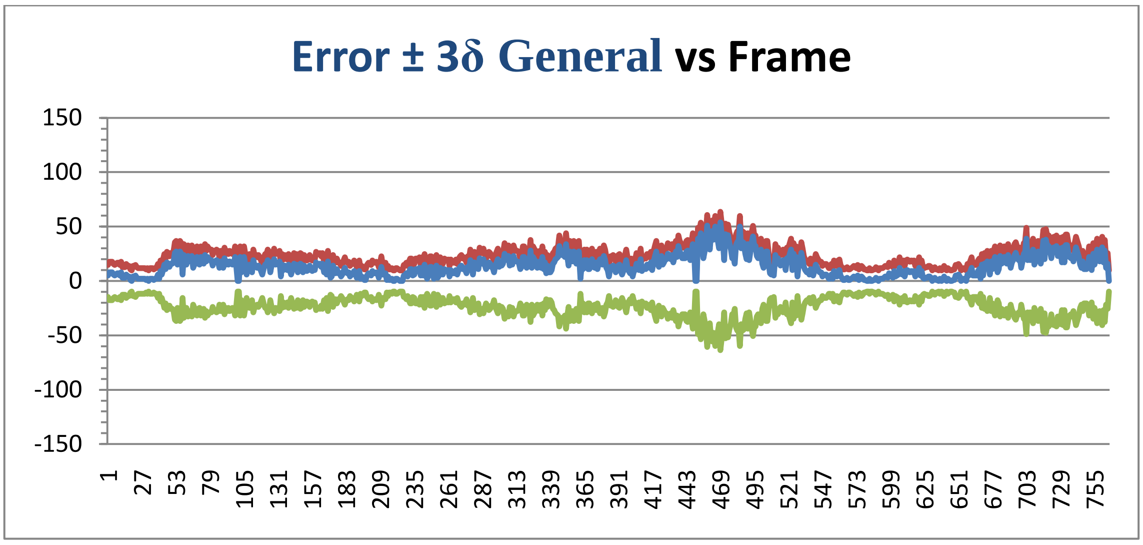

TN vs Frame
250
200
150
100
50
0
1
32
63
94
125
156
187
218
249
280
311
342
373
404
435
466
497
528
559
590
621
652
683
714
745


FN vs Frame
5
4
3
2
1
0
1
31
61
91
121
151
181
211
241
271
301
331
361
391
421
451
481
511
541
571
601
631
661
691
721
751

Calculating err function with local standard deviation of each distribution



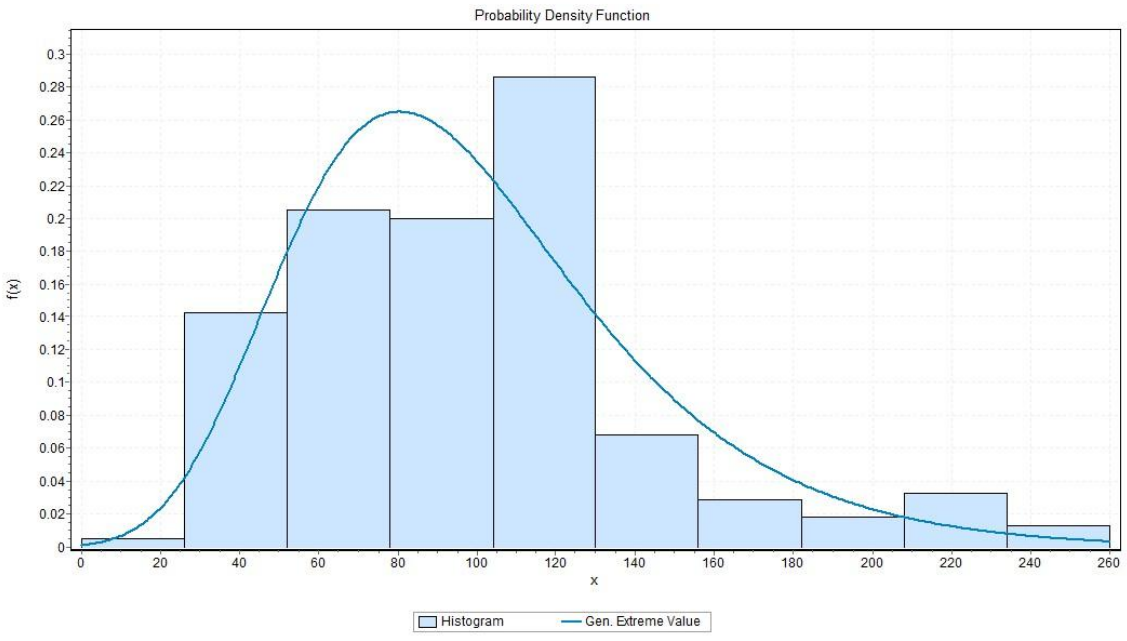
Calculating err function with Overall standard deviation of each distribution



Static Calculation and Graphs

The Matched IP output data are fitted to distribution with Anderson Darling fitting tests.

The best fitted distribution is for Gen Extreme Value, the parameters are:



$$k = -0.0301 \;\; \delta = 36.15 \;\; \mu = 79.39$$

7.1.4 4th Test - ROI with Scale comparison

In this test we exactly follow the third test procedure besides adding another comparison parameter. We use the "Scale" parameter of each IP beside the array error to make the matched point more restrict. Other conditions are exactly the same as previous test.

Results

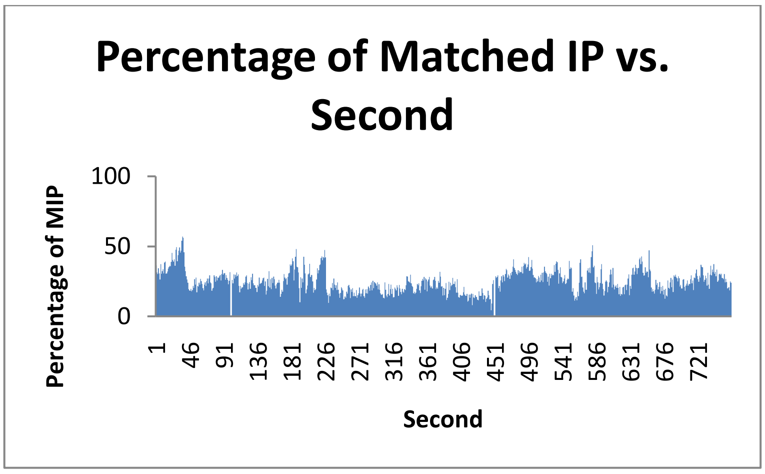
Total extracted IP in 766 frames database = 128,768

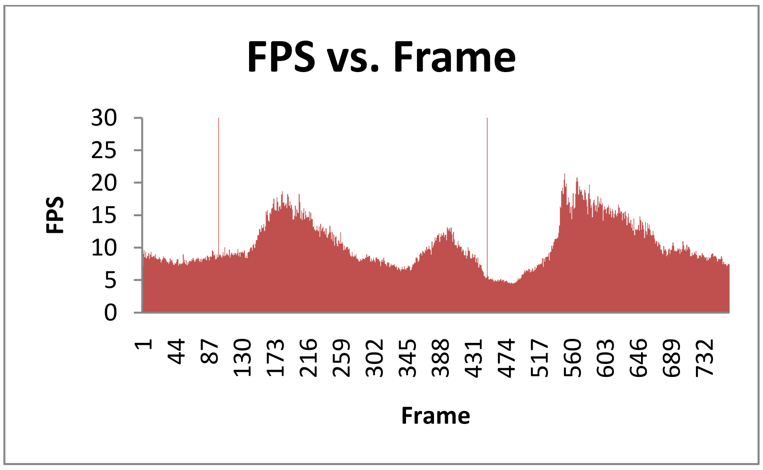
Average frames' Interest Points = 168.337183

Average frames' Match Points = 42.5254902

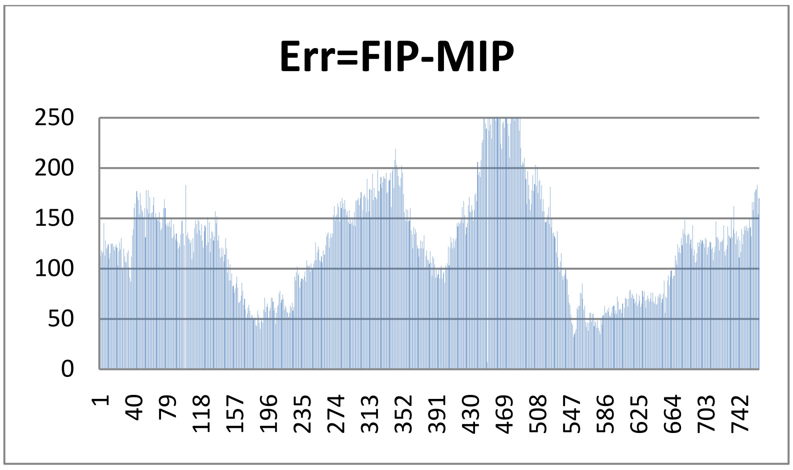
Average FPS = 10.53350344

Output Graphs





Error Graph



$$\Delta Error = Frame\ IP - Matched\ IP$$

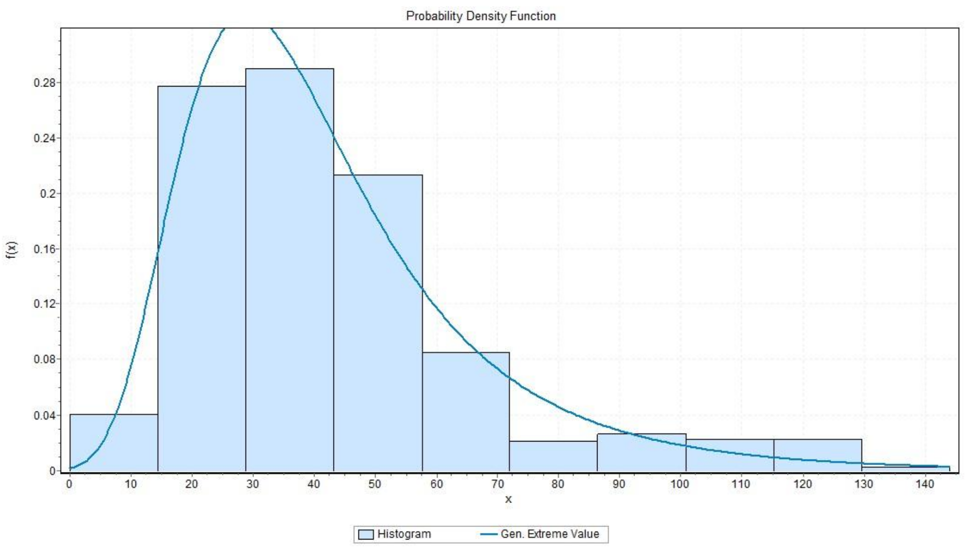
$$Average\ \Delta Error = 125.8052288$$

Static Calculation and Graphs

The Matched IP output data are fitted to distribution with Anderson Darling fitting tests.

Assuming Continues dataset:

The best fitted distribution is for Gen. Extreme Value, the parameters are:



$$k = 0.12719 \quad \delta = 16.235 \quad \mu = 30.836$$

7.1.5 5th Test - ROI group points

In this test we use an innovative way for selecting the matched points. At first we define the ROI around each IP , then instead of comparing the point to point inside the relevant ROI in previous frame, we define a centroid for all the points inside the previous frame ROI and then the comparison for matching process is calculated with current frame IP and the centroid ROI point in previous frame. The comparison is based on Surf descriptor 64bit array as well.

Results

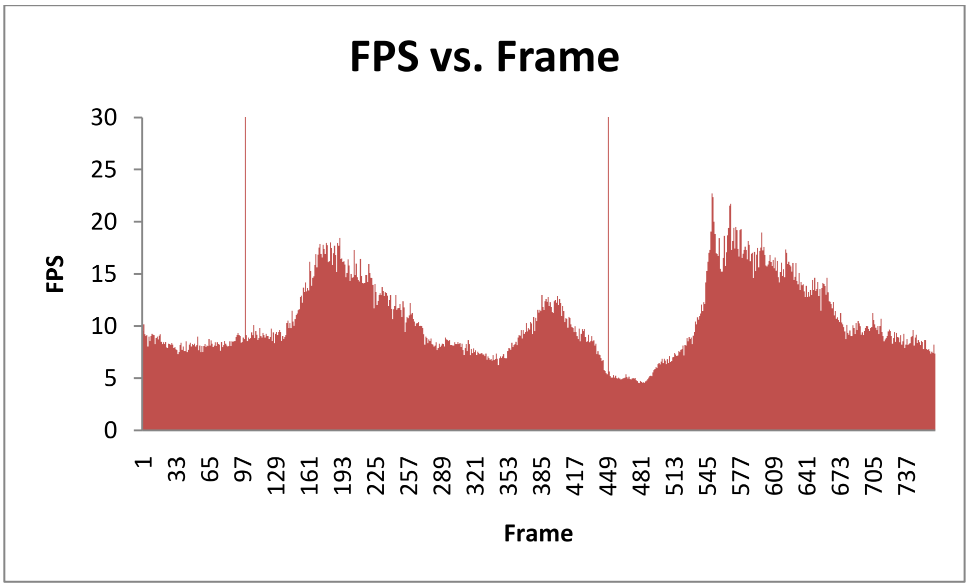
Total extracted IP in 766 frames database = 128,768

Average frames' Interest Points = 168.330719

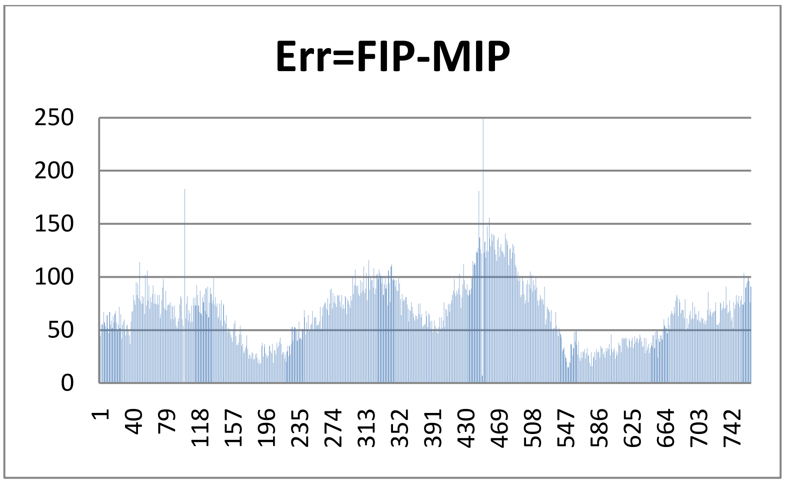
Average frames' Match Points = 101.8653595

Average FPS = 10.62197058

Output Graph



Error Graph



$$\Delta Error \ = \ Frame\ IP - Matched\ IP$$

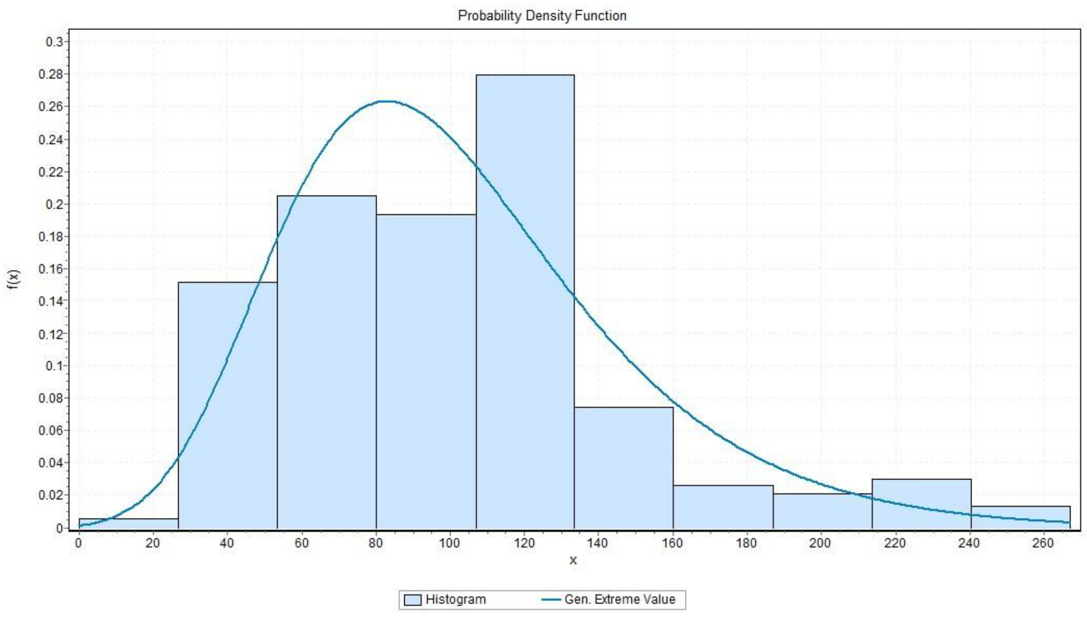
$$Average\ \Delta Error \ = \ 66.46535948$$

Static Calculation and Graphs

The Matched IP output data are fitted to distribution with Anderson Darling fitting tests.

Assuming Continues dataset:

The best fitted distribution is for Gen.Extreme Value, the parameters are:



$$k = -0.03662 \ \ \delta = 37.369 \ \ \mu = 81.605$$

7.2 Set B

In the following three tests, we compare the current frame not only with previous frame but also with two previous frame as well. It means if we are in frame “t”, the IP will compare with frame “$t - 1$” at first, and if any points don’t flagged as match point then the same matching search will be held for frame “$t - 2$” with current frame “t”.

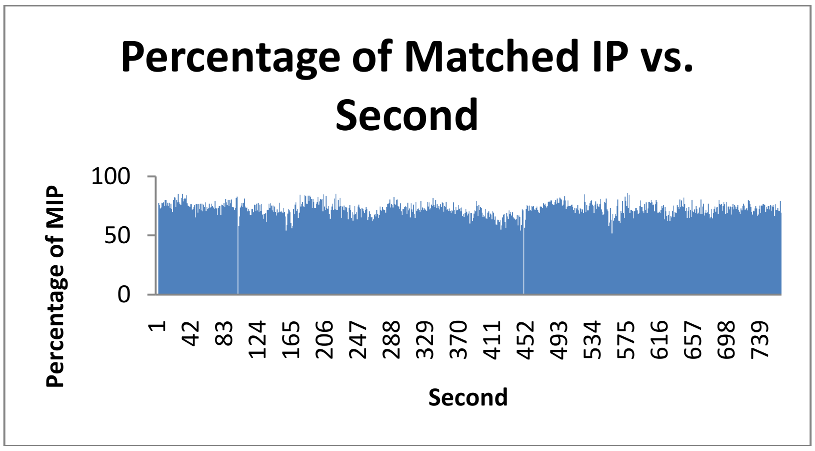
The other set tests’ conditions are exactly same as set A. it means with Fast Hessian detector and Surf descriptor. The tests 6, 7 and 8 are exactly correspondence to tests 1, 3 and 5 respectively.

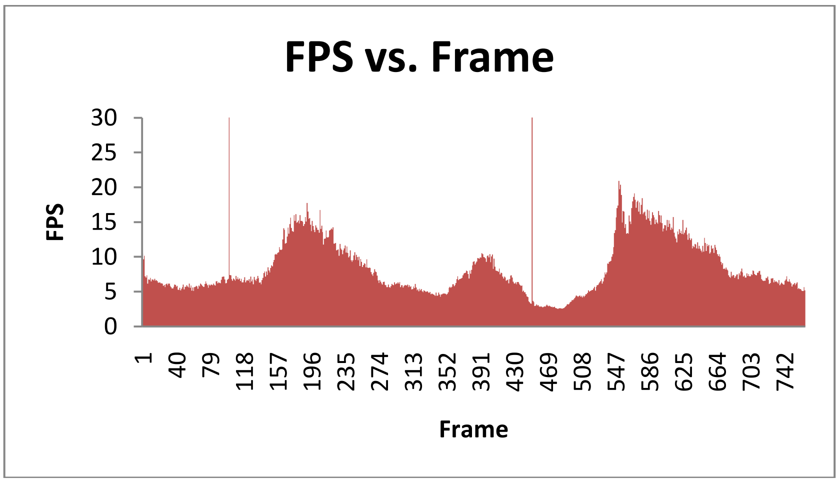
7.2.1 6th Test – Total Point to Point-2frames

As it was explained in previous chapter, the test is exactly same as the 1st test with the difference of checking 2 previous frames for matching the IP.

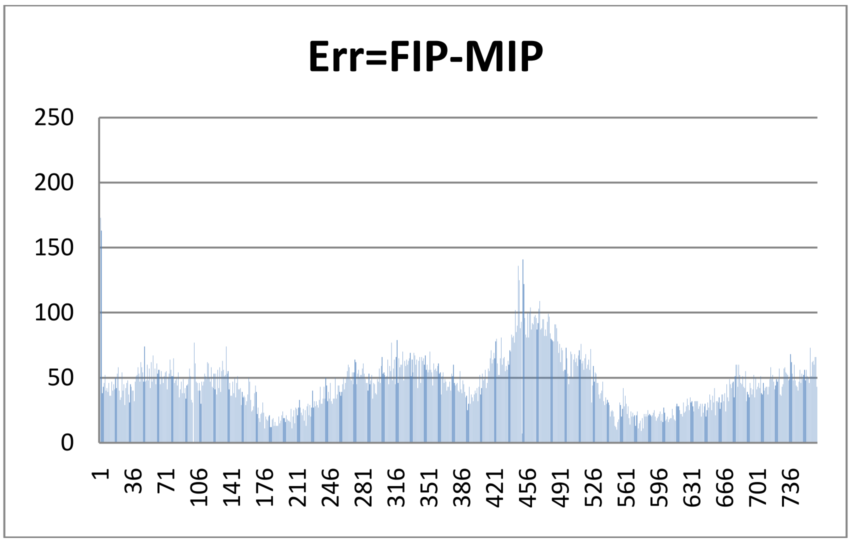
Results
Total extracted IP in 766 frames database = 128,930
Average frames' Interest Points = 168.3159269
Average frames' Match Points = 122.1736292
Average FPS = 8.49976936

Output Results





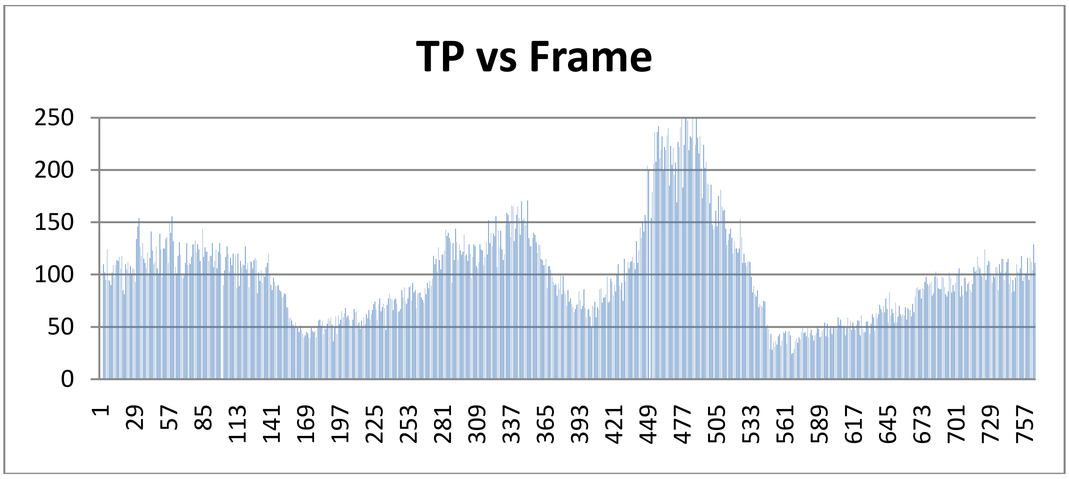
Error Graph

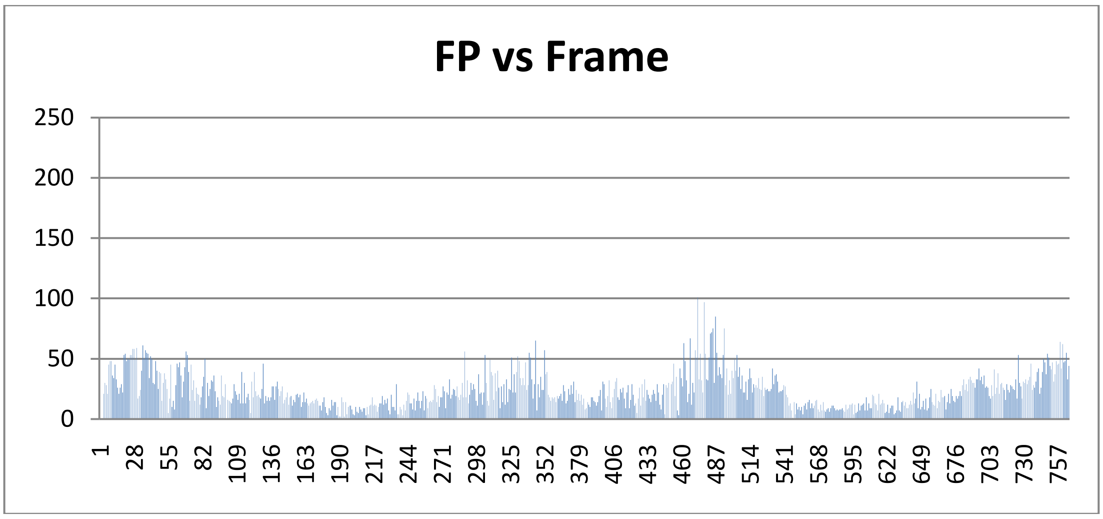


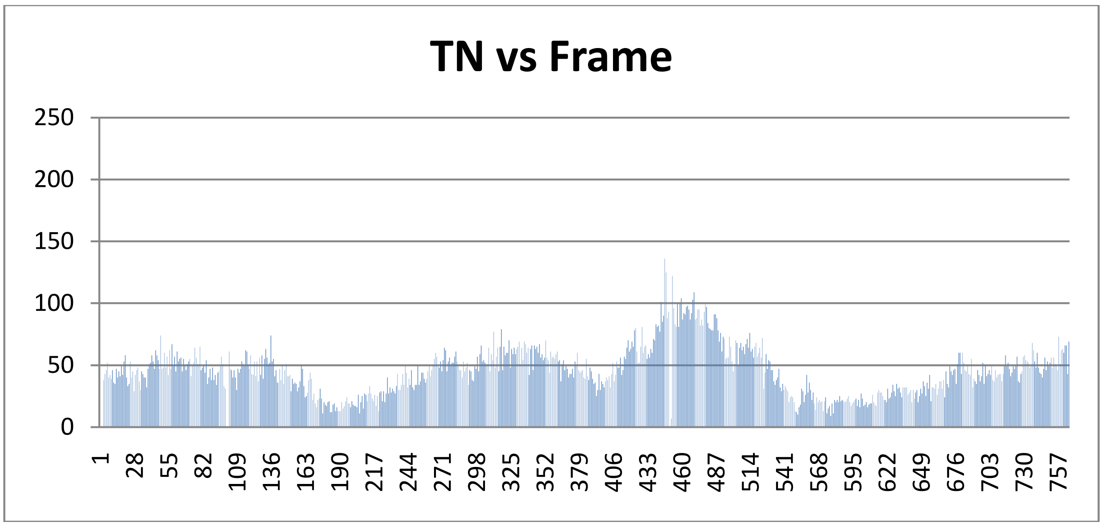
$$\Delta Error = Frame\ IP - Matched\ IP$$
$$Average\ \Delta Error = 46.1124183$$

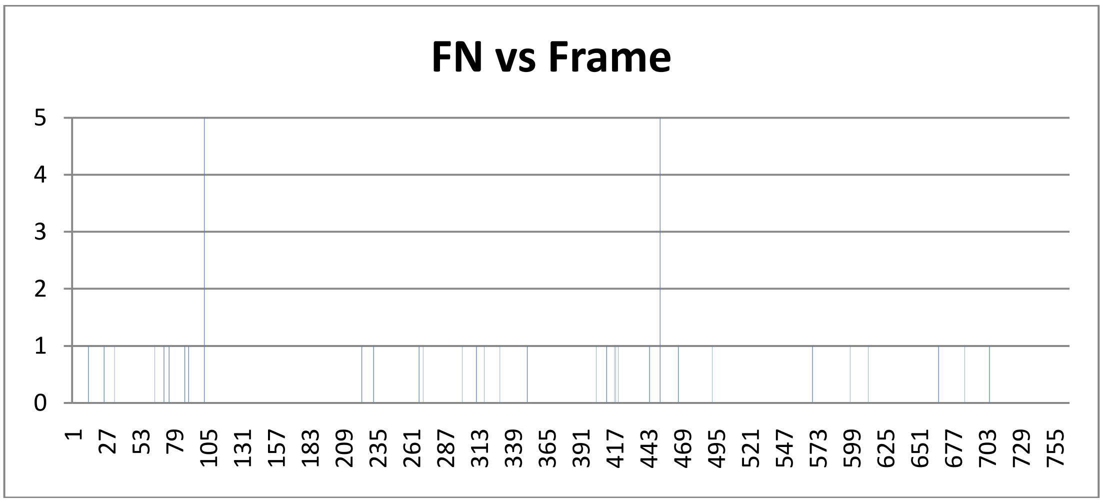
Calculating Error with Fundamental and Homography Matrices (TP, FP, TN and FN)

The average for TP: 58.5570235 %
The average for FP: 13.60660574 %
The average for TN: 27.05362924 %
The average for FN: 0.134927 %

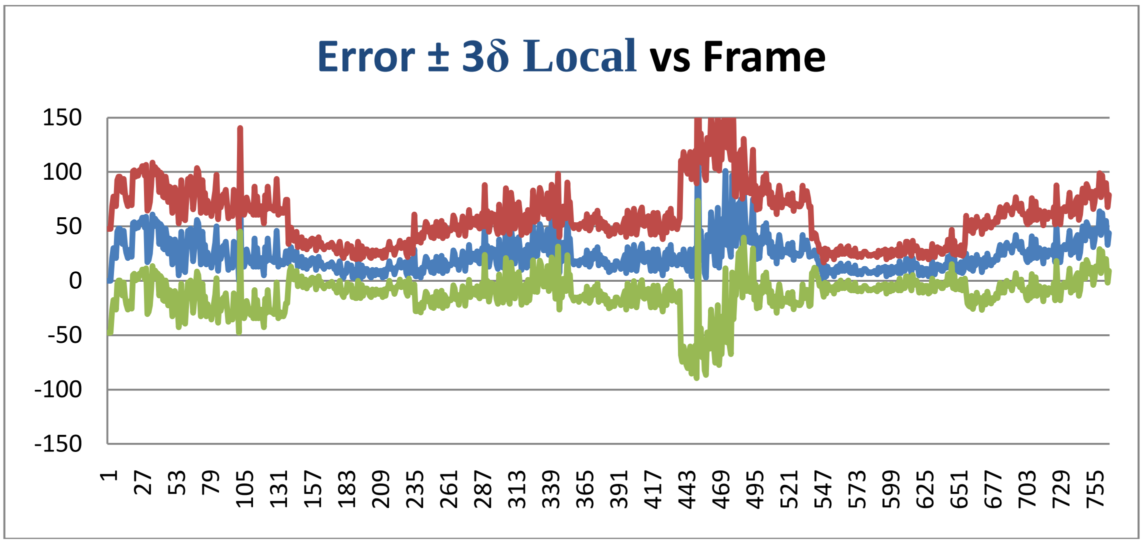



FP vs Frame
250
200
150
100
50
0
1
28
55
82
109
136
163
190
217
244
271
298
325
352
379
406
433
460
487
514
541
568
595
622
649
676
703
730
757

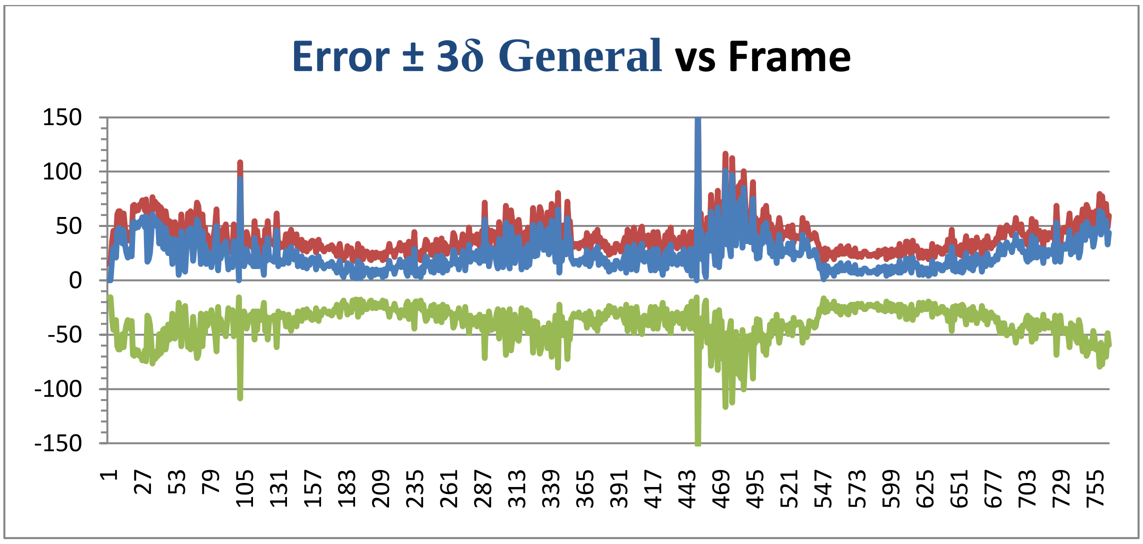

TN vs Frame
250
200
150
100
50
0
1
28
55
82
109
136
163
190
217
244
271
298
325
352
379
406
433
460
487
514
541
568
595
622
649
676
703
730
757


FN vs Frame
5
4
3
2
1
0
1
27
53
79
105
131
157
183
209
235
261
287
313
339
365
391
417
443
469
495
521
547
573
599
625
651
677
703
729
755

Calculating err function with local standard deviation of each distribution

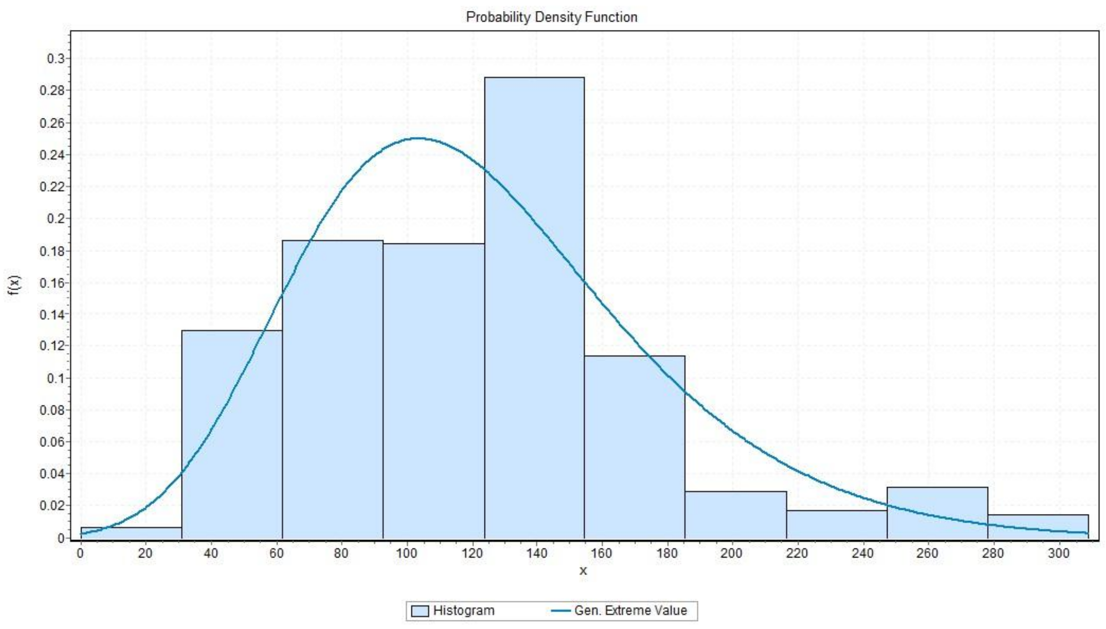


Calculating err function with Overall standard deviation of each distribution



Static Calculation and Graphs

The Matched IP output data are fitted to distribution with Anderson Darling Smirnov fitting tests.
The best fitted distribution is for Gen.Extreme Value, the parameters are:



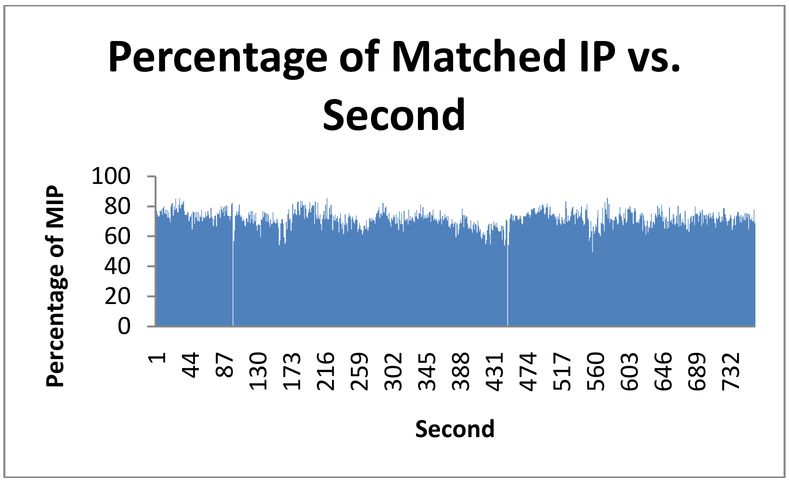
$$k = -0.08698 \;\; \delta = 45.659 \;\; \mu = 99.46$$

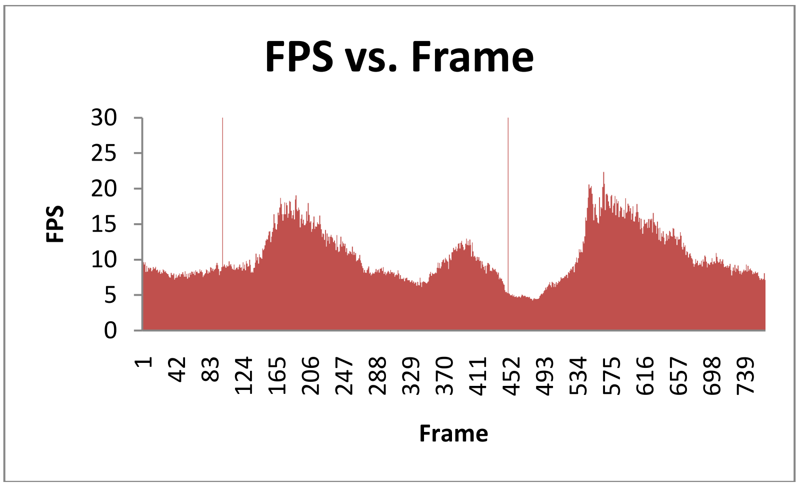
7.2.2 7th Test - ROI Points 2frames

In this test, we used the two previous techniques as sixth test. The basic model here is on ROI for each IP same as 3rd test.
It means after extracting the IP in current frame, an ROI is defined for current, it will be used for previous and two previous frames as well ("$t-1$","$t-2$"). So if an IP doesn't match to a point in previous frame $(t-1)$ then we search in $(t-2)$ ROI frame for new matching.

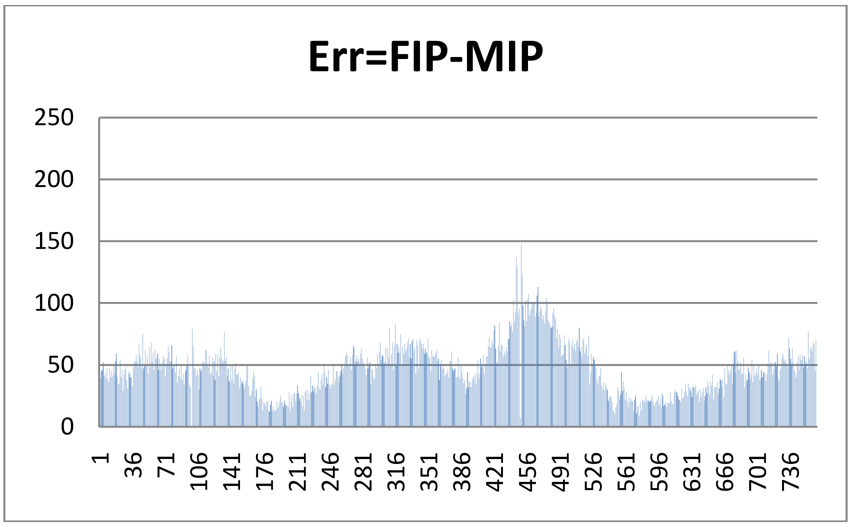
Results
Total extracted IP in 766 frames database = 128,768
Average frames' Interest Points = 168.324183
Average frames' Match Points = 121.4960733
Average FPS = 10.57236191

Output Results





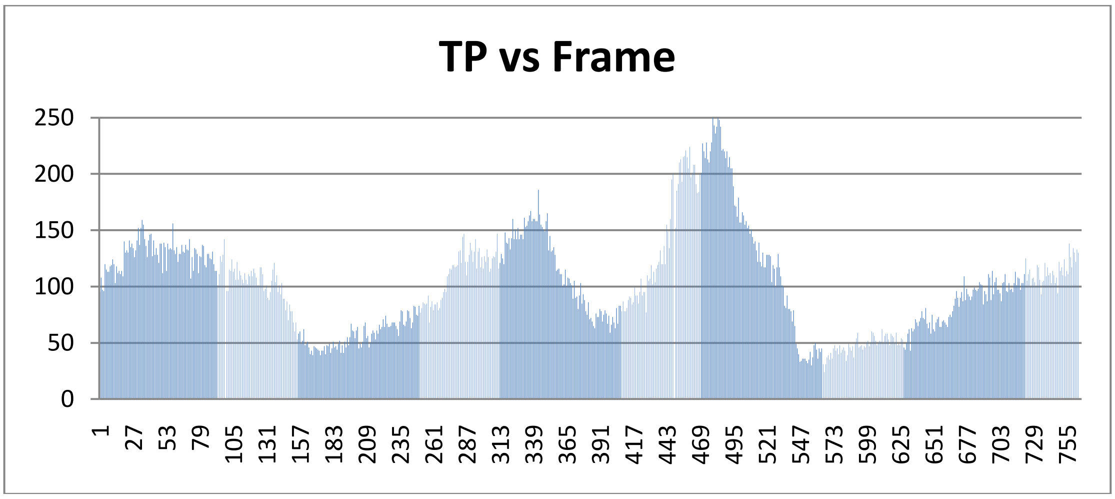
Error Graph

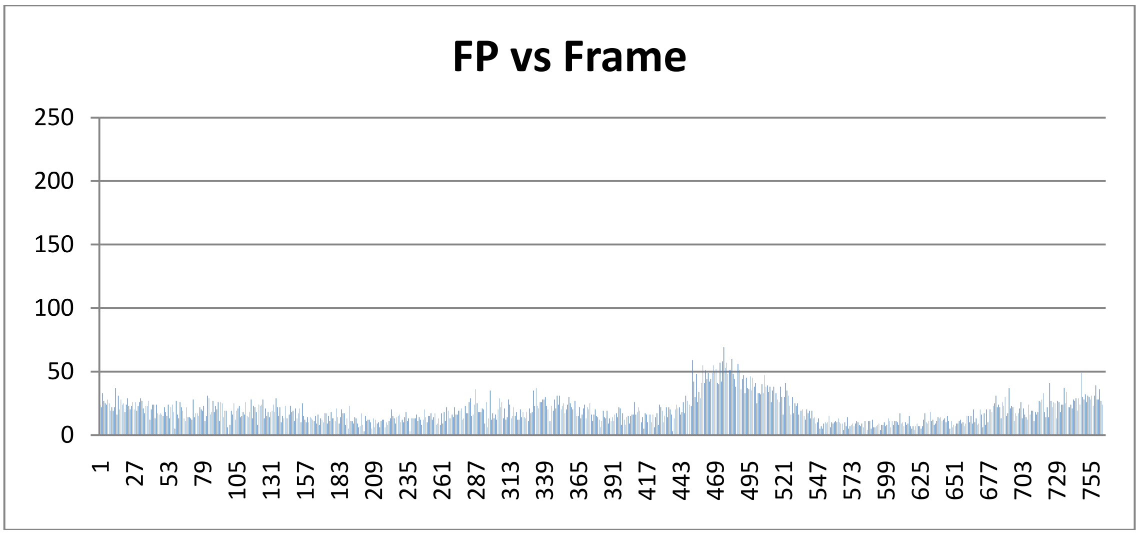


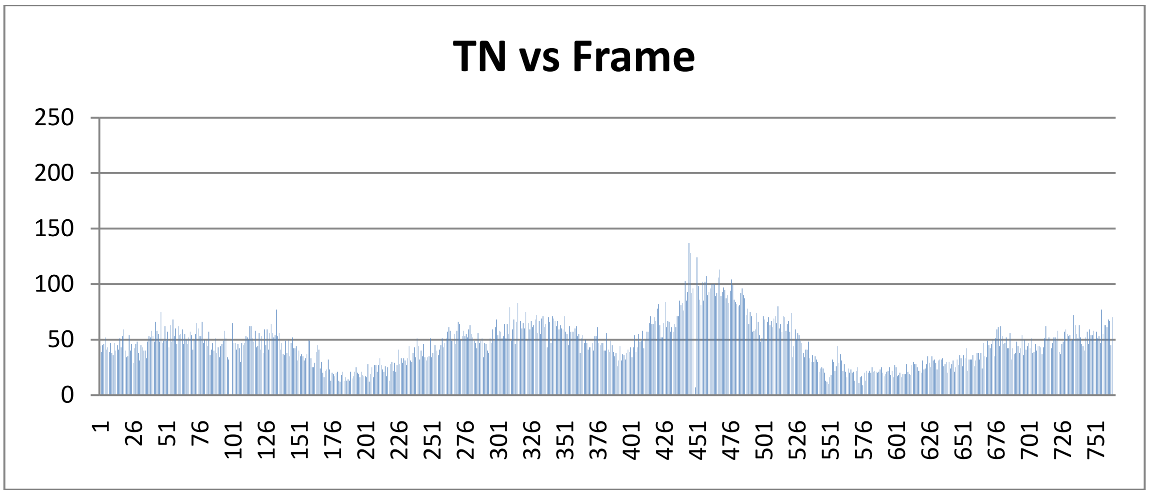
$$\Delta Error \;=\; Frame\ IP - Matched\ IP$$
$$Average\ \Delta Error \;=\; 46.76732026$$

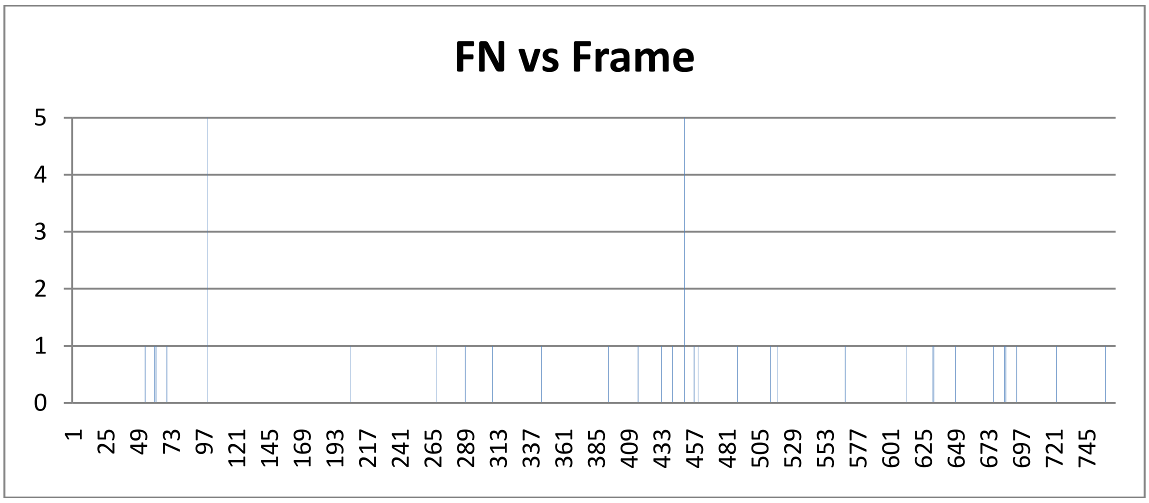
Calculating Error with Fundamental and Homography Matrices (TP, FP, TN and FN)

The average for TP: 102.3795812
The average for FP: 19.11649215
The average for TN: 46.4934555
The average for FN: 0.325916

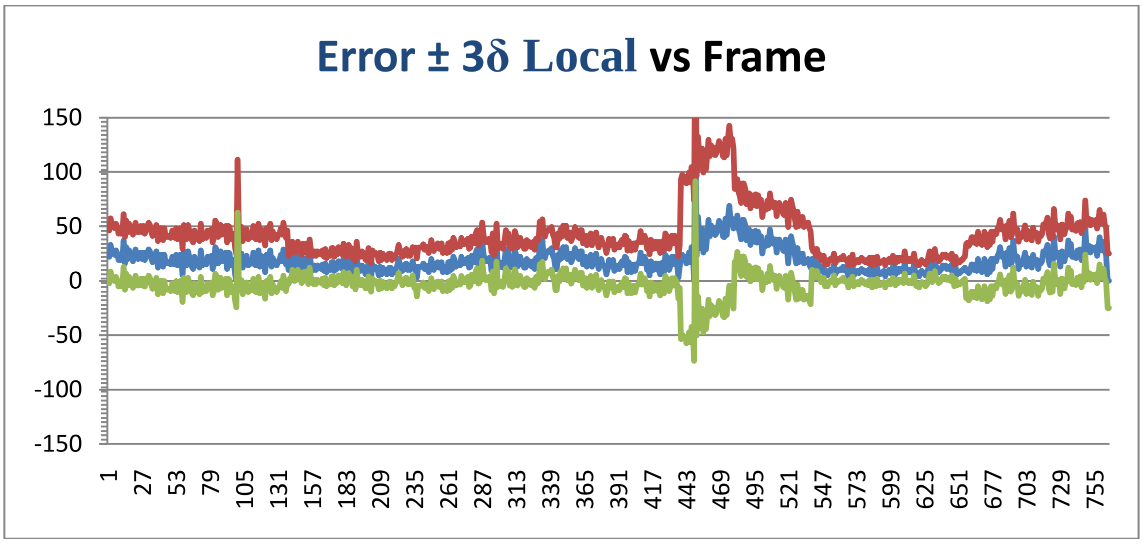



FP vs Frame
250
200
150
100
50
0
1
27
53
79
105
131
157
183
209
235
261
287
313
339
365
391
417
443
469
495
521
547
573
599
625
651
677
703
729
755

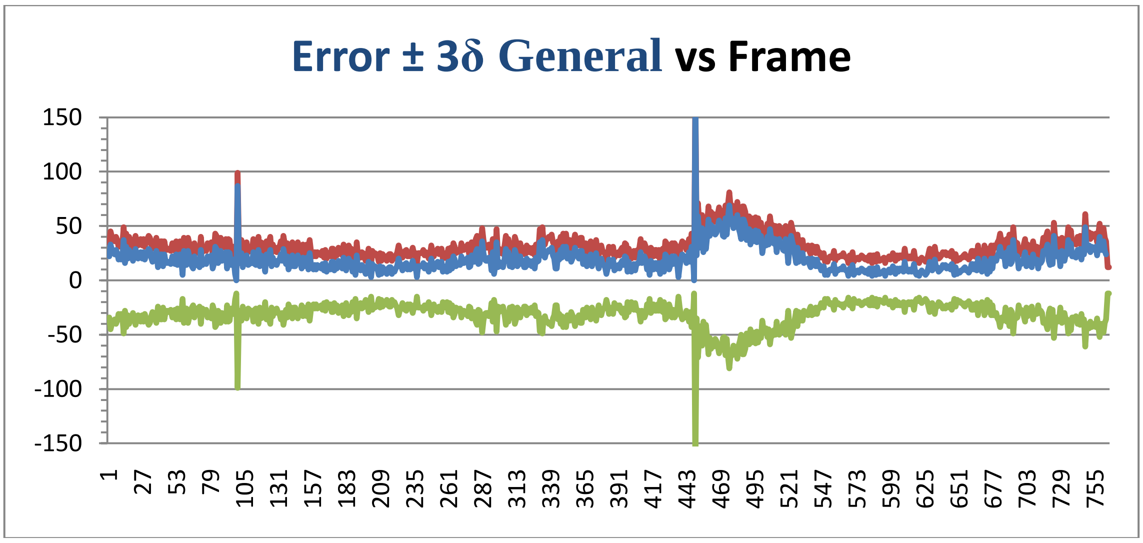

TN vs Frame
250
200
150
100
50
0
1
26
51
76
101
126
151
176
201
226
251
276
301
326
351
376
401
426
451
476
501
526
551
576
601
626
651
676
701
726
751


FN vs Frame
5
4
3
2
1
0
1
25
49
73
97
121
145
169
193
217
241
265
289
313
337
361
385
409
433
457
481
505
529
553
577
601
625
649
673
697
721
745

Calculating err function with local standard deviation of each distribution



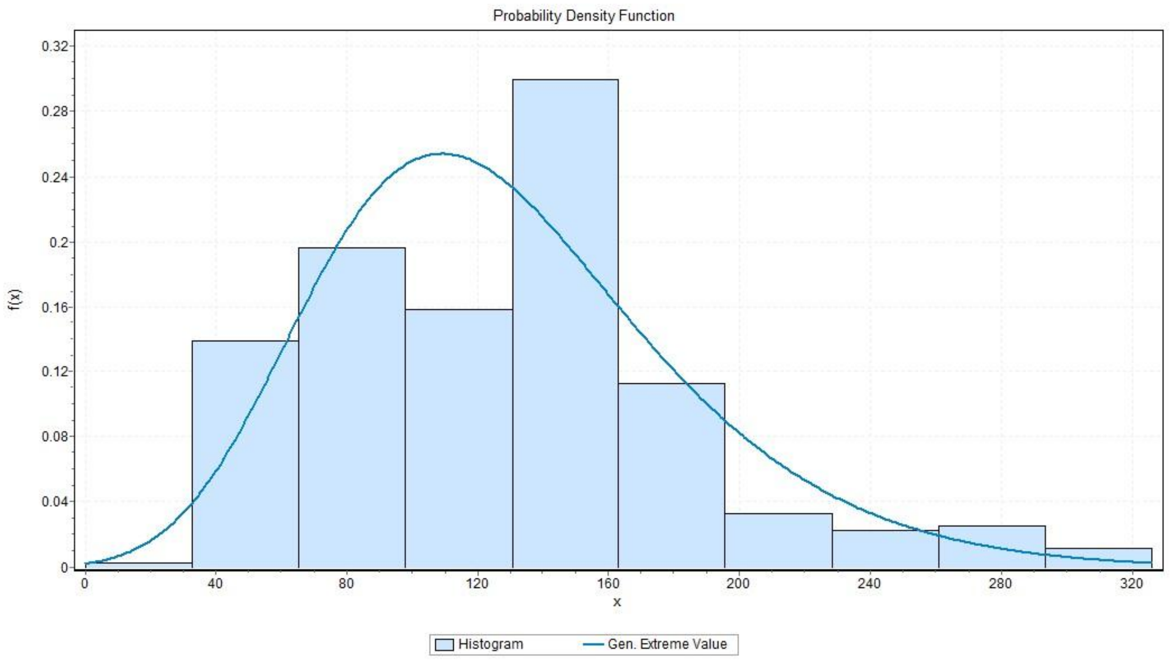
Calculating err function with Overall standard deviation of each distribution



Static Calculation and Graphs

The Matched IP output data are fitted to distribution with Anderson Darling fitting tests.

The best fitted distribution is for Gen. Extreme Value and the parameters are:



$k = -0.08768 \;\; \sigma = 47.445 \;\; \mu = 104.7$

7.3 Set C

8 Discussion, Conclusion and Future Development

9 Bibilography