THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

MULTIPLE VISUAL OBJECT LOCALIZATION

Master's Thesis

ABDULLAH KUZHAN

İSTANBUL, 2012

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

INSTITUTE OF SCIENCE

COMPUTER ENGINEERING DEPARTMENT

MULTIPLE VISUAL OBJECT LOCALIZATION

Master's Thesis

ABDULLAH KUZHAN

Supervisor: ASSISTANT PROFESSOR KEMAL EGEMEN OZDEN

İSTANBUL, 2012

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

INSTITUTE OF SCIENCE
COMPUTER ENGINEERING

Name of the thesis: Mulyiple visual object localization
Name / Last Name of the Students: Abdullah Kuzhan
Date of the Defense of Thesis: 03.08.2012

The thesis has been approved by The Graduate School of Applied Science

Assoc. Prof.Tunç BOZBURA
Graduate School Director
Signature

I certify that this thesis meets all the requirements as a thesis for the degree of Master of Arts.

Asst. Prof. V. Çağrı Güngör
Program coordinator
Signature

This is to certify that we have read this thesis and we find it fully adequate in scope, quality and content, as a thesis for the degree of Master of Arts.

Examining Committee Members	Signature
Thesis Supervisor Asst. Prof. Kemal Egemen Ozden	----------
Member Asst. Prof. Tevfik Aytekin	----------
Member Asst. Prof. Alper Tunga	----------

ABSTRACT

MULTIPLE VISUAL OBJECT LOCALIZATION

Abdullah Kuzhan

Computer Engineering Department

Thesis Supervisor: Assistant Professor Dr. Kemal Egemen Özden

August 2012, 70

This study deals with the multiple object detection in image processing. There are several ways to detect multiple object but we tried a different approach than others. Different than all other matcher algorithms our approach can find multiple same objects. We detected SURF and SIFT keypoint and extracted keypoint descriptors. Applying normalized cross correlation to these descriptors found possible matches. Using local keypoint's scale, rotation and translation informations estimated center point of query image to target image. Using a Window we found local maxima of target image. These local maximas means a matched image of query image.

Keywords: Multiple Object Detection, Image Processing, Object localization

1. INTRODUCTION

Nowadays, there are many augmented reality applications which tries to localize objects and puts some advertisement to scenes. Most of these very famous applications use GPS, gyroscope, accelerometer or some other technologies. These technologies are helpful to understand camera position but they are not enough to object localization.

Gyroscope is a device for measuring or maintaining orientation, based on the principles of angular momentum. Mechanically, a gyroscope is a spinning wheel or disk in which the axle is free to assume any orientation. Although this orientation does not remain fixed, it changes in response to an external torque much less and in a different direction than it would without the large angular momentum associated with the disk's high rate of spin and moment of inertia. Since external torque is minimized by mounting the device in gimbals, its orientation remains nearly fixed, regardless of any motion of the platform on which it is mounted. An accelerometer is a device that measures proper acceleration, also called the four-acceleration. This proper acceleration is associated with the weight of a test mass. For example, an accelerometer on a rocket far from any gravitational influences that is accelerating through space due to the force from its engine, will measure the rate of change of the velocity of the rocket relative to any inertial frame of reference, because such changes require application of a (rocket) force that can be felt (as weight), for any mass. GPS can pass glass or some kind of plastic stuffs but cannot pass mountains or tunnels. Also cannot works correctly at city centers because of skyscrapers. For this reason DGPS has been improved. DGPS uses a network of fixed, ground-based reference stations to broadcast the difference between the positions indicated by the satellite systems and the known fixed positions. These stations broadcast the difference between the measured satellite pseudo ranges and actual (internally computed) pseudo ranges, and receiver stations may correct their pseudo ranges by the same amount. The digital correction signal is typically broadcast locally over ground-based transmitters of shorter range. Even so that GPS or DGPS cannot provide information to localize an object, it just can help us to find where we are.

Other way to localize an object, creating 3d model of interested place and detect object at the model. This can be done by structure from motion approach. Structure from motion (SfM)

refers to the process of finding the three-dimensional structure of an object by analyzing local motion signals over time. Humans perceive a lot of information about the three-dimensional structure in their environment by moving through it. When the observer moves and the objects around him move, information is obtained from images sensed over time. Finding structure from motion presents a similar problem as finding structure from stereo vision. In both instances, the correspondence between images and the reconstruction of 3D object needs to be found. To find correspondence between images, features such as corner points (edges with gradients in multiple directions) need to be tracked from one image to the next. The feature trajectories over time are then used to reconstruct their 3D positions and the camera's motion.

Many problems of computer vision can be solved by finding point correspondences between images using local features. Examples are object recognition, depth reconstruction and self-localization. The first step in computing local features consists of detecting salient locations such as corners, blobs etc. From the neighborhood regions of these interest points, image features are then calculated, yielding a descriptor for each one.

Object localization is an important problem in computer vision. There are lots of ways to detect objects and localize them. Simply detecting interested points from query image and trying to match them to training images interested points. Absolutely there are some matcher algorithms too (Flann, Knn, BruteForce etc.).

Our approach can be baseline of other technologies and methods. We decided to detect object and localize them using local image features. Any Gyroscope, accelerometer or GPS based applications can use our model to localize objects. Also structure from motion type applications basically detects features and tries to match them, so these applications can use our approach too.

We intended to scale, rotation and translation invariant object localization using local image features. The analysis of image features such as size (scale), or angle (rotation) provides significant cues in the process of image understanding. Each local image feature has own coordinate system, rotation and scale information.

Several applications have been developed using image descriptors. For instance famous CBIR system tineye.com can search images with url or uploaded an image file. EBay has more like this application which can detect object look likes to each other. Using Bing image search you

can filter images just have faces. There are many other applications image stitching, augmented reality, content based image retrieval and so on.

Our aim was to develop a mobile Application which will be detect a cinema poster at real time. When people went to the cinema always checks cinema posters. Our Application would be detect poster and got information from IMDB or some other useful sites with a mobile phone. Sadly we could not implement a mobile Application but however we developed a c++ Application which has been tested with many different ways and hopefully results was satisfactory.

We choose this approach because of any other matching techniques tries to find best match and it cannot detect multiple same objects. Our approach widely detects objects even if they are multiple same objects. Mid-point estimation is a new method to detect object and localize them and proudly we invented it. We are going to introduce this new method and explain it widely in this research.

We used OpenCV (open source computer vision) in this research. OpenCV has lots of ways to match objects. SURF matcher, SIFT matcher, BruteForce matcher and some others. All of these methods cannot match multiple objects. By design all of them try to find best match for a query image descriptor and best match's mean only one match will be made. We obligated to develop a new algorithm to locate multiple objects.

1.1 OPENCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision, developed by Intel, and now supported by Willow Garage and Itseez. It is free for use under the open source BSD license. The library is cross-platform. It focuses mainly on real-time image processing. If the library finds Intel's Integrated Performance Primitives on the system, it will use these proprietary optimized routines to accelerate it.

Officially launched in 1999, the OpenCV project was initially an Intel Research initiative to advance CPU-intensive applications, part of a series of projects including real-time ray tracing and 3D display walls. The main contributors to the project included a number of optimization experts in Intel Russia, as well as Intel's Performance Library Team. In the early days of OpenCV, the goals of the project were described as

i. Advance vision research by providing not only open but also optimized code for basic vision infrastructure. No more reinventing the wheel.
ii. Disseminate vision knowledge by providing a common infrastructure that developers could build on, so that code would be more readily readable and transferable.
iii. Advance vision-based commercial applications by making portable, performance-optimized code available for free with a license that did not require being open or freeing them.

Applications;

a) 2D and 3D feature toolkits
b) Egomotion estimation
c) Facial recognition system
d) Gesture recognition
e) Human–computer interaction (HCI)
f) Mobile robotics
g) Motion understanding
h) Object identification
i) Segmentation and Recognition
j) Stereopsis Stereo vision: depth perception from 2 cameras
k) Structure from motion (SFM)
l) Motion tracking
m) To support some of the above areas, OpenCV includes a statistical machine learning library that contains:
n) Boosting
o) Decision tree learning
p) Gradient boosting trees
q) Expectation-maximization algorithm
r) k-nearest neighbor algorithm
s) Naive Bayes classifier
t) Artificial neural networks
u) Random forest
v) Support vector machine (SVM)

1.1.1 Detecting the Scale-Invariant SURF Features

When trying to match features across different images, we are often faced with the problem of scale changes. That is, the different images to be analyzed can be taken at a different distance from the objects of interest, and consequently, these objects will be pictured at different sizes. If we try to match the same feature from two images using a fixed size neighborhood then, because of the scale change, their intensity patterns will not match.

To solve this problem, the concept of scale-invariant features has been introduced in computer vision. The main idea here is to have a scale factor associated with each of the detected feature points. In recent years, several scale-invariant features have been proposed and this recipe presents one of them, the SURF features. SURF stands for Speeded up Robust Features, and as we will see, they are not only scale-invariant features, but they also offer the advantage of being computed very efficiently.

OpenCV has a function called cv::drawKeypoints to draw keypoints.

```
// Draw the keypoints with scale and orientation information
cv::drawKeypoints(image, // original image
keypoints, // vector of keypoints
featureImage, // the resulting image
cv::Scalar(255,255,255), // color of the points
cv::DrawMatchesFlags::DRAW_RICH_KEYPOINTS); //flag
```

Table 1.1: Keypoint Drawing Parameters

image	source image
keypoints	Keypoints from source image
featureImage	Its content depends on flags value what is drawn in output image
scalar	Color of keypoints

Flags	Default=0->For each keypoint only the center point will be drawn (without the circle around keypoint with keypoint size and orientation.
	DRAW_OVER_OUTIMG =1->Matches will be drawn on existing content of output image.
	NOT_DRAW_SINGLE_POINTS = 2, // Single keypoints will not be drawn.
	DRAW_RICH_KEYPOINTS = 4 // For each keypoint the circle around keypoint with keypoint size and orientation will be drawn.

The resulting image with the detected feature that is produced by the drawing function is:

Figure 1.1: Drawing keypoints.

As can be seen in the preceding screenshot, the size of the keypoint circles resulting from the use of the DRAW_RICH_KEYPOINTS flag is proportional to the computed scale of each feature. The SURF algorithm also associates an orientation with each feature to make them rotation-invariant. This orientation is illustrated by a radial line inside each drawn circle.

If we take another picture of the same object but at a different scale, the feature detection results in:

Figure 1.2: Scaled keypoints.

By carefully observing the detected keypoints, it can be seen that the change in size of corresponding circles is proportional to the scale change. As an example, consider the bottom part of the upper-right window. In both images, a SURF feature has been detected at that

location and the two corresponding circles (of different sizes) contain the same visual elements.

Image derivatives of an image can be estimated using Gaussian filters. Those filters make use of an σ parameter defining the aperture (size) of the kernel. As we know, this σ corresponds to the variance of the Gaussian function used to construct the filter, and it then implicitly defines a scale at which the derivative is evaluated. Indeed, a filter having a larger σ value smoothed out the finer details of the image. This is why we can say that it operates at a coarser scale.

Now, if we compute, for instance, the Laplacian of a given image point using Gaussian filters at different scales, and then different values are obtained. Looking at the evolution of the filter response for different scale factors, we obtain a curve which eventually reaches a maximum value at some σ value. If we extract this maximum value for two images of the same object taken at two different scales, the ratio of these two σ maxima will correspond to the ratio of the scales at which the images were taken. This important observation is at the core of the scale-invariant feature extraction process. That is, scale-invariant features should be detected as local maxima in both the spatial space (in the image) and the scale space (as obtained from the derivative filters applied at different scales).

SURF implements this idea by proceeding as follows. First, to detect the features, the Hessian matrix is computed at each pixel. This matrix measures the local curvature of a function and is defined as:

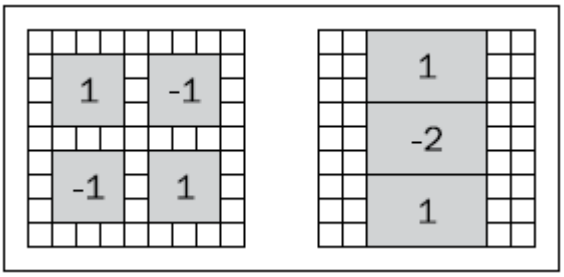
$$\mathrm{H}(\mathrm{x},\mathrm{y}) = \begin{bmatrix} \frac{\delta^2 I}{\delta x^2} & \frac{\delta^2 I}{\delta x \delta y} \\ \frac{\delta^2 I}{\delta x \delta y} & \frac{\delta^2 I}{\delta y^2} \end{bmatrix} \qquad (1.1)$$

The determinant of this matrix gives the strength of this curvature. The idea is therefore to define corners as image points with high local curvature (that is, high variation in more than one direction). Since it is composed of second-order derivatives, this matrix can be computed using Laplacian Gaussian kernels of different scale σ. This Hessian then becomes a function of three variables: H(x,y,σ). A scale-invariant feature is therefore declared when the determinant of this Hessian reaches a local maximum in both spatial and scale space (that is,

3x3x3 non-maxima suppression needs to be performed). However, this determinant must have a minimum value as specified by the first parameter in the constructor of the cv::SurfFeatureDetector class.

The calculation of all of these derivatives at different scales is computationally costly. The objective of the SURF algorithm is to make this process as efficient as possible. This is achieved by using approximated Gaussian kernels involving only few integer additions. These have the following structure:

Figure 1.3: Surf kernels



The kernel on the left is used to estimate the mixed second derivatives, while the right one estimates the second derivative in the vertical direction. A rotated version of this second kernel estimates the second derivative in the horizontal direction. The smallest kernels have a size of 9x9 pixels corresponding to $\sigma \approx 1.2$. Kernels of increasing size are successively applied. The exact amount of filter that is applied can be specified by additional parameters of the SURF class. By default, 12 different sizes of kernels are used (going up to size 99x99). Note that the fact that integral images are used guarantees that the sum inside each lob can be computed by using only 3 additions independently of the size of the filter.

Once the local maxima is identified, the precise position of each detected interest point is obtained through interpolation in both scale and image space. The result is then a set of feature points localized at sub-pixel accuracy and to which is associated a scale value.

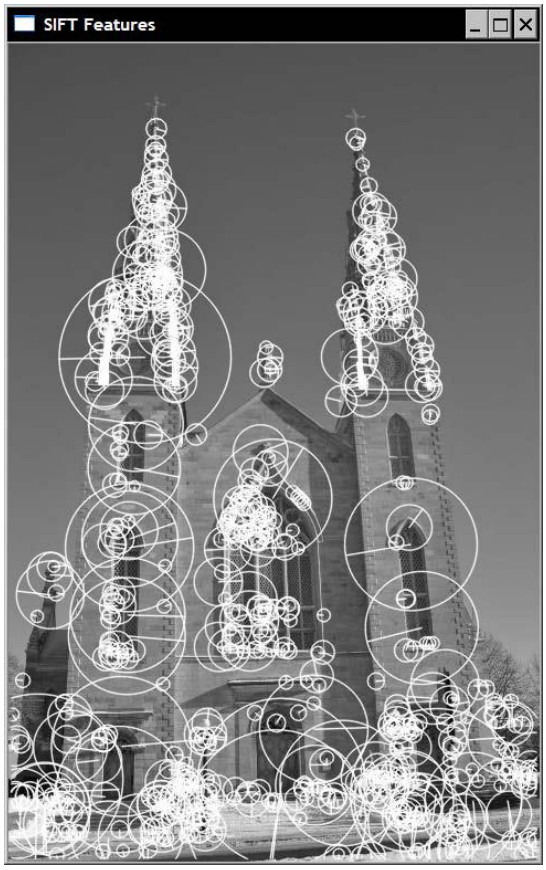
The SURF algorithm has been developed as an efficient variant of another well-known scale invariant feature detector called SIFT (for Scale-Invariant Feature Transform). SIFT also detects features as local maxima in image and scale space, but uses the Laplacian filter response instead of the Hessian determinant. This Laplacian at different scales is computed

using difference of Gaussian filters. OpenCV has a wrapper class that detects these features and it is called in a way similar to the SURF features:

```
// vector of keypoints

std::vector<cv::KeyPoint> keypoints;

// Construct the SURF feature detector object

cv::SiftFeatureDetector sift(

0.03, // feature threshold

10.); // threshold to reduce

// sensitivity to lines

// Detect the SURF features

sift.detect (image, keypoints);
```

The results are also very similar:

Figure 1.4: Drawing keypoints.



However, since the computation of the feature point is based on floating-point kernels, it is generally considered to be more accurate in terms of feature localization in space and scale. Although, for the same reason, it is also more computationally expensive.

1.1.2 Describing SURF Features

The SURF algorithm, discussed in the preceding recipe, defines a location and a scale for each of the detected features. This scale factor can be used to define the size of a window around the feature point such that the defined neighborhood would include the same visual information no matter what scale the object to which the feature belongs has been pictured. In addition, the visual information included in this neighborhood can be used to characterize the feature point to make it distinguishable from the others.

This recipe will show you how to describe a feature point's neighborhood using compact descriptors. In feature matching, feature descriptors are usually N-dimensional vectors that describe a feature point, ideally in a way that is invariant to change in lighting and to small perspective deformations. In addition, good descriptors can be compared using a simple distance metric (for example, Euclidean distance). Therefore, they constitute a powerful tool to use in feature matching algorithms.

The following code is a pattern similar to the one used for feature detection. OpenCV 2 proposes a general class which defines a common interface for the extraction of the various feature point descriptors that are available. To follow up on the preceding recipe, here we use the one proposed in the SURF algorithm. Based on the std::vector of cv::Keypoint instances obtained from feature detection, the descriptors are obtained as follows:

```
// Construction of the SURF descriptor extractor

cv::SurfDescriptorExtractor surfDesc;

// Extraction of the SURF descriptors

cv::Mat descriptors1;

surfDesc.compute (image1, keypoints1, descriptors1);
```

The result is a matrix (that is, a cv::Mat instance) which will contain as many rows as the number of elements in the keypoint vector. Each of these rows is an N- dimensional descriptor vector. In the case of the SURF descriptor, by default, it has a size of 64. This vector characterizes the intensity pattern surrounding a feature point. The more similar the two feature points, the closer their descriptor vectors should be.

These descriptors are particularly useful in image matching. Suppose, for example, that two images of the same scene are to be matched. This can be accomplished by first detecting features on each image, and then extracting the descriptors of these features. Each feature descriptor vector in the first image is then compared to all feature descriptors in the second image. The pair that obtains the best score (that is, the lowest distance between the two vectors) is then kept as the best match for that feature. This process is repeated for all features in the first image. This is the most basic scheme that has been implemented in OpenCV as the cv::BruteForceMatcher. It is used as follows:

```
// Construction of the matcher

cv::BruteForceMatcher<cv::L2<float>> matcher;

// Match the two image descriptors

std::vector<cv::DMatch> matches;

matcher.match(descriptors1, descriptors2, matches);
```

This class is a subclass of the cv::DescriptorMatcher class defining the common
interface for different matching strategies. The result is a vector of cv::DMatch instances which is the structure used to represent a match pair. Essentially, the cv::DMatch data structure contains a first index referring to an element in the first vector of descriptors, and a second index referring to the matching feature in the second vector of descriptors. It also contains a real value representing the distance between the two matched descriptors. This distance value is used in the definition of operator< comparing two cv::DMatch instances.

```
cv::Mat imageMatches;

cv::drawMatches(

image1, keypoints1, // 1st image and its keypoints
```

```
image2, keypoints2, // 2nd image and its keypoints
Matches, // the matches
ImageMatches, // the image produced
cv::Scalar (255,255,255)); // color of the lines
```

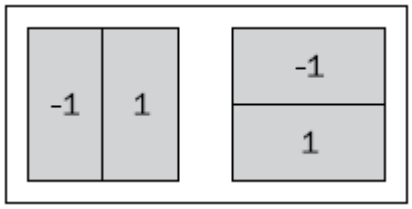
That produces the following image:

Figure 1.5: Matched keypoints

As can be seen, most of these matches correctly link a point on the left with its corresponding image point on the right. One can notice some errors due to the fact that the observed building has a symmetrical façade which makes some of the local matches ambiguous (the topmost match is one example of wrongly matched features).

Good feature descriptors must be invariant to small changes in illumination, in viewpoint, and to the presence of image noise. Therefore, they are often based on local intensity differences. This is the case of the SURF descriptors which apply the following simple kernels inside a larger neighborhood around a keypoint:

Figure 1.6: Surf kernels



The first one simply measures the local intensity difference in the horizontal direction(designated as dx), and the second measures this difference in the vertical direction (designated as dy). The size of the neighborhood used to extract the descriptor vector is defined as 20 times the scale factor of the feature (that is, 20σ). This square region is then split into 4x4 smaller square sub-regions. For each sub-region, the kernel responses dx and dy are computed at 5x5 regularly spaced locations (the kernel size being 2σ). All of these responses are summed as follows in order to extract four descriptor values for each sub region:

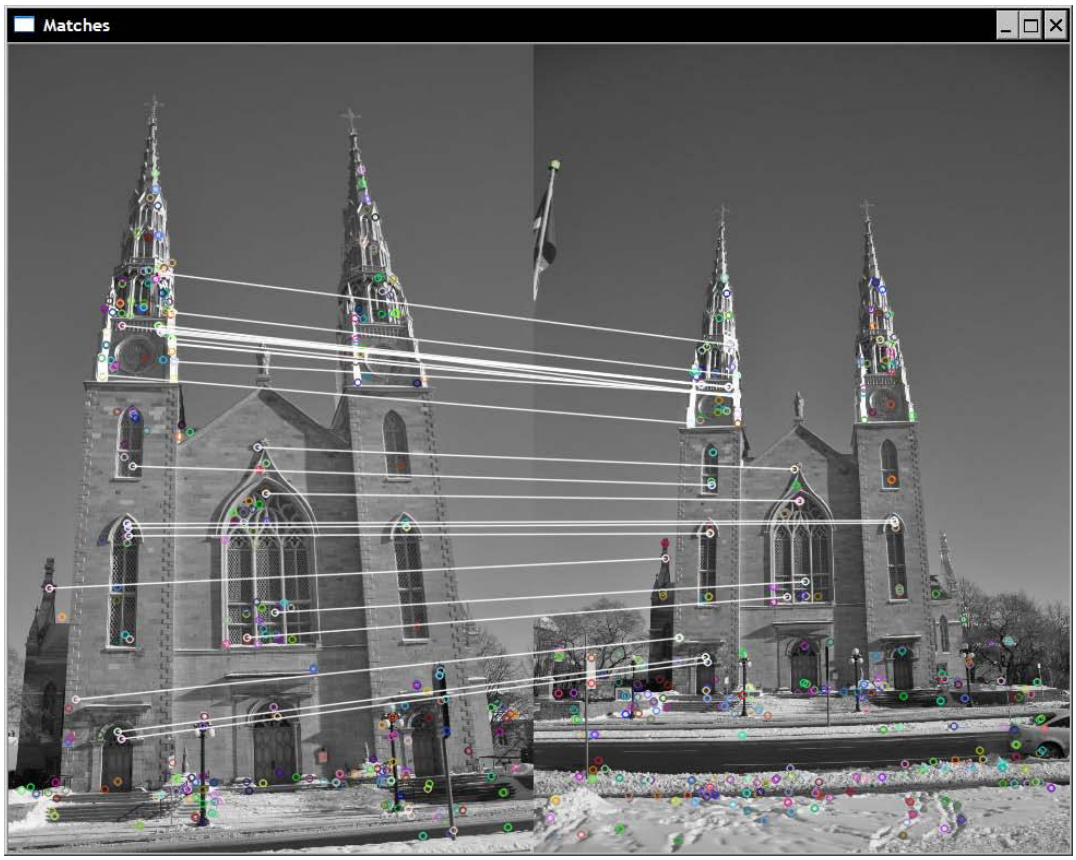
$$[\ \sum dx\ \sum dy\ \sum |dx|\ \sum |dy|\ ] \qquad (1.2)$$

Since there are 4x4=16 sub-regions, we have a total of 64 descriptor values. Note that in order to give more importance to the neighboring pixel values closer to the keypoint, the kernel responses are weighted by a Gaussian centered at the keypoint location (with a σ=3.3).

The dx and dy responses are also used to estimate the orientation of the feature. These values are computed (with a kernel size of 4σ) within a circular neighborhood of radius 6σ at locations regularly spaced by intervals of σ. For a given orientation, the responses inside a certain angular interval (π/3) are summed, and the orientation giving the longest vector is defined as the dominant orientation.

With the SURF features and descriptors, scale-invariant matching can be achieved. Here is an example showing the matches in a match pair containing two images at different scales:

Figure 1.7: SURF matches in different scales.



The SIFT algorithm also defines its own descriptor. It is based on the gradient magnitude and orientation computed at the scale of the considered keypoint. As for the SURF descriptors, the scaled neighborhood of the keypoint is divided into 4x4 sub-regions. For each of these regions, an 8-bin histogram of gradient orientations (weighted by their magnitude and by a global Gaussian window centered at the keypoint) is built. Therefore, the descriptor vector is made of the entries of these histograms. There are 4x4 regions and 8 bins per histogram, which leads to a descriptor of length 128.

As for feature detection, the difference between SURF and SIFT descriptors is mainly speed and accuracy. Since SURF descriptors are mostly based on intensity differences, they are faster to compute. However, SIFT descriptors are generally considered to be more accurate in finding the right matching feature.

1.2 HOMOGRAPHY

A 2D point (x; y) in an image can be represented as a 3D vector x =(x1; x2; x3) where x = x1/x3 and y = x2/x3. This is called the homogeneous representation of a point and it lies on the projective plane P2. A homography is an invertible mapping of points and lines on the projective plane P2. Other terms for this transformation include collineation, projectivity, and planar projective transformation. Hartley and Zisserman provide the specific definition that a homography is an invertible mapping from P2 to itself such that three points lie on the same line if and only if their mapped points are also collinear. They also give an algebraic definition by proving the following theorem: A mapping from P2 → P2 is a projectivity if and only if there exists a non-singular 3x3 matrix H such that for any point in P2 represented by vector x it is true that its mapped point equals Hx. This tells us that in order to calculate the homography that maps each xi to its corresponding x0 i it is sufficient to calculate the 3x3 homography matrix, H.

It should be noted that H can be changed by multiplying by an arbitrary non-zero constant without altering the projective transformation. Thus H is considered a homogeneous matrix and only has 8 degrees of freedom even though it contains 9 elements. This means there are 8 unknowns that need to be solved for.

Typically, homographies are estimated between images by finding feature correspondences in those images. The most commonly used algorithms make use of point feature correspondences, though other features can be used as well, such as lines or conics.

1.2.1 Relation to Other Geometric Transformations

One good way to understand homographies is to put them into the context of other geometric transformations. The homography transformation has 8 degrees of freedom and there are other simpler transformations that still use the 3x3 matrix but contain specific constraints to reduce the number of degrees of freedom. This section presents a hierarchy of transformations leading to the homography and will show how homographies can be broken down into an aggregation of these simpler transformations.

1.2.1.1 Isometry

An isometry is a transformation that preserves Euclidian distance. This means that the distance between two points in one image will be the same as the distance between their corresponding points in the mapped image. The same goes for the angles between lines and areas. Isometries are made up of only 2D rotations and 2D translations and therefore have only 3 degrees of freedom. An isometry can be written as:

$$x' = \begin{pmatrix} R & t \\ 0^T & 1 \end{pmatrix} x \qquad (1.3)$$

where R is a 2x2 rotation matrix, t is a translation 2-vector and 0^T is a row of 2 zeros.

1.2.1.2 Similarity Transformation

A similarity transform is similar to an isometry except it also contains isotropic scaling. Isotropic means that the scaling is invariant with respect to direction. The scale adds an additional degree of freedom so a similarity transform contains 4 degrees of freedom overall. Like with isometries, angles are not affected by this transformation. The distance between points are no longer invariant, but the ratio of distances is preserved under similarity transformations since any scale change cancels out. A similarity transform can be written as:

$$x' = \begin{pmatrix} sR & t \\ 0^T & 1 \end{pmatrix} x \qquad (1.4)$$

Where s is a scalar and represents the isotropic scaling.

1.2.1.3 Affine Transformation

An affine transformation is like a similarity transform but instead of a single rotation and isotropic scaling it is a composition of two rotations and two non-isotropic scalings. It contains two more degrees of freedom than the similarity transformation; one for the angle specifying the scaling direction and one for the ratio of the scaling parameters. Unlike the similarity transformation, an affine transformation does not preserve the distance ratios or the angles between lines. There still are some invariants though, as parallel lines in one image remain parallel in the mapped image, and the ratios of lengths of parallel line segments and areas are preserved. An affine transformation can be written as:

$$x' = \begin{pmatrix} A & t \\ 0^T & 1 \end{pmatrix} x \qquad (1.5)$$

Where A is a 2x2 non-singular matrix.
A can be decomposed as:

$$A = R(\Theta)R(-\phi)DR(\phi) \qquad (1.6)$$

Where R(Θ) and R(ϕ) are rotation matrices for Θ and ϕ respectively and D is a diagonal matrix:

$$D = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix} \qquad (1.7)$$

Where λ_1 and λ_2 can be considered as two scaling values.

The matrix A is thus a concatenation of a rotation by ϕ, a scaling by $\lambda_1$in the x direction, a scaling by λ_2 in the y direction, a rotation back by -ϕ and then another rotation by Θ.

1.2.1.4 Projective Transformation

Finally we come to projective transformations or homographies which have already been defined above. The projective transformation is a non-singular linear transformation of homogeneous coordinates. This transformation would be non-linear with inhomogeneous coordinates and this is what makes the use of homogeneous coordinates so valuable. Projective transformations contain two more degrees of freedom than affine transformations as now the matrix has nine elements with only their ratio significant. None of the invariants from the affine transformation mentioned above hold in the projective case, though the fact that if three points lie on the same line in one image, they will be collinear in the other still holds. A projective transformation can be written as:

$$x' = \begin{pmatrix} A & t \\ 0^T & v \end{pmatrix} x \qquad (1.8)$$

Where $v = (v1, v2)^T$.

The key difference between the affine and projective transformation is the vector v, which is null in the affine case. This vector is responsible for the non-linear effects of the projectivity. For affinities, the scalings from A are the same everywhere in the plane, while for projectivities scaling varies with the position in the image. Similarly, for affinities the orientation of a transformed line depends only on the orientation of the original line while for projectivities the position of the original line on the plane also effects the transformed line's orientation.

A projective transformation can be decomposed into a chain of the previously mentioned transformations:

$$H = HSHAHP = \begin{pmatrix} sR & t \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} U & 0 \\ 0^T & 1 \end{pmatrix} \begin{pmatrix} I & 0 \\ v^T & v \end{pmatrix} = \begin{pmatrix} A & t \\ v^T & v \end{pmatrix} \qquad (1.9)$$

Here HS represents a similarity transformation, HA represents an affinity and HP represents projectivity. A = sRU + tv^T and U is an uppertriangular matrix normalized as det U = 1. For this decomposition to be valid, v cannot equal 0. If s is selected as positive then this decomposition is unique.

1.2.1.5 Perspective Projection

So far this hierarchy has dealt with 2D to 2D (or plane to plane) transformations. Another transformation that is widely studied is perspective projection which is a projection of 3D points in space to 2D points. This is the projection occurring when cameras take images of the world and display the result on an image plane. A perspective projection can be represented with homogeneous coordinates by a 3x4 camera matrix P such that:

$$x = PX \qquad (1.10)$$

Where x is an image point represented by a homogeneous 3-vector and X is a world point represented by a homogeneous 4-vector. The camera matrix P has 11 degrees of freedom, which is the same as the number of degrees of freedom of a 3x4 matrix defined up to an arbitrary scale. These degrees of freedom, or parameters, can be broken down into two

categories: 5 internal and 6 external parameters. The 5 internal camera parameters are often represented by a matrix K:

$$K = \begin{pmatrix} a_x & s & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \qquad (1.11)$$

Here a_x and $a_y$represent the focal lengths of the camera in terms of pixel dimensions in the x and y directions respectively, (x0; y0) is the principal point on the image plane and s is a skew parameter. The 6 external parameters relate the camera orientation to a world coordinate system and consist of 3 rotations (represented by a 3x3 matrix R) and 3 translations (represented by a 3-vector t). Thus the camera matrix P can be represented as:

$$P = K\,[Rjt] \qquad (1.12)$$

Hartley and Zisserman note that some assumptions can be made about the camera model in order to reduce the number of degrees of freedom. Assuming the camera has square pixels, and thus equal scales in both the x and y directions allows one to set $a_x = a_y = \alpha$. Also in many cases s can be set to 0. Even with making these assumptions, the perspective projection will have 9 degrees of freedom which is one more than a homography which has 8.

1.2.2 Algorithms for Homography Estimation

1.2.2.1 Basic DLT Algorithm

The Direct Linear Transform (DLT) algorithm is a simple algorithm used to solve for the homography matrix H given a sufficient set of point correspondences. Since we are working in homogeneous coordinates, the relationship between two corresponding points x and x' can be re-written as:

$$c\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = H\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \qquad (1.13)$$

Where c is any non-zero constant, $(u \quad v \quad 1)^T$ represents x', $(x \quad y \quad 1)^T$

Represents x, and H = $\begin{pmatrix} h1 & h2 & h3 \\ h4 & h5 & h6 \\ h7 & h8 & h9 \end{pmatrix}$ (1.14)

Dividing the first row of equation (2.1) by the third row and the second row by the third row we get the following two equations:

$$-h1x - h2y - h3 + (h7x + h8y + h9)u = 0 \quad (1.15)$$

$$-h4x - h5y - h6 + (h7x + h8y + h9)u = 0 \quad (1.16)$$

Equations (2.2) and (2.3) can be written in matrix form as:

$$Aih = 0 \quad (1.17)$$

Where Ai = $\begin{pmatrix} -x & -y & -1 & 0 & 0 & 0 & ux & uy & u \\ 0 & 0 & 0 & -x & -y & -1 & vx & vy & vz \end{pmatrix}$ (1.18)

And h=$(h1 \quad h2 \quad h3 \quad h4 \quad h5 \quad h6 \quad h7 \quad h8 \quad h9)^T$ (1.19)

Since each point correspondence provides 2 equations, 4 correspondences are sufficient to solve for the 8 degrees of freedom of H. The restriction is that no 3 points can be collinear (i.e., they must all be in "general position"). Four 2x9 Ai matrices (one per point correspondence) can be stacked on top of one another to get a single 8x9 matrix A. The 1D null space of A is the solution space for h.

In many cases we may be able to use more than 4 correspondences to ensure a more robust solution. However many point correspondences are used, if all of them are exact then A will still have rank 8 and there will be a single homogeneous solution. In practice, there will be

some uncertainty, the points will be inexact and there will not be an exact solution. The problem then becomes to solve for a vector h that minimizes a suitable cost function.

1.2.2.2 Homogeneous Linear Least Squares

We will frequently encounter problems of the form

$$Ax = 0 \qquad (1.20)$$

known as the Homogeneous Linear Least Squares problem. It is similar in appearance to the inhomogeneous linear least squares problem

$$Ax = \qquad (1.21)$$

in which case we solve for x using the pseudo inverse or inverse of A. This won't work with Equation 15. Instead we solve it using Singular Value Decomposition (SVD).
Starting with equation 13 from the previous section, we first compute the SVD of A:

$$A = U\sum V^T = \sum_{i=0}^{9} \sigma_i u_i v_i \qquad (1.22)$$

When performed in Matlab, the singular values σ_i will be sorted in descending order, so σ_9 will be the smallest. There are three cases for the value of σ_9:

i. If the homography is exactly determined, then $\sigma_9 = 0$, and there exists a homography that fits the points exactly.
ii. If the homography is over determined, then $\sigma_9 >= 0$. Here σ_9 represents a "residual" or goodness of fit.
iii. We will not handle the case of the homography being underdetermined.

From the SVD we take the .right singular vector. (a column from V) which corresponds to the smallest singular value, σ_9. This is the solution, h, which contains the coefficients of the homography matrix that best fits the points. We reshape h into the matrix H, and form the equation x2 ~ Hx1.

1.2.2.3 Homogeneous Linear Least Squares Alternate Derivation

Starting again with the equation Ah = 0, the sum squared error can be written as,

$$f(h) = \frac{1}{2}(Ah - 0)^T (Ah - 0) \quad (1.23)$$

$$f(h) = \frac{1}{2}(Ah)^T (Ah) \quad (1.24)$$

$$f(h) = \frac{1}{2}(h)^T (A)^T Ah \quad (1.25)$$

Taking the derivative of if with respect to h and setting the result to zero, we get

$$\frac{d}{dh} f = 0 = \frac{1}{2}(A^T A + (A^T A)^T)h \quad (1.26)$$

$$0 = A^T Ah \quad (1.27)$$

Looking at the Eigen-decomposition of $A^T A$, we see that h should equal the eigenvector of $A^T A$ that has an eigenvalue of zero (or, in the presence of noise the eigenvalue closest to zero). This result is identical to the result obtained using SVD, which is easily seen from the following fact,

Fact 1 given a matrix A with SVD decomposition $A = U\sum V^T$, the columns of V correspond to the eigenvectors of $A^T A$.

2. LITERATURE REVIEW

2.1 OBJECT RECOGNITION FROM LOCAL SCALE-INVARIANT FEATURES

An object recognition system has been developed that uses a new class of local image features. The features are invariant to image scaling, translation, and rotation, and partially invariant to illumination changes and affine or 3D projection. These features share similar properties with neurons in inferior temporal cortex that are used for object recognition in primate vision. Features are efficiently detected through a staged filtering approach that identifies stable points in scale space. Image keys are created that allow for local geometric deformations by representing blurred image gradient sin multiple orientation planes and at multiple scales. The keys are used as input to a nearest-neighbor indexing method that identifies candidate object matches. Final verification of each match is achieved by finding a low-residual least-squares solution for the unknown model parameters. Experimental results show that robust object recognition can be achieved in cluttered partially-occluded images with a computation time of fewer than 2 seconds.

Object recognition is widely used in the machine vision industry for the purposes of inspection, registration, and manipulation. However, current commercial systems for object recognition depend almost exclusively on correlation-based template matching. While very effective for certain engineered environments, where object pose and illumination are tightly controlled, template matching becomes computationally infeasible when object rotation, scale, illumination, and 3D pose are allowed to vary, and even more so when dealing with partial visibility and large model databases.

An alternative to searching all image locations for matches is to extract features from the image that are at least partially invariant to the image formation process and matching only to those features. Many candidate feature types have been proposed and explored, including line segments , groupings of edges, and regions, among many other proposals. While these features have worked well for certain object classes, they are often not detected frequently enough or with sufficient stability to form a basis for reliable recognition.

There has been recent work on developing much denser collections of image features. One approach has been to use a corner detector (more accurately, a detector of peaks in local image variation) to identify repeatable image locations,

Around which local image properties can be measured. Zhang et al used the Harris corner detector to identify feature locations for epipolar alignment of images taken from differing

viewpoints. Rather than attempting to correlate regions from one image against all possible regions in a second image, large savings in computation time were achieved by only matching regions centered at corner points in each image.

For the object recognition problem, Schmid & Mohr also used the Harris corner detector to identify interest points, and then created a local image descriptor at each interest point from an orientation-invariant vector of derivative-of-Gaussian image measurements. These image descriptors were used for robust object recognition by looking for multiple matching descriptors that satisfied object based orientation and location constraints. This work was impressive both for the speed of recognition in a large database and the ability to handle cluttered images.

The corner detectors used in these previous approaches have a major failing, which is that they examine an image at only a single scale. As the change in scale becomes significant, these detectors respond to different image points. Also, since the detector does not provide an indication of the object scale, it is necessary to create image descriptors and attempt matching at a large number of scales. This paper describes an efficient method to identify stable key locations in scale space. This means that different scalings of an image will have no effect on the set of key locations selected.

Furthermore, an explicit scale is determined for each point, which allows the image description vector for that point to be sampled at an equivalent scale in each image. A canonical orientation is determined at each location, so that matching can be performed relative to a consistent local 2D coordinate frame. This allows for the use of more distinctive image descriptors than the rotation-invariant ones used by Schmid and Mohr, and the descriptor is further modified to improve its stability to changes in affine projection and illumination.

Other approaches to appearance-based recognition include eigen space matching, color histograms , and receptive field histograms . These approaches have all been demonstrated successfully on isolated objects or presegmented images, but due to their more global features it has been difficult to extend them to cluttered and partially occluded images. Ohba & Ikeuchi successfully apply the eigen space approach to cluttered images by using many small local eigen-windows, but this then requires expensive search for each window in a new image, as with template matching.

2.1.1 Key Localization

We wish to identify locations in image scale space that are invariant with respect to image translation, scaling, and rotation, and are minimally affected by noise and small distortions. Lindeberg has shown that under some rather general assumptions on scale invariance, the Gaussian kernel and its derivatives are the only possible smoothing kernels for scale space analysis.

To achieve rotation invariance and a high level of efficiency, we have chosen to select key locations at maxima and minima of a difference of Gaussian function applied in scale space. This can be computed very efficiently by building an image pyramid with resampling between each level. Furthermore, it locates key points at regions and scales of high variation, making these locations particularly stable for characterizing the image. Crowley & Parker and Lindeberg have previously used the DoG (difference of Gaussian) in scale space for other purposes. In the following, we describe a particularly efficient and stable method to detect and characterize the maxima and minima of this function.

As the 2D Gaussian function is separable, its convolution with the input image can be efficiently computed by applying two passes of the 1D Gaussian function in the horizontal and vertical directions.

For key localization, all smoothing operations are done using $\sigma = \sqrt{2}$, which can be approximated with sufficient accuracy using a 1D kernel with 7 sample points.

2.1.1.1 SIFT Key Stability

To characterize the image at each key location, the smoothed image A at each level of the pyramid is processed to extract image gradients and orientations. At each pixel, Aij, the image gradient magnitude, Mij, and orientation, Rij, are computed using pixel differences.

The pixel differences are efficient to compute and provide sufficient accuracy due to the substantial level of previous smoothing. The effective half-pixel shift in position is compensated for when determining key location.

Robustness to illumination change is enhanced by thresholding the gradient magnitudes at a value of 0.1 times the maximum possible gradient value. This reduces the effect of a change

in illumination direction for a surface with 3D relief, as an illumination change may result in large changes to gradient magnitude but is likely to have less influence on gradient orientation.

The stability of the resulting keys can be tested by subjecting natural images to affine projection, contrast and brightness changes, and addition of noise. The location of each key detected in the first image can be predicted in the transformed image from knowledge of the transform parameters. This framework was used to select the various sampling and smoothing parameters given above, so that maximum efficiency could be obtained while retaining stability to changes.

Table 2.1: For various image transformations applied to a sample of 20 images, this table gives the percent of keys that are found at matching locations and scales (Match %) and that also match in orientation (Ori %).

Image transformation	**Match %**	**Ori %**
A. Increase contrast by 1.2	89.0	86.6
B. Decrease intensity by 0.2	88.5	85.9
C. Rotate by 20 degrees	85.4	81.0
D. Scale by 0.7	85.1	80.3
E. Stretch by 1.2	83.5	76.1
F. Stretch by 1.5	77.7	65.0
G. Add 10% pixel noise	90.3	88.4
H. All of A,B,C,D,E,G.	78.6	71.8

2.1.2 Local Image Description

Given a stable location, scale, and orientation for each key, it is now possible to describe the local image region in a manner invariant to these transformations. In addition, it is desirable to make this representation robust against small shift sin local geometry, such as arise from affine or 3D projection.

One approach to this is suggested by the response properties of complex neurons in the visual cortex, in which a feature position is allowed to vary over a small region while orientation and spatial frequency specificity are maintained. Edelman, Intrator & Poggio have performed experiments that simulated the responses of complex neurons to different 3D views of computer graphic models, and found that the complex cell outputs provided much better discrimination than simple correlation-based matching. This can be seen, for example, if an affine projection stretches an image in one direction relative to another, which changes the relative locations
of gradient features while having a smaller effect on their orientations and spatial frequencies.

The SIFT features improve on previous approaches by being largely invariant to changes in scale, illumination, and local affine distortions. The large number of features in a typical image allow for robust recognition under partial occlusion in cluttered images. A final stage that solves for affine model parameters allows for more accurate verification and pose determination than in approaches that rely only on indexing.

An important area for further research is to build models from multiple views that represent the 3D structure of objects. This would have the further advantage that keys from multiple viewing conditions could be combined into a single model, thereby increasing the probability of finding matches in new views. The models could be true 3D representations based on structure-from-motion solutions, or could represent the space of appearance in terms of automated clustering and interpolation (Pope & Lowe). An advantage of the latter approach is that it could also model non-rigid deformations.

2.2 DISTINCTIVE IMAGE FEATURES FROM SCALE-INVARIANT KEYPOINTS

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. This paper describes image features that have many properties that make them suitable for matching differing images of an object or scene. The features are invariant to image scaling and rotation, and partially invariant to change in

illumination and 3D camera viewpoint. They are well localized in both the spatial and frequency domains, reducing the probability of disruption by occlusion, clutter, or noise. Large numbers of features can be extracted from typical images with efficient algorithms. In addition, the features are highly distinctive, which allows a single feature to be correctly matched with high probability against a large database of features, providing a basis for object and scene recognition.

The cost of extracting these features is minimized by taking a cascade filtering approach, in which the more expensive operations are applied only at locations that pass an initial test. Following are the major stages of computation used to generate the set of image features:

a) **Scale-space extreme 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.
b) **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.
c) **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.
d) **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.

The development of image matching by using a set of local interest points can be traced back to the work of Moravec (1981) on stereo matching using a corner detector. The Moravec detector was improved by Harris and Stephens (1988) to make it more repeatable under small image variations and near edges. Harris also showed its value for efficient motion tracking and 3D structure from motion recovery (Harris, 1992), and the Harris corner detector has since been widely used for many other image matching tasks. While these feature detectors are usually called corner detectors, they are not selecting just corners, but rather any image location that has large gradients in all directions at a predetermined scale.

The initial applications were to stereo and short-range motion tracking, but the approach was later extended to more difficult problems. Zhang et al. (1995) showed that it was possible to match Harris corners over a large image range by using a correlation window around each corner to select likely matches. Outliers were then removed by solving for a fundamental matrix describing the geometric constraints between the two views of rigid scene and removing matches that did not agree with the majority solution. At the same time, a similar approach was developed by Torr (1995) for long-range motion matching, in which geometric constraints were used to remove outliers for rigid objects moving within an image.

The ground-breaking work of Schmid and Mohr (1997) showed that invariant local feature matching could be extended to general image recognition problems in which a feature was matched against a large database of images. They also used Harris corners to select interest points, but rather than matching with a correlation window, they used a rotationally invariant descriptor of the local image region. This allowed features to be matched under arbitrary orientation change between the two images. Furthermore, they demonstrated that multiple feature matches could accomplish general recognition under occlusion and clutter by identifying consistent clusters of matched features.

The Harris corner detector is very sensitive to changes in image scale, so it does not provide a good basis for matching images of different sizes. Earlier work by the author (Lowe, 1999) extended the local feature approach to achieve scale invariance. This work also described a new local descriptor that provided more distinctive features while being less sensitive to local image distortions such as 3D viewpoint change. This current paper provides a more in-depth development and analysis of this earlier work, while also presenting a number of improvements in stability and feature invariance.

There is a considerable body of previous research on identifying representations that are stable under scale change. Some of the first work in this area was by Crowley and Parker (1984), who developed a representation that identified peaks and ridges in scale space and linked these into a tree structure. The tree structure could then be matched between images with arbitrary scale change. More recent work on graph-based matching by Shokoufandeh, Marsic and Dickinson (1999) provides more distinctive feature descriptors using wavelet coefficients.

The problem of identifying an appropriate and consistent scale for feature detection has been studied in depth by Lindeberg (1993, 1994). He describes this as a problem of scale selection, and we make use of his results below.

Recently, there has been an impressive body of work on extending local features to be invariant to full affine transformations (Baumberg, 2000; Tuytelaars and Van Gool, 2000; Mikolajczyk and Schmid, 2002; Schaffalitzky and Zisserman, 2002; Brown and Lowe, 2002). This allows for invariant matching to features on a planar surface under changes in orthographic 3D projection, in most cases by resampling the image in a local affine frame. However, none of these approaches are yet fully affine invariant, as they start with initial feature scales and locations selected in a non-affine-invariant manner due to the prohibitive cost of exploring the full affine space. The affine frames are also more sensitive to noise than those of the scale-invariant features, so in practice the affine features have lower repeatability than the scale-invariant features unless the affine distortion is greater than about a 40 degree tilt of a planar surface (Mikolajczyk, 2002). Wider affine invariance may not be important for many applications, as training views are best taken at least every 30 degrees rotation in viewpoint (meaning that recognition is within 15 degrees of the closest training view) in order to capture non-planar changes and occlusion effects for 3D objects.

While the method to be presented in this paper is not fully affine invariant, a different approach is used in which the local descriptor allows relative feature positions to shift significantly with only small changes in the descriptor. This approach not only allows the descriptors to be reliably matched across a considerable range of affine distortion, but it also makes the features more robust against changes in 3D viewpoint for non-planar surfaces. Other advantages include much more efficient feature extraction and the ability to identify larger numbers of features. On the other hand, affine invariance is a valuable property for matching planar surfaces under very large view changes, and further research should be performed on the best ways to combine this with non-planar 3D viewpoint invariance in an efficient and stable manner.

Many other feature types have been proposed for use in recognition, some of which could be used in addition to the features described in this paper to provide further matches under differing circumstances. One class of features are those that make use of image contours or region boundaries, which should make them less likely to be disrupted by cluttered backgrounds near object boundaries. Matas et al., (2002) have shown that their maximally-

stable extremal regions can produce large numbers of matching features with good stability. Mikolajczyk et al., (2003) have developed a new descriptor that uses local edges while ignoring unrelated nearby edges, providing the ability to find stable features even near the boundaries of narrow shapes superimposed on background clutter. Nelson and Selinger (1998) have shown good results with local features based on groupings of image contours. Similarly, Pope and Lowe (2000) used features based on the hierarchical grouping of image contours, which are particularly useful for objects lacking detailed texture.

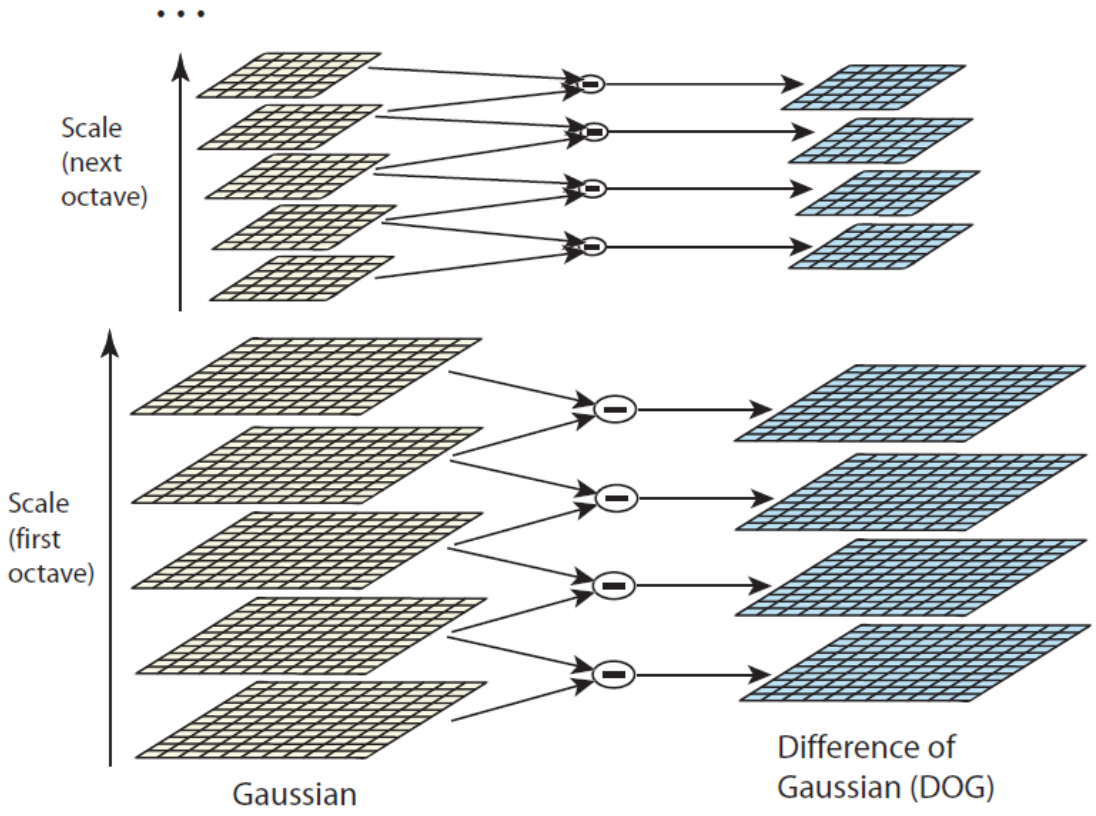
The history of research on visual recognition contains work on a diverse set of other

image properties that can be used as feature measurements. Carneiro and Jepson (2002) describe phase-based local features that represent the phase rather than the magnitude of local spatial frequencies, which is likely to provide improved invariance to illumination. Schiele and Crowley (2000) have proposed the use of multidimensional histograms summarizing the distribution of measurements within image regions. This type of feature may be particularly useful for recognition of textured objects with deformable shapes. Basri and Jacobs (1997) have demonstrated the value of extracting local region boundaries for recognition. Other useful properties to incorporate include color, motion, figure-ground discrimination, region shape descriptors, and stereo depth cues. The local feature approach can easily incorporate novel feature types because extra features contribute to robustness when they provide correct matches, but otherwise do little harm other than their cost of computation. Therefore, future systems are likely to combine many feature types.

2.2.1 Detection of Scale-Space Extrema

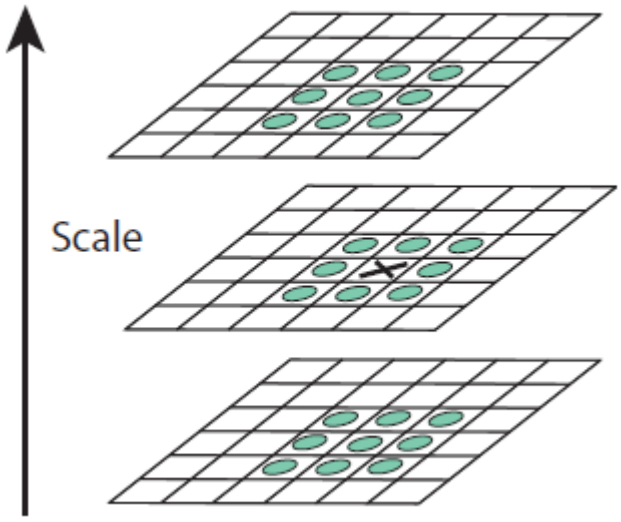
For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled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

Figure 2.1: Scale space.



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).

Figure 2.2: Maxima and minima of the difference-of-Gaussian images are detected.



In order to detect the local maxima and minima of D(x, y), 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.

2.2.2 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.

The initial implementation of this approach (Lowe, 1999) simply located keypoints at the location and scale of the central sample point. However, recently Brown has developed a method (Brown and Lowe, 2002) for fitting a 3D quadratic function to the local sample points to determine the interpolated location of the maximum, and his experiments showed that this provides a substantial improvement to matching and stability.

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 and Mohr (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.

2.2.3 The Local Image Descriptor

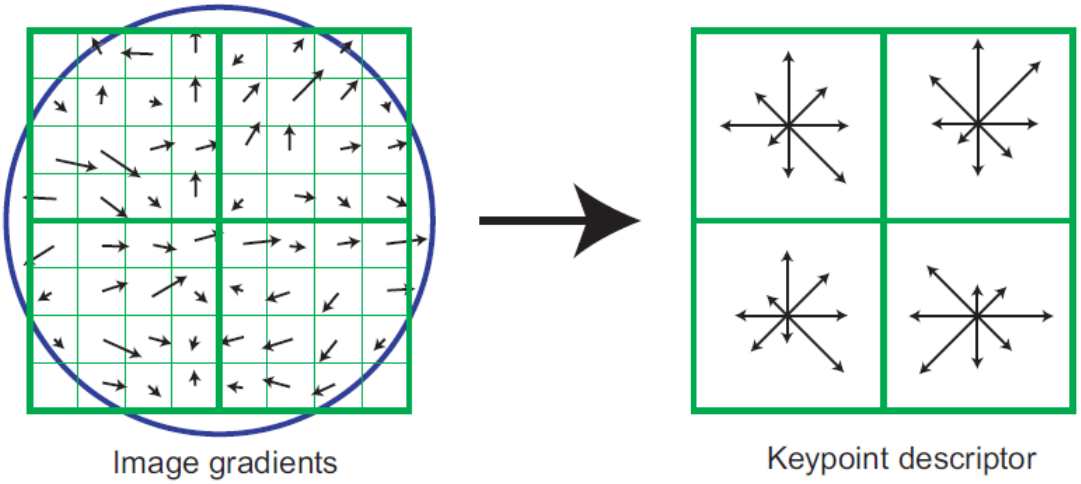
The previous operations have assigned an image location, scale, and orientation to each keypoint. These parameters impose a repeatable local 2D coordinate system in which to describe the local image region, and therefore provide invariance to these parameters. The next step is to compute a descriptor 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. For example, recognition accuracy for 3D objects rotated in depth by 20 degrees increased from 35% for correlation of gradients to 94% using the complex cell model. Our implementation described below was inspired by this idea, but allows for positional shift using a different computational mechanism.

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 sub regions, as

shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes 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.

Figure 2.3: Local image descriptor.



2.3 SIFT ALGORITHM

SIFT is quite an involved algorithm. It has a lot going on and can become confusing, so we can split up the entire algorithm into multiple parts. Here's an outline of what happens in SIFT;

a) **Constructing a scale space:** This is the initial preparation. You create internal representations of the original image to ensure scale invariance. This is done by generating a "scale space".
b) **LoG Approximation:** The Laplacian of Gaussian is great for finding interesting points (or key points) in an image. But it's computationally expensive. So we cheat and approximate it using the representation created earlier.
c) **Finding keypoints:** With the super-fast approximation, we now try to find key points. These are maxima and minima in the Difference of Gaussian image we calculate in step 2

d) **Get rid of bad key points:** Edges and low contrast regions are bad keypoints. Eliminating these makes the algorithm efficient and robust. A technique similar to the Harris Corner Detector is used here.

e) **Assigning an orientation to the keypoints**: An orientation is calculated for each key point. Any further calculations are done relative to this orientation. This effectively cancels out the effect of orientation, making it rotation invariant.

f) **Generate SIFT features:** Finally, with scale and rotation invariance in place, one more representation is generated. This helps uniquely identify features. Lets say you have 50,000 features. With this representation, you can easily identify the feature you're looking for (say, a particular eye, or a sign board).

2.4 SURF ALGORITHM

SURF (Speeded-Up Robust Features) is a fast and robust algorithm for local, similarity invariant image representation and comparison. Similarly to the SIFT approach , SURF selects interest points of an image from the salient features of its linear scale-space, and then builds local features based on the image gradient distribution. The main interest of the SURF approach lies in its fast computation of approximate differential operators in the scale-space, based on Integral Image Representation and Box Filters, enabling real-time applications such as tracking and object recognition.

The SURF algorithm is composed of three consecutive steps:

a) interest point detection,

b) interest point description,

c) Feature matching.

Like the SIFT method the first two steps rely on a scale-space and first and second order differential operators. The originality of the SURF method is that these operations are speeded-up by the use of an integral image and box filters techniques.

In the detection step, the local maxima of the Hessian determinant operator applied to the scale-space are computed to select interest point candidates. These candidates are then

validated if the response is above a given threshold. Both scale and location of these candidates are then refined using an iterated procedure to fit a quadratic function. Typically, a few hundreds interest points are detected in a digital image of 1 Mega-pixels.

The purpose of the second step described in the local representation section is to build a descriptor that is invariant to view-point changes of the local neighborhood of the point of interest. Recall that the location of this point in the scale-space provides invariance to scale and translation changes. To achieve rotation invariance, a dominant orientation is defined by considering the local gradient orientation distribution, estimated with Haar wavelets. Making use of a spatial localization grid, a 64-dimensional descriptor is then built, corresponding to a local histogram of the Haar wavelet responses.

Classically, the third step matches the descriptors of both images. Exhaustive comparisons are performed here by computing Euclidean distance between all potential matching pairs. A nearest-neighbor distance-ratio matching criterion is then used to reduce mismatches, combined with a RANSAC-based technique for geometric consistency checking (epipolar geometry with the ORSA algorithm). After these filters eliminating all suspectedly spurious matches, one can be reasonably sure that the remaining matches are real and correspond to the same scene seen from different viewpoints.

Algorithm step-by-step;

a) Computation of the integral image of the input image.

b) Interest points detection:

   i. Computation of the discrete Hessian operator at several scales using box-filters.

   ii. Selection of maxima responses of the determinant of the Hessian matrix in scale space.

   iii. Refinement of the corresponding interest point location by quadratic "interpolation".

   iv. Storage of the interest point with its Laplacian sign.

c) Local descriptors construction:

   i. Estimation of the dominant orientation of each interest point;

ii. Computation of the descriptor (16 x 4 vector) corresponding to the scaled and oriented neighborhood of the interest point;

d) Image matching:

i. Matching the SURF descriptors of both images by a nearest neighbour criterion inspired from the SIFT algorithm, speeded-up by a priori imposing that the sign of the Laplacian is the same for corresponding descriptors.

ii. discarding matches based on geometric consistency checking (ORSA);

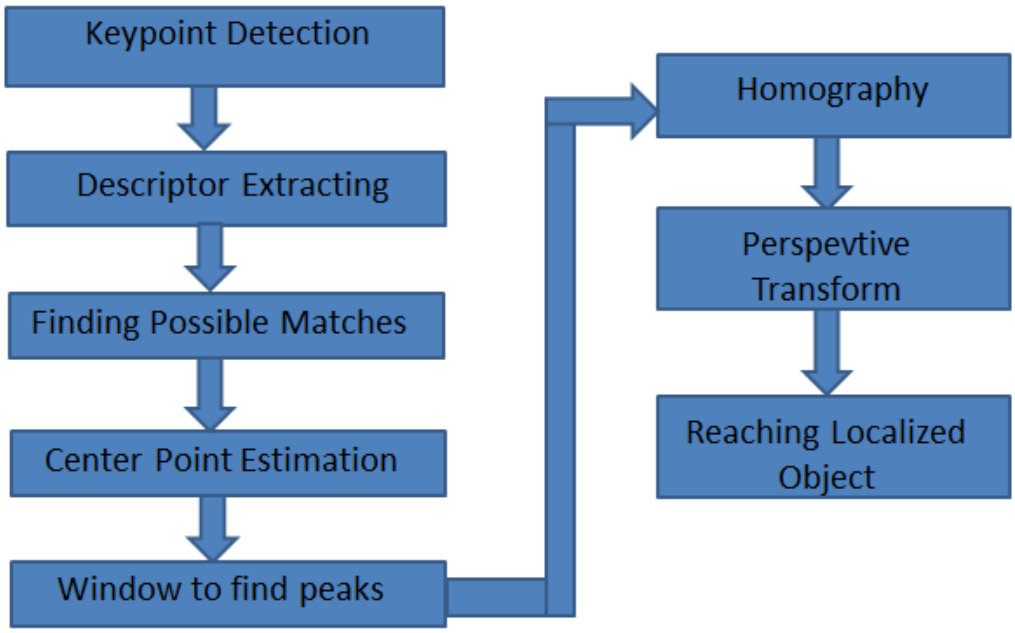
3. METHODOLOGIES

There are many way to localize objects. In state of art many augmented reality application use GPS, Gyroscope and Accelerometer. Actually these technologies are not helpful to locate objects but simultaneously locate objects. Some applications just like “Layar” use GPS to locate objects. When you turned your phones camera to some place Layar tries to find related shops around that direction and use GPS, Accelerometer and Gyroscope. But in fact Layar kind of applications cannot localize an object, just behave localization.

Other way to localize an object to use structure from motion. Structure from motion applications can generate 3d model from 2d images. Thus, we can localize our query image on this 3d model. An application has been developed by Noah Snavely which models Collezione.

OpenCV has been used in this research. Our aims to achieve multiple object detection with local features. First of all detecting interested points (keypoints) then extracting descriptor from these keypoints. These are basic Operations of image processing. Our idea starts with descriptor matching, there are several ways to match descriptors but all of them find only one match point for a single point. Our idea is to match multiple objects, so we need to find multiple matches for a single point on demand. We used our matching method for this purpose which uses normalized cross correlation. Thus we can find possible matches between images. Center point estimation is key operation of this research, i will explain it in this chapter. After estimation center point using local matches we get an area which has center point estimations. But these estimations can be wrong or deviation can be happen because surf keypoint is not invariance to perspective effects. So we need to find a area which has most of estimated points. We used a window to find these matches with a threshold value. If threshold has been passed match has been found. Then applying homography with these match points we can found exact matches between two images.

Figure 3.1: Our algorithm to object localization.



I will explain step by step what we have done in this research;

3.1 KEYPOINT DETECTION

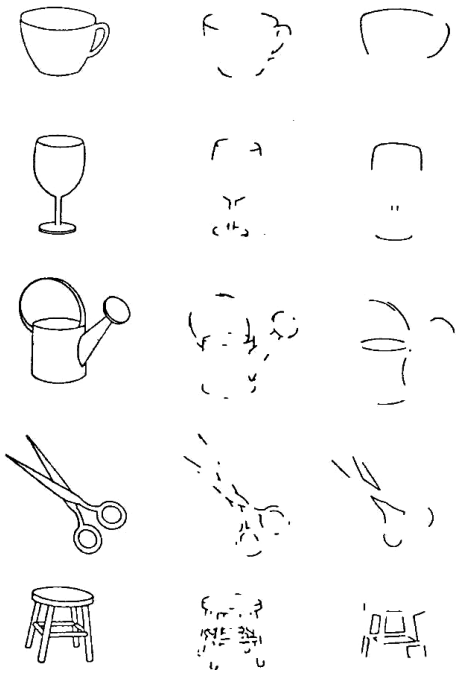
Good features should have the following properties:

i. 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.
ii. Distinctiveness: The intensity patterns underlying the detected features should show a lot of variation such that features can be distinguished and matched.
iii. 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).
iv. 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.

v. Accuracy: The detected features should be accurately localized, both in image location, as with respect to scale and possibly shape.

vi. Efficiency: Preferably, the detection of features in a new image should allow for time-critical applications.

Figure 3.2:Corners and junctions for keypoint detection.



After corners, the second most intuitive local features are blobs. In the area of computer vision, blob detection refers to visual modules that are aimed at detecting points and/or regions in the image that differ in properties like brightness or color compared to the surrounding. Used in The difference of Gaussians, The Laplacian of Gaussian, The determinant of the Hessian.

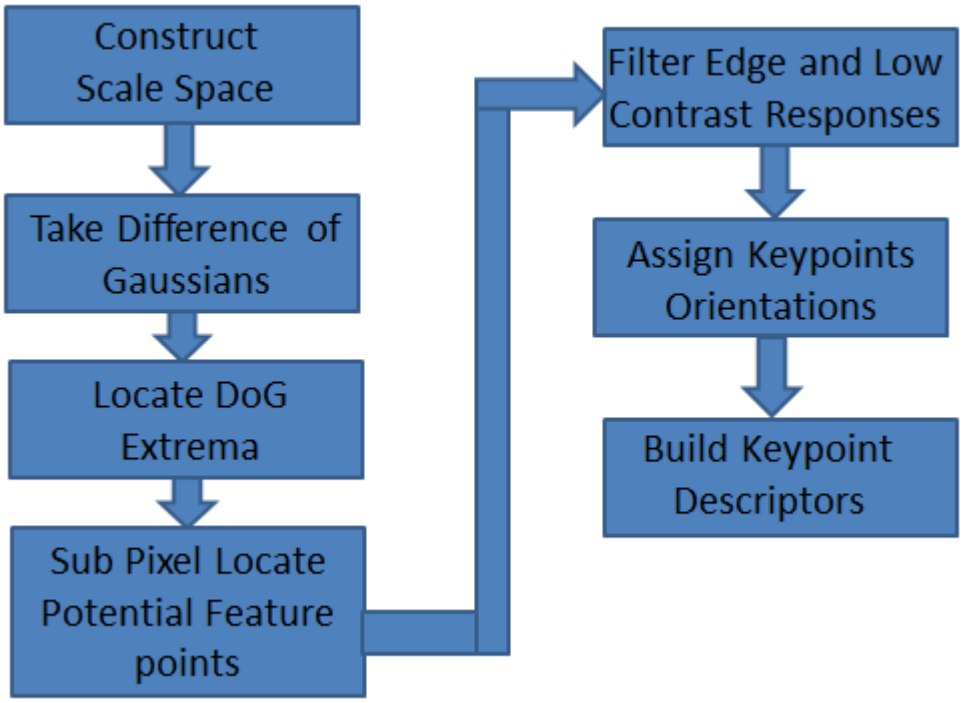
Figure 3.3: Soccer player tracking (blob application)

We used SIFT and SURF keypoint detectors to develop our research.

SIFT keypoints can be detect in 6 step;

i. Constructing a scale space
ii. LoG Approximation
iii. Finding keypoints
iv. Get rid of bad key points
v. Assigning an orientation to the keypoints
vi. Generate SIFT features

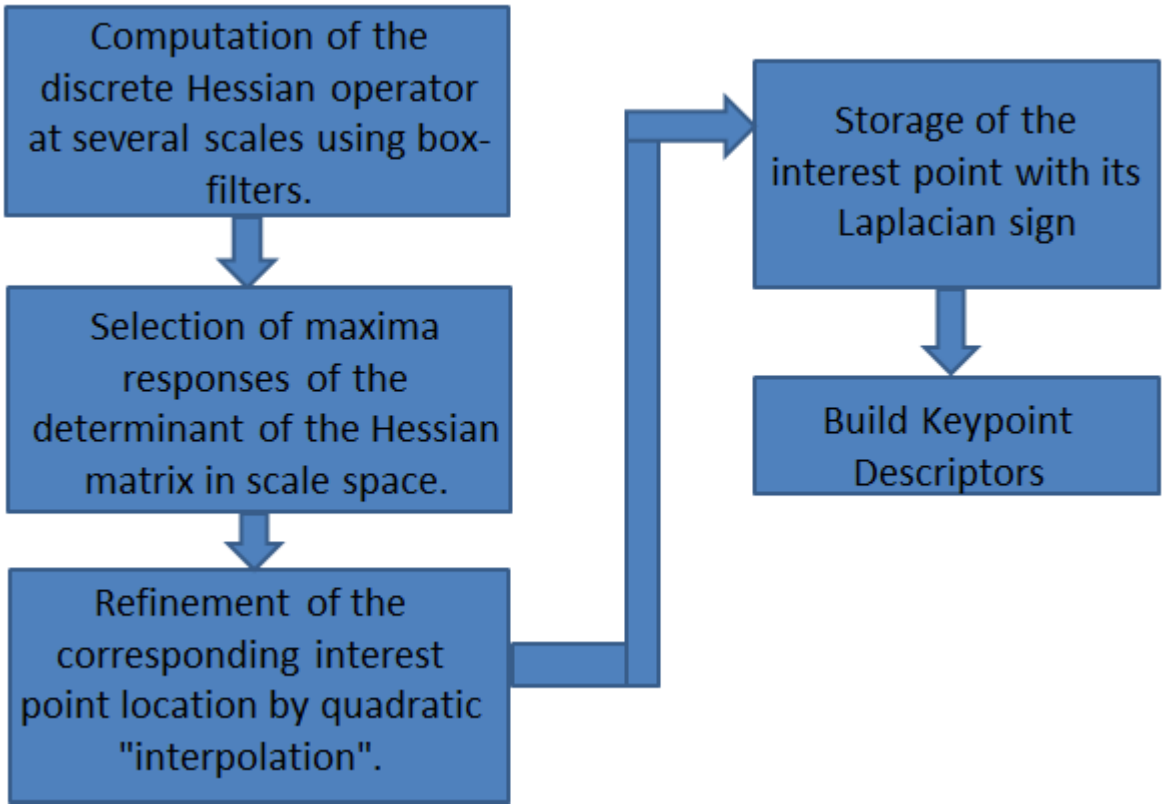
Figure 3.4:Sift keypoint detection steps



SURF keypoints detection;

i. Computation of the discrete Hessian operator at several scales using box-filters
ii. Selection of maxima responses of the determinant of the Hessian matrix in scale space
iii. Refinement of the corresponding interest point location by quadratic "interpolation"
iv. Storage of the interest point with its Laplacian sign

Figure 3.5: Surf keypoint detection steps



3.2 DESCRIPTOR EXTRACTING

SIFT descriptor extracting; first a set of orientation histograms are created on 4x4 pixel neighborhoods with 8 bins each. These histograms are computed from magnitude and orientation values of samples in a 16 x 16 region around the keypoint such that each histogram contains samples from a 4 x 4 sub-region of the original neighborhood region. The magnitudes are further weighted by a Gaussian function with σ equal to one half the width of the descriptor window. The descriptor then becomes a vector of all the values of these histograms. Since there are 4 x 4 = 16 histograms each with 8 bins the vector has 128 elements. This vector is then normalized to unit length in order to enhance invariance to affine changes in illumination. To reduce the effects of non-linear illumination a threshold of 0.2 is applied and the vector is again normalized.

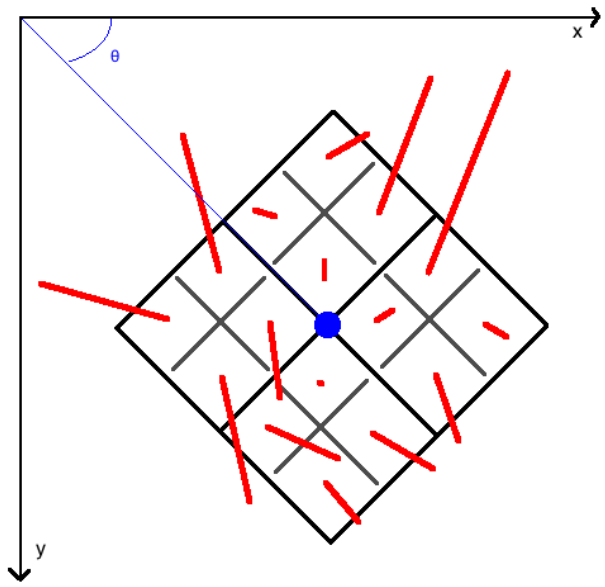
Although the dimension of the descriptor, i.e. 128, seems high, descriptors with lower dimension than this don't perform as well across the range of matching tasks and the computational cost remains low due to the approximate BBF method used for finding the nearest-neighbor. Longer descriptors continue to do better but not by much and there is an

additional danger of increased sensitivity to distortion and occlusion. It is also shown that feature matching accuracy is above 50% for viewpoint changes of up to 50 degrees. Therefore SIFT descriptors are invariant to minor affine changes. To test the distinctiveness of the SIFT descriptors, matching accuracy is also measured against varying number of keypoints in the testing database, and it is shown that matching accuracy decreases only very slightly for very large database sizes, thus indicating that SIFT features are highly distinctive.

SURF Descriptor Extracting; A SURF descriptor is a 16 x 4 vector, representing normalized gradient statistics (mean and absolute mean values) extracted from a spatial grid divided into 4-by-4 cells. For a given an interest point, as illustrated in the figure below, the corresponding square grid is centered on ($x_i y_i$), aligned accordingly toΘ_i and with size.

The SURF descriptor is obtained by concatenating the 16 vectors computed for every sub-region and by normalizing the corresponding 64-dimensional vector so that it's Euclidean norm is unitary, thus making the descriptor invariant to affine contrast changes.

Figure 3.6: Surf descriptor



3.3 FINDING POSSIBLE MATCHES

Finding possible matches is an important point of our algorithm. We are going to try match to images, we already detected keypoints and extracted descriptors from these keypoints. Now we need to find possible matches. All of other matcher algorithms implemented in OpenCV by default try to find best matches. Our aim is to detect and localize objects widely even if multiple same objects. In this point, all of other matcher algorithms inadequate to detect multiple same objects. We developed our matcher algorithm which applies normalized cross correlation between two image descriptors.

Descriptors can be used as float vectors.

D_1 = [10,12,25,36,25,1,2,6,93,…] (3.1)

D_2= [19,12,5,33,14,5,78,41,6,…] (3.2)

We applied normalized cross correlation to these vectors.

Cross Correlation = $\sum \boldsymbol{D_1^i}\, \boldsymbol{D_2^i}$ (3.3)

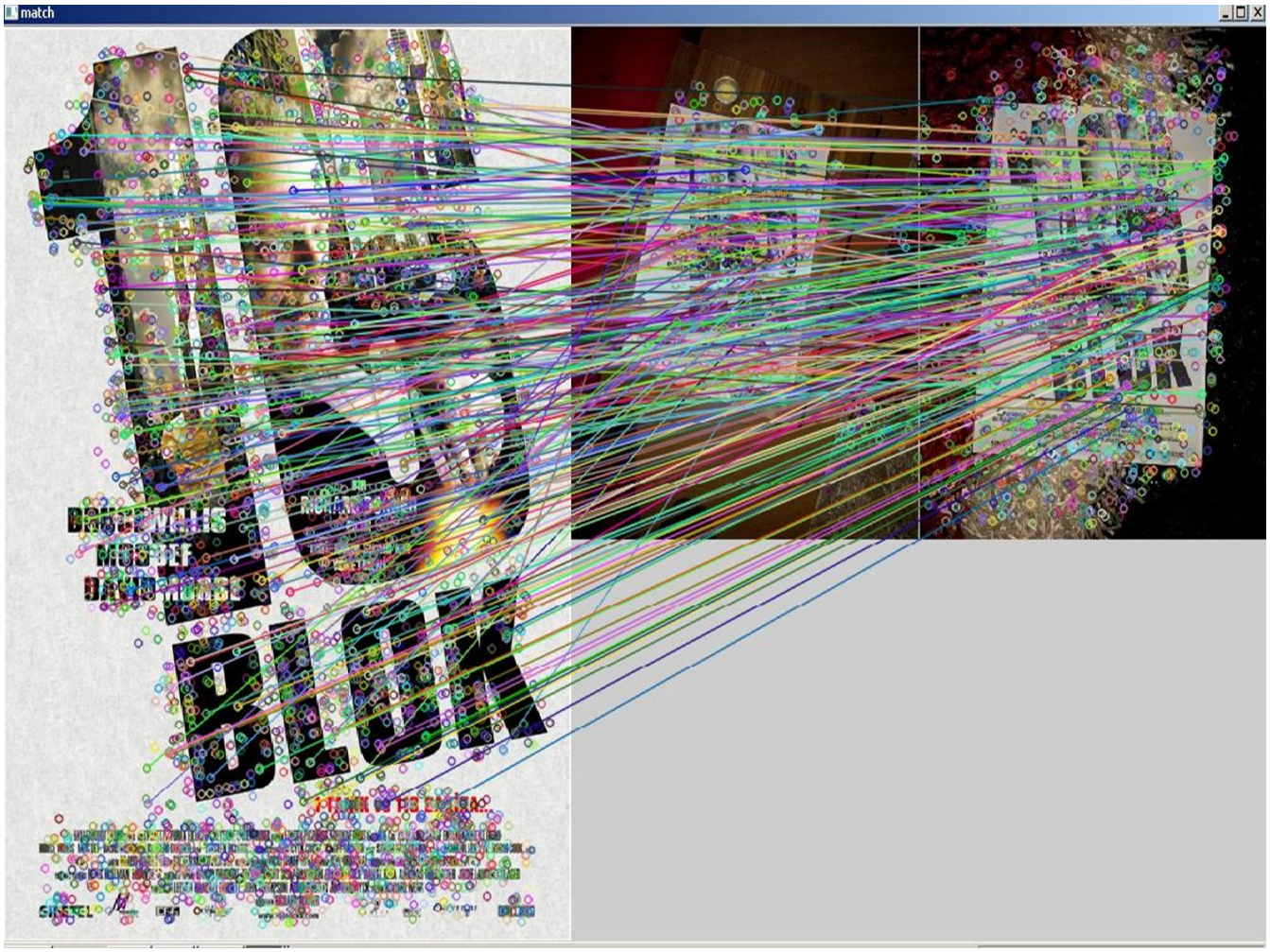
Normalized Cross Correlation: $\frac{\sum \mathbf{D_1^i D_2^i}}{\sqrt{(\sum \mathbf{D_1^i})^2 (\sum \mathbf{D_2^i})^2}}$ (3.4)

NCC~1, successful

NCC~0, unsuccessful

Result of normalized cross correlation can be accepted successful how much close to 1. But take into consideration some distortion some calculations cannot be exactly 1. We can choose a threshold close to 1 just like 0,95,then we get matches.

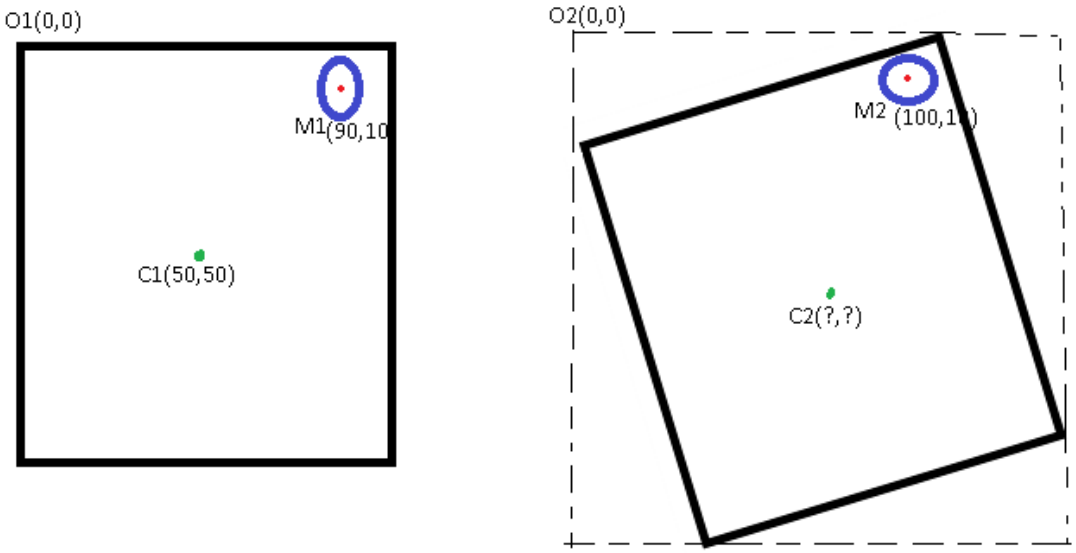
Figure 3.7: Drawing possible matches.



3.4 MID POINT ESTIMATION

Mid-point estimation approach brought up by us. Our algorithms important part is mid-point estimation. We have two images and also have 2 keypoints, we know possibly these two keypoints matches to each other. First image is our query image and second one training image. We know keypoints and their local features (rotation, scale, etc.). We want to estimate query image's mid-point on training image using matched keypoints. This can be achieved using local features of keypoints. I will explain it using Figure 3.3.

Figure 3.8: Mid-point estimation, M1,C1 and M2 known parameters, C2 will be calculated.



C1 is coordinate system of first image.

O1 is center point of first image.

M1 is keypoint of first image.

C2 is coordinate system of second image.

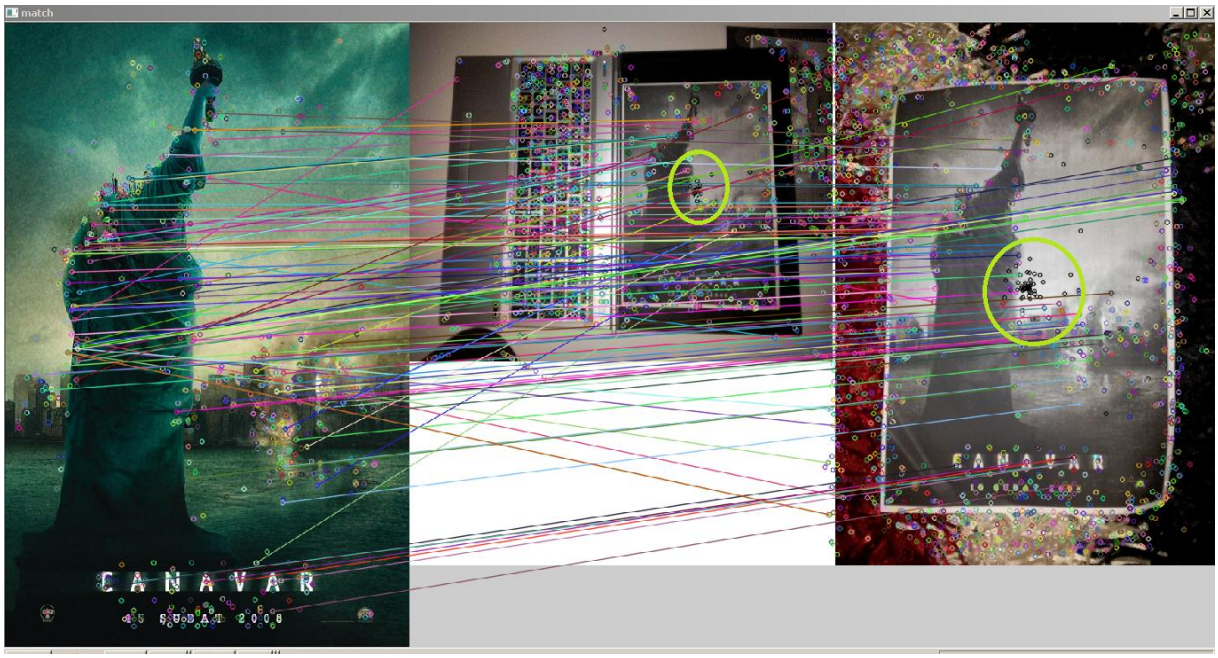
O2 is center point of second image.

M2 is keypoint of second image.

C1,M1 and M2 known so we need to estimate C2 using these features. We can formulate situations below;

$$\mathbf{C_2} = \mathbf{M_{T2}} + \frac{\mathbf{S_{M1}}}{\mathbf{S_{M2}}}\mathbf{R^T_{M2}}(\boldsymbol{\Theta})\mathbf{R_{M1}}(\boldsymbol{\Theta})(\mathbf{C_1} - \mathbf{M_1}) \quad (3.5)$$

Figure 3.9: Center point estimations has been made and circled with a green circle.



3.5 WINDOW TO FIND POSSIBLE MATCHES

By center point estimation operation we expect that all estimated points will be closed to each other in best case. Our research invariant to rotation and scale. Depending on camera orientation estimated points can be sparse and spread. We can handle this situation with a window and threshold value, window will be visit all image. İf the window area has estimated points more than threshold value, it means match has been caught. As you can see below around the noise there are black circles.

Figure 3.10: Mid-point estimation has been made successfully, can be seen estimated center points around the nose.

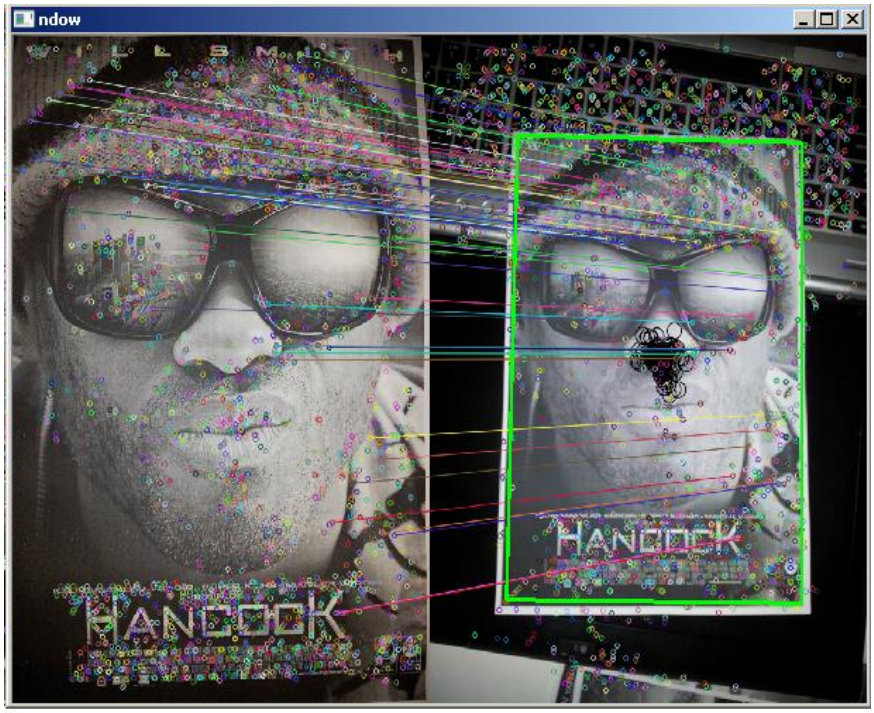


Figure 3.11: Window detects a possible match area.

3.6 APPLYING HOMOGRAPHY

We get possible match points with using a window but it does not mean that we got exact matches. Some incorrect match can be found inside of window. We need to eliminate those incorrect possible matches. At this point we can apply homography. At least 4 points required to find a homography by default. You can see possible matches before homography and after homography below;

Figure 3.12: Before homography you can see incorrect matches.

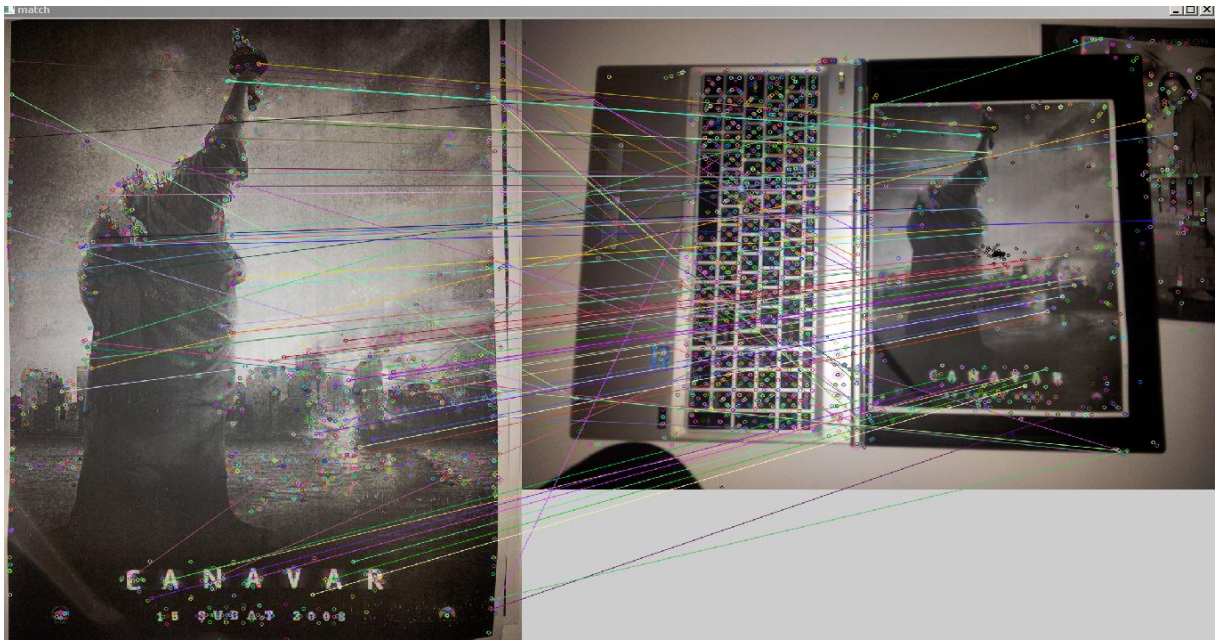
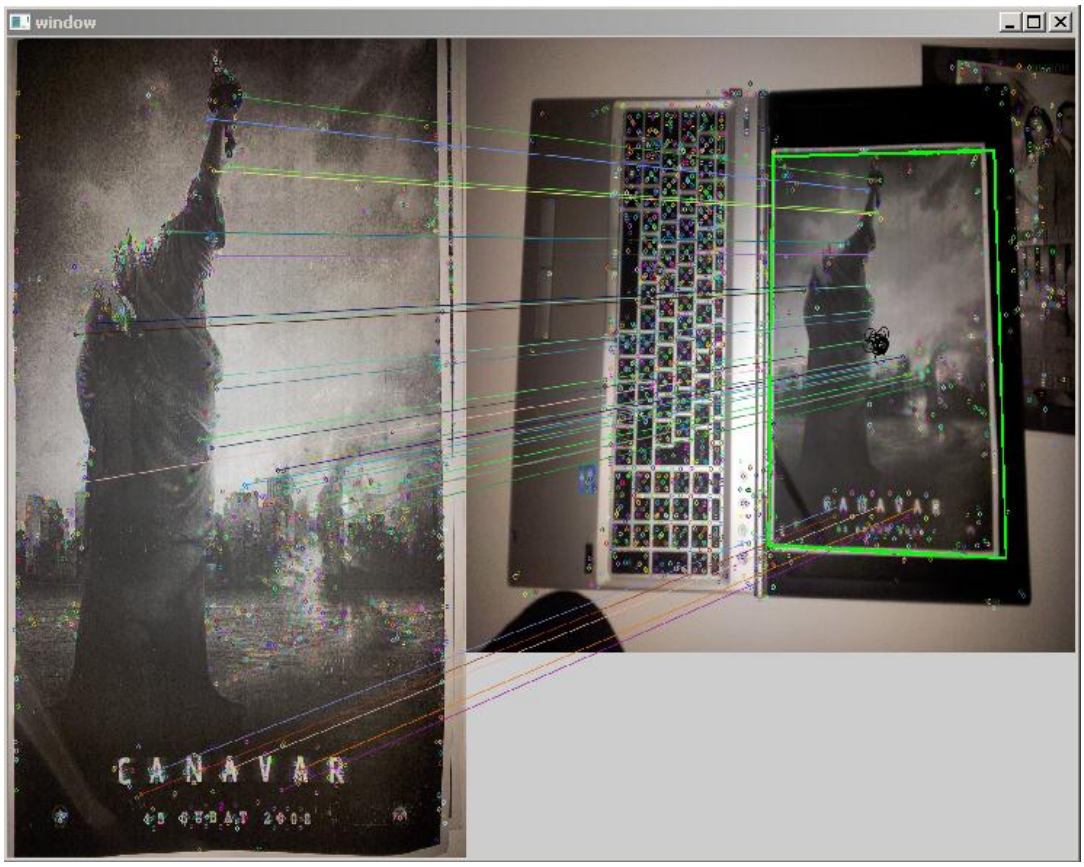


Figure 3.13: After applying homography all incorrect matches has disappeared.



3.7 APPLYING PERSPECTIVE TRANSFORM

After applying homography, we get correct matches. In this point we already got matches, so we can draw a rectangle around the object. We used perspective transform to draw this rectangle. At least 4 point required to draw rectangle with perspective transform. These points are query image's corner points.

Figure 3.14:Query image's corner points. These points will be used to find target image's corner points.

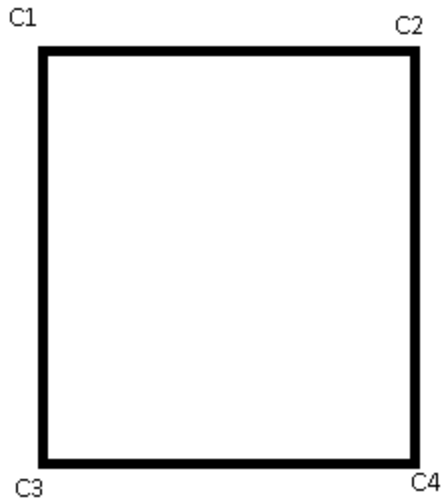
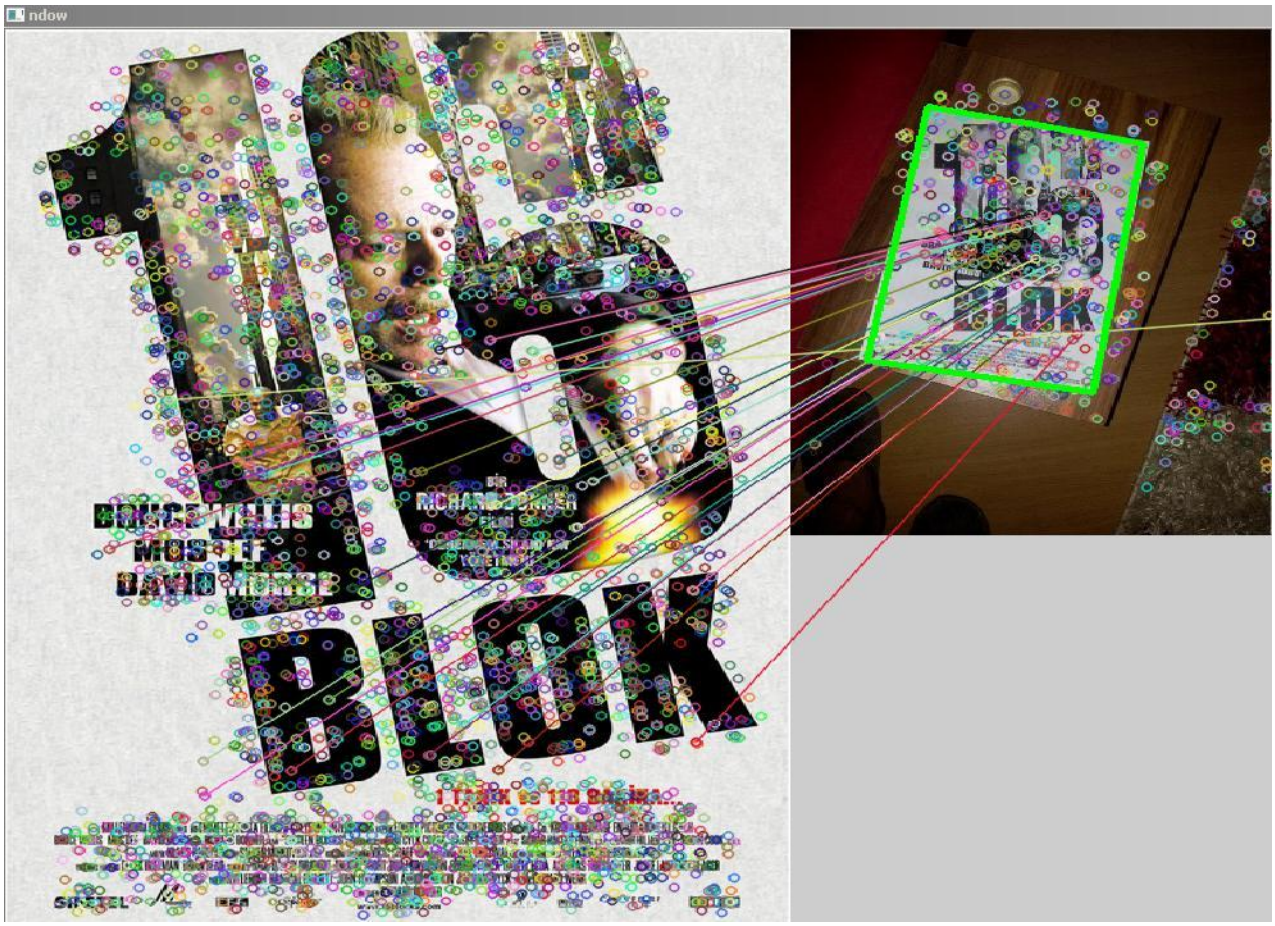


Figure 3.15: Perspective transform applied to target image.



4. TEST RESULTS

4.1 OBJECT LOCALIZATION

We have tested our approach on many different image sets. SIFT and SURF keypoints tested for object localization. You can see below a matched sample in background there are several objects which are discarded by algorithm.

Figure 4.1: Object localization sample in background there are several object which are discarded.

4.2 MULTIPLE OBJECT LOCALIZATION

Multiple object detection is not applicable with opencv's default matcher algorithms. Our matcher algorithm can handle this problem. You can see below 2 of object has been detected by algorithm.

Figure 4.2: Multiple object detection

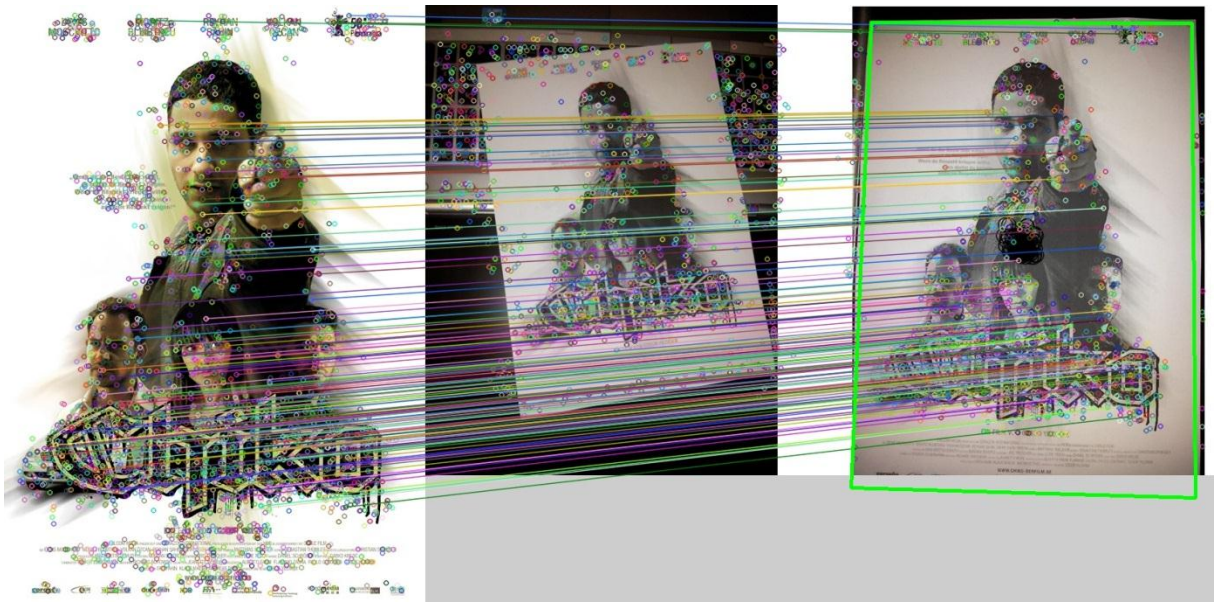
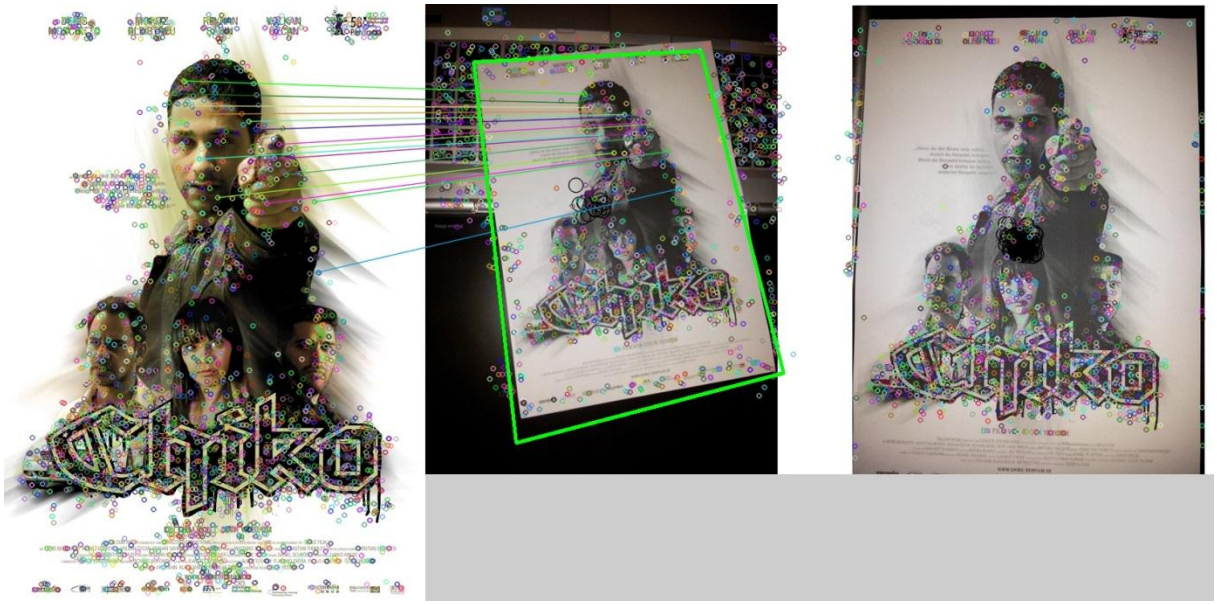


Figure 4.3: Multiple object detection second (object)



4.3 SIFT AND SURF CAMERA ORIENTATION COMPARISON

Our algorithm is not affine invariant, changing camera orientation affect our algorithm. As shown at Figure 4.4 SIFT is not resistible to affine invariant. SURF also is not resistible but gives more acceptable results.

Figure 4.4: SIFT and SURF camera orientation comparison results. SIFT was unsuccessful but SURF had relevant results.

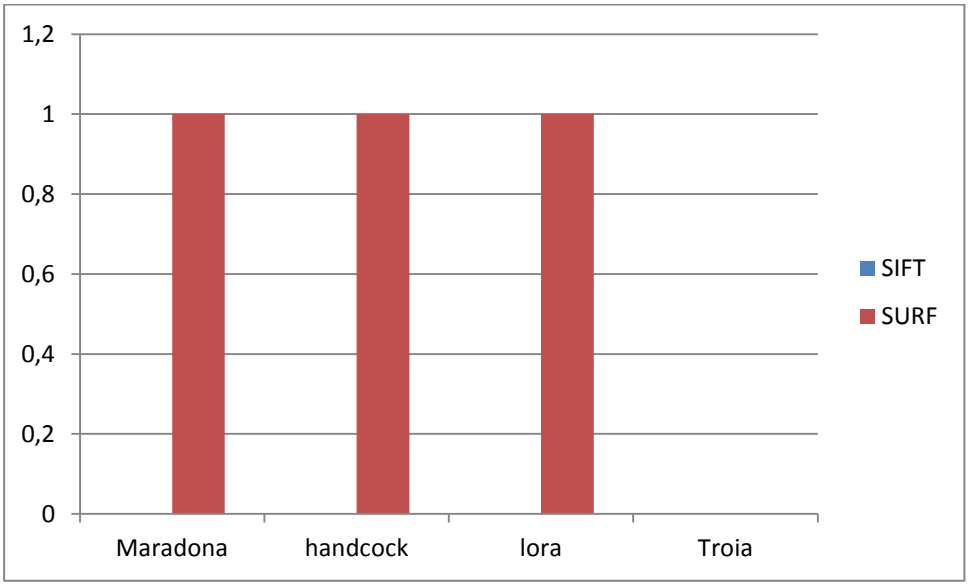
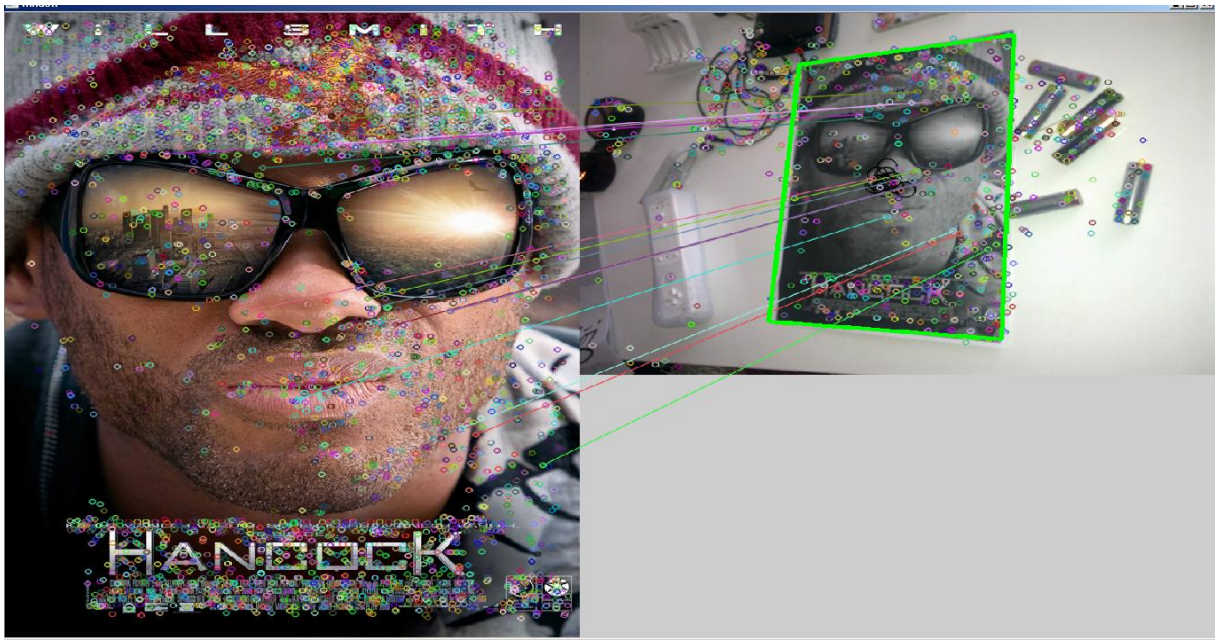


Figure 4.5: An example of SIFT which has no relevant result.

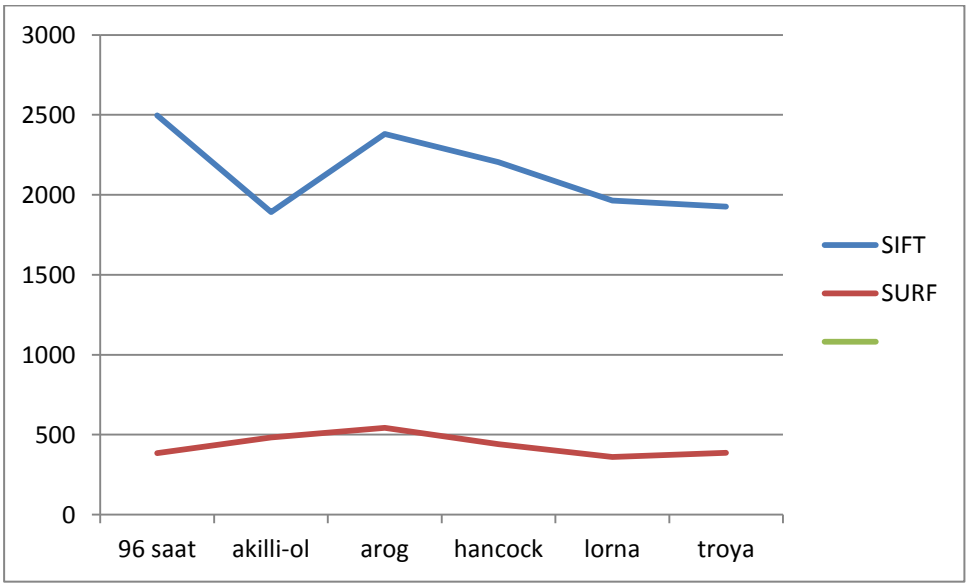
Figure 4.6: Surf illustration. Camera orientation has been changed but using SURF features we could accomplished to locate handcock poster.



4.4 SIFT AND SURF KEYPOINT DETECTION TIME COMPARING

SIFT and SURF keypoint detection times calculated. As we can see at Figure 4.7 SIFT takes much more time.

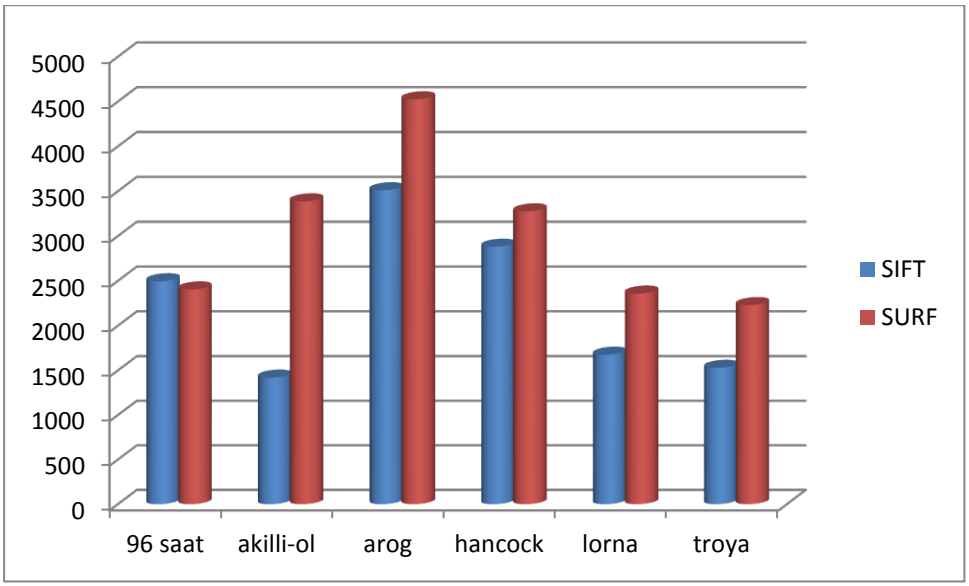
Figure 4.7: SURF keypoint detection takes less time than SIFT.



4.5 SIFT AND SURF DETECTED KEYPOINT NUMERICAL COMPARISON

SIFT and SURF keypoints counts calculated for each poster. SURF can detect much more keypoints than SIFT as shown Figure 4.8.

Figure 4.8: SIFT and SURF detected keypoints comparison numerically.



4.6 ROTATION INVARIANCE COMPARISON

Rotation invariance of our algorithm has been tested various ways. Figure 4.11 shows results;

Figure 4.9: -15 to 20 degree test results. 1 matched, 0 unmatched.

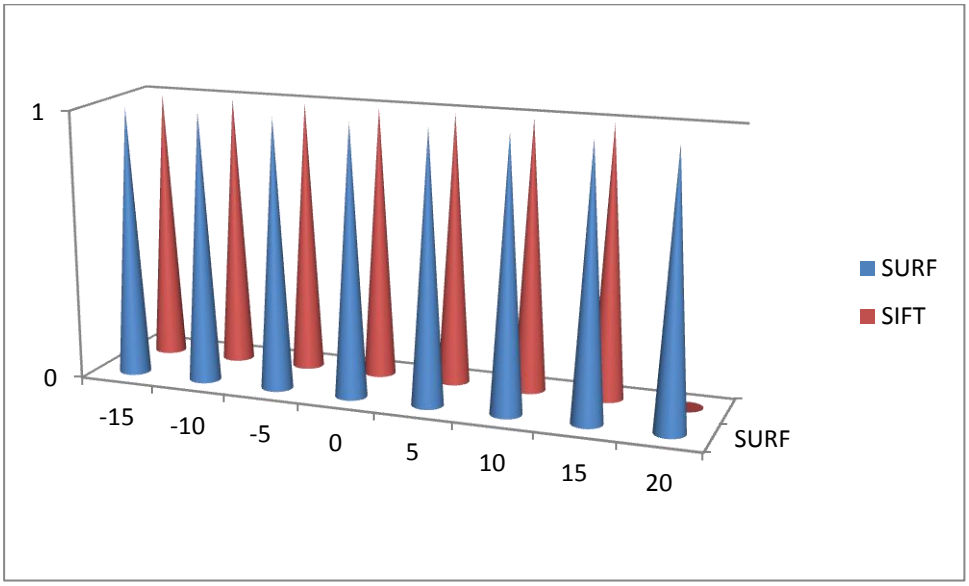


Figure 4.10: 10 degree rotated SURF test result.

4.7 SCALE INVARIANCE COMPARISON

Scale invariance of our algorithm has been tested various ways. Figure 4.11 shows results;

Figure 4.11: 0 to 70 percentage scale test results. 1 matched, 0 unmatched.

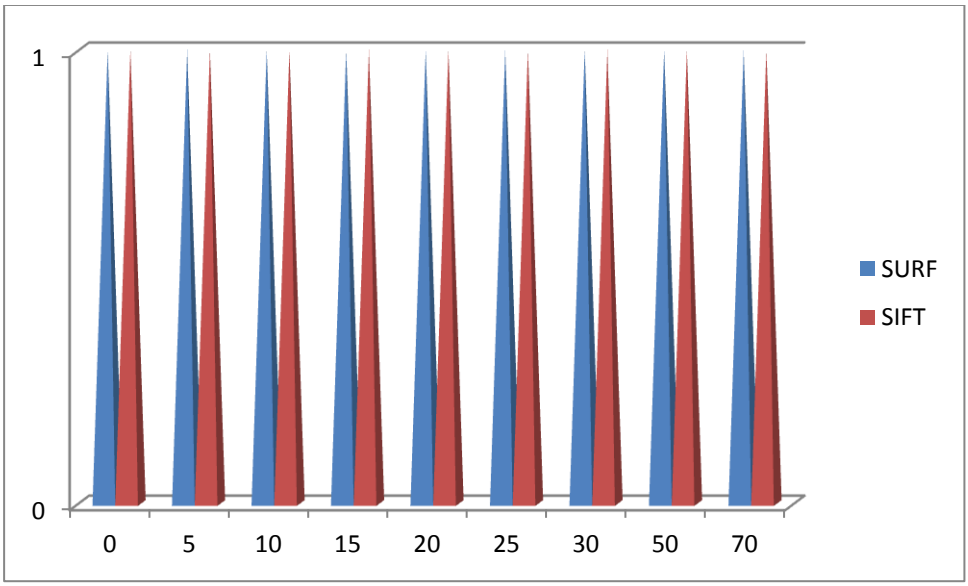
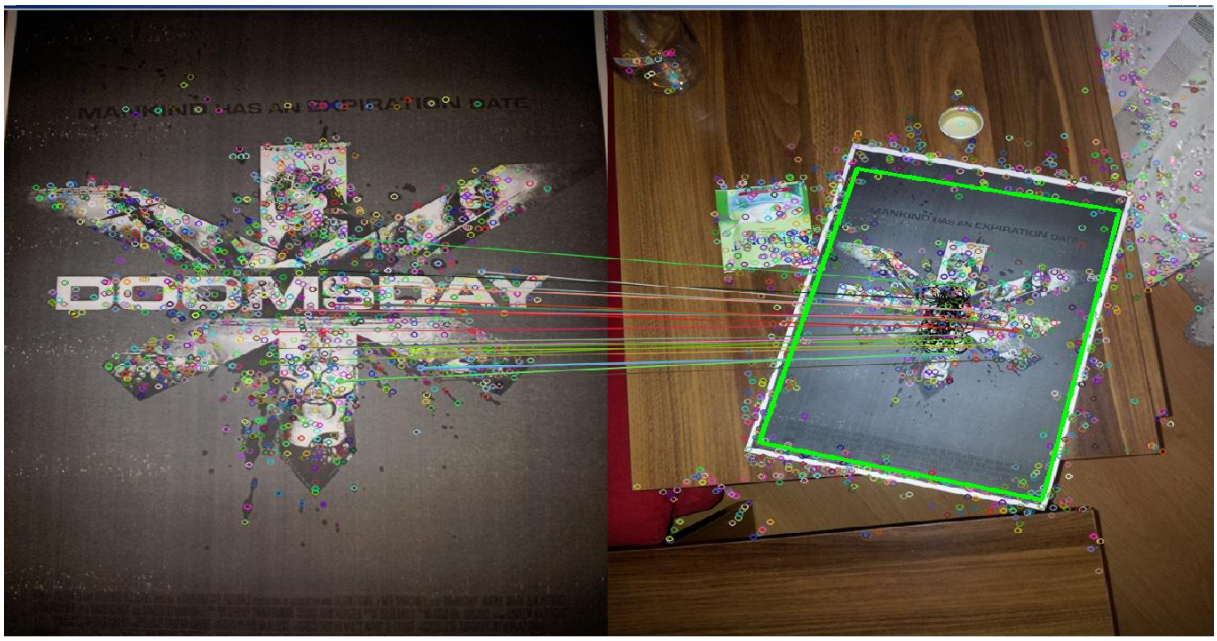


Figure 4.12: SURF test result of 50% scale.



4.8 TRANSLATE INVARIANCE COMPARISON

Translate invariance of our algorithm has been tested various ways. Figure 4.13 shows results;

Figure 4.13: Test result of translated images. 1 matched , 0 unmatched.

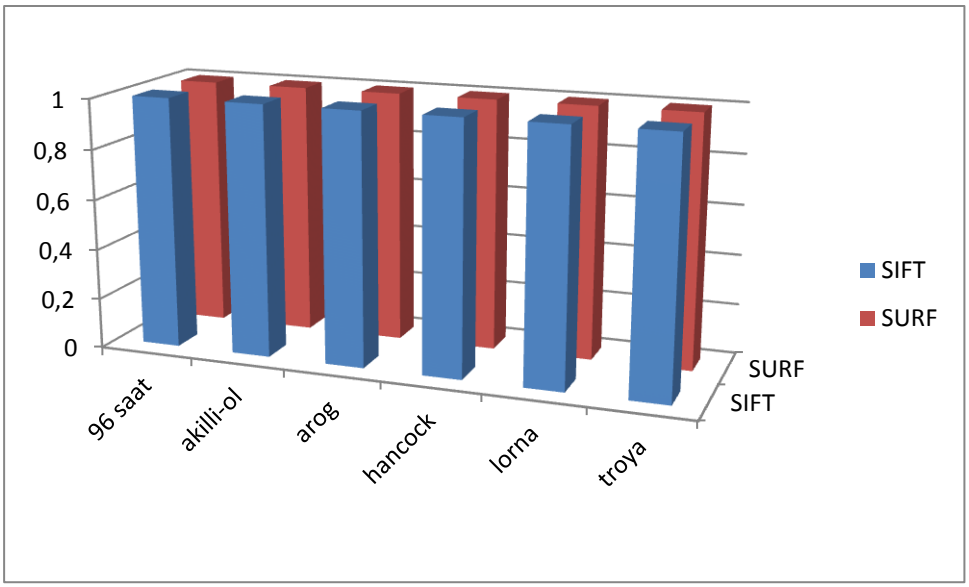
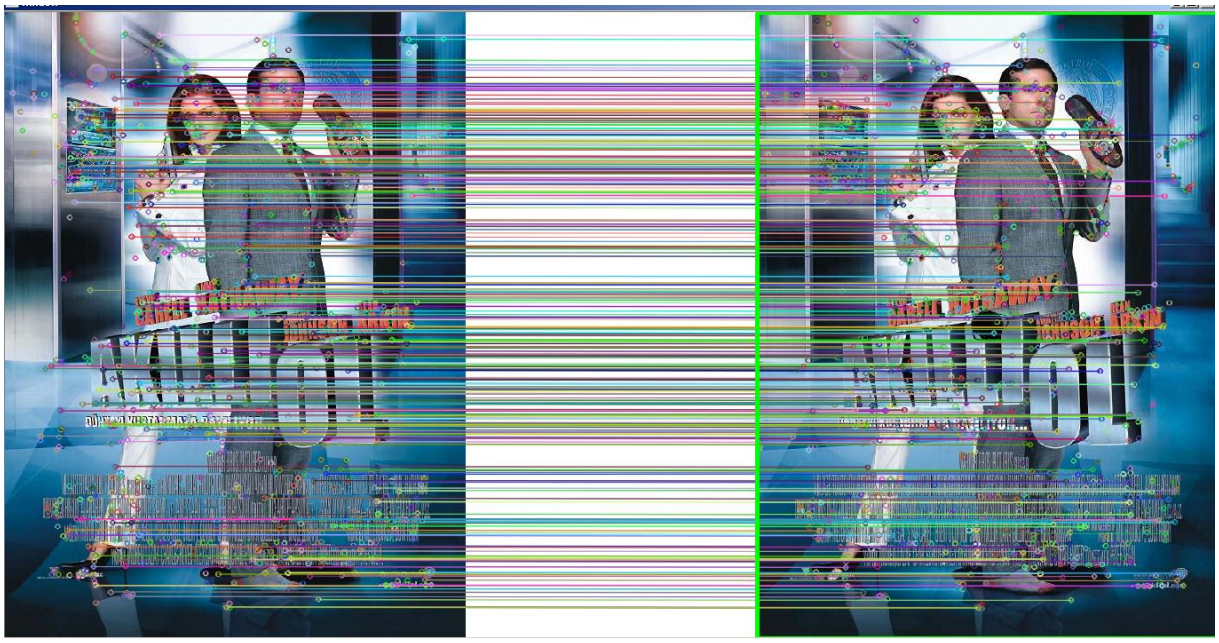


Figure 4.14: Second image translated from right side and even if translated images can be match with our algorithm.



4.9 SIFT COMPARISON

We used 64 images to compare SIFT results. Test results will be explained step by step.

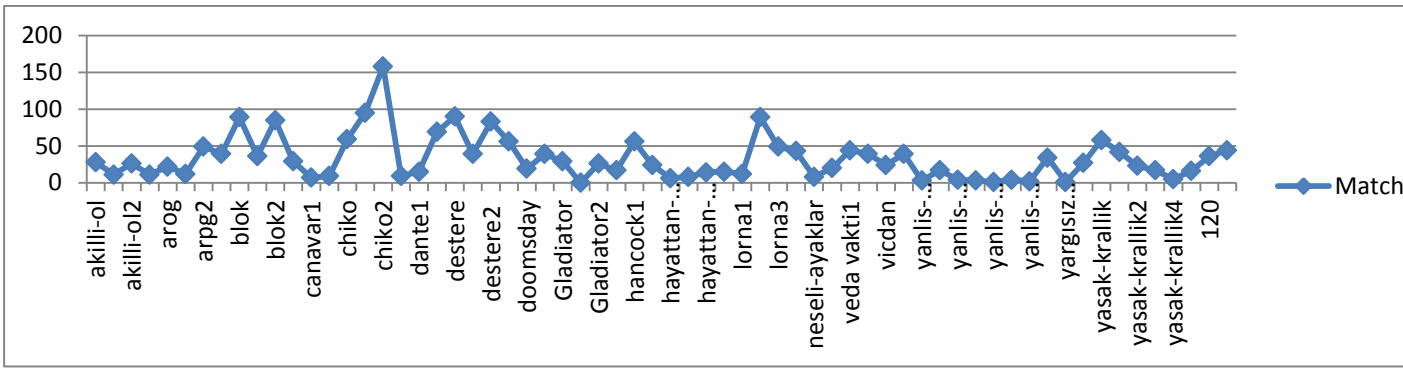
4.9.1 One to One Object Match Comparison

In this test we tried to match one single object to another single object. Match size and success of matching's graphics will be shown by turns;

4.9.1.1 Match size Comparison

Images matched to each other and match sizes calculated. Graphic shows image by image match sizes;

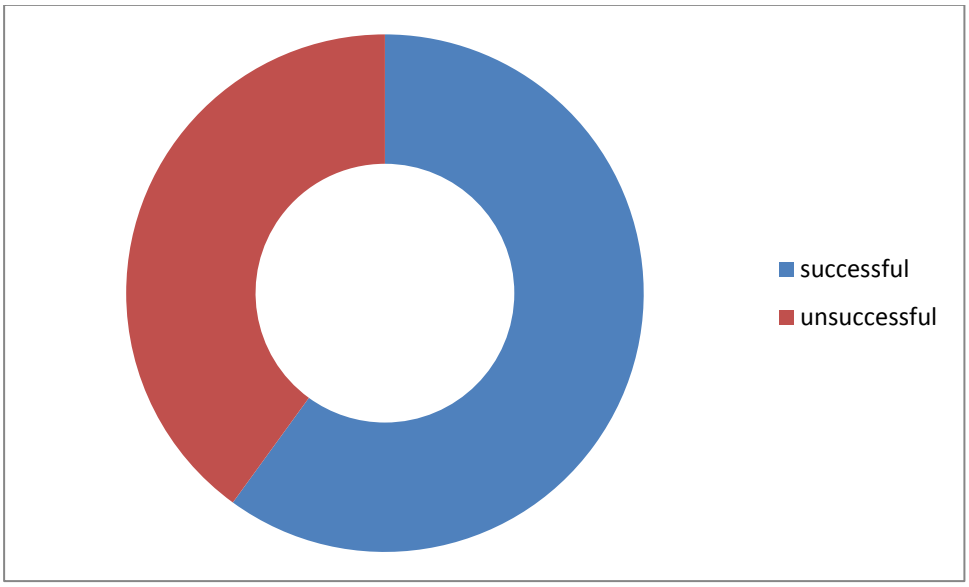
Figure 4.15: Match sizes of SIFT training set.



4.9.1.2 Success Comparison

Images matched to each other. Graphic shows image by image match success rates.

Figure 4.16: 39 successful, 25 unsuccessful, total 64. Totally 60% succeed.



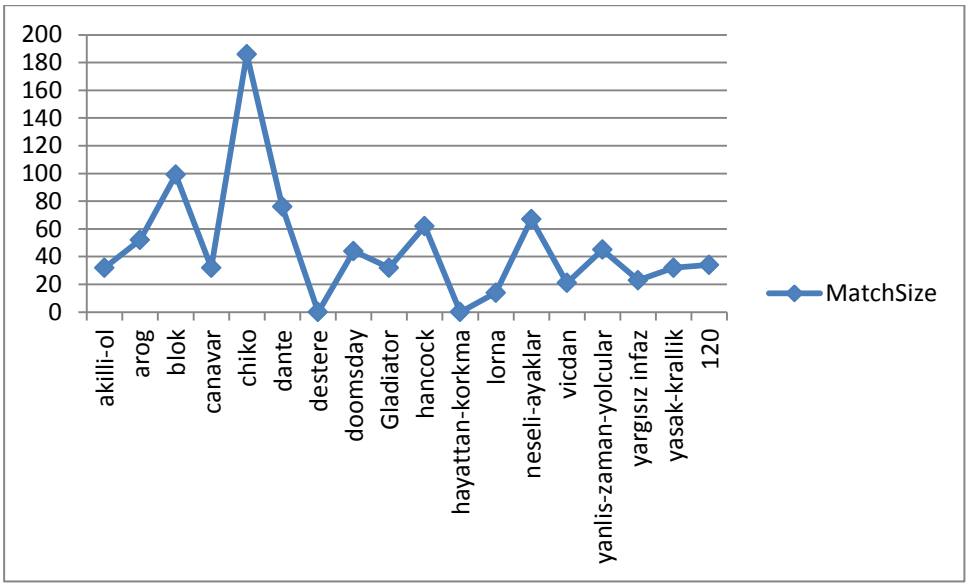
4.9.2 Multiple Object Match Comparison

In this test we tried to match one single object to multiple same objects. Match size and success of matching's graphics will be shown by turns;

4.9.2.1 Match Size Comparison

Images matched to each other and match sizes calculated. Graphic shows image by image match sizes;

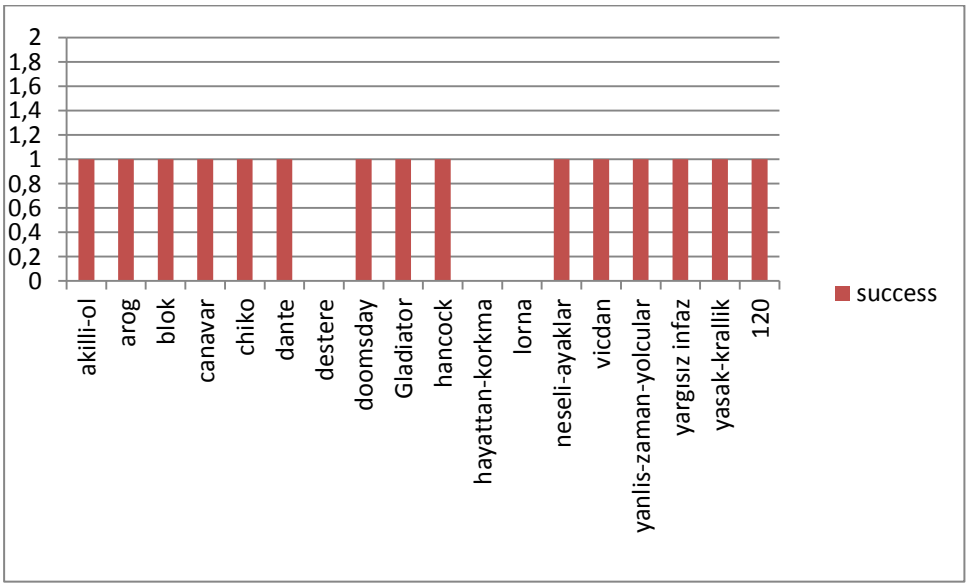
Figure 4.17: SIFT multiple object detection tests' match sizes.



4.9.2.2 Success Comparison

Images matched to each other. Graphic shows image by image match success rates.

Figure 4.18: SIFT multiple object detection success results.



4.10 SURF Comparison

We used 64 images to compare SIFT results. Test results will be explained step by step.

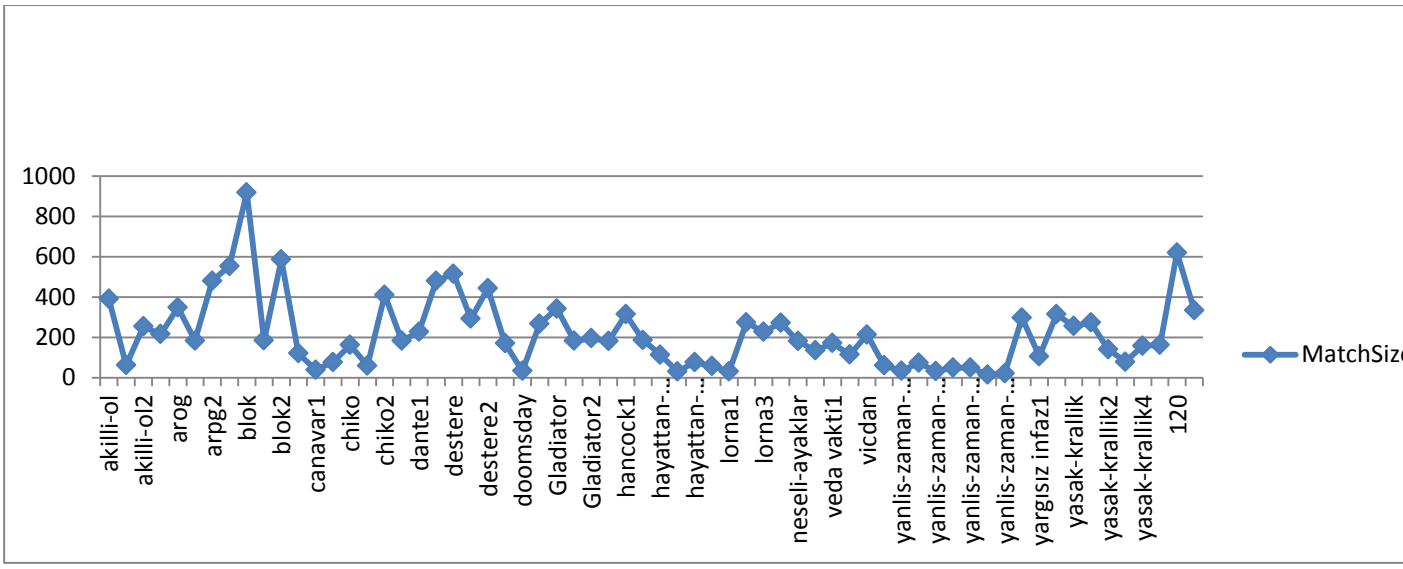
4.10.1 One to One Object Match Comparison

One to one image matching test results will be shown.

4.10.1.1 Match size Comparison

Images matched to each other and match sizes calculated. Graphic shows image by image match sizes;

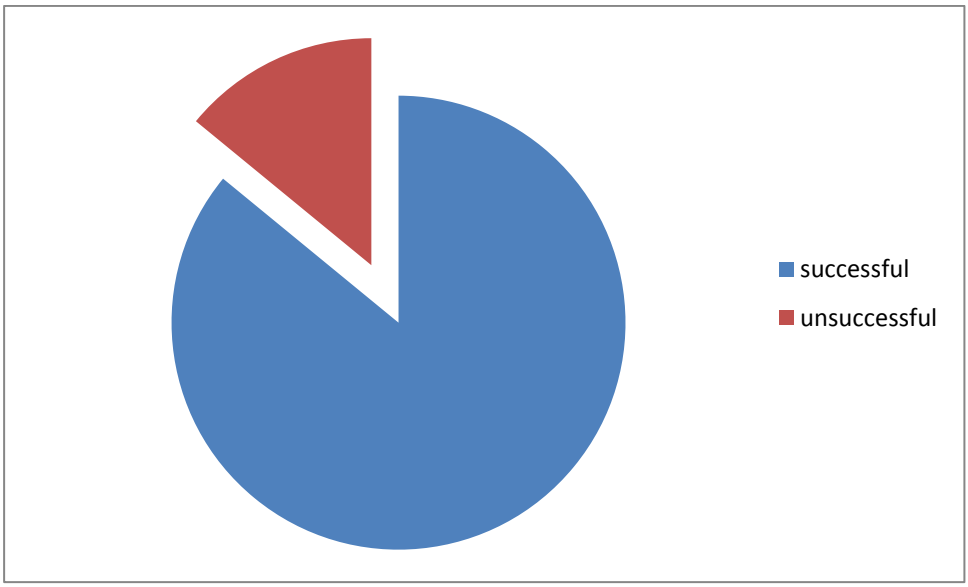
Figure 4.19: Match sizes of SURF training set.



4.10.1.2 Success Comparison

Images matched to each other. Graphic shows image by image match success rates.

Figure 4.20: 55 successful, 9 unsuccessful, total 64. Totally 85% succeed.

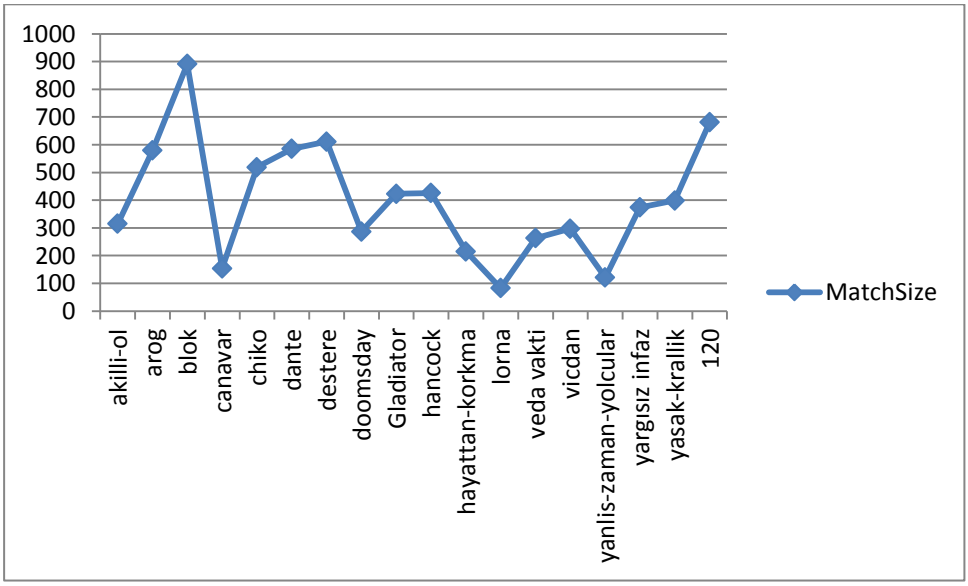


4.10.2 Multiple Object Match Comparison

4.10.2.1 Match Size

Images matched to each other and match sizes calculated. Graphic shows image by image match sizes;

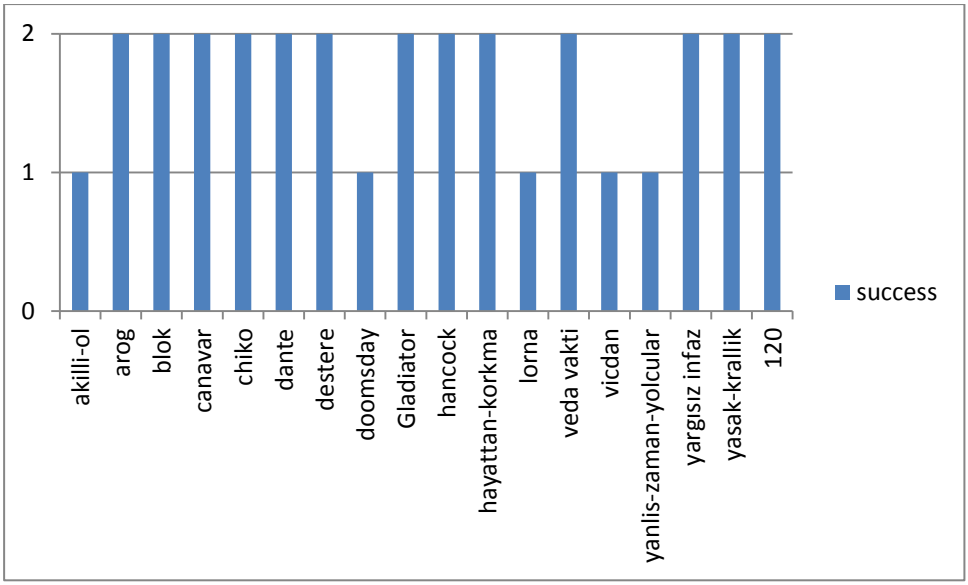
Figure 4.21: SURF multiple object detection test's match sizes.



4.10.2.2 Success Comparison

Images matched to each other. Graphic shows image by image match success rates.

Figure 4.22: Multiple object detection applied to 18 different images. 2 match success, 1 match half success, 0 unsuccessful.



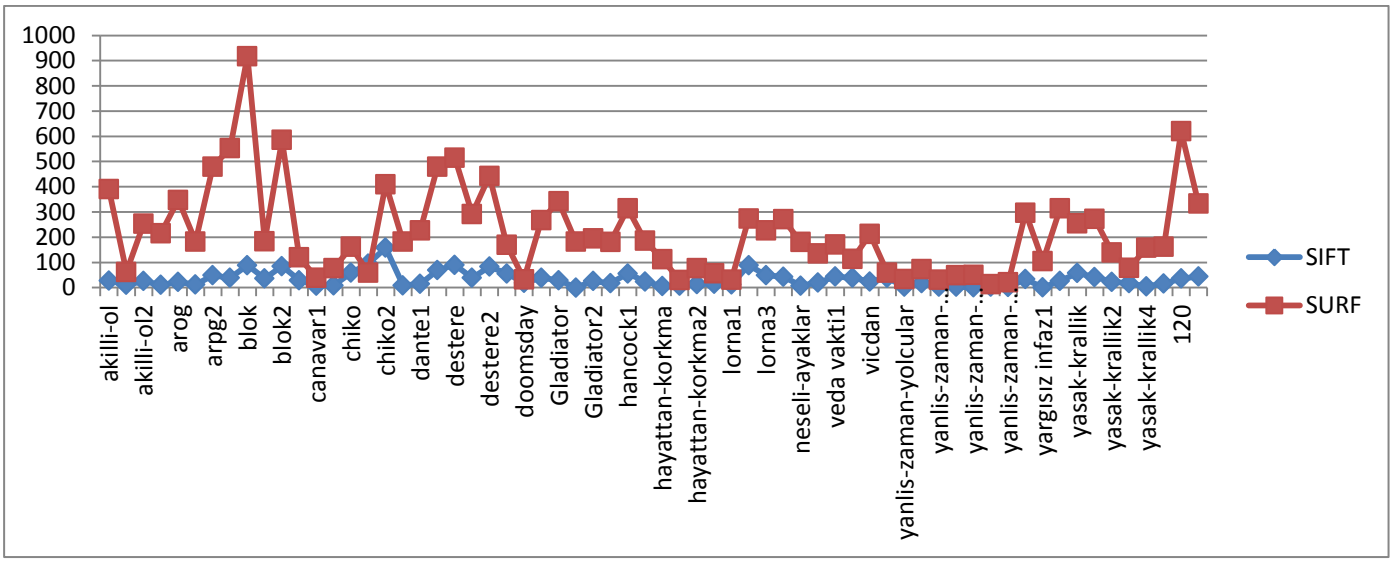
4.11 SIFT- SURF COMPARISON

Same training set have been used on both SIFT and SURF tests. Comparison results will be shown step by step.

4.11.1 Match Size

Images matched to each other and SIFT and SURF match sizes calculated. Graphic shows image by image match sizes;

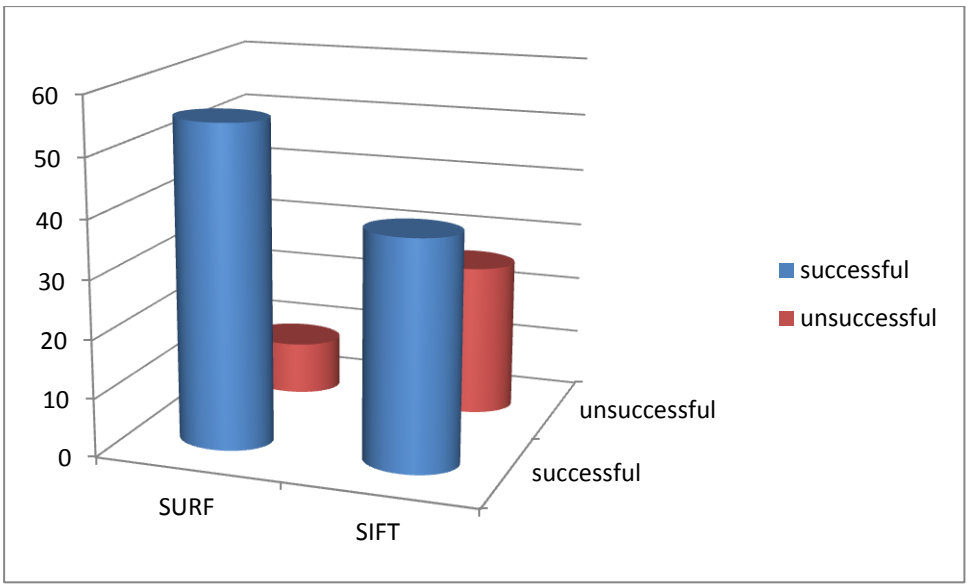
Figure 4.23: Match sizes compared to each other.



4.11.2 Success Comparison

Images matched to each other. Graphic shows image by image match success rates.

Figure 4.24: Success comparison between SIFT and SURF.



5. DISCUSSION

As we mentioned before our approach uses local image features to match images. Our approach scale and rotation invariant but there are many problem to solve like blurring, brightness, camera orientation, excessive scale, illumination etc. These are common problem of image processing. Our approach is also affected by these problems.

In computer science, an inverted index (also referred to as postings file or inverted file) is an index data structure storing a mapping from content, such as words or numbers, to its locations in a database file, or in a document or a set of documents. We could save our descriptors in a inverted file and search them on demand. We did not implement it during this research.

Many applications can be develop with our approach. Augmented reality applications in the shop centers, museum applications, cinema poster applications, online content based image search applications and so on. We could not implement an application during this research but will develop it soon.

6. CONCLUSION

In this research, we have evaluated SIFT and SURF algorithm for detecting and describing local gray scale image features using OpenCV. Both of them succeeded but SURF results was better than SIFT. Especially for the multiple object detection SURF worked perfectly. It can be seen chapter 4 by test results.

We have pointed out that our algorithm can detect multiple same object and also scale and rotation invariant. Any rotation and scale change does not affect our approach. Also any object detection can be made with our approach.

We have showed that some of basic object detection problems can affect our approach , just like camera orientation. Each of them has solutions but in this research our goal is not finding solutions to these problems.

Finally, our approach has been successfully applied to many different image set and tested. Results showed us to using local features and estimating center point on target image method can be applied to many applications.

REFERENCES

Books

Laganière, R., 2011. *Opencv 2 computer vision application programming cookbook*.1. Edition. Birmingham. Packt Publishing.

Huamán, A., 2011. *The opencv tutorials release 2.3.*

Bradski, G. & Kaehler A., 2008. *Learning opencv - computer vision with opencv library*. 1. Edition. O'reilly.

Other Sources

Oyallon, E. & Rabin, J., 2012. Surf: speeded-up robust features.

Sinha, U., 2010. Constructing a scale space.

Sinha, U., 2010. Sift: scale invariant feature transform.

Sinha, U., 2010. Laplacian of gaussian approximation.

Sinha, U., 2010. Assigning keypoint orientation.

Sinha, U., 2010. Eliminate edges and low contrast regions.

Bay, H., Tuytelaars, T., & Gool,L., 2008. Surf: speeded up robust features.

Tuytelaars, T. & Mikolajczyk, K., 2008. Local invariant feature detectors: a survey.

Zhang, Y., Jia, Z. & Chen, T., 2008. Image retrieval with geometry-preserving visual phrases.

Dubrofsky, E., 2007. Homography Estimation.

Nist´er, D. & Stew´enius, H., 2006. Scalable recognition with a vocabulary tree.

Lowe, David G., 2004. Distinctive image features from scale-invariant keypoints.

Lowe, David G., 1999.Object recognition from local scale-invariant features. *Proc. of the international conference on computer vision*. Corfu.

Sivic, J. & Zisserman, A., 2003. Video google: a text retrieval approach to object matching in videos. *Proceedings of the ninth IEEE internationalconference on computer vision*.

Romeny, H., 1996. Introduction to scale-space theory: multiscale geometric image analysis.

CONTENTS

FIGURES