Chapter 6
Proposed Improvement in Image Registration for Augmented Reality

6.1 Introduction

Summary of the enhancements/improvements done for Image Registration methods till 2017: In last three decades, many approaches have been proposed for extracting invariant regions of interest in an image and these approaches are usually classified according to their efficiency to handle affine transformations. For example, the Harris point detector [Harris and Stephens 1988] is considered to be rotation invariant. The Harris-Laplace and Hessian-Laplace feature detectors [Mikolajczyk and Schmid 2001, Mikolajczyk and Schmid 2004] are invariant to rotations and scale change. Certain moment-based detectors [Lindeberg and Garding 1997, Baumberg 2000], Harris-Affine and Hessian-Affine feature detectors [Mikolajczyk and Schmid 2002, Mikolajczyk and Schmid 2004], Edge-based region detector (EBR) [Tuytelaars and Gool 2004], Intensity-based region detector (IBR) [Tuytelaars and Gool 2004], Maximally Stable Extremal Regions (MSER) [Matas et al. 2004] etc. are designed to be invariant to affine transformations. However, a comparative study conducted by [Mikolajczyk et al. 2005] between Harris-Affine, Hessian-Affine, EBR, IBR and MSER detectors demonstrated that performance of all the five detectors moderately declines as the viewpoint change increases, however, in many scenarios MSER performed better than the other detectors followed by Hessian-Affine.

In an attempt to design an affine invariant feature detection algorithm, Alvarez and Morales [Alvarez and Morales 1997] introduced an affine morphological multi-scale analysis to extract corners in an image. Tuytelaars and Gool [Tuytelaars and Gool 1999, Tuytelaars and Gool 2000] proposed two approaches for detecting image features in an affine invariant way. The former approach extracted Harris points and used the nearby edge for defining a parallelogram region. The latter approach initiated by extracting local intensity extrema and an ellipse was defined for the region determined by significant changes in the intensity profiles. Laptev and Lindeberg [Laptev and Lindeberg 2003] developed a method for finding elliptical blobs in an image for hand tracking. One of the most prominent and widely used feature detector is the Scale Invariant Feature Transform (SIFT) feature detector [Lowe 2004]. SIFT first performs

keypoint detection in an image at multiple resolutions in linear scale space and then assigns a descriptor vector with each keypoint by defining a local histogram of image gradient orientations around it. SIFT Detector-Descriptor combination results in keypoints that are scale, translation and rotation invariant. However, SIFT is not fully affine invariant and it works well only up to a 30° change in viewpoint angle between two images being matched [Yu and Morel 2011].

Another well-known Detector-Descriptor combination is Speeded Up Robust Features (SURF) feature detector [Bay et al. 2008]. SURF, is considered as an improved version of SIFT in terms of runtime efficiency. In a comparison study carried out by Gauglitz et al. [Gauglitz et al. 2011] between SIFT and SURF in different conditions of scale, viewpoint and illumination changes, it has been shown that SURF performs better than SIFT in all scenarios with fewer but sufficient number of detected keypoints. In an attempt to design a fully affine invariant feature detector, Yu and Morel [Yu and Morel 2011] proposed another enhancement in SIFT where image views obtained by changing two camera axis orientation parameters i.e. latitude and longitude angles were simulated. These image parameters were then clubbed with SIFT evaluated parameters involving simulated scale and normalized rotation and translation parameters, to make the detector work well under different affine conditions.

A number of detectors have been used in the past for performing image registration in a markerless Augmented Reality (AR) system. For example: Yuan et al. [Yuan et al. 2006] used Harris-Affine and Hessian-Affine feature extraction approach for designing a projective reconstruction technique using natural features. Gomez and Karatas [Gomez and Karatas 2014] made use of MSER features for detecting and tracking text in natural scenes. Chen et al. [Chen et al. 2007] proposed a system initialization algorithm for markerless AR using SIFT keypoints. SIFT features are also used by Li and Chen [Li and Chen 2010] for developing a markerless AR system for E-commerce applications. Similarly, ASIFT and SURF features are also used in literature [Ham and Golparvar-Fard 2013, Paz et al. 2012] for defining stable features of interests in an image, which are further used for providing reliable estimations in designing a markerless AR system.

Need of the proposed improvement: As revealed from the studies proposed till date, there still seems a need for developing a fully affine invariant feature detection procedure for performing image registration in an AR system. Drawbacks that still need to be resolved includes, high computation complexity and the ability of the detector to provide robust results in extreme changing imaging conditions like viewpoint change, illumination change etc. In this chapter, we propose a way to handle some of these limitations by providing an improved implementation of MSER for detecting stable regions of interest in an image. Selection of MSER detector for this can be reasoned on the results of the comparative study between the

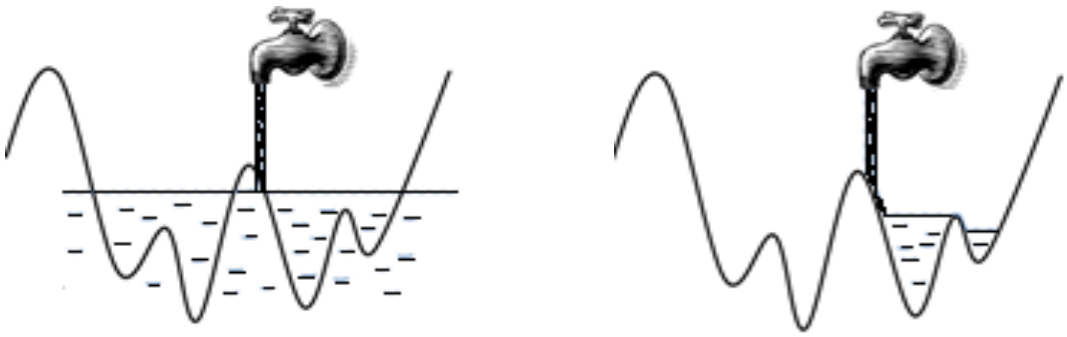
six feature detectors (Harris-Affine. Hessian-Affine, MSER, SIFT, ASIFT and SURF) that seem suitable for performing image registration in an AR system [Gupta and Rohil 2017a, also reproduced in thesis chapter 4].

6.2 Proposed Improvement to Maximally Stable Extremal Regions

6.2.1 Feature Detection

Standard MSER: MSERs are the connected components in an image having extremal regions either with higher (bright) or lower (dark) intensity properties than all the pixels on its outer boundary. The regions are defined solely by an extremal property of the intensity function in the region and on its outer boundary. The ordering of pixels intensities is done under monotonic transformations, due to which MSERs are considered to be stable features [Matas et al. 2004]. Standard MSER algorithm follows a four step procedure for extracting stable regions of interest in an image [Chapter 3, Section 3.1.1]. The concept can be explained informally as follows: Imagine a gray-level image *I* that could be thresholded to a maximum number of *k* levels. Pixels with intensity value below a threshold are treated as 'black' and those above or equal are considered as 'white'. Now, if all the thresholded images are seen as an image sequence represented as I_{it}, with frame *i* corresponding to threshold *t*, white image appears as first in the image sequence. Iteratively as the images are thresholded, black spots corresponding to local intensity minima appears to grow and regions corresponding to two local minima tends to merge at some point, subsequently forming a whole black image as the final output. The set of all connected components of all frames in the image sequence is the set of all maximal regions. Minimal regions could be obtained by inverting the intensity of *I* and running the same process.

Linear-MSER: The version of MSER chosen for experiments is the one proposed by Nister and Stewenius [Nister and Stewenius 2008], where a different analogy is chosen for maintaining the connected components of pixels, resulting in lower computational complexity. Figure 6.1 shows the difference between the two analogies where Figure 6.1(a) and Figure 6.1(b) describes the Standard MSER and Linear-MSER analogies respectively. The analogy chosen by Linear-MSER is a true flood fill approach (Figure 6.1(b)) where the water fills all the basins not at once but spills over to other parts as they become accessible to the current body of water i.e. the next pixel looked upon from the current pixel shouldn't have a lower grey level value, if it does, then the pixel with lower grey value is accessed first and the current pixel is stored for later processing. The reason behind this can be explained by considering the case of a ridge pixel having access to several edges with lower grey level pixels. In such a case the order of processing of pixels proposed by Nister and Stewenius [Nister and Stewenius 2008] goes wrong and the algorithm works in standard way of finding MSER's in an image.



Standard analogy used (MSER)

Analogy proposed by Nister and Stewenius (Linear-MSER)

(a) **(b)**

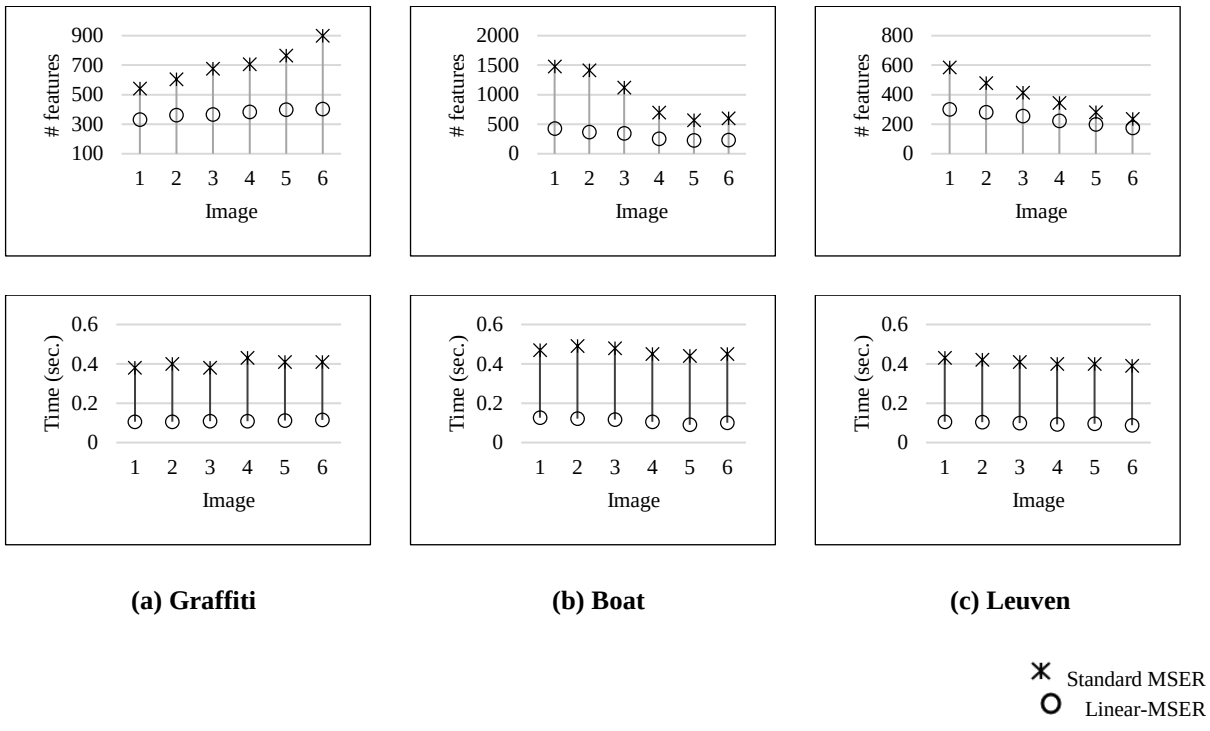
Fig. 6.1. Immersion Analogy used by (a) Standard MSER [Matas et al. 2004] and (b) Linear-MSER [Nister and Stewenius 2008]

In Linear-MSER [Nister and Stewenius 2008], worst case execution time is $O((n + ed)\log(g))$, where n and g symbolizes the number of pixels and grey levels in the image respectively and ed represents the number of edges in the image graph, where the value of $ed \approx 2n$ for four connected pixel images. Considering the grey level and connectivity as constant, MSERs are extracted in $O(n)$ time from an image. The comparative evaluation between standard and Linear MSER in terms of number of detected features and time taken for three random image-sets from Mikolajczyk dataset [Mikolajczyk 2007, Appendix 1] is shown in figure 6.1. Graphical representation of results clearly depicts the much lower computational complexity of Linear-MSER with very less difference in the number of detected keypoints. Though, the number of keypoints detected by Linear-MSER are less than those detected by standard MSER, the keypoint count in case of Linear-MSER remains stable even under changing imaging conditions.

Difference between Linear-MSER & Standard MSER: The main difference between the standard MSER algorithm [Matas et al. 2004] and Linear-MSER [Nister and Stewenius 2008] is the analogy they follow for building and maintaining the component tree. Figure 6.1 shows different analogies adopted by the two methods. For better understanding of the adopted analogies, the image in the figure is represented as a topographic map where peaks and valleys determine the image local maxima and minima in terms of intensity values respectively. In standard MSER [Matas et al. 2004], the level of water is lifted uniformly at all places i.e., make an assumption that the landscape is constructed as a porous field so that the water level is proportionate everywhere, or equivalently, assume that each local minima of the landscape has a small opening, allowing the water to enter everywhere. Therefore, following this analogy, standard MSER algorithm accesses complete pixels of an image at once and track of the connected components is kept

using union find algorithm. However, in Linear-MSER [Nister and Stewenius 2008], a single connected component of pixels is maintained i.e. water first fills up the basin where it was initially poured and then spills over to other parts as they become accessible to the current body of water (Figure 6.2).

MSLinear-MSER: In the proposed detector, Multi-scale Linear-MSER (MSLinear-MSER), regions of interest are extracted from the image at multiple resolutions of an image. This is done by forming a scale pyramid from the original image as shown in Figure 6.3 (Figure shows an example of scale pyramid formation for four octaves with four level stack of scaled images). Image is divided into octaves (a given image is incrementally convolved with Gaussians to produce images separated by a constant factor k in scale space. The set of these separated images is called an octave) and each octave is divided into multiple number of level stack of scaled images. Within an octave, the size of the image remains same but at each level the image is scaled down by a factor of k ($k = \sqrt{2}$, [Lowe 2004]) and stable regions of interest are extracted at all levels using the Linear-MSER algorithm. The same process is followed for multiple octaves where at each octave the image size is resampled by removing every alternate pixel in each row



(a) Graffiti **(b) Boat** **(c) Leuven**

Fig. 6.2. Comparative Results of performance of Standard MSER and Linear-MSER in terms of number of features detected (above) and time taken (below) for Graffiti, Boat and Leuven Image-Set

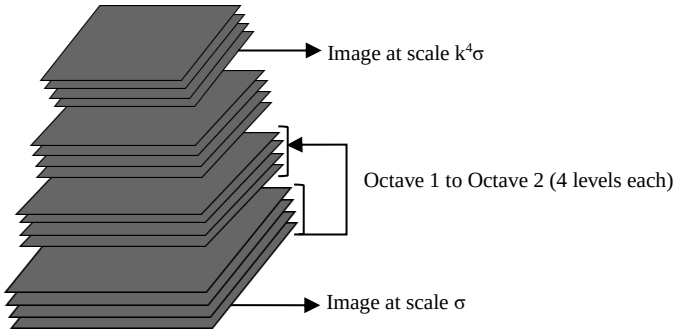


Fig. 6.3. Scale Pyramid formation

and column. The image that is chosen for resampling at each octave is the one with twice the value of scale (σ) from the bottom image. Refinement of detected regions of interest is done once all the octaves are processed in order to remove duplicate regions with same size and location. The location requirement that is used for elimination is that the centroid distance should be smaller than four pixels in the finer grid.

Complexity Analysis for Linear-MSER given by the authors [Nister and Stewenius 2008] for the algorithm is linear in terms of number of pixels. Maximum time taken by the algorithm is in terms of $log\ g$ time (where g is the number of grey-levels) to access the available pixels and to save the accessed pixels. Therefore, the worst case execution time is confined by $O((np + ne)log(g))$, where np is the number of pixels and ne is the number of edges in the image graph (such as $e \approx 2n$ for four-connected images). Now if, g, the number of grey-levels and ne, the number of edges in the image graph are considered as constants, the overall complexity of the algorithm evaluates to $O(np)$, linear time. On similar steps, complexity analysis for MSLinear-MSER is evaluated as:

$$O((np) + \left(\frac{np}{4}\right) + \left(\frac{np}{16}\right) + \left(\frac{np}{64}\right) + \left(\frac{np}{256}\right) + (\frac{np}{1024})),$$

as for scale pyramid formation containing 6 octaves, where for each octave subsampling is done by removing every alternate pixel in each row and column and np represents the total number of pixels. Therefore, the overall complexity of the algorithm evaluates to $O(np)$, linear time (Pseudocode for MSLinear-MSER is given in Appendix D.1).

Advantages of MSLinear-MSER: MSLinear-MSER detector results in interest points which are invariant to both scale and affine transformations. The detector uses characteristic scale selection in order to achieve scale invariance and output regions of the detector are normalised using affine shape adaption algorithm [Lowe 2004] to make them affine invariant.

Parameters of the detector are summarized as:

- MSLinear-MSER is implemented with 6 octaves with 5 levels each (Results for various combinations of octaves and levels per octave are shown in Table 6.2. In correlation, the reason for choosing 6 octaves with 5 levels each is also specified).
- Orientation computation is done for each detected keypoint.

Proposed improvement in the terms of feature description: MSLinear-MSER is combined with two widely used feature descriptors, SIFT and SURF, for describing the extracted region of interest uniquely among the others. The performance and comparative analysis of MSLinear-MSER detector with the two descriptors is explained in the next subsections.

6.2.2 Descriptor Evaluation

SIFT Descriptor: For every detected keypoint in an image, SIFT descriptor initially computes gradient magnitude and orientation for the neighboring pixels surrounding the keypoint. In order to avoid sudden changes in feature description, the gradient magnitude is weighted using a gradient window to give more priority to gradients located near the center of descriptor. All boundary affects such as sample shifts from one histogram to another or from one orientation bin to another is avoided by using trilinear interpolation to distribute the selected neighboring pixel's gradient value into appropriate histogram bins. This is done by multiplying each bin entry by a weight of $1 - q$ for each dimension, where q is the distance of the neighboring pixel from the bin central as computed in units of the histogram bin spacing. SIFT descriptor is a 128 dimensional vector [Lowe 2004] where a 16×16 sample array computed from the neighboring pixels is summarized to give a 4×4 descriptor where orientation histogram for each sub-region is associated with 8 orientation bins [Chapter 3, Section 3.1.2].

SURF Descriptor: Orientation is assigned for every neighboring pixel of the extracted keypoint by considering a circular region around it. A square region is constructed for descriptor extraction after computing all orientations and is centered on the extracted keypoint. The square region is oriented along the dominant orientation selected by computing Haar wavelet responses in both horizontal (dx) and vertical (dy) directions and then calculating the sum of all responses within a sliding orientation window covering an angle of $\pi/3$ [Bay et al. 2008]. SURF descriptor is a 64 dimensional vector [Chapter 3, Section 3.1.2] and takes less time for feature computation and matching as compared to SIFT descriptor.

Intension for performance comparison of Detector+Descriptor combinations: Once a set of interest points are detected using MSLinear-MSER detector from an image, there is a need to define their surrounding neighborhood by using a suitable descriptor for discriminative matching which is insensitive to local image deformations. Therefore to run a comparative performance evaluation between Linear-

MSER and the proposed detector, MSLinear-MSER, both the detectors are combined with SIFT and SURF detectors. The results for the comparative study are analyzed with respect to the number of correct matches and time taken by each Detector + Descriptor combination, i.e. four combinations, Linear-MSER combined with SIFT and SURF descriptor, represented as Linear-MSER+SIFT and Linear-MSER+SURF respectively, and similarly, MSLinear-MSER combined with SIFT and SURF descriptor, represented as MSLinear-MSER+SIFT and MSLinear-MSER+SURF respectively.

Discussion on expected performance for the combinations: As SIFT and SURF descriptors are proven to work well under affine deformations, the performance enhancement of Linear-MSER and MSLinear-MSER detectors is expected to improve as SIFT and SURF descriptors tend to distinctively define the neighborhood of the extracted keypoint and makes it possible to tracks the same feature in subsequent image frames under affine transformations. These descriptors also add illumination invariance property to the extracted keypoint. This improves the efficiency of image matching tasks and thereby, would definitely increase the efficiency of image registration procedure.

6.3 Augmented Reality System Building Mechanism

In order to check the accuracy of the proposed feature detector for analyzing and estimating the position of a virtual object that is to be placed in the real scene, an offline process is carried out in order to build a markerless AR System. The workflow of the system is described as a three stage process.

6.3.1 Feature Detection

Feature detection deals with finding regions of interest in an image and then describing descriptor vectors for each extracted feature. Descriptor vector defines some distinguished properties of an interest point that allows it to be correctly matched in follow subsequent images. Here in this research, MSLinear-MSER+SIFT is used for identifying regions of interest in an image while developing the AR system. Choice of this detector-descriptor combination is supported by the comparative evaluation discussed in Section 6.5.

6.3.2 Descriptor Matching

Once keypoint detection is done for an image, it is matched with an offline constructed feature map [Wientapper et al. 2011]. Here, dot product of two descriptor vectors is used as the comparison criteria. Dot product calculation is done between two set of descriptors where first set describes the keypoints detected in the camera image arranged as a an $N \times 128$ matrix, where N is the number of keypoints

extracted from the image and *128* values defines the descriptor vector associated with each keypoint. Second matrix represents the keypoints from the map file arranged as a transposed matrix and hence forms a $128 \times M$ matrix, where M represents the count of map file keypoints. Output of this multiplication is an $N \times M$ matrix from each of the keypoint combinations. Rows of the resulting matrix represents camera image keypoints, whereas columns indices to a point in the map file [Tam and Fiala 2012].

Best matches are picked among correspondences between each keypoint in camera image and map file. Two thresholds (relative and absolute) values are considered before dismissing or accepting the match. Also, multiple matches to a same point in map file are removed and only one match is retained.

6.3.3 Pose recovery

After performing descriptor matching between camera image and map file, every detected keypoint in the camera image corresponds to a point in the map file, allowing the position estimation of virtual object in the real world. Pose recovery is performed by Perspective-n-Point (PnP) algorithm [Gao et al. 2003] and Random Sample Consensus (RANSAC) algorithm [Fischler and Bolles 1981] is used to reduce the effects of outliers resulted from faulty correspondences between camera image points and map file points.

6.4 Methodology & Experimental Setup

6.4.1 Methodology

Which statistics/metrics to use and how: Table 6.1 describes the corresponding tables and figures listing in the chapter with respect to comparative evaluation of standard MSER with Linear-MSER, MSLinear-MSER execution for different octaves and Levels per octave combination and Linear-MSER with MSLinear-MSER.

6.4.2 Experimental Setup

Language, Software and Tools used for implementation and system specification: The experiments are carried out using single threaded code on a computer with 16GB RAM and Intel® Core ™ i5-3470 CPU@3.20ghz × 4 processor with cache size of 6144 KB.

Implementation details: Basic considerations of MSER are reanalyzed and some enhancements in the standard version are incorporated to make it more stable and affine invariant for feature detection procedures in real time applications. The main contribution of the work is the development of an

Table 6.1. Tables & Figures Representing Respective Performance Evaluation

	Table / Figure	Comments
Standard MSER and Linear-MSER	Figure 6.2	Figure 6.2 presents the results of comparative performance of Standard and Linear-MSER in terms of number of features detected and time taken.
MSLinear-MSER for different octaves and Levels per octave combination	Table 6.2	Results are displayed for octave and level combination of Octave:5 Level:4, Octave:6 Level:5 and Octave:7 Level:5
Linear-MSER and MSLinear-MSER	Figure 6.4	Figure 6.4 presents the results of comparative performance of Linear-MSER and MSLinear-MSER combined with SIFT and SURF descriptor, in terms of number fo correct matches between image pair and time taken.

appropriate detector-descriptor combination using MSER for AR applications. MSER detector is implemented in two forms, Linear-MSER and MSLinear-MSER, for extracting stable regions of interest in an image. SIFT and SURF descriptors are used in combination with the two detectors for analyzing the performance of the methods in terms of time complexity, affine invariant property and accurate correspondences between image pairs. The outcome shows that MSLinear-MSER+SIFT detector-descriptor combination works efficiently under various imaging conditions. To demonstrate the efficiency of MSLinear-MSER+SIFT detector, an AR system prototype is developed using the same approach. Please refer to Appendix D for more implementation details.

6.5 Data Reporting

Dataset used for experiments: The experiments are performed on Mikolajczyk dataset [Mikolajczyk 2007, Appendix A.1] as described in Chapter 4 [Section 4.6, Figure 4.2], containing eight image-sets with six images in each set (total 48 images). These images are varying under five imaging conditions i.e. viewpoint change, scale change, image blur, illumination change and JPEG compression. Among these five imaging conditions, three (viewpoint change, scale change and image blur) have two image-sets each. One image-set contains a set of structured images and the other contains a set of natural images.

For MSLinear-MSER, experiments are carried out at varied number of both octaves and levels per octave for all image-sets. Outcome for all image-sets is tabulated in Table 6.2, where comparisons between different variations of octaves and levels per octave is done using MSLinear-MSER+SIFT in terms of number of correct matches and time complexity. Tabulated results can be used to specify the reason behind choosing the number of octaves as six and level of stacked images within each octave as five (values in bold style font in Table 6.2). For seven octaves and five scales per octave, the time

Table 6.2. Performance of MSLinear-MSER+SIFT with respect to number of octaves and level of stacked images per octave

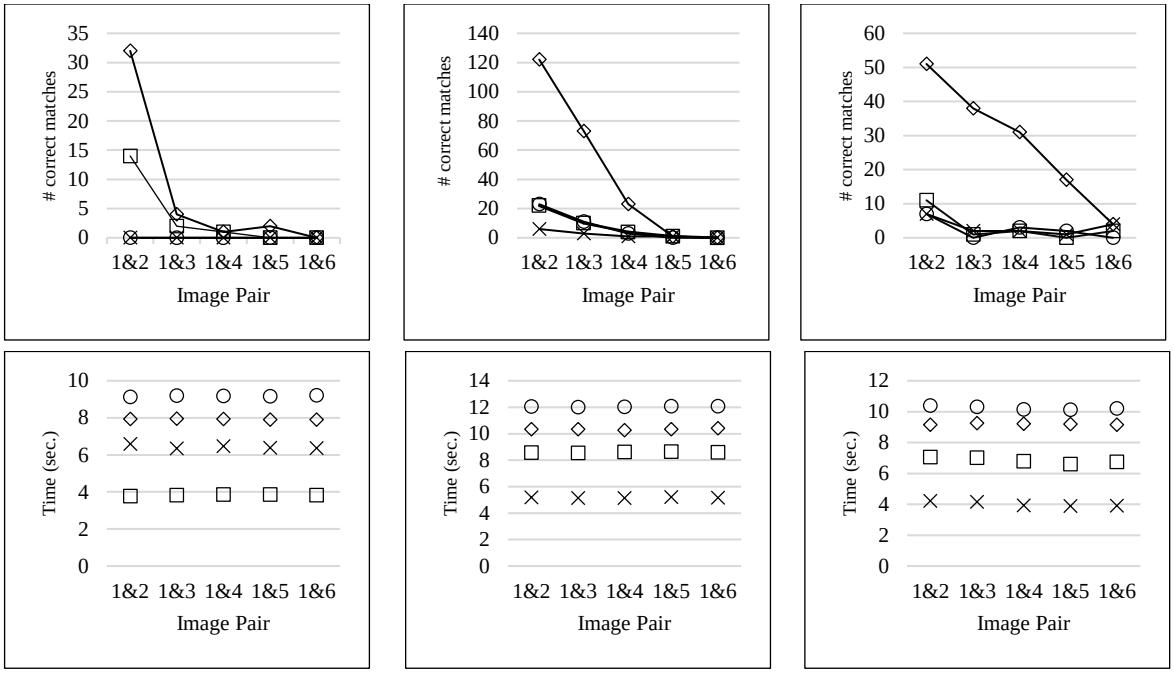
Image-Set	Image Pair	MSLinear-MSER+SIFT 5 Octave 4 level		MSLinear-MSER+SIFT 6 Octave 5 level		MSLinear-MSER+SIFT 7 Octave 5 level	
		cm*	T#	cm*	T#	cm*	T#
Graffiti	1&2	32	7.95	**32**	**8.76**	37	22.44
	1&3	4	7.96	**4**	**8.8**	6	22.46
	1&4	1	7.94	**1**	**8.75**	2	22.51
	1&5	0	7.92	**2**	**8.73**	2	22.44
	1&6	0	7.92	**0**	**8.77**	3	22.53
Wall	1&2	122	10.36	**123**	**11.34**	139	29.6
	1&3	73	10.36	**73**	**11.29**	83	29.62
	1&4	23	10.29	**23**	**11.23**	21	29.62
	1&5	1	10.36	**2**	**11.42**	2	31.41
	1&6	0	10.42	**0**	**11.39**	0	30.2
Boat	1&2	51	9.17	**56**	**9.69**	59	26.28
	1&3	38	9.27	**39**	**9.62**	43	27.24
	1&4	31	9.22	**33**	**9.68**	37	26.51
	1&5	17	9.21	**19**	**9.62**	19	26.31
	1&6	4	9.17	**5**	**9.69**	5	26.32
Bark	1&2	60	6.83	**59**	**7.31**	62	18.68
	1&3	35	6.80	**38**	**7.27**	40	18.62
	1&4	30	6.83	**31**	**7.31**	37	18.72
	1&5	19	6.76	**19**	**7.31**	26	18.76
	1&6	2	6.79	**11**	**7.29**	15	18.76
Bikes	1&2	49	10.71	**52**	**11.33**	56	30.99
	1&3	47	10.67	**49**	**11.36**	48	30.94
	1&4	29	10.72	**30**	**11.38**	30	30.93
	1&5	34	10.74	**38**	**11.28**	38	30.91
	1&6	28	10.75	**31**	**11.32**	31	30.37
Trees	1&2	65	11.60	**67**	**12.43**	69	33.8
	1&3	44	11.71	**47**	**12.45**	48	33.67
	1&4	19	11.59	**24**	**12.38**	23	33.63
	1&5	19	11.56	**21**	**12.29**	20	33.34
	1&6	14	11.63	**18**	**12.26**	19	32.75
Leuven	1&2	34	8.29	**60**	**8.76**	43	23.98
	1&3	20	8.29	**33**	**8.72**	24	23.96
	1&4	25	8.27	**37**	**8.74**	30	23.87
	1&5	16	8.26	**25**	**8.72**	23	23.78
	1&6	17	8.26	**20**	**8.72**	21	23.8
Ubc	1&2	110	8.07	**113**	**8.62**	132	23.32
	1&3	88	8.07	**91**	**8.6**	102	23.26
	1&4	42	8.08	**46**	**8.57**	50	23.19
	1&5	34	8.05	**38**	**8.57**	38	23.15
	1&6	19	7.99	**21**	**8.54**	19	22.9

cm* = Correct number of Matches
T# = Time Taken (in seconds**)**

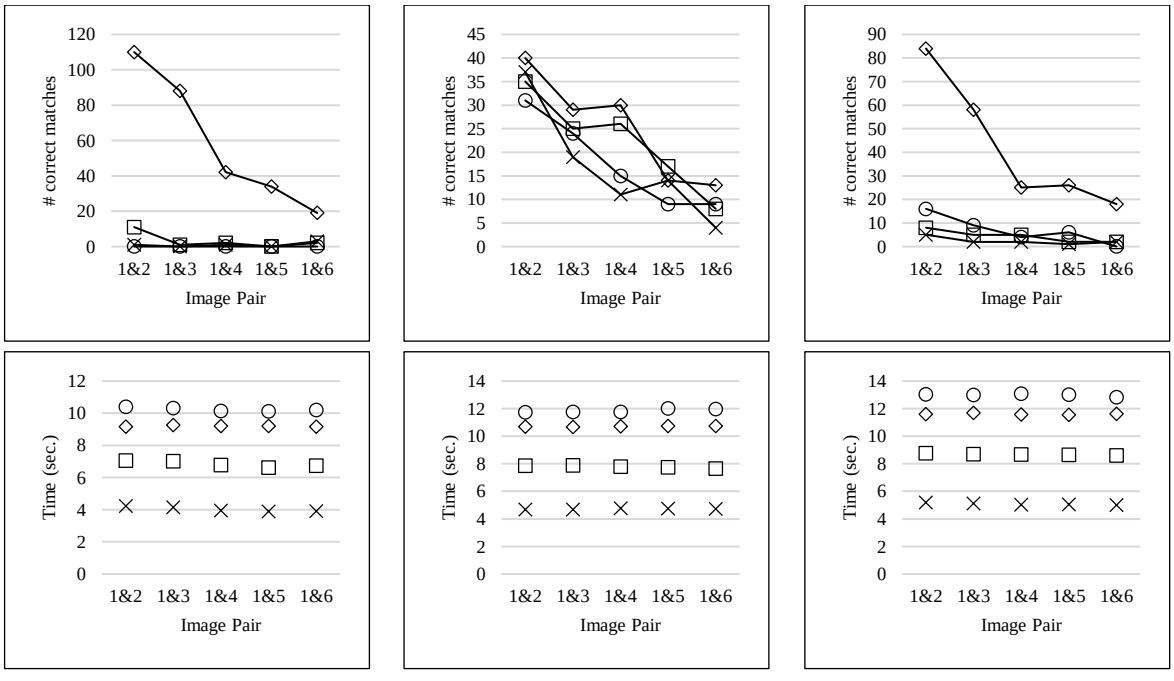
complexity of the process increases with a great value, however, there is not much difference in the number of correct matches that are extracted between image pairs. Similarly, for five octaves and four scales per octave, the time complexity remains comparable but the number of correct matches between images suffers in few cases.

For the four detector-descriptor combinations (Linear-MSER+SIFT, Linear-MSER+SURF, MSLinear-MSER+SIFT and MSLinear-MSR+SURF), comparative study for performance evaluation is done in terms of number of correct matches and time taken. For this, MSLinear-MSER detector is executed with 6 octaves and 5 levels per each octave. Figure 6.4 shows a graphical representation of results for all eight image-sets. The linear representation shows that for all cases MSLinear-MSER detector when combined with SIFT descriptor generates a recognizable high number of correct matches

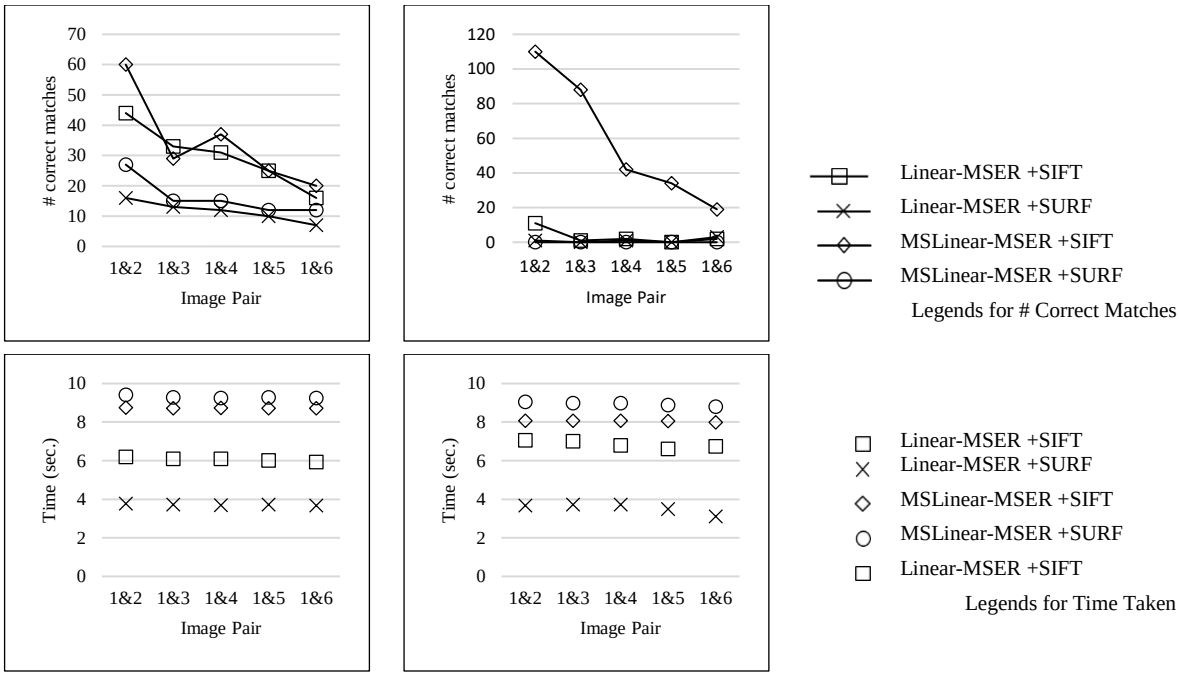
between image pairs as compared to the rest three techniques. Also, the time taken by MSLinear-MSER+SIFT process is not much high as compared to Linear-MSER+SIFT and Linear-MSER+SURF feature detectors. Likewise, when MSLinear-MSER+SIFT outcome is compared to MSLinear-MSER+SURF process, the number of correct correspondences between image pair is considerably low and the time taken for processing is almost equal.



(a) Graffiti **(b) Wall** **(c) Boat**



(d) Bark **(e) Bikes** **(f) Trees**



(g) Leuven **(h)Ubc**

Fig. 6.4. Comparative Results in terms of number of correct matches and time taken for image pairs by Linear-MSER and MSLinear-MSER for the Mikolajczyk Dataset [Mikolajczyk 2007, Appendix A.1]

Figure 6.5 displays the results of MSLinear-MSER+SIFT procedure for identifying correspondences between two image pairs in Leuven and Ubc image-set respectively. Here, in extreme changing conditions of illumination and JPEG image quality, the method is significantly efficient in terms of correspondences that are exhibited between two images. The main drawback of MSLinear-MSER+SIFT method is the

(a) **(b)**

Fig. 6.5. MSLinear-MSER+SIFT (6 Octave 5 level): Correspondences obtained between two images (a) Leuven Image-Set (Image 1&6) (b) Ubc Image-Set (Image 1&6) are 20 and 21 respectively

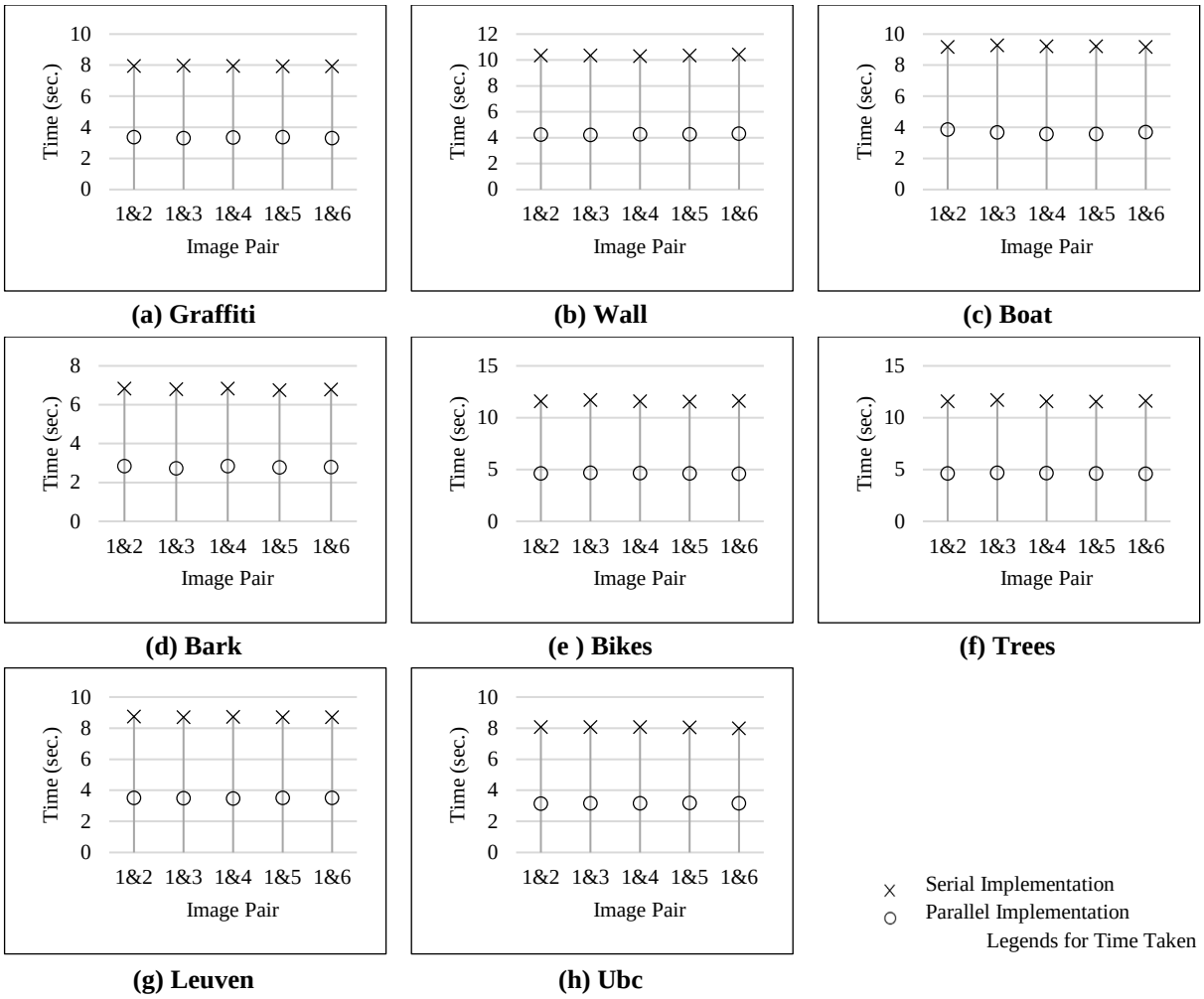


Fig. 6.6. Time Taken (in seconds) by Serial and Parallel implementation of MSLinear-MSER+ SIFT

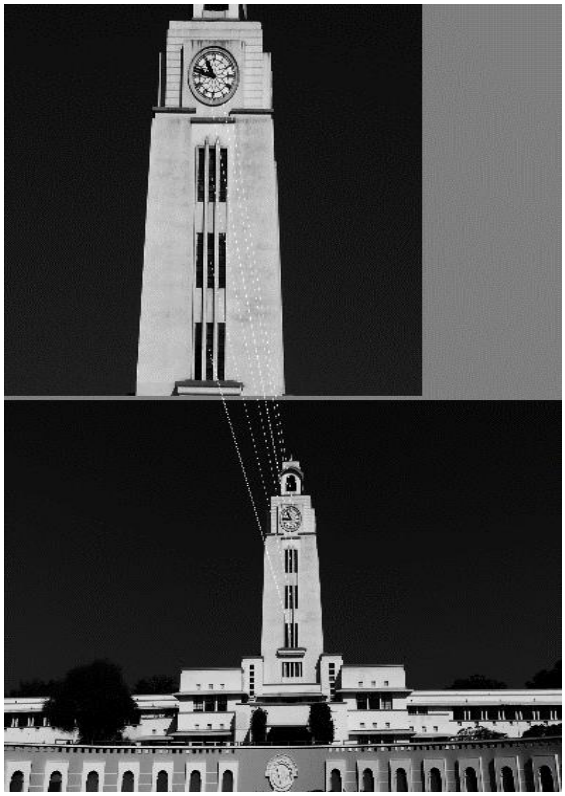
time complexity that makes it incapable for real time applications. To overcome this disadvantage, the method is implemented using parallel implementation with Open MP resulting in almost 50% of time reduction in time. Figure 6.6 shows the comparative results for serial and parallel implementation of the method. The method could be made faster by incorporating more degree of parallelism.

MSLinear-MSER+SIFT detector is further used for building an offline AR system prototype for analyzing the accuracy of the method in identifying correct position of virtual object in a real scene. The scenario taken into consideration is given in Figure 6.7, where the images are displayed as (a) map file corresponding to the position of the virtual object, (b) Real Scene camera capture, (c) evaluation of correspondences between (a) and (b), and finally (d) displays the augmented image where a yellow circle appears on the tower. The experiment is carried out on two videos with duration of eight and six seconds respectively. Both videos contained frames at varying rotation angles up to 90°. The evaluation of rotation

angle is done by choosing first video frame as the reference image (say at angle 0°) and rotation angle for rest of the images is evaluated with respect to it. Figure 6.8 represents the rotation angle evaluation result for two images from Figure 6.9, where Figure 6.8(a) represents rotation angle evaluation for Figure 6.9(a) and Figure 6.9(b) and Figure 6.8(b) represents rotation angle evaluation for Figure 6.9(a) and Figure 6.9(i)

(a) Map File (Reference Image)

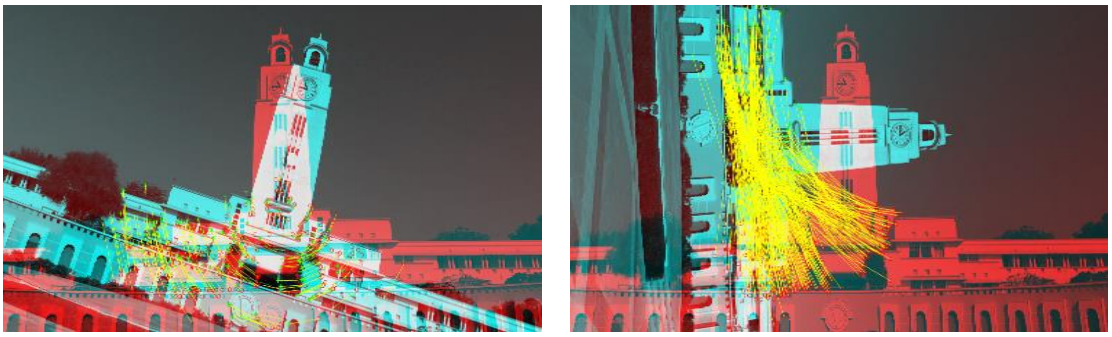
(b) Camera Input



(c) Keypoint matching output between the map file and the camera image.

(d) Augmented Image (yellow circle appears on the tower)

Fig. 6.7. AR System prototype



(a) **(b)**

Fig. 6.8. Angle of Rotation Estimation (a) Between the images shown in Figure 6.9 (a) and 6.9 (b). (b) Between the images shown in Figure 6.9 (a) and 6.9 (i)

(a) Initial Image **(b) Rotation Angle 20°** **(c) Rotation Angle 30°**

(d) Rotation Angle 50° **(e) Rotation Angle 70°** **(f) Rotation Angle 80°**

(g) Rotation Angle 85° **(h) Rotation Angle 90°** **(i) Rotation Angle 90°**

Fig. 6.9. Augmented Views of the scene where a yellow circle appears on the tower in different frames of a video

Table 6.3. Precision values for correct augmentation

	Video1	Video 2
Total number of frames:	244	185
Number of frames with correct augmentation:	225	180
Number of frames with incorrect augmentation:	19	05
Precision:	0.92213	0.9729

and are equated at a variation of $20^{\circ}$and 90° angle respectively from the reference frame. In Figure 6.9, augmented view for different frames of video is displayed. The rotation angle of displayed images vary from a range of 20° to $90^{\circ}$and it can be seen that even at a rotation angle of 90°, virtual object (yellow circle) is placed at the right position in the real scene i.e. on the tower (as mapped from the map file which is serving as the reference image for evaluating the position of virtual object). Precision values calculated for correct augmentation in all video frames for the two videos is given in Table 6.3 and the evaluated results exhibits good accuracy of the method.

6.6 Result Analysis and Interpretation

AR for all practical purposes requires extensive computation, accurate view alignment and real-time performance. To address some of these limitations, an improved method of feature detection is proposed and discussed in this chapter using MSER. For extracting the regions of interest in an image, a faster variation of MSER is chosen as the method uses a true flood fill approach for building and maintaining the component tree and hence sustain true worst-case linear time complexity. The performance evaluation of this detector, referred as Linear-MSER, is done with the standard MSER algorithm in terms of number of keypoints detected in an image and time taken for image-sets containing images with varied imaging conditions like viewpoint, scale and illumination change (Figure 6.2). The results show stable performance of Linear-MSER in terms of keypoint detection with much less processing time.

Therefore, for the proposed feature detection procedure, MSLinear-MSER, Linear-MSER is implemented at multiple scales of an image using scale pyramid formation of multiple octaves and level of scaled images per octave in order to increase the affine invariance properties of the detector. MSLinear-MSER is tested at various combinations of octaves and level of scaled images per octave to analyze its best performance at a particular combination (Table 6.2). The two detectors, Linear-MSER and MSLinear-MSER, are then combined separately with SIFT and SURF feature descriptors for image matching performance comparison (Figure 6.4). Performance evaluation is done under varying imaging conditions like changes in viewpoint, scale, blur, illumination and JPEG compression.

Results show that, MSLinear-MSER+SIFT performs best over the other three detector-descriptor

combinations when executed at 6 octaves and 5 levels per each octave in terms of number of correspondences found between an image pair and does reasonably well when compared in terms of time complexity (Figure 6.4). Moreover, individual time complexity of the detector serve as a main drawback for it to be used in any AR application, thereby, the detector is implemented using parallel implementation to further lower its time complexity. The comparative results of serial and parallel implementation of the detector represents almost 50% time reduction, i.e. twice speedup (Figure 6.6) and could be made faster by adding more degree of parallelism. This observation is true for all image-sets taken into consideration, containing images that are affine transformed in one way or other. Moreover, MSLinear-MSER+SIFT promising performance can be summarized with respect to the following parameters:

1. Execution time: Since the SIFT descriptor is a 128 dimensional vector and the method is processing at multiple resolutions, so to achieve better performance along the subsequent two parameters (i.e. number of correct correspondences between image pairs and affine invariance), the time taken in serial implementation is reasonably justified and is comparable to MSLinear-MSER+SURF and LinearMSER+SIFT approach but the time taken by the parallel implementation is less than or at least comparable to the time taken by the serial implementation of MSLinear-MSER+SIFT when it is executed with 6 octaves and 5 levels per each octave.

2. Number of correct correspondences between image pairs: Since the number of correspondences detected are accurate and stable, hence more the detected correspondences the better is the performance of the algorithm. So in all cases the number of correspondences detected between image pairs by MSLinear-MSER+SIFT are more than the other three detectors when it is executed with 6 octaves and 5 levels per each octave.

The number of correspondences also depends upon the quality of images. On visual inspection of the image-sets we observe that the quality of Graffiti and Wall image-sets ranges from fair to good. Same is the case for bark and boat image-sets. However for blur imaging condition, the Bikes image-set is of low quality and only first three images of Trees image-set are of good quality. For Leuven image-set quality of images is good and for Ubc, only first image among the six images is of good quality. This visual quality is also reflected in the number of correspondences detected. For example, in Figure 6.4(e), number of correspondences detected doesn't vary much for four methods. This is due to the poor quality of images. However these images are visually similar and are distorted only by high degree of blur change.

3. Affine invariance: Among the eight image-sets, five image-sets i.e. Graffiti, Wall, Boat, Bark and Leuven can be considered as if they are variants of an image after applying some affine transformation. For all these image-sets and even for other three image-sets MSLinear-MSER+SIFT outperforms in terms

of number of correct matches between image pairs. Hence we could state MSLinear-MSER+SIFT method is more affine invariant when it is executed with 6 octaves and 5 levels per each octave.

To exhibit the efficiency of MSLinear-MSER+SIFT in AR, a prototype of an AR system is also developed using this approach and is discussed in this chapter. The procedure and setup to develop the system is given in Section 6.3 and Section 6.5 respectively. The results are validated by using precision metric for interpreting the accuracy of the approach (Table 6.3).

6.7 Summary

In this chapter, an improved implementation of MSER feature detector is discussed to overcome few shortcomings of existing feature detectors that deals with extensive computation, flawed image matching and are unsuitable for designing real time applications. Also, this chapter discusses the procedure for designing an AR system using the proposed detector. Next chapter presents the proposed novel feature descriptor based on local elliptical sampling of keypoint neighboring pixels.