VISION BASED ADAPTIVE OBSTACLE DETECTION, ROBUST TRACKING AND 3D RECONSTRUCTION FOR AUTONOMOUS UNMANNED AERIAL VEHICLES

By

Sushil Pratap Bharati

Submitted to the graduate degree program in Department of Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements for the degree of Master of Science.

Dr. Guanghui Wang, Chairperson

Committee members

Dr. Suzanne Shontz

Dr. Bo Luo

Date defended: 01/22/2018

The Thesis Committee for Sushil Pratap Bharati certifies
that this is the approved version of the following thesis :

VISION BASED ADAPTIVE OBSTACLE DETECTION, ROBUST TRACKING AND 3D RECONSTRUCTION FOR AUTONOMOUS UNMANNED AERIAL VEHICLES

Dr. Guanghui Wang, Chairperson

Dr. Suzanne Shontz,

Dr. Bo Luo,

Date approved: 01/22/2018

Abstract

Autonomous navigation of UAVs in real-time with vision-based camera system requires obstacle detection and tracking mechanism along with 3D reconstruction of a scene for the depth analysis. Though the problem is very interesting, obstacle detection and tracking along with the fundamental matrix estimation for 3D reconstruction still pose a big challenge. To address these issues, a vision-based fast and robust obstacle detection and tracking approach is proposed by integrating a salient object detection strategy within a kernelized correlation filter (KCF) framework. Moreover, an adaptive obstacle detection technique is proposed to refine the location and boundary of the object when the confidence value of the tracker drops below a predefined threshold. In addition, a reliable post-processing technique is implemented for an accurate obstacle localization. Similarly, detection of outliers present in noisy image pairs for a robust fundamental matrix estimation and further 3D reconstruction of a scene helps significantly in obstacle avoidance in UAVs. Given a noisy stereo image pair obtained from the mounted stereo cameras and initial point correspondences between them, reprojection residual error and 3-sigma principle together with robust statistic based Q_n estimator (RES-Q) is proposed to efficiently detect outliers and estimate the fundamental matrix with superior accuracy. The proposed approach has been extensively tested through quantitative and qualitative evaluations on a number of challenging datasets. The experiments demonstrate that the proposed approach significantly outperforms the state-of-the-art methods in terms of tracking speed and accuracy. Also, RES-Q is found to be robust than the other classical outlier detection algorithms for both symmetric and asymmetric random noise assumptions.

Acknowledgements

I would like to express my humble gratitude to my advisor Dr. Richard Wang, who has guided me throughout my Master's program at the The University of Kansas and my committee members Dr. Suzanne Shontz and Dr. Bo Luo for providing their valuable suggestions and guidelines to construct this thesis. I would also like to pay my sincere respect towards all the faculty and staff members of the EECS department at the University of Kansas. Pursuing my Master's degree at KU has been a great journey. The challenging course work together with an exciting research have helped me learn a lot personally as well as professionally. There have been days when the research would not produce great results and I would ponder over the winding codes and try to dig into minute details and wished everything would be sorted out easily. But during the course of time, I have realized that patience and grit are the only success formulae to produce a quality research. I cannot emphasize more on how helpful Dr. Wang has been throughout my academic career. He never hesitated to sacrifice his weekends or to rigorously review the research papers to meet the deadlines of which I shall always be thankful to him. I would also like to take an opportunity to thank my parents and family for constantly supporting me on this incredible journey. Also, my friends and colleagues at Lawrence have been a great support. Without them, I cannot imagine staying away from home and focus on my academics. Finally, I would like to thank the graduate students in our Computer Vision group at the University of Kansas under Dr. Wang's supervision. Most of the learnings happened with them.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Automating visual detection and tracking of moving objects by intelligent autonomous systems, such as unmanned aerial vehicles (UAVs), has been an active research topic for the past decades in computer vision. The research has diverse applications extending from military, surveillance, security systems, aerial photography, search and rescue, object recognition, auto-navigation to human-machine interactions [1]. Recently, computer vision is being extensively used in roadside vehicle positioning and tracking as well as in intelligent transportation systems for the vehicle as well as the passengers' safety [2], [3]. Due to its emerging multidisciplinary usage, a handsome number of companies are developing their own UAV systems, such as Google's Project Wing, Amazon Prime Air and DHL's parcelcopter.

However, designing such intelligent UAVs is pragmatically challenging. It is vital to keep track of other UAVs, birds, airplanes or other possible flying objects during the flight of an autonomous UAV. Hence, identifying such potential obstacles precisely and localizing them in real-time for successful collision avoidance and autonomous navigation is essential. Recognizing such possible threats in a real-time and embedding such computationally sound algorithm in flying UAVs demands an ample amount of research and engineering.

Among all recent advancements in technology, vision-based sense and avoid system is becoming a more popular choice since cameras are light-weight and low-cost as well as they provide richer information of the surrounding than other available sensors, thus appropriate for UAVs with limited payload capacity. A successful sense and avoid system should be able to automatically detect a possible obstacle which may be present in the path of the flying UAVs and track it in order to prevent a possible collision.

Also, the popularity of vehicle mounted stereo cameras for recognition and localization in intelligent vehicles is increasing tremendously. Application of computer vision algorithms in object tracking [4, 5] and 3D reconstruction using stereo cameras for depth estimation [6], 3D lane detection [7], traffic sign recognition [8], pedestrian detection and tracking with night vision [9], driver assistance [10] have been extensively studied in the field of computer vision. Stereo vision, along with the sensor fusion, has been used in a number of security and safety applications as the trend of autonomous and intelligent vehicles is increasing. In order to successfully exploit the benefits of stereo cameras, one needs to estimate the fundamental matrix accurately from the noisy matching point correspondences in the stereo image pair. Thus, the obtained fundamental matrix can be successfully used to reproduce 3D reconstruction of the scene for further analysis, such as determining the distance to the other vehicles ahead of the autonomous vehicle, buildings or obstacles ahead, traffic lights, poles, and pedestrians. However, noises in the images captured by these stereo cameras are inevitable due to unpredictable disturbances in the camera system. For example, an intelligent vehicle might be running through a rough terrain or the stereo cameras might be affected due to the presence of dust, rain, fog, irregular illumination, reflectance, occlusion, and other surrounding electromechanical or electromagnetic interferences. Similarly, instances like inconsistent feature extraction, depth discontinuities or cyclic patterns also add to this. Thus, a real-time and robust algorithm is required to filter the correct matching points from the noisy outliers for an accurate fundamental matrix estimation which helps in precise 3D reconstruction of the scene for further applications in an intelligent vehicle.

1.1 Motivation

Unmanned Aerial Vehicles (UAVs) are widely used for commercial, industrial or scientific purposes like aerial photography, drone based agriculture, surveillance and faster product deliveries. It is well understood that UAVs can make its way through challenging route with the help of transponder-based collision avoidance system in the absence of air traffic or skycrapers, tall towers, trees, bridges or statues. But in the presence of air traffics such as aeroplanes, other flying

objects like drones, parachutes or birds and tall obstacles, UAV requires a visual guide. Absence of such visual assistance may lead to catastropic accidents which may further result in loss of life or assets or both.

Generally, UAVs are lightweigth and small. They have their own flying modules and operating systems on board. Therefore, installation of radars in an UAV for navigation through obstacles on its path is not pragmatic because radars are bulky and usually have big transmitters and receivers. Moreover, radars are highly susceptible to electromagnetic interferences. Simlarly, using obstacle sensors is not a good choice in UAVs due to their feeble singal coverage and inaccuracies. One of the best ways to assist UAVs in such scenario is to add a small, light-weight wide angle camera or a pair of stereo cameras as deemed neccessary along with the flight system. This would overcome the challenges of maneuvering through the complex path and barely impact the overall weight of an UAV. Additionally, installation of image processing softwares or firmwares on UAVs is much cost-effective than the installation of giant radar units.

It is also important to note that such compact arrangements in UAV can also assist with the onboard radar system if present. Visual aids in UAVs will effectively help in the autonomous navigation in case of false alarms or miss detections during radar data processing. Hence, object detection, depth estimation and tracking using computer vision aided mechanism is the most suitable and reliable method in an autonomous navigation of UAVs.

1.2 Goals

Provided a scenery or a landscape, human eyes tend to first notice the characteristic features they could sense from the entire view [11]. These characteristic features, which help the human brain distinguish between a particular object and its background could be the basis for a successful sense and avoid algorithm that segregates the object from its background.

On the other hand, given an initially detected position of an object in the initial frame, an intelligent system should correctly localize the position of the moving object throughout the sequence. However, most of the previous works focused only on object detection or object tracking, rather

than creating an intelligent system capable of simultaneous detection, depth estimation and tracking in real-time. Thus, a novel and intelligent vision-based system that can automatically detect, localize, and track the objects in high speed is necessary.

In addition, it is required to manually initialize most of the trackers with the position of the target in the first frame making them an incomplete automated system. Since manual labeling is required in such trackers, they are not suitable for fully autonomous UAVs. Moreover, such trackers are not fit for long-term tracking purposes. Similarly, most of the trackers assign a fixed size bounding box only to track a part of the moving object in a scene, although the object changes its shape and size throughout a sequence. Such approaches are inappropriate for estimating the shape and size of the obstacle and therefore rectifying the path of UAVs to avoid possible collisions with the obstacle. Some of the tracking by detection methods aim to provide a changing bounding box according to object's shape and size but they perform slower and thus inapt for real-time implementation.

Furthermore, the estimation of a fundamental matrix from feature point correspondences between a stereo image pair of the same scene is one of the most important steps in the field of 3D computer vision. It is considered a vital step since the fundamental matrix stores all the geometric information for the relative transformation between an image pair and helps in the projective reconstruction of a scene. In general, a stereo image pair is taken with the help of two cameras with different orientations. However, the two-view images can also be taken using a single camera with appropriate motion dynamics such as rotation with translation. It is shown in [12] that an estimation of such a 3×3 fundamental matrix is governed by the epipolar geometry between the camera orientations.

It would have been a lot easier to estimate the fundamental matrix if there were no erroneous matching point correspondences between the image pair. Pragmatically, a number of mismatched points are present in two views and a reliable outlier detection algorithm should be employed to filter the outliers efficiently. The matching point pair is considered to be an outlier if it violates the epipolar constraint or comes from different 3D coordinates. The epipolar constraint is merely

a geometrical constraint between the stereo image pair. It can be understood as if a feature point in one image is matched with the feature point in the other image, then both the points must lie in their respective epipolar lines. The epipolar lines are determined using the point coordinates and the fundamental matrix. Thus, the main goal of an outlier detection algorithm is to validate the match points against such constraints and also check if the point pairs come from the same 3D coordinates. Those pairs of matching points that follow these constraints are termed as inliers, otherwise they are called outliers.

Thus, it is very important to find an algorithm that is robust to general noise models in the stereo image pair for accurate fundamental matrix estimation and 3D reconstruction in intelligent vehicle applications like in UAVs. Similarly, on the quest to find such algorithms one should also make sure that the algorithm performs with reliable efficiency. Otherwise, the unnoticed outliers would severely impact the accuracy of the estimated fundamental matrix and the UAV shall possess inaccuracies which could be risky.

1.3 Contribution

In this thesis, we propose a fast, reliable and accurate object localization and tracking approach for the autonomous navigation of the flying UAVs by integrating the techniques for salient object detection [13] with the kernelized correlation filter [14]. Our approach achieves better detection and tracking results compared to the state-of-the-art method in terms of speed and accuracy as demonstrated in our experimentation chapter. Fig. 1.1 shows the result of the proposed method along with SAMF [15] and KCF [14]. It can be clearly observed that, although the appearance of the flying object undergoes deformations, illumination or scale variations, the proposed method accurately confines the concerned object compared to other peer trackers.

Additionally, we propose a reprojection residual error based iterative approach RES-Q for robust estimation of the fundamental matrix. RES-Q successfully exploits both the Gaussian, as well as the other generalized noise model, which are often present in the duo image pair captured by a stereo cameras mounted on an UAV. The proposed approach exploits the reprojection residual

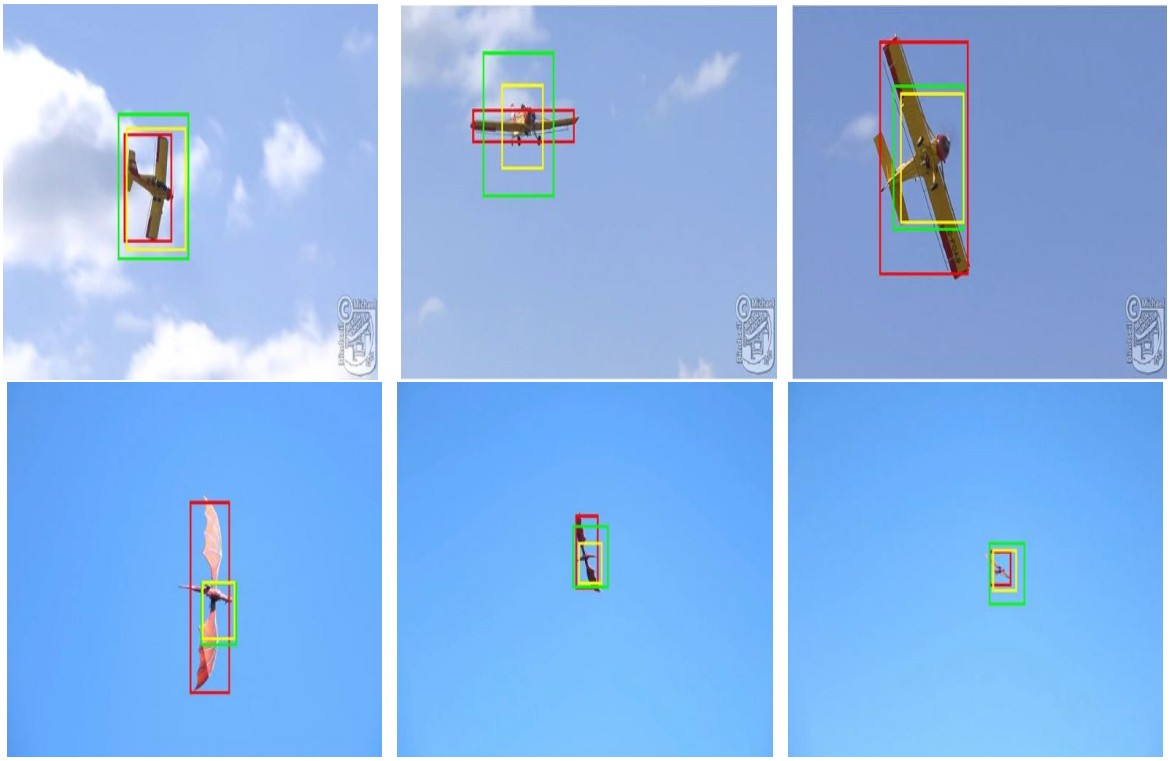


Figure 1.1: Frames demonstrating our proposed approach in action. Our algorithm quickly adjusts to variations in shape, size and illumination of the object. Color code: 'red' marks our approach, 'yellow' marks SAMF and 'green' marks KCF.

error as a confidence measure of the fundamental matrix. Similarly, the algorithm also utilizes an efficient robust statistics based estimator Q_n [16], which performs much better than the MAD [17] estimator in the Gaussian as well as other noise distributions for outlier elimination to estimate an accurate fundamental matrix for 3D computer vision tasks. Compared to the state-of-the-art outlier detection techniques, RES-Q is more efficient as well as accurate which is demonstrated via several experiments on both synthetic data and standard real datasets available online. Fig. 1.2 shows the efficiency of choosing RES-Q.

The main contributions of this thesis are listed below:

- The proposed approach correctly localizes and generates an adaptive bounding box in real-time despite varying shape and size of the object throughout the sequence.

- Our approach, by integrating the detection and tracking strategies together and forming a

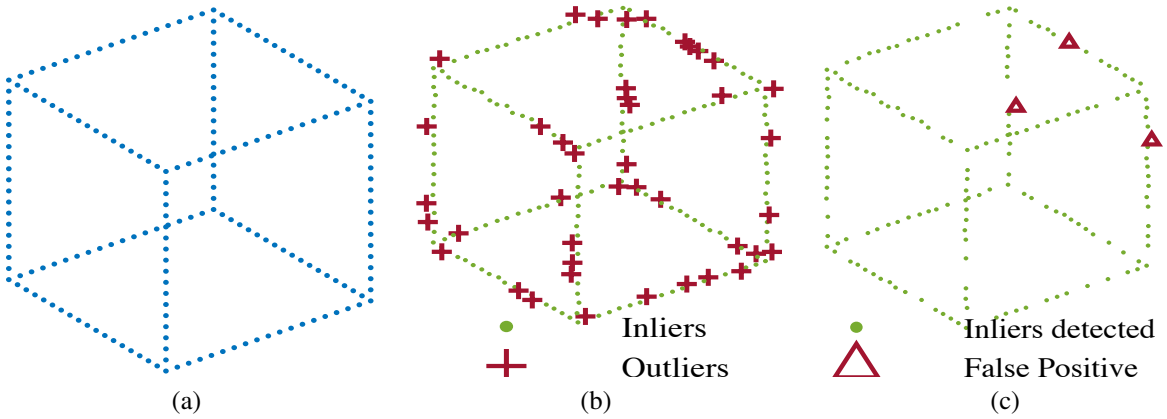


Figure 1.2: RES-Q rejects most of the outliers successfully. (a) Original simulated cube. (b) Cube with inliers and 20% outliers. (c) Resulting cube with detected inliers where most of the outliers present earlier are rejected after running through RES-Q.

closed loop system, achieves long-term error-free tracking.

- The proposed approach, by training the filter from previous frames, tracks the object in subsequent frames without the need of any computationally expensive supervised training for the detection.

- RES-Q efficiently removes the outliers from matching point correspondences, determines fundamental matrix accurately and reconstructs the scene.

- The proposed system is fully automated, accurate and has superior real-time speed without requiring any sort of manual intervention.

A part of this thesis - [4] has been published in IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI), [5] has been published in IEEE Transactions on Intelligent Vehicles and [18] has been submitted to IEEE Transactions on Intelligent Vehicles at the time of writing this thesis.

Chapter 2

Literature Review

2.1 Object Detection

Previous research on saliency map-based object detection can be broadly classified into two methods — top-down and bottom-up. In the top-down methods [19–21], detection is executed on the reduced search space since all the possible objects in an image are localized. But these methods are unrealistic for real-time object detection because they are mostly task-driven and accompanied by supervised learning. On the other hand, bottom-up methods [13, 22–25] compare the feature contrast of the salient region with the background contrast by using the low-level features (like the color, contrast, shape, texture, gradient and spatio-temporal features) from an image. Such methods have higher possibilities to fail in the case of complex images as they do not have prior knowledge of the localization of the object or the number of objects present in an image. In contrast, the top-down methods require proper training before detection.

Object detection in [22,26] used a geodesic saliency map by looking at the contrast of an image and calculating the distance of each pixel from the background seeds to segment a region in the image. A supervised regression-based segmentation approach in [24] used binary classifier but was limited to detecting single objects in an image. Instead of scanning an image with sliding windows, a ranked list of innumerable proposal windows in an image was proposed in [27, 28]. Such methods improved the recall rate but failed to correctly localize an object in a given scene. A minimum barrier saliency map was generated using a raster scanning method in [13] which performed better than the geodesic saliency map. However, this method used the entire image for generating saliency map which was exploited to look only around the region where the object has

more probability to be found compared to its previous location.

2.2 Object Tracking

Classical tracking approaches can be categorized as generative and discriminative models. In the generative trackers [19, 20, 29–31], we can represent the targets as a set of basis vectors in a subspace and the trackers search for regions similar to previously tracked targets while the discriminative trackers [14, 32, 33] uses binary classification to differentiate the background with the desired target. It has been mathematically proven that the asymptotic error of a discriminative model is lower than that of a generative model [34].

Tracking by detection approaches [31,33,35] provide a new concept for detection and tracking, however, such approaches suffer from the well-known stability-plasticity dilemma [36], where the drifting of an object in the later frames cannot be rectified since the classifier cannot be trained with stable samples, like that of the first frame. Thus, these approaches barely identify the noisy images with occlusion. Henriques *et al.* [14] harnessed the circulant structure of the samples in the tracking problem with an aid of a kernelized correlation filter (KCF). This method is computationally inexpensive as it transforms the correlation operation in the spatial domain to the frequency domain by exploiting the circulant structure and Parseval's identity and yields only O(nlogn) complexity. It is based on a common principle that the circular matrices (used in an algorithm for kernel ridge regression) that performs correlation in the spatial domain is equal to performing element-wise multiplication in the frequency domain according to the Fourier Transformation principles. However, experiments show that the algorithms using correlation filtering fail to track an obstacle for a larger period of time.

In addition, it is required to manually initialize most of the generative and discriminative trackers with the position of the target in the first frame making them an incomplete automated system. Since manual labeling is required in such trackers, they are not suitable for fully autonomous UAVs. Moreover, such trackers are not fit for long-term tracking as discussed before. Similarly, most of the trackers assign a fixed size bounding box only to track a part of the moving object in

a scene, although the object changes its shape and size throughout a sequence. Such approaches are inappropriate for estimating the shape and size of the obstacle and therefore rectifying the path of UAVs to avoid possible collisions with the obstacle. Some of the tracking by detection methods aim to provide a changing bounding box according to object's shape and size but they perform slower and thus inapt for real-time implementation.

Since most of the previous work required supervised training, correlation filters, though adept at object tracking, seemed inappropriate for real-time object tracking. The minimum output sum of a squared error (MOSSE) filter [37] and its derivatives [38–40] was found to be computationally efficient for real-time object tracking as correlation filter was trained on gray-scale images in this approach. Subsequently, an ample of research has been done in the correlation filter-based tracking. As a result, MOSSE filter was improved in [41] by introducing a kernel-based correlation filter trained on gray-scale images meeting high tracking speed in benchmark datasets [42].

Henriques *et al.* integrated Gaussian and polynomial kernels together with multi-channel HoG features in [14] to achieve higher accuracy and speed than most of the state-of-the-art discriminative and generative trackers. However, their method suffered from its inability to deal with scale variations because of the fixed template size. Li *et al.* [15] tried to solve this problem by combining adaptive templates and HoG features in SAMF tracker. To adapt with the changing size and appearance of the object, Danelljan *et al.* [43] used HoG features in a multiscale correlation filter in DSST tracker. However, all these trackers are prone to mishandling the cases of occlusion and camera instability throughout the sequence.

A part-based tracking algorithm using a correlation filter was proposed in [44] to deal with the occlusion. Only some part of the object is visible during a partial occlusion and part-based tracker exploits this feature to successfully handle the partial occlusion. However, such algorithms fail if the object undergo complete occlusion (becomes invisible) between certain consecutive frames. Correlation between temporal contexts was used in [36] to estimate the translation and scale change of the objects. This approach also used a re-detection scheme by training a fern classifier to handle tracking failures for long-term tracking. However, the proposed approach made their trackers run

slower. Some other detectors [23, 25, 45] and trackers [46, 47] rely on deep learning techniques to improve the accuracy of the trackers and thus require large-scale training database making them slower and unsuitable for real-time applications.

2.3 Fundamental Matrix Estimation

Researchers have proposed several robust outlier detection methodologies to correctly identify and eliminate the outliers in a given noisy data points for the fundamental matrix estimation. One of the most celebrated and widely used robust outlier detection methods is RANSAC [48]. In RANSAC, the fundamental matrix is initially guessed using the minimal set of points. Next, each matching feature points is tested against a hypothesized model to choose an inlier set which gives an error below a set threshold and a new fundamental matrix is determined for successive iteration. By the end of the algorithm, RANSAC would choose an inlier set with the maximum number of data points. However, RANSAC requires a preset threshold which set to a very low value might result in a very few number of inliers with increased iterations depending on the accuracy of the match points. Similarly, if the threshold is set to some higher value with respect to the quality of the match points, the fundamental matrix cannot be accurately estimated. These inconsistencies make RANSAC unsuitable for the scenarios where there are approximately less than 50% inliers. Moreover, the lower the number of inliers, the higher the computational cost of RANSAC due to the increased number of iterations.

One of the RANSAC based methods which improved on computational cost is PROSAC [49]. This improvement is possible as PROSAC keeps track of the quality measures of the point correspondences while determining the fundamental matrix. This technique is absent in RANSAC. In addition, the matches are sorted in their non-increasing order of quality scores and subsets of seven points are progressively chosen unless a minimum number of inliers are found to estimate the fundamental matrix. Since the algorithm starts with the match points having the best quality scores, PROSAC converges in fewer iterations than RANSAC. Though PROSAC is comparatively faster than RANSAC, it undergoes the same limitations as in RANSAC because it requires a preset

threshold to vote the match points as inliers.

The other method MLESAC [50], instead of maximizing the number of inliers, utilizes a median based approach and a different cost function unlike RANSAC. Inliers are assigned a fitness score whereas the outliers are assigned a constant weight. Next, expectation maximization is performed to maximize the maximum likelihood estimates of the normal distribution for the inliers and the uniform distribution for the outliers. The solution that produces the least median of residuals is considered to be the set of inliers. However, MLESAC possesses the same limitations as in RANSAC and also requires the noise parameters as a prior knowledge.

Similarly, MAPSAC [51] tried to improve MLESAC using posterior estimation maximization of the fundamental matrix and the matching point correspondences under Bayesian statistics. MAPSAC uses a new evaluation technique to determine the consistency of the solution. Guided MLESAC [52] replaced the random search in MLESAC or RANSAC with a guided search and significantly reduced the required number of iterations. ARRSAC [53] removed the outliers in real-time thus making the algorithm computationally inexpensive than RANSAC. To do so, ARRSAC claimed a new framework for the real-time robust estimation by utilizing efficient adapting techniques for faster performance over a wide range of the inlier ratios.

In order to solve the problem of automatically determining the preset threshold for separating inliers from outliers, a value proportional to the median of the residuals was used as a threshold in LMedS [54]. LMedS also elegantly depicted the use of median as a robust outlier detection estimator [55, 56] and accurate fundamental matrix estimation.

Recently, the machine learning algorithms, as well as the neural networks, have been extensively used for efficient outlier detection and robust fundamental matrix estimation. A one-class support vector machine-based pre-selection algorithm on matching correspondences obtained using SIFT [57], together with the maximization of a soft decision objective function to refine obtained inliers set, was used to estimate the fundamental matrix in [58]. A robust estimation technique least trimmed squares (LTS) regression was used in [59] to track the outliers which deviate away from majority linear model. This technique was successfully applied to remove the outliers

resulting from the occlusion. Cluster based outlier removing methodology was proposed in [60] that solved outlier detection problems in complex datasets with an ample of clusters and varied densities. An unsupervised boosting approach was tested in [61] to improve the accuracy of the ensemble outlier detection algorithms. Moreover, autoencoder ensembles were used in [62] for unsupervised outlier detection. However, one of the major problems with such approaches is that they are extremely sensitive to noise and often require a lot of datasets to train. Similarly, as the model hyper parameters increase, there is a significant impact on their performance speed which makes them unsuitable for real-time applications in an intelligent vehicle. Therefore, a robust statistical based model is required in order to perform with minimal payload and apt accuracy.

Several robust estimation techniques were discussed in [16], [63], [64]. The motivation to use such robust estimators is that they are simplistic, computationally inexpensive and robust as demonstrated by their influence function and 50% breakdown point in [54].

Chapter 3

Salient Object Detection and KCF Tracking

Object detection resembles much with segmentation of an object from the background. Segmentation usually requires classification of background pixels and foreground pixels in an image. One of the techniques used is the contrast difference between the foreground pixels from the background pixels [22]. Some of the other methods use cues like uniqueness, rarity, region uniformity and spatial compactness. Usually, a distance function between two pixels is computed to distinguish salient object in a scene from the background. Such distance function involves a path cost function minimization which directly depends on the intensity between the end pixels.

3.1 Distance Function

The distance transform [65] is generally used for geometric and morphological characteristics analysis of an object in a digital image [13,22,66,67]. Rosenfeld *et. al.* in [65] has defined distance transform as the function of a distance. Let **P** be the set of all integers (i,j) and the function f maps **P** into some non-negative integers, then f is

- Positive definite, if $f(x,y) = 0$, if and only if x=y and x, y $\in$ **P**
- Symmetric, if $f(x,y) = f(y,x)$ for all x,y $\in$ **P**
- Triangular, if $f(x,z) \leq f(x,y) + f(y,z)$ for all x, y, z $\in$ **P**

The function f that satisfies above three conditions is termed as distance function. This distance function is also known as *pseudo-metric* in some of the object detection and segmentation

algorithms. f is an absolute metric function [67] if it satisfies an additional condition along with the above three conditions.

- Positive, if $f(x,y) > 0$ for all $x \neq y$ and all x, y $\in \mathbf{P}$

Distance function can be computed either using Euclidean disance or with some variants of Djikstra's algorithm. In the first method, Euclidean distance or second norm is calculated between a pixel and the background pixel, whereas, in the second approach distance is calculated using the discrete sets of image pixels with Djikstra's algorithm or its variations. Two basic image segmentation methods [67] - Watershed and Relative Fuzzy Correctedness uses some or the other variants of Djikstra's algorithm for computing distance function. Such segmentation methods use graph theory to represent image pixels, intensities and distance functions.

An image intensity function $f : \mathbf{P} \rightarrow \mathsf{R}^l$ where x$\in \mathbf{P}$ are space elements or *spels* [67]. The image intesity at the pixel x is represented as a l-dimensional vector of attributes like color or gradients. For the segmentation task, an adjacency relation is determined with the *spels* to generate a distance function among all the adjacent *spel* pairs. Adjacent structure generated in this fashion is termed as *scene* [67]. Generally, a rectangular scene is considered such that $\mathbf{P} = \prod_{i=1}^{m} \{a_i\}$ for m-adjacency.

3.1.1 Graph theory equivalence in image processing

One of the common ways to represent a finite set of vertices $\mathbf{V}$ and the edges $\mathbf{E}$ that comprises these vertices is using a graph $\mathbf{G} = \{V, E\}$ where $V \in \mathbf{V}$ and $E \in \mathbf{E}$. Thus, for a distance function $f : \mathbf{P} \rightarrow \mathsf{R}^l$, vertices are *spels* and edges are represented using adjacency relation of an image *scene*.

A set of edges or paths is represented as $\pi =< \pi(0), \pi(1), \ldots, \pi(n) >$ for all $\{\pi(i)\} \in \mathbf{V}$ in a connected graph. Hence, the path that connects two vertices c and d can be represented as $\pi =< \pi(0), \pi(1), \ldots, \pi(n) >$ where $\pi(0) =$c and $\pi(n) =$d. Since there may be multiple paths connecting vertex c with d, we can represent the set of all these paths by $\prod_{cd}$ for our convenience.

For the segmentation task, we require a path with minimum cost between any two given vertices. Thus, a cost function, denoted by $\lambda(\pi) \geq 0$ is usually associated with every path in π. For

instance, $f_\lambda : G \rightarrow [0,\infty]$ is defined as a function that calculates the minimum path between two pixels as $f_\lambda = \min\{\lambda(\pi)\}$.

For f_λ to be considered as a *pseudo metric* distance function, it must satisfy the following criteria

- $\lambda(\pi) = \lambda(<\pi(n),\pi(n-1),\ldots,\pi(0)>)$
- $\lambda(\pi) \leq \lambda(<\pi(0),\pi(1),\ldots,\pi(i)>) + \lambda(<\pi(i),\pi(i+1),\ldots,\pi(n)>)$ for all $0 \leq i \leq n$.

for every possible set of path π.

Since the pseudo metric distance function f_λ is symmetric definite and triangular, it is used as a distance transform for image segmentation [66]. The cost function λ can either be defined either in terms of vertices or edges. It is important to understand that geodesic distance transform is based on edge cost function (weight map) and minimum barrier distance transform is based on vertex cost function (weight map).

3.1.2 Geodesic Distance Transform

Geodesic distance transform is based on geodesic distance function. The minimum cost function between pixels c and d is defined in terms of weight map such that $\lambda : \mathbf{E} \rightarrow (0,\infty)$. Thus, the geodesic distance function, also known as path length function, is defined as

$$f(\lambda) = \sum\big(<\pi(0),\pi(1),\ldots,\pi(n)>\big) = \sum_{i=1}^{n} \lambda\big(\{\pi(i),\pi(i+1)\}\big) \tag{3.1}$$

Geodesic distance transform was successfully used in [22] by representing an image via undirected weighted graph $\mathbf{G} = \{V,E\}$ where $V \in \mathbf{V}$ are image pixels or patches $\{V_p\}$ or a virtual background node $\{V_b\}$ and $E \in \mathbf{E}$ are edges connecting them. Thus, the geodesic saliency distant function [22] is computed as

$$f_\lambda = \min_{V_{p_1}=I, V_{p_2},\ldots,V_{p_n}=V_b} \sum_{i=1}^{n} wt(V_{p_i}, V_{p_{i+1}}) \tag{3.2}$$

such that $(V_{p_i}, V_{p_{i+1}}) \in \mathbf{E}$ and $wt(V_{p_i}, V_{p_{i+1}})$ is the required cost function to minimize. This cost function takes the following three priors into account.

1. Backgroundness cue - Boundary pixels do not contain an object pixel.
2. Connectivity Cue - Background pixels are homogeneous and can therefore be easily connected.
3. Contrast prior - Contrast variation between the background and the non-background image pixel.

However, distance transform calculated using such priors have limitations. For example, if the salient object touches the boundary then the calculated cost becomes higher which impacts object detection. Also, there might be intensity variation on the background pixels and small-weight-accumulation problem [22] may arise. The cost function parameter $\lambda(\pi)$ assumes independent image intensity when $\lambda(\pi)$ is low. So, when $\lambda(\pi)$ is higher, the random noises in the image may affect object detection.

3.2 Minimum Barrier Distance (MBD) Transform

Minimum barrier distance transform considers a vertex cost function such that $\lambda : \mathbf{V} \rightarrow (0, \infty)$. Also, the mimimum barrier distance function or the path length function is defined as

$$f_\lambda = \min_{\pi(I) \in \prod_{a,b}} \left(\lambda^+(\pi) - \lambda^-(\pi) \right) = \min_{\pi(I) \in \prod_{a,b}} \left(\max_{i=0,1,\ldots,n} \{\lambda(\pi(i))\} - \min_{i=0,1,\ldots,n} \{\lambda(\pi(i))\} \right) \quad (3.3)$$

This cost function is also termed as minimum barrier strength [66] and is a pseudo-metric cost function [67]. f_λ is real-valued, bounded and its domain is a subset of pixel intensity values in a given 2D image. One of the advantages of minimum barrier distance transform over the other distance transforms is that the function remains constant unless it finds a stronger barrier i.e. path

that has higher intensity difference than previously chosen seed. It is also important to know that the minimum barrier distance transform requires an algorithm proposed in [66] and cannot be calculated using Djikstra's algorithm. Total order of complexity to calculate minimum barrier distance is O(mnlogn) where m is the total number of distinct pixel intensity values and n is the total number of pixels in the image. It is reported by [66] that it takes 0.5 seconds in average to calculate the minimum barrier distance transform in an image of size 300 x 200.

The other advantage of minimum barrier distance (MBD) transform is that it can be calculated without any prior information of the intensity distribution of the pixels unlike Euclidean based distance function. Thus, it requires no training and performs real-time. Also, minimum barrier distance transform is robust to any noise in the image.

There are several variations of MBD transform like exact MBD [66], approximate MBD [67] and raster scanned MBD [13]. Exact MBD is slower compared to other variations. Approximate MBD transform performance decreases as the image gets noisy and more blur. However, raster scanned MBD performs at 80 frames per second (fps) and is robust to noise. Thus, we have used raster scanned MBD in our thesis for real-time salient object detection.

3.3 Fast Minimum Barrier Distance Transform

Fast Minimum Barrier Distance (MBD) transform is a relatively faster iterative approximation of the exact MBD transform. Fast MBD has a linear complexity and approximates the distance function qualitatively comparable to that calculated by exact MDB.

Fast MBD cacluates the distance function with the help of raster scanning as well as inverse raster scanning. Considering 4-adjacency in a 2D image, the top and the immediate left pixel are considered for calculating the distance transform during raster scanning. Similarly, the bottom and the immediate right pixel are considered during inverse raster scanning.

Raster and inverse raster scanning process consists of two paths - (1) path from background seed to the adjacent pixel π_y and (2) path from adjacent pixel to the chosen pixel $\pi_{\{y,x\}}$. Let the entire path cost function be defined as $\lambda(\pi) = \lambda(\pi_y) + \lambda(\pi_{\{y,x\}}$, then the distance function for a

single scan is given by

$$f_\lambda = \max_{i=0,1,\ldots,n}\{\lambda(\pi(i))\} - \min_{i=0,1,\ldots,n}\{\lambda(\pi(i))\} = \max\{C(y), I(x)\} - \min\{D(y), I(x)\} \tag{3.4}$$

where $C(y)$ and $D(y)$ are the highest and the lowest pixel values on the entire path π that is from a chosen pixel to the background in a single raster scan. Let f_F be the final MBD map and updated on each raster scan by the following rule

$$f_F = \min\left(f_F, f_\lambda\right) \tag{3.5}$$

$C(y)$ and $D(y)$ are updated on each raster and inverse raster scan as f_F gets updated due to change in the path. For n iterations comprising of a raster scan and an inverse raster scan, each iteration terminates when $f_\lambda = f_F$. It is observed that the error between exact MBD and Fast MBD decreases on each iteration and generally becomes negligible after 3 iterations (2 raster scan and 1 inverse raster scan) as demonstrated in [13].

3.4 Saliency Map

Saliency map is a probability map where each pixel in an image is marked based on its probability to be a salient object. The more the pixel is probable to be classified as a salient, the more the intensity. One can expect to see the brighter salient object and darker background in a saliency map. To do so, several unique cues on an input image is considered.

- Color cue - Histogram and statistical distributions such as chi-squared distributions are used to compare the color distribution of a group of pixels with that of the others to generate a saliency map.

- Edge cue - Canny or Sobel edge detectors are used to compare the density of the edges among a group of pixels with that of the others to generate a saliency map.

- Spectral cue - Fast Fourier Transform of the different sizes of chunk of pixels is used to compare the intensities of the group of pixels in the fourier domain to generate a saliency map.

- Superpixel cue - The color or texture features of similar pixels are used to generate a saliency map.

Generally, saliency map of an image can be computed in three distinct ways [68].

1. Saliency sum: In this method, sum of the absolute difference between the intensity of a chosen pixel and all other pixel is computed. The final value determines the saliency of the chosen pixel. Similarly, the salinecy value of all the pixels is calculated and thus a saliency map is generated. Mathematically, if f_i be the intensity of any pixel in an image and f_x be the intensity of the chosen pixel, saliency of the chosen pixel $\mathbf{S}$ is given by $\mathbf{S} = \sum_{i=1, i \neq x}^{n} |f_x - f_i|$ when n is the total number of pixels in an image of which saliency map is to be generated. Thus, the process has order of complexity of O(n).

2. Frame difference sum: In this method, sum of the absolute difference between the intensity of a chosen pixel in current frame and all other pixels in the previous frame of a video sequence is computed. The final value of all the pixels in the current frame is calculated and thus a saliency map is generated for the current frame. Mathematically, if f_i be the intensity of any pixel in an image in the previous frame and f_x be the intensity of the chosen pixel in the current frame, saliency of the pixel $\mathbf{S}$ is given by $\mathbf{S} = \sum_{i=1, i \neq x}^{n} |f_x - f_i|$ when n is the total number of pixels in an image of which saliency map is to be generated. Thus, the process has order of complexity of O(n).

3. Coordinate saliency difference: In this method, saliency sum is used to calculate the saliency map for each pixel. Then, the frame difference sum is performed on the saliency value of each pixel in the current frame as well as the previous frame. Mathematically, if f_i be the intensity of any pixel in an image and f_x be the intensity of the chosen pixel, saliency of

the chosen pixel $\mathbf{S}$ is given by $\mathbf{S} = \sum_{i=1, i \neq x}^{n} |f_x - f_i|$ when n is the total number of pixels in an image of which saliency map is to be generated. Then if $\mathbf{S}'$ be the saliency map of the previous frame calculated similarly as $\mathbf{S}$, the final saliency map of the current frame is calculated as $|\mathbf{S} - \mathbf{S}'|$.

3.5 Fast MBD Saliency Map

Fast MBD saliency map is generated independently on each color channels of an input image. First, each color channel is converted to a grayscale (intensity) image. Next, for each color channel, fast MBD transform is performed as described in equations 3.4 and 3.5. Background pixels are chosen as the seeds with zero level of intensity to start with. Finally, fast MBD transform is calculated for each pixel for each channel.Thereafter, the saliency map of each channel is pixel wise added and normalized. Fig. 3.1 shows the results of applying Fast MBD in an image.

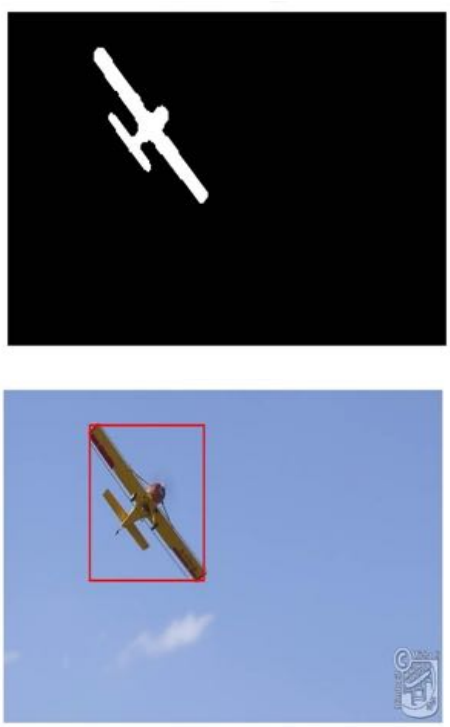


Figure 3.1: A saliency map generated using Fast MBD and an input image with bounding box over the detected object.

3.6 Kernelized Correlation Filter (KCF) Tracking

Kernelized correlation filter (KCF) based tracker [14] is a discriminative tracker that utilizes both positive and negative samples for training its classifier. As discussed earlier, the positive samples come from the salient object in a scene and the negative samples come from the background. Then, a classifier is trained to correctly classify the images patches/pixels for accurate object tracking by detection. However, one of the key problems that a classifier needs to face is the increasing complexity with the increasing number of samples. To handle this, a large number of online trackers try to cut down the negative samples or use similar negative samples which leads to redundancy making the classification task inefficient. To mitigate this problem, KCF operates in a fourier domain (correlation framework) which lowers the complexity to implement both linear as well as non-linear regression using kernels and use plenty of positive and negative samples to train the tracker. This is shown to increase the efficiency and accuracy of the tracker and can perform in real-time.

3.6.1 Regularized Least Square (RLS)

Regularized least square (RLS) is widely used in the field of machine learning to train the model with minimum error rate. In KCF, a classifier is trained using RLS to discriminate between the positive and the negative samples. RLS is usually devised to a ridge regression problem to avoid overfitting of the learned parameters which can then be solved using a system of linear equations.

Let $\mathbf{X}$ and $\mathbf{Y}$ be the sets of random variables. Let the set of training samples be $\mathbf{T} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ such that $x_i \in \mathbf{X}$ and $y_i \in \mathbf{Y}$. Also, let $\mathbf{B}(.)$ be a binary classifier, then RLS is mathematically expressed as

$$RLS = \min_{B} \sum_{i}^{n} \mathbf{L}(y_i, \mathbf{B}(x_i)) + \lambda ||\mathbf{W}||^2 \tag{3.6}$$

where $\mathbf{L}(y_i, \mathbf{B}(x_i))$ is a loss function to minimize, $\mathbf{W}$ is the weight parameters to learn and λ is the regularization parameter to check overfitting. The loss function chosen for our purpose is

$\mathbf{L}(y_i, \mathbf{B}(x_i)) = (y_i - \mathbf{B}(x_i))^2$. In [69], diagonalization technique is used for closed form expression as

$$\mathbf{W} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y} \tag{3.7}$$

where $\mathbf{I}$ is an identity matrix. For complex input features and output labels, transpose is replaced by hermitian as

$$\mathbf{W} = (\mathbf{X}^H\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^H\mathbf{Y} \tag{3.8}$$

where $\mathbf{X}^H = (\mathbf{X}^*)^T$ is the hermitian of $\mathbf{X}$ and $\mathbf{X}^*$ is the complex conjugate of $\mathbf{X}$.

3.6.2 Circulant Matrix

This concept in KCF tracking arises from its use in [41] where a positive sample and a limited negative sample is used to generate large number of positive and negative dense samples by simple cyclic shifts to train the classifier using RLS. This technique helps to solve the limitations of availability of a huge number of data to train the classifier accurately.

Let $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ be a vector of n input samples. The circulant matrix derived from x is given by

$$M(\mathbf{x}) = \begin{bmatrix} x_1 & x_2 & x_3 & \ldots & x_n \\ x_n & x_1 & x_2 & \ldots & x_{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_2 & x_3 & x_4 & \ldots & x_1 \end{bmatrix} \tag{3.9}$$

In order to generate a circulant matrix, let us define a permuation matrix $\mathbf{P}$ as

$$\mathbf{P} = \begin{bmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & 0 \end{bmatrix} \tag{3.10}$$

Now, $M(\mathbf{x})$ can be represented as $\mathbf{P}.\mathbf{x}^T$. Similarly, the i^{th} row can be obtained using $\mathbf{P}^i.\mathbf{x}^T$. Negative i will shift the vector in the reverse direction. Now $\mathbf{X}$ in equation 3.8 can be replaced by $M(\mathbf{x})$ which consists of upper rows created by shifting $\mathbf{x}$ in the positive direction and the lower rows created by shifting $\mathbf{x}$ in the negative direction.

It is important to note that $M(\mathbf{x}).\mathbf{y}$ is equivalent to performing convolution of $\mathbf{x}$ with $\mathbf{y}$. Since convolution in the spatial domain is equivalent to multiplication in the frequency domain, we have

$$M(\mathbf{x}).\mathbf{y} = \mathbf{F}^{-1}(\mathbf{F}^*(\mathbf{x}) \odot \mathbf{F}(\mathbf{y})) \tag{3.11}$$

where $\mathbf{F}(.)$ computes the Fourier transform and $\mathbf{F}^{-1}(.)$ computes the inverse Fourier Transform. $\odot$ represents element-wise multiplicaiton.

Mathematical operations like sum, product or inverse of a circulant matrix preserves the circulant structure which helps to save memory space as we do not need to store the entire matrix but just a vector would suffice. The convolution is replaced by element-wise multiplication which reduces time complexity. Thus, the circulant matrix structure helps to make KCF tracker perform faster compared to the other state-of-the-art trackers.

3.6.3 RLS Implementation with Circulant Matrix

The circulant matrix $M(\mathbf{x})$ is easily computed from the training vector $\mathbf{x}$ in the frequency domain with the help of a discrete fourier transform matrix $\mathbf{V}$ which is a constant and a square matrix. Let $\hat{\mathbf{x}} = \mathbf{F}(\mathbf{x})$ be the fourier transform of the training vector $\mathbf{x}$, then eigen-decomposition of $\mathbf{X}$ can be represented as

$$\mathbf{X} = \mathbf{V}\mathbf{D}\mathbf{V}^H \tag{3.12}$$

where a diagonal matrix $\mathbf{D} = diagonal(\hat{\mathbf{x}})$ is formed using $\hat{\boldsymbol{\lambda}}$ as

$$\mathbf{X}\hat{\boldsymbol{\lambda}} = \mathbf{Z}\hat{\boldsymbol{\lambda}} \tag{3.13}$$

$\mathbf{D}$ has eigen vector $\hat{\boldsymbol{\lambda}}$ as its diagonal elements. Eigenvalue decomposition of $\mathbf{X}$ results in eigenvalues $\mathbf{Z}$ as shown in equation 3.13. This process is termed as diagonalization. Since $\mathbf{V}$ is independent of $\mathbf{X}$, we can compute $\mathbf{X}^H\mathbf{X}$ from equation 3.12 as

$$\mathbf{X}^H\mathbf{X} = (\mathbf{V}\mathbf{D}\mathbf{V}^H)^H(\mathbf{V}\mathbf{D}\mathbf{V}^H) = \mathbf{V}\mathbf{D}^*\mathbf{V}^H\mathbf{V}\mathbf{D}\mathbf{V}^H \tag{3.14}$$

Since $\mathbf{V}^H\mathbf{V} = \mathbf{I}$, equation 3.14 becomes

$$\mathbf{X}^H\mathbf{X} = \mathbf{V}\mathbf{D}^*\mathbf{D}\mathbf{V}^H \tag{3.15}$$

which can be expressed as

$$\mathbf{X}^H\mathbf{X} = \mathbf{V}\{diagonal(\hat{\mathbf{x}})^* diagonal(\hat{\mathbf{x}})\}\mathbf{V}^H \tag{3.16}$$

which further reduces to

$$\mathbf{X}^H\mathbf{X} = \mathbf{V}\{diagonal(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}})\}\mathbf{V}^H \tag{3.17}$$

Note that $(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}})$ is an autocorrelation of $\mathbf{x}$ which is equivalent to power spectrum of the signal in the frequency domain $\hat{\mathbf{W}}$. Now, the weight parameters $\mathbf{W}$ in equation 3.8 can be expressed in the frequency domain using equation 3.17 as

$$\hat{\mathbf{W}} = diagonal\{(\hat{\mathbf{x}}^* \odot \hat{\mathbf{y}})/(\hat{\mathbf{x}}^* \odot \hat{\mathbf{x}} + \lambda)\} \tag{3.18}$$

Thus obtained $\hat{\mathbf{W}}$ is the regularized correlation filter coefficients in the frequency domain [37]. The correlation filter coefficients can be converted to the spatial domain using the inverse fourier transform. The computation complexity of performing discrete fourier transform is O(nlogn) and all the other mathematical operations like division and multiplications are computed element-wise with O(n). Thus, it is much simpler than computing RLS which has a complexity of $O(n^3)$ and involves matrix product and inverison.

3.7 Non-linear Regression

For tracking by detection purposes, we need to train a non-linear regression model for the uninterrupted continuous output rather than a binary classifier. Thus, a kernel based non-linear classifier is discussed in this section in detail. Kernel matrix is used to derive a solution for a non-linear regression model using the kernel matrix representation [70] and regularized least squares with kernels [41] without affecting the time complexity.

3.7.1 Kernel Matrix

Input vector $\mathbf{x}$ can be easily represented in higher dimensional feature space $\mathbf{H}(\mathbf{x})$ using a kernel k and applying the kernel trick as described in [70].

The simplest kernel k that measures the similarity between two input vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ can be represented using their inner products as

$$\langle \mathbf{x}_1 \mathbf{x}_2 \rangle = \sum_{i=1, j=c}^{n} x_i . x_j \tag{3.19}$$

where c is any constant. Such inner products can be computed by a multiplication of the cosine of the angles between two vectors in higher dimensional feature space $mathbfH(x)$ if the input samples are normalized. Thus equation 3.19 can be represented in the higher dimension space as

$$k(\mathbf{x}_1, \mathbf{x}_2) = \langle \mathbf{x}_1, \mathbf{x}_2 \rangle = \langle \mathbf{H}(\mathbf{x_1}), \mathbf{H}(\mathbf{x_2}) \rangle \tag{3.20}$$

Now, the constrained optimization problem can be solved using Lagrange's multiplier α and the loss function in equation 3.6 for KCF becomes

$$\mathbf{L}(\mathbf{B}, c, \alpha) = \frac{1}{2}||\mathbf{B}||^2 - \sum_{i}^{n} \alpha_i (y_i(\langle x_i, \mathbf{B} \rangle + c) - 1) \tag{3.21}$$

where $\mathbf{B}$ is the regularized least square binary classifier (primal variable in the Lagrangian equation). c is a constant and another primal variable in the set of linear equations. In order to obtain an optimized kernel distance function, equation 3.21 needs to maximized in terms of α which is a dual variable and minimized in terms of primal variables $\mathbf{B}$ and c. This is termed as a dual optimization problem in KCF tracking.

Differentiating equation 3.21 with respect to $\mathbf{B}$, α and c and representing the results in terms of higher dimensional space $\mathbf{H}(\mathbf{x})$, we obtain

$$\mathbf{B} = \sum_{i} \alpha_i \mathbf{H}(x_i) \tag{3.22}$$

$$f(\mathbf{y}) = \mathbf{B}^T \mathbf{y} = \sum_{i}^{n} \alpha_i k(\mathbf{y}, x_i) \tag{3.23}$$

where $f(\mathbf{y})$ is a new kernelized classifier. In non-linear regression, as the number of training samples increases, the complexity of our kernelized classifier increases too. This problem is solved using kernel trick which is described in the following section.

3.7.2 Kernelized RLS

Let us define a kernel matrix $\mathbf{K}$ which contains all the possible inner products between the input vectors $\mathbf{K} = k(x_i, x_j)$. Now, the dual variable α also known as the kernel weight can be defined in terms of kernel matrix $\mathbf{K}$, regularization parameter λ and labels $\mathbf{y}$ as

$$\alpha = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y} \tag{3.24}$$

where $\mathbf{K}$ and $\mathbf{I}$ has the same dimension. Similar to the exploitation of circulant marix structure in RLS in the fourier domain, we want the kernel matrix to be circulant. It is observed in [14] that the kernel matrix should satisfy the following condition for the circulant properties to hold.

$$k(x_1, x_2) = k(\mathbf{P}x_1, \mathbf{P}x_2) \tag{3.25}$$

where $\mathbf{P}$ is the permutation matrix defined in the previous section. Hence, the kernel matrix can be evaluated in the fourier domain using a kernel correlation defined as

$$k_i^{x_1,x_2} = k(x_2, \mathbf{P}^{i-1}x_1) \tag{3.26}$$

where $k(.,.)$ is simply the inner products between $\mathbf{P}^{i-1}x_1$ and x_2. Equation 3.26 can be expressed in terms of higher dimensional feature space $\mathbf{H}(\mathbf{x})$ as

$$k_i^{x_1,x_2} = \mathbf{H}^T(x_2)\mathbf{H}(\mathbf{P}^{i-1}x_1) \tag{3.27}$$

Thus, equation 3.24 can now be represented using the fourier transformation as

$$\alpha = \mathbf{F}^{-1}(\hat{\mathbf{y}}(\hat{k^{x,x}} + \lambda)^{-1}) \tag{3.28}$$

where $\mathbf{F}(\mathbf{y}) = \hat{\mathbf{y}}$ is the fourier transform of output labels and $\mathbf{F}(\mathbf{k^{xx}}) = \hat{k^{xx}}$ is the fourier transform of the kernel auto-correlation vector. Equation 3.28 is the kernelized RLS weight vector to train the classifier. Traditional kernel methods require O(n^4) operations whereas equation 3.28 is the closed form solution with only O(n^2logn) complexity.

3.7.3 Kernelized RLS Filter

Once the kernelized RLS wight vector trains the classifier $f(\mathbf{y})$ in equation 3.23 with the training labels $\mathbf{y}$, we again reap the benefits of circulant matrix structure.

Let RLS binary classifier defined in 3.22 contain kernel matrix $\mathbf{K}$. Then, a new classifier kernel

matrix $\mathbf{K^y}$ consists of cyclic shifts of base input training sample vector $\mathbf{x}$ and the base output testing sample vector $\mathbf{y}$ such that

$$k_{i,j}^{x,y} = k(\mathbf{P}^{j-1}\mathbf{y}, \mathbf{P}^{i-1}\mathbf{x}) \tag{3.29}$$

where $k_{i,j}^{x,y}$ is the kernel correlation between $\mathbf{x}$ and $\mathbf{y}$. Let $k^{x,y}$ be the kernel correlation vector consisting of all the elements $k_{i,j}^{x,y}$. Now, the new classifier kernel matrix can be expressed as

$$\mathbf{K^y} = \mathbf{M}(k^{x,y}) \tag{3.30}$$

The kernel classifier $f(\mathbf{y})$ (equation 3.23) can now be defined as

$$f(\mathbf{y}) = (\mathbf{K^y})^T \alpha \tag{3.31}$$

Using equation 3.30 and the fourier domain representation, our new kernel classifier or KRLS filter becomes

$$f(\mathbf{y}) = \mathbf{F}^{-1}(\hat{k}^{x,y} \odot \hat{\alpha}) \tag{3.32}$$

where $\mathbf{F}^{-1}$ is the inverse fourier transform, $\mathbf{F}(k^{x,y}) = \hat{k}^{x,y}$ and $\mathbf{F}(\alpha) = \hat{\alpha}$.

3.8 Kernel Correlation Association

Since computation of kernels involves two independent vectors and their relative shifts, it can be slower during non-linear regression. To mitigate this problem, we can represent the kernel as a circulant matrix provided it satisfies equation 3.25. One of the kernel that satisfies this property is a radial basis function kernels which is described in this section.

3.8.1 Inner Product Kernel Function

A kernel $k(\mathbf{x_1}, \mathbf{x_2})$ is said to be a inner product kernel function represented as $Q(\mathbf{x_1}^T \mathbf{x_2}) = k(\mathbf{x_1}, \mathbf{x_2})$ if it satisfies equation 3.26 as

$$k_i^{(\mathbf{x_1}, \mathbf{x_2})} = k(\mathbf{x_2}, \mathbf{P}^{i-1}\mathbf{x_1}) = Q(\mathbf{x_2}^T \mathbf{P}^{i-1}\mathbf{x} - \mathbf{1}) \tag{3.33}$$

Now, if $\mathbf{x_1}$ and $\mathbf{x_2}$ is operated element wise via inner product kernel function $Q(.)$, kernel correlation vector can be expressed (from equation 3.30) as

$$k^{(\mathbf{x_1}, \mathbf{x_2})} = Q(\mathbf{M}(\mathbf{x_1})\mathbf{x_2}) \tag{3.34}$$

Applying the kernel trick and the diagonalization technique plus evaluating the kernel correlation vector in the frequency domain, we get

$$k^{\mathbf{x_1}, \mathbf{x_2}} = Q(\mathbf{F}^{-1}(\mathbf{M}(\mathbf{x_1})\mathbf{x_2})) = Q(\mathbf{F}^{-1}(\hat{\mathbf{x_1}}^* \odot \hat{\mathbf{x_2}})) \tag{3.35}$$

3.8.2 Radial Basis Kernel Function

For the sample vectors $\mathbf{x_1}$ and $\mathbf{x_2}$, radial basis kernel function $\mathbf{Rb}(.)$ that follows equation 3.26 is defined as

$$k_i^{\mathbf{x_1}, \mathbf{x_2}} = k(\mathbf{x_2}, \mathbf{P}^{i-1}\mathbf{x_1}) = \mathbf{Rb}(||\mathbf{x_2} - \mathbf{P}^{i-1}\mathbf{x_1}||^2) = \mathbf{Rb}(||\mathbf{x_1}||^2 + ||\mathbf{x_2}||^2 - 2\mathbf{x_2}^T \mathbf{P}^{i-1}\mathbf{x_1}) \tag{3.36}$$

Now, if $\mathbf{x_1}$ and $\mathbf{x_2}$ is operated element wise via radial basis kernel function $Q(.)$, kernel correlation vector can be expressed (from equation 3.30) as

$$k^{(\mathbf{x_1}, \mathbf{x_2})} = \mathbf{Rb}(\mathbf{M}(\mathbf{x_1})\mathbf{x_2}) \tag{3.37}$$

Applying the kernel trick and the diagonalization technique plus evaluating the kernel correla-

tion vector in the frequency domain, we get

$$k^{\mathbf{x_1},\mathbf{x_2}} = \mathbf{Rb}(\mathbf{F}^{-1}(\mathbf{M}(\mathbf{x_1})\mathbf{x_2})) = \mathbf{Rb}(||\mathbf{x_1}||^2 + ||\mathbf{x_2}||^2 - 2\mathbf{F}^{-1}(\hat{\mathbf{x_1}}^* \odot \hat{\mathbf{x_2}})) \tag{3.38}$$

where the notations hold the same meaning as defined in the earlier sections.

One of the widely used radial basis kernel function is Gaussian kernel $k(\mathbf{x_1},\mathbf{x_2}) = exp(-\frac{1}{\sigma^2})||\mathbf{x_1} - \mathbf{x_2}||^2)$, where σ is the standard deviation. The Gaussian kernel can be evaluated in the frequency domain as

$$k^{\mathbf{x_1},\mathbf{x_2}} = exp\bigg(-\frac{1}{\sigma^2}\big(||\mathbf{x_1}||^2 + ||\mathbf{x_2}||^2 - 2\mathbf{F}^{-1}(\hat{\mathbf{x_1}}^* \odot \hat{\mathbf{x_2}})\big)\bigg) \tag{3.39}$$

which has O(nlogn) complexity.

3.9 Cosine Window Transformation

Digital images are highly non-periodic whereas correlation in the Fourier domain requires smooth periodic transition between the left and the right edges or the top and the bottom edges of the image. To solve this non-periodicity problem in any given raw image, cosine window transformation was suggested in [37].

Cosine window transformation $x^c_{i,j}$in any given pixel $x_{i,j}$ in an input image is given by

$$x^c_{i,j} = \left(x_{i,j} - \frac{1}{2}\right)\sin\left(\frac{\pi i}{n}\right)\sin\left(\frac{\pi j}{n}\right) \tag{3.40}$$

Now for the multiple input channels, one of the advantages of operating in the Fourier domain is that we can easily process the kernel function for each of the multiple image channels in the Fourier domain independently and later add the results to produce the final multiple channel kernel function.

Chapter 4

Real-time Obstacle Tracking by Detection, Experiments and Results

In this chapter, the details of the proposed strategy for a fast and robust object tracking by detection is discussed. We will discuss in detail about a fast, reliable and accurate object localization and tracking approach for the autonomous navigation of the flying UAVs by integrating the techniques for salient object detection [13] with the kernelized correlation filter [14]. We will also explain the importance of salient object detection scheme as it is important for the autonomous UAVs to first detect the salient object in the scene to avoid obstacles or collision. Since salient object detection is not able to properly determine the shape, post-processing technique needs to applied. Further, refinement process is elaborated in this chapter to threshold the salient object to precisely pin point the object and feed to an integrated tracker for the tracking purpose. Experiments and results are then analytically as well as qualitatively analysed to compare the state-of-the-art competing tracker with our approach. A flowchart of the proposed technique is shown in Fig. 4.1.

4.1 Proposed Approach

First, a saliency map **S** of an entire image is generated to segment the salient object out from the background and auto-initialize the tracker with the current location of the salient object for tracking in the consecutive frame. In this process, we generate a saliency map, post-process the generated saliency map using the proposed post-processing technique to segment the salient object and feed the location of the salient object to initialize the tracker. Next, the filter starts training itself on the salient object on each frame while tracking of the object runs simultaneously until a low peak

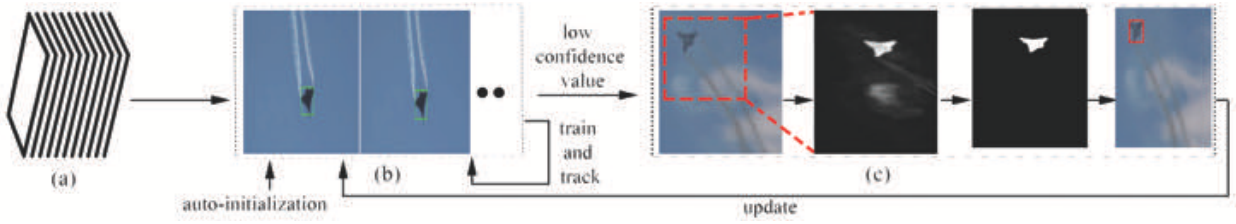


Figure 4.1: Illustration of our approach: (a) input frames, (b) correlation filter and tracking, (c) (from left to right) frame to redetect the salient object, saliency map generated, thresholded binary image using our post processing technique and new bounding box detected on the object.

of filter response (confidence value) is observed. Confidence value measures the resemblance of the object in the consecutive frame compared to the previous frame where the object was being tracked. Once such low confidence value is observed for the tracker, our proposed adaptive detection approach is applied to re-detect the object. The re-detection scheme is important because it helps to increase the confidence value of the tracker to track in the later frames by re-estimating the accurate position and the size of the object being tracked. To do so, we determine an adaptive search region **R** based on the confidence value; the area of **R** is progressively increased to re-detect the object being tracked as the confidence value drops lower. This re-detection scheme is much similar to the detection process as performed in the first frame. A slight variation is that we generate **S** only for search region **R** instead of an entire frame and update the tracker accordingly for a smooth training of the KCF filter throughout the tracking process.

4.1.1 Automatic Salient Object Detection

This section best explains how our tracker is auto-initialized in the first frame of any given sequence. Most of the trackers need to be provided with a ground truth of the initial frame to let know the whereabouts of the concerned object and further to continue tracking the object in the consecutive frames, thus requiring manual initialization. However, our tracker independently initializes from the very first frame and continues smooth tracking throughout the sequence.

To initialize our tracker in the first frame, salient object detection algorithm is run on an entire image as we are unaware where the salient object to track is initially located. Inspired by the

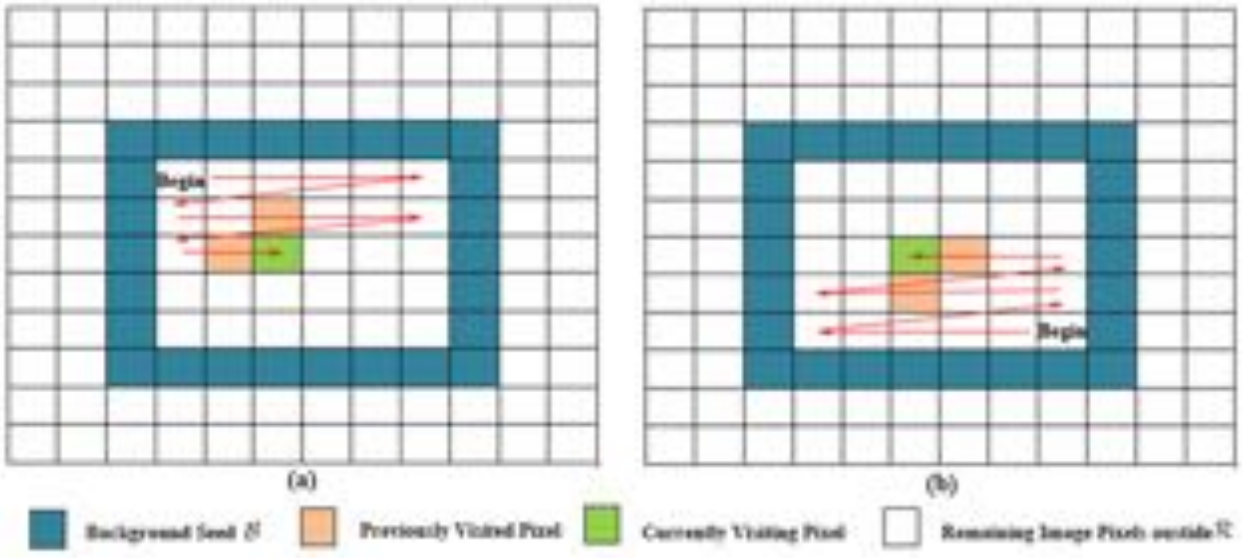


Figure 4.2: Updation method for **R** (a) Raster scanning. (b) Inverse raster scanning.

Minimum Barrier Distance (MBD) transform, we formulate the salient object detection as finding the shortest distance from pixel $p_{i,j}$ to the set of pixels **B** along the image boundary. For simplicity, we consider a single-channel digital image **I** and 4-adjacent neighboring pixels to calculate the distance of $p_{i,j}$ to **B**. For instance, the neighbors of $p_{i,j}$ are $p_{i-1,j}$, $p_{i+1,j}$, $p_{i,j-1}$ and $p_{i,j+1}$. A path $\mathbf{P} = <\mathbf{P}(0),\mathbf{P}(1),...,\mathbf{P}(k)>$ on **I** is a sequence of pixels where consecutive pairs of pixels are adjacent. Given a distance cost function **D**, the distance map **M** for each pixel in **I** is best defined as

$$\mathbf{M}(p_{i,j}) = \min_{\delta \in \Pi_{\mathbf{B},p_{i,j}}} \mathbf{D}(\delta) \tag{4.1}$$

where $\Pi_{\mathbf{B},p_{i,j}}$ denotes a set of all possible paths connecting elements in **B** with $p_{i,j}$. In [22], the geodesic distance is used for **D**, but in [13, 71] a formula more robust to noise is proposed which is found to be the most appropriate approach four our purpose. The cost function is given as

$$\mathbf{C}_\mathbf{I}(\mathbf{P}) = \max_{i=0}^{n} \mathbf{I}(\mathbf{P}(i)) - \min_{i=0}^{n} \mathbf{I}(\mathbf{P}(i)) \tag{4.2}$$

Each pixel $p_{i,j}$ is visited during the raster scan as well as the inverse raster scan (Fig. 4.2). During the raster scan, we update pixel values of two adjacent neighbors $p_{i,j-1}$ and $p_{i-1,j}$, whereas

Figure 4.3: From left to right: Original frame, generated saliency map, binary image and the object boundary using our object detection technique.

in the inverse raster scan, the values of $p_{i,j+1}$ and $p_{i+1,j}$ are updated. The updates take place by

$$\mathbf{M}\left(p_{i,j}\right) = min\{\mathbf{D}\left(x\right), \mathbf{C_I}\left(\mathbf{Z}\left(p'_{i,j}\right).\left\langle p'_{i,j}, p_{i,j}\right\rangle\right)\} \tag{4.3}$$

where $\mathbf{Z}\left(p'_{i,j}\right).\left\langle p'_{i,j}, p_{i,j}\right\rangle$ is an appended path from $p'_{i,j}$ to $p_{i,j}$ with currently assigned path for $p'_{i,j}$, *i.e.*, $\mathbf{Z}\left(p'_{i,j}\right)$.

Let us denote $\mathbf{Z}\left(p'_{i,j}\right).\left\langle p'_{i,j}, p_{i,j}\right\rangle$ with $\mathbf{Z}_{i,j}$.

$$\mathbf{C_I}\left(\mathbf{Z}_{i,j}\right) = max\{\mathbf{H}\left(p'_{i,j}\right), \mathbf{I}\left(p_{i,j}\right)\} - min\{\mathbf{L}\left(p'_{i,j}\right), \mathbf{I}\left(p_{i,j}\right)\} \tag{4.4}$$

where $\mathbf{H}\left(p'_{i,j}\right)$ and $\mathbf{L}\left(p'_{i,j}\right)$ are the highest and the lowest pixel values on $\mathbf{Z}\left(p'_{i,j}\right)$, respectively. Each iteration of the raster/inverse raster scan updates $\mathbf{H}$ and $\mathbf{L}$ if the path assignment is found to be changed. The final outcome is a saliency map $\mathbf{S}$ for the entire image, where a certain post processing needs to be done to obtain a binary image for the final object detection and successfully initialize the tracker. An example of a saliency map is shown in Fig. 4.3

4.1.2 Post Processing

Post processing helps to enhance the quality of the saliency map $\mathbf{S}$. For successful tracking, it is vital to obtain a binary image from which we can segment a foreground salient object from its background. It is inefficient to apply a direct threshold to $\mathbf{S}$ because of the presence of different levels of noise content, relative size of objects and background, illuminance and reflectance in

different frames. Such wide variations necessitate to choose an adaptive threshold technique that successfully handles these subtle entities. Hence, in our approach, we rely on inter-class variance maximization [72] to find an optimal value for the global threshold and obtain a binary image.

Let $\mathbf{H}$ be the histogram of $\mathbf{S}$ that contains pixels of intensity levels $\mathbf{L} \in [0, l-1]$ and η_i be the numbers of pixels of intensity i where $i \in \mathbf{L}$. Then,

$$\mathbf{H} = \sum_{i=0}^{l-1} \eta_i \tag{4.5}$$

Let $\mathbf{H}_n$ be the normalized histogram, for every threshold value t, $t \in \mathbf{L}$, we define two classes $\mathbf{C}_1 \in \mathbf{H}_{n_i}, i \in [0,t]$ and $\mathbf{C}_2 \in \mathbf{H}_{n_i}, i \in [t+1, l-1]$, such that

$$\mathbf{P_1} = P(\mathbf{C}_1) = \sum_{i=0}^{t} \mathbf{H}_{n_i}, \tag{4.6}$$

$$\mathbf{P_2} = P(\mathbf{C}_2) = \sum_{i=t+1}^{l-1} \mathbf{H}_{n_i} = 1 - \mathbf{P_1} \tag{4.7}$$

The mean intensity of pixels in $\mathbf{C}_1$ is given by

$$m_1 = \sum_{i=0}^{t} i.P(i/\mathbf{C}_1) = \sum_{i=0}^{t} i.\frac{P(\mathbf{C}_1/i).P(i)}{P(\mathbf{C}_i)} = \frac{1}{\mathbf{P_1}} \sum_{i=0}^{t} i.\mathbf{H}_{n_i} \tag{4.8}$$

where $\mathbf{P}(\mathbf{C}_1/i) = 1, \mathbf{P}(i) = \mathbf{H}_{n_i}$. Similarly mean intensity of pixels in $\mathbf{C}_2$ is given by

$$m_2 = \frac{1}{\mathbf{P_2}} \sum_{i=t+1}^{l-1} i.\mathbf{H}_{n_i} \tag{4.9}$$

Let m_g represent mean global intensity and m_t represent mean intensity upto t level. Then, the inter-class variance is derived as in [72].

$$\sigma_b^2 = \mathbf{P_1}.(m_1 - m_g)^2 + \mathbf{P_2}.(m_2 - m_g)^2 = \frac{m_g.\mathbf{P_1} - m_t}{\mathbf{P_1}.(1 - \mathbf{P_1})} \tag{4.10}$$

For each $t \in \mathbf{L}$, we calculate $\sigma_b^2(t)$ and optimal threshold t_{opt} for $\mathbf{S}$ is given by

$$\sigma_b^2(t_{opt}) = \max_{0<t<l-1} \sigma_b^2(t) \tag{4.11}$$

By applying this method, we successfully obtain a binary image where the salient object is distinguishably highlighted from the background and the noisy image background is eliminated. Now, the tracker is able to correctly locate the required salient object in an analyzed scene as shown in Fig. 4.3 and begin tracking in the consecutive frames. The auto-initialization algorithm is given in Algorithm 1.

Algorithm 1 Auto-initialization

Input: first frame (f)
Output: co-ordinates (x,y,width,height) of salient object in f

1: Generate saliency map $\mathbf{S}$ using equation (1)-(4)
2: Compute normalized histogram $\mathbf{H}_n$ of $\mathbf{S}$
3: Divide into two groups with probabilities $\mathbf{P_1}$ and $\mathbf{P_2}$ as in equations (6), (7)
4: **for** threshold level t=1 to maximum intensity in $\mathbf{S}$ **do**
5: Compute global intensity $m_g = \mathbf{P_1} m_1 + \mathbf{P_2} m_2$ using equations (8), (9)
6: Compute mean intensity upto level t $m_t = \sum_{i=0}^{t} i\mathbf{P_i}$
7: Compute σ_b^2 using equation (10)
8: **end for**
9: Derive optimal threshold using equation (11)
10: Draw bounding box covering maximum area of the salient object in optimally thresholded binary image
11: **return** salient object coordinates

4.1.3 Object Tracking

This section aims to introduce the correlation filters briefly as well as the tracking mechanism for further understanding of the proposed technique.

Correlation filter-based trackers use filters trained on previously tracked objects and their immediately surrounding background for tracking the object. Usually, a small test window is selected on the object that needs to be tracked [37]. Thereafter, tracking of an object and training of the filter is performed simultaneously in the consecutive frames. The filter is correlated over a search

window in an adjacent frame to obtain a correlation map. The peak value in the correlation map helps to determine the position of the object being tracked in this frame. However, computational efficiency can be significantly increased by performing the correlation in the frequency domain. To perform a correlation in the frequency domain, Fast Fourier Transform (FFT) of the filter is element-wise multiplied with a two-dimensional FFT of the input image. This is possible because an element-wise multiplication in the frequency domain is equivalent to the correlation in the spatial domain.

The correlation G between the FFT of $\mathbf{R}$ (denoted by I) where the object needs to be tracked and the FFT of the filter (denoted by H) is given by

$$G = I \odot H^* \tag{4.12}$$

where $\odot$ is element-wise multiplication and * denotes complex conjugate. We can use inverse Fourier transform to transform the correlation output back to the spatial domain.

It can be derived from the properties of the circulant matrices that any $m \times 1$ vector $\mathbf{v}$ can be diagonalized using Discrete Fourier Transform (DFT) [73]. Hence,

$$\mathbf{C} = F.diag(\hat{\mathbf{v}}).F^H \tag{4.13}$$

where $\hat{\mathbf{v}}$ denotes the DFT of $\mathbf{v}$, $\hat{\mathbf{v}} = \mathbf{F}(\mathbf{v})$, $\mathbf{F} = \sqrt{n}F\mathbf{v}$ and F is a constant matrix independent of $\mathbf{v}$.

Motivated by [14], we use ridge regression along with kernelized correlation filter to implement the tracking. Our main aim is to find a function $f(\beta) = \alpha^T \beta$ to minimize the squared error between the training samples $x_i \in \mathbf{X}$ and the output $y_i \in \mathbf{Y}$. If we transform our linear object of interest $\mathbf{O}$ to a non-linear feature space $\phi(\mathbf{O})$ and apply a kernel trick [74] on it, we have,

$$\alpha = \sum_i \delta_i.\phi(\mathbf{O}) \tag{4.14}$$

Here, we need to optimize parameter δ_i for the least squared error. Representing $\phi(\mathbf{O})$ in terms of

dot products,

$$\phi^T(\mathbf{O})\phi(\mathbf{O}') = \mathbf{K}(\mathbf{O},\mathbf{O}') \tag{4.15}$$

where $'$ denotes the cyclic shifts and $\mathbf{K}$ is a Gaussian kernel function. Therefore,

$$f(\beta) = \alpha^T\beta = \sum_{i=1}^{n} \delta_i \mathbf{K}(\beta, \mathbf{O}_i) \tag{4.16}$$

The solution to equation (16) as derived in [75] is

$$\sigma = (\mathbf{K} + \lambda I)^{-1}\mathbf{Y} \tag{4.17}$$

where σ is the vector of coefficients δ_i representing the solution in the dual space. In [14], given a condition $\mathbf{K}(\mathbf{O},\mathbf{O}') = \mathbf{K}(\mathbf{MO},\mathbf{MO}')$ where $\mathbf{M}$ is a permutation matrix, $\mathbf{K}$ is shown to be circulant and can be diagonalized for faster computation.

Since we need to find $f(\beta)$ on multiple image locations that can be arranged in a circulant matrix, let us define $\mathbf{K}^\beta$ as the kernel matrix between every training samples and candidate patches that are cyclic shifts of $\mathbf{O}$ and β. We have,

$$\mathbf{K}^\beta = \mathbf{C}(k^{\mathbf{O}\beta}) \tag{4.18}$$

where $\mathbf{C}(k^{\mathbf{O}\beta})$ is the kernel correlation of $\mathbf{O}$ and β. Now, the regression function for all candidate patches is given by

$$f(\beta) = (\mathbf{K}^\beta)^T\delta \tag{4.19}$$

Diagonalizing for efficient computation,

$$f(\hat{\beta}) = \hat{k}^{\mathbf{O}\beta} \odot \delta \tag{4.20}$$

where $f(\hat{\beta})$ represents the DFT of $f(\beta)$ and $\odot$ denotes element-wise operation.

The confidence values of the filter are monitored to determine the adaptive search region for the

salient object detection in our tracking approach. A tracker is less confident about the object being tracked when its confidence value (measures the similarity of object in two consecutive frames while tracking) drops below a certain defined threshold. Hence, we acquire a re-detection scheme to mitigate such off-guarded tracker. Therefore, we avidly monitor the confidence values throughout the tracking process and adjust our detection region adaptively as the tracker's confidence value surge lower. If the confidence value drops too low, we set our search region **R** to an entire frame. Thus, such settlements help precise tracking of the obstacle that undergo variations in shape, size, rotation, camera instability and illumination.

4.1.4 Refinement

During the course of tracking, tracker may lose the track of the object being tracked due to several inconsistencies such as abrupt motion dynamics, undefined perturbations, camera instability, projection/separation of similar or disparate foreign objects into the scene or nearby the object being tracked. Most of the trackers are unable to handle such complicacies efficiently. Thus, to effectively monitor such circumstances, we have implemented a refinement approach in our tracker.

It is observed that the peak of filter response (confidence value) drops below certain threshold when our tracker is unable to correctly track the object in a subsequent frame. Therefore, a proper refining approach to correct the tracker in such situations was found to be necessary. One of the approaches that could be considered is to undergo salient object detection approach (as described in subsection A) on an entire image to relocate the object and update the tracker with necessary correction in the given frame. However, such an attempt is computationally expensive when applied on each individual frame or on an entire frame iteratively while tracking an object. Therefore, in our approach we adaptively generate region **R** depending on the confidence value to run the detection algorithm. Detailed analysis on the measures for selecting a certain peak of filter response value for adaptive refinement process is presented in Section 4.2.

One of the examples that best demonstrates the refinement process used in our approach is shown in Fig. 4.4. When the object being tracked considerably changes in shape, size, illumination

Figure 4.4: Refinement process: (From left to right) Selected region around the prime object based on the peak value for the refinement process (brown bounding box), the chosen area for refinement (zoomed in for better view), saliency map generated for the chosen area, post-processed binary image, prime salient object re-detected (notice the change in the size bounding box before and after the refinement process) to update the tracker.

or reflectance the peak of response drops lower as the current frame significantly differ from the previous frames. Thus, our refinement approach comes into action. During the refining process, our approach successfully selects the region **R** around the prime object being tracked and applies salient object detection algorithm (as described in subsection 4.1.1) only in this selected region, thus making our approach computationally efficient. Further, the generated saliency map is post-processed (as described in subsection 4.1.2) to relocate the prime object. The correct coordinates are further updated to the running tracker for successful tracking as shown in Fig. 4.1. The tracking algorithm is given in Algorithm 2.

Algorithm 2 Real-time Salient Object Tracking

Input: a sequence of images
for each frame f **do**
  if (first frame) **then**
    Auto-initialize using Algorithm 1
    continue
  end if
  observe $f(\hat{\beta})$ (equation (20))
  if ($f(\hat{\beta}) <$ set_confidence_value) **then**
    Adaptively define **R** around last known coordinates of the object being tracked
    Generate saliency map **S** of **R** using equation (1)-(4)
    Postprocess **S** using steps 2-10 of Algorithm 1
  end if
  Update the tracker with new coordinates
end for

4.2 Experiments

We have implemented the proposed algorithm using C++ and OpenCV v.3.0. All the experiments were performed on a Intel(R) Xeon(R) W3530 PC with 2.80 GHz processor and 4 GB RAM. The competing tracker's codes were also experimented on the same PC and downloaded from the respective author's web page.

4.2.1 Dataset

Our approach is tested on 25 challenging video sequences where the object is subjected to variations of scale, partial occlusion, axial and planar rotation, illumination variation and camera instability. The experimented sequences are namely; airplane_001 (200 frames), airplane_004 (200 frames), airplane_005 (200 frames), airplane_006 (200 frames), airplane_007 (200 frames), airplane_ 011 (300 frames), airplane_012 (300 frames), airplane_013 (300 frames), airplane_015 (300 frames), airplane_016 (300 frames), big_2 (382 frames) from [76], Dog (127 frames), planestv_1 (223 frames), planestv_2 (200 frames), planestv_3 (300 frames), planestv_4 (350 frames), planestv_5 (200 frames), planestv_6 (230 frames), planestv_7 (250 frames), planestv_8 (260 frames), planestv_9 (410 frames), Skater (160 frames), youtube_1 (216 frames), youtube_2 (475 frames) together with youtube_3 (301 frames) from publicly available videos on the web to test our algorithm on several types of objects. Dog and Skater datasets have been chosen to observe our tracker's performance on general objects other than aerial flying units. We manually annotated the ground truth for each of the chosen datasets to perform quantitative analysis as described in the latter sections. Datasets along with their annotated ground truth is made available on the author's web page[1].

4.2.2 Determination of Suitable Peak of Filter Response

It is essential to find a suitable value for a peak of response of the tracking filter to ensure the success of object tracking, utilize our object re-detection approach to rectify the proper coordinates of

[1] http://www.ittc.ku.edu/cviu/tracking.html

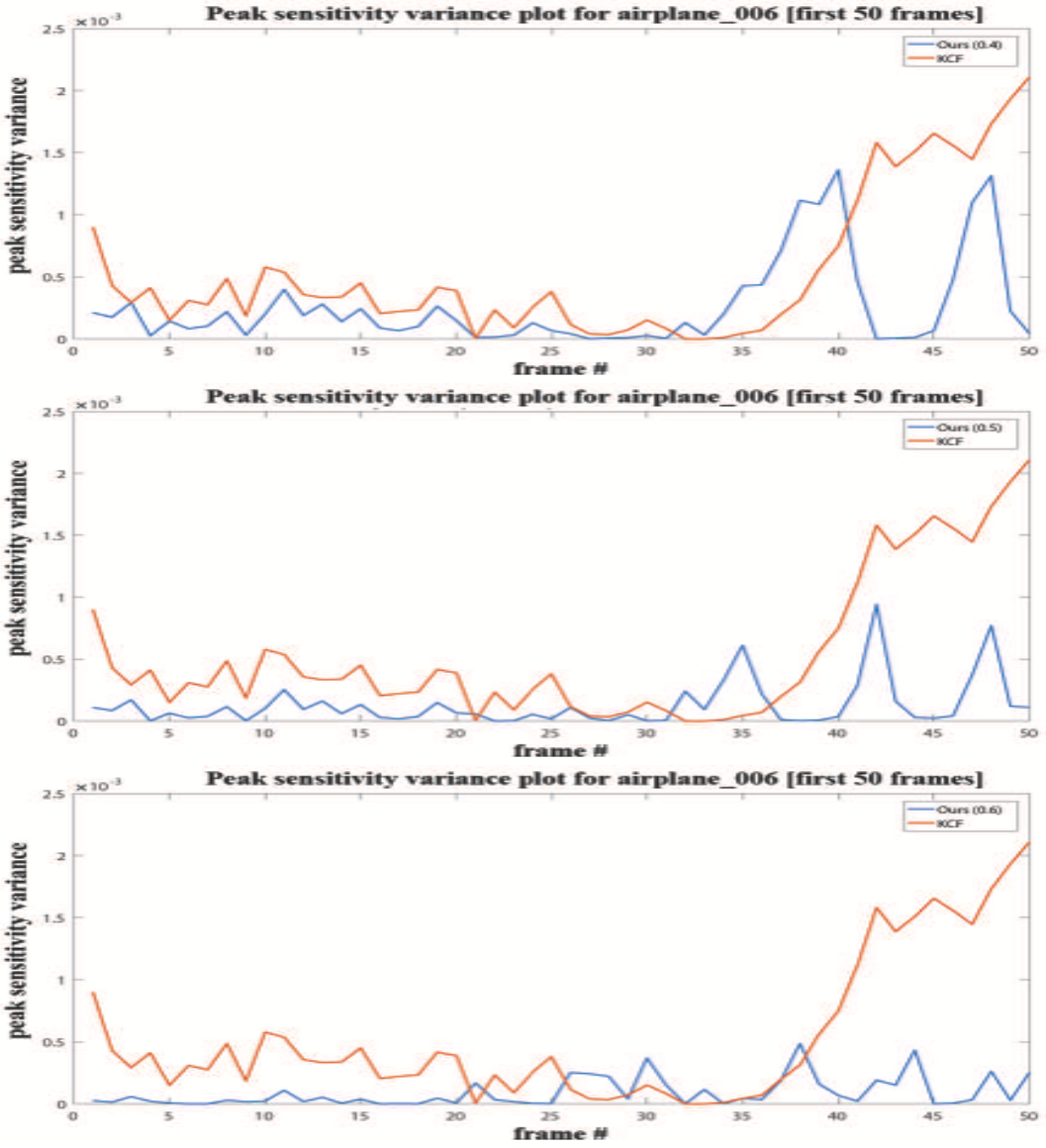


Figure 4.5: Peak sensitivity curve demonstrating high peak of sensitivity variance (bad) of KCF versus low peak of sensitivity variance (good) of our approach in first 50 frames of airplane_006 dataset. (Best viewed in color)

the object being tracked, and further update the tracker with the new object localization parameters. We designed an experiment, in which, we set a certain value for peak of filter response (confidence value) and apply our re-detection approach when the peak of filter response falls to the set value or below it. Since we need a metric to compare and evaluate our plots against the KCF plot for several peak of filter response values, we propose a peak sensitivity variance $p_{var} = \frac{(p_i - p_m)^2}{n}$ where p_i is the peak of the filter response value for i^{th} frame in a given dataset, p_m is a mean peak of filter response value and n is the total number of frames in the dataset chosen. Thus, p_{var} gives the measure of the variation of sensitivity of the tracker's filter response in the i^{th} frame from its mean value. We prefer lower p_{var} value (throughout the frames in any given sequence) which shows that the tracker is able to correctly and consistently track the position of the object in most of the frames and does not lose track of the tracked objects i.e. the bounding box does not deviate away from the tracked object which would otherwise change its peak of response value on this frame (p_i) from mean peak value (p_m), thus increases the sensitivity metric p_{var}.

We observed that as we increased the peak of response value for re-detection, the tracker performed better, i.e., lower peak sensitivity variance in most of the observed frames as shown in Fig. 4.5. This is due to the re-detection scheme being performed more on the frames as soon as the peak of response value would fall to such higher values. In contrast, we found the speed of tracker decreased sharply as more re-detection were being performed. Thus, a suitable balance between the speed of the reliable tracker and p_{var} is needed to be determined. Several experiments on our 25 challenging datasets demonstrated average tracking speed of 83.02 fps, 115.53 fps, 122.18 fps for the set of peak of filter response values 0.4, 0.5 and 0.6 respectively. Though, running the re-detection scheme for lower peak of filter response values reduces p_{var}, it severely impacts our tracking speed. Thus, a peak of filter response value 0.5 was chosen for adaptive re-detection to maintain a sound balance between p_{var} and the tracking speed.

The higher filter response values are observed when the tracker tracks the object properly. Fig. 4.6 shows two such experiments where filter response is plotted throughout 100 frames in two chosen datasets, namely; airplane_011 and youtube_3. It can be clearly observed that the peak of

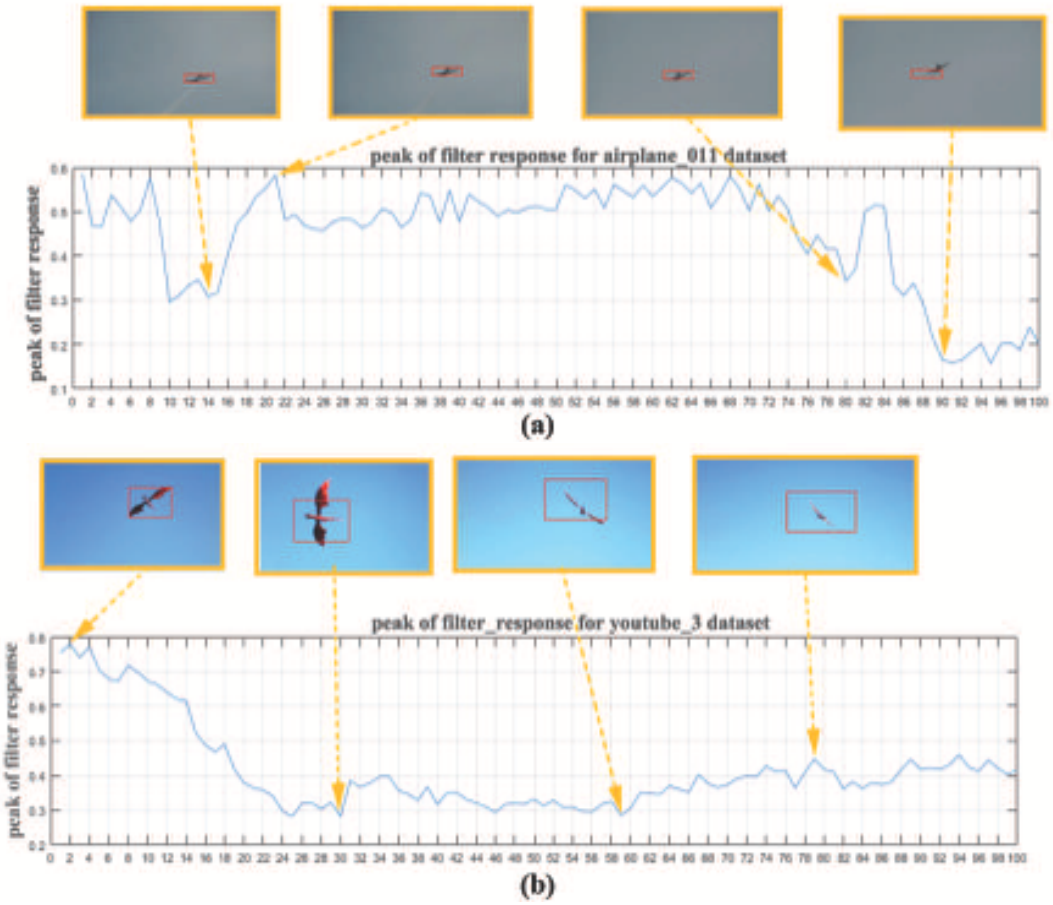


Figure 4.6: Peak of filter response of KCF in (a) airplane_011 and (b) youtube_3 dataset showing fall in the peak values of the filter leading to inaccurate tracking of the object when the object changes its shape, size or illumination.

filter response for KCF remains stable as long as the object being tracked do not change much in shape, size or illumination whereas the peak of filter response falls drastically in circumstances like partial occlusion, scale variation, or inability to track properly. Since the sudden fall or rise in the peak of filter response accounts for higher peak sensitivity variance, it is not a good characteristic for long-term stable trackers as explained above. Several experiments as discussed in later sections clearly demonstrate that our approach is robust to such scale variations or partial occlusions. Comparative study of peak sensitivity variance of KCF versus our approach in Fig. 4.5 further bolsters our claim. The sudden rise observed in the peak sensitivity variance curve in our method is due to re-detection scheme being performed once the tracker tend to lose the object being tracked or the object changes its scale which is essential for a reliable object tracker.

4.2.3 Comparison with State-of-the-Art Trackers

In quantitative analysis, we run the six competing trackers along with our approach on 25 datasets and report the average performance. We use measures like precision rate (PR), success rate (SR) and central location error (CLE) to compare our approach with other competing trackers. CLE is defined as the Euclidean distance between the central coordinates of the the ground truth bounding box and that of the tracker's output. Thus, for a better performance of the tracker, a lower value of CLE is preferred. PR is defined as the percentage of frames in which CLE is lower than a given threshold. A threshold value of 20 pixels is used in this thesis for the evaluation as suggested in [42]. Tracking results are considered to be successful if $\frac{(a_t \cap a_g)}{(a_t \cup a_g)} > \theta$, where $\theta \in [0,1]$, a_t and a_g denote the areas of the bounding boxes of the tracker's output and the ground truth, respectively. Thus, SR is defined as the percentage of frames where the overlap rates are greater than a threshold θ. Generally, θ is set to 0.5, which means a 50% overlap ratio threshold.

Similarly, one pass evaluation (OPE) and temporal robust evaluation (TRE) experiments can be performed for a sound evaluation of any tracker [42]. For OPE, each tracker is run from the first frame till the last frame and compared with the ground truth. TRE slightly differs from OPE as the sequence is randomly divided into several portions (20 in our experimentation) and the tracker is

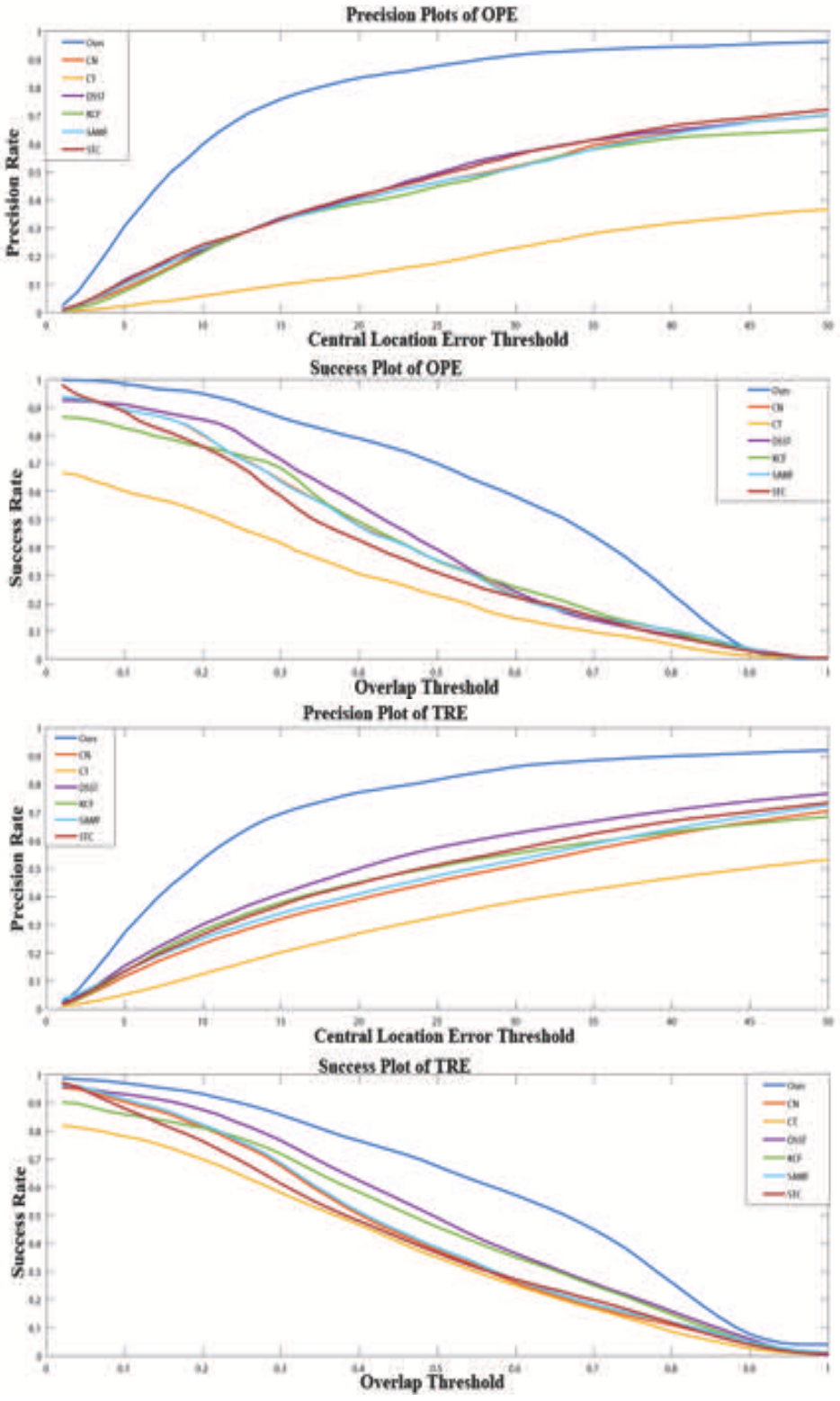


Figure 4.7: OPE and TRE curve demonstrating the average precision rate and the success rate of the proposed and 6 competing trackers over 25 video sequences. (Best viewed in color)

Table 4.1: Quantitative analysis of the proposed and 6 other competing trackers on 25 test sequences. The best and the second best results are highlighted using bold-face and underline font-styles, respectively.

	Ours	CT	STC	CN	DSST	SAMF	KCF
Average Precision Rate (TRE)	**0.82**	0.31	0.47	0.45	<u>0.51</u>	0.45	0.46
Average Success Rate (TRE)	**0.76**	0.37	0.41	0.42	<u>0.49</u>	0.43	0.46
Average Precision Rate (OPE)	**0.77**	0.19	<u>0.45</u>	0.44	<u>0.45</u>	0.43	0.42
Average Success Rate (OPE)	**0.6**	0.27	0.38	0.41	<u>0.43</u>	0.42	0.40
CLE (in pixels)	**13**	170	<u>47</u>	75	59	79	87
Average Speed (fps)	**115.53**	29.13	27.21	27.50	4.97	6.54	<u>71.69</u>

thereafter run on each portion and finally compared with their respective ground truths.

The quantitative evaluation between our approach versus six competing visual trackers: CT [77], STC [78], CN [79], DSST [43], SAMF [15] and KCF [14] is shown in Table 4.1. It can be observed from OPE and PRE values that our method outperforms the competing trackers. Similarly, our approach also has the least CLE and real-time speed performance.

Experimental results performed during OPE against the six competing trackers on all the 25 video sequences are also reported. PR and SR are tabulated in 4.2 and 4.3 respectively. The tables clearly demonstrate that our approach is more accurate in the almost all of the experimented challenging datasets. Similarly, our approach stands best among the competing trackers by a great margin – 20 out of 25 sequences in PR evaluation and 17 out of 25 sequences in SR evaluation.

Fig. 4.7 shows the precision as well as success rate plots for OPE as well as TRE experimented over all the challenging datasets. It is clear from the plots that our approach is significantly better than the other trackers compared. In summary, we can verify that our approach is superior in robustness to the other competing trackers by observing the precision rate plot and our method is also more adaptive to the variations in shape and size of the object being tracked in a given video sequence as demonstrated by success rate plot.

Table 4.2: Precision rate of the proposed and the 6 competing trackers on 25 sequences. The best and the second best results are highlighted using bold-face and underline font styles, respectively.

	Ours	CN	CT	DSST	SAMF	STC	KCF
airplane_001	**0.92**	0.20	0.20	0.26	0.21	0.38	0.12
airplane_004	**0.79**	0.44	0.25	0.50	0.26	0.42	0.37
airplane_005	**0.81**	0.33	0.19	0.32	0.21	0.36	0.27
airplane_006	**0.92**	0.54	0.22	0.54	0.20	0.65	0.53
airplane_007	**0.76**	0.61	0.18	0.36	0.15	0.46	0.37
airplane_011	**0.90**	0.43	0.27	0.28	0.8	0.31	0.25
airplane_012	0.74	0.15	0.20	**0.88**	0.20	0.83	0.81
airplane_013	**0.89**	0.32	0.20	0.32	0.21	0.26	0.12
airplane_015	**0.82**	0.73	0.35	0.58	**0.82**	0.49	0.79
airplane_016	**0.83**	0.76	0.18	0.73	0.75	0.65	0.45
big_2	0.89	0.82	0.31	**0.91**	0.84	0.85	0.85
planestv_1	0.86	**0.90**	0.46	0.85	0.75	**0.90**	0.84
planestv_2	**0.77**	0.37	0.37	0.42	0.35	0.36	0.49
planestv_3	**0.89**	0.33	0.14	0.14	0.30	0.60	0.13
planestv_4	**0.55**	0.03	0.12	0.08	0.15	0.02	0.10
planestv_5	**0.64**	0.03	0.09	0.03	0.09	0.16	0.23
planestv_6	**0.73**	0.47	0.15	0.61	0.70	0.34	0.53
planestv_7	**0.75**	0.27	0.24	0.49	0.30	0.14	0.38
planestv_8	**0.88**	0.64	0.18	0.80	0.63	0.66	0.34
planestv_9	**0.52**	0.04	0.02	0.21	0.20	0.14	0.03
youtube_1	**0.89**	0.88	0.41	0.09	0.80	0.86	0.87
youtube_2	**0.81**	0.65	0.40	0.56	0.70	0.66	0.46
youtube_3	**0.69**	0.07	0.06	0.07	0.08	0.12	0.26
Dog	0.42	0.25	0.31	**0.66**	0.30	0.29	0.42
Skater	0.57	**0.60**	0.53	0.59	0.50	0.44	0.57

Table 4.3: Success rate of the proposed and the 6 competing trackers on 25 sequences. The best and the second best results are highlighted using bold-face and underline font styles, respectively.

	Ours	CN	CT	DSST	SAMF	STC	KCF
airplane_001	**0.78**	0.17	0.20	0.27	0.12	<u>0.32</u>	0.12
airplane_004	**0.54**	<u>0.53</u>	0.49	0.42	0.49	0.47	0.44
airplane_005	**0.57**	0.20	0.15	0.26	0.21	<u>0.39</u>	0.23
airplane_006	**0.54**	0.43	0.19	0.45	0.43	<u>0.49</u>	0.43
airplane_007	<u>0.53</u>	0.46	0.13	**0.55**	<u>0.53</u>	0.47	0.49
airplane_011	**0.77**	0.34	0.21	0.29	<u>0.73</u>	0.33	0.20
airplane_012	0.47	0.31	0.18	<u>0.59</u>	0.34	0.48	**0.70**
airplane_013	**0.70**	0.19	0.24	<u>0.30</u>	0.11	0.27	0.16
airplane_015	**0.71**	<u>0.65</u>	0.45	0.59	0.62	0.49	0.54
airplane_016	**0.73**	0.58	0.22	0.56	<u>0.65</u>	0.62	0.58
big_2	0.61	0.58	0.30	**0.65**	0.57	0.58	<u>0.63</u>
planestv_1	0.26	**0.86**	0.66	<u>0.80</u>	0.76	0.78	0.78
planestv_2	**0.59**	0.41	0.38	<u>0.57</u>	0.45	0.30	0.45
planestv_3	**0.79**	<u>0.47</u>	0.42	0.43	0.42	0.43	0.38
planestv_4	**0.76**	0.35	0.23	0.27	0.30	<u>0.43</u>	0.31
planestv_5	**0.72**	0.32	<u>0.36</u>	0.26	0.27	0.15	0.41
planestv_6	**0.58**	<u>0.48</u>	0.35	0.34	0.32	0.31	0.37
planestv_7	**0.63**	0.28	0.30	<u>0.49</u>	0.11	0.17	0.41
planestv_8	<u>0.47</u>	0.37	0.24	**0.54**	0.29	0.46	0.27
planestv_9	**0.71**	0.37	0.25	<u>0.45</u>	0.41	0.32	0.21
youtube_1	<u>0.44</u>	0.41	0.32	0.18	0.43	0.18	**0.55**
youtube_2	**0.56**	<u>0.45</u>	0.15	0.40	0.35	0.25	0.43
youtube_3	**0.62**	0.24	0.10	0.22	0.22	0.16	<u>0.29</u>
Dog	<u>0.33</u>	0.15	0.15	**0.39**	0.10	0.24	0.19
Skater	0.51	**0.57**	<u>0.55</u>	<u>0.55</u>	0.53	0.47	0.54

4.2.4 Speed Comparison

As presented in Table 4.1, our algorithm (implemented in C++) achieves an average speed of 115.53 frames per second (fps) whereas KCF achieved 132.87 fps when implemented in C++ and 71.69 fps when implemented in MATLAB on the 25 challenging video sequences. One of the major disadvantages of KCF, despite its good speed, is that it fails to track the object in the following frames once it loses track of the object in a given frame, thus making it unreliable for real-time tracking. However, our approach is able to re-detect the object if it loses track of the object due to the proposed re-detection scheme, thus making our algorithm more apt for such purpose. Moreover, KCF is not adaptive to variations in shape and size of the object being tracked and draws a fixed bounding box around them. In contrast, our approach accurately adapts to such variations in shape and size of the object being tracked and adjusts the bounding box accordingly thus making our approach more suitable for sense-and-avoid systems. Similarly, compared to CN, STC and CT, our approach stands out by more than three times (3x) faster than their average speeds. Similarly, DSST and SAMF clearly does not fit for real-time object tracking due to their very low speed. Hence, this tremendous advantage in the speed and long-term tracking ability makes our algorithm more suitable for real-time object tracking than the compared state-of-the-art trackers.

4.2.5 Qualitative Evaluation

In this subsection, the qualitative comparisons of our approach against the 6 competing trackers is presented.

4.2.5.1 Scale Variations and Partial Occlusion

One of the most challenging task for a real-time tracker is to continuously track the object through several occlusions as well as able to cope-up with the changing shape and size of the object. A robust tracker should be least affected by the partial occlusion as well as changing dimension of the

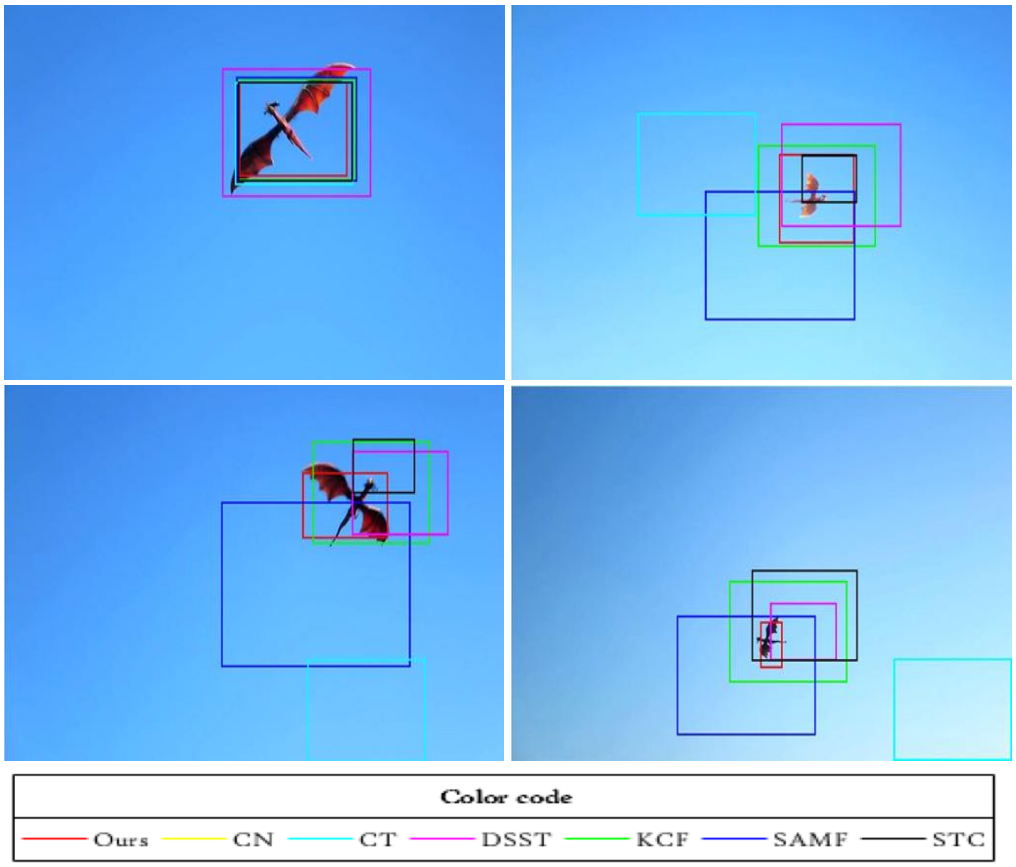


Figure 4.8: Tracking results of our approach and the output of 6 competing trackers during scale variations in youtube_dataset_3.

object being tracked. During the experimentation, we compared all the competing trackers through these test cases.

In Fig. 4.8 and Fig. 4.9, we can observe that our tracker is extremely adaptive to the change in scale as well as during partial occlusion respectively. It is experimentally found that though all the competing trackers produce acceptable outputs in the first few frames, as the object changes its shape (frame #1148, frame #1222 and #1317 in youtube_dataset_3) or undergoes partial occlusion (frame #91, frame #109 and frame #121 in airplane_005), some of the trackers fail. For instance, CT, SAMF, KCF and STC are unable to keep the scale variation in their account. However, our method quickly adjusts to changing appearance and size of the object. Similarly, only STC and our method are found to be invariant of partial occlusion. Almost all the other trackers fail in this

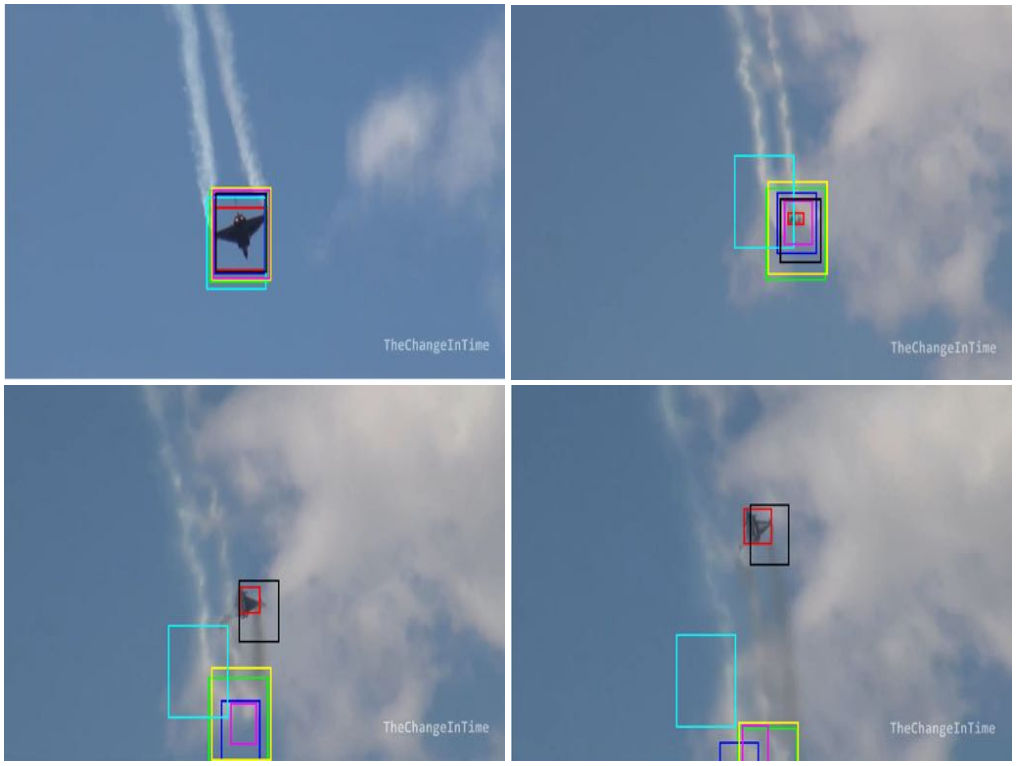


Figure 4.9: Tracking results of our approach and the output of 6 competing trackers during partial occlusions in airplane_005. (Color code same as in Fig. 4.8.)

situation.

4.2.5.2 Rotations

It is very important for the real-time tracker to be able to perform if the object being tracked undergo rotations. During the experiments, we decided to observe the performance of the competing trackers in different types of rotations - axial rotation and planar rotation.

In Fig. 4.10, trackers are tested on the basis of axial rotation (frame #41, frame #182, frame #270 and frame #288 in planestv_4) and in Fig. 4.11 the trackers are tested on the basis of planar rotation (frame #87, frame #203, frame #325 and frame #372 in big_2) dynamics. It can be seen that CT is not apt for both axial or planar rotations. Though KCF is able to perform well (except for scale changes) during axial rotation, it fails against planar rotations. Despite such complicated rotation dynamics, our method stands out among all the other competing trackers, thus proving it

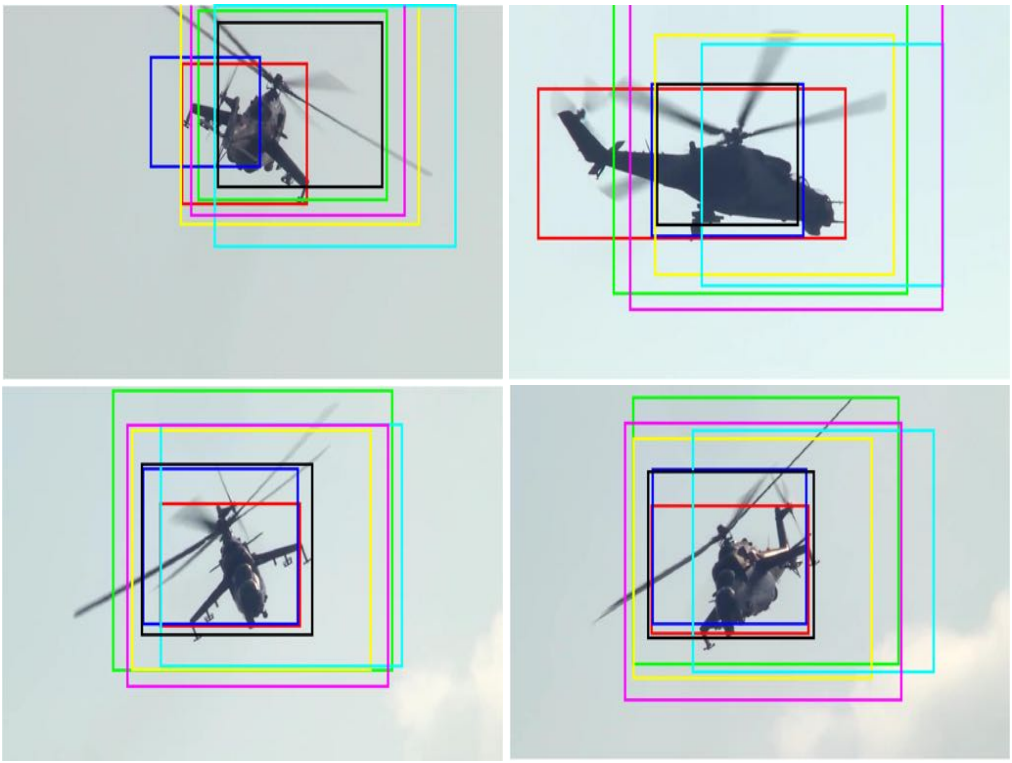


Figure 4.10: Tracking results of our approach and the output of 6 competing trackers during axial rotations in planestv_4. (Color code same as in Fig. 4.8.)

to be more robust.

4.2.5.3 Illumination Variation

A robust object tracking system should be able to account for the change in illumination. A flying obstacle can easily reflect differently in different frames, thus changing the illumination of the object being tracked unpredictably. The tracker which is not trained properly to handle such change of scene due to illumination may fail during the course of tracking and find hard to recover. To check the robustness of the competing trackers, we selected one of the datasets where a flying airplane reflects the sunlight directly on the camera, thus producing the perfect condition for varying illumination as the tracker is tracking the object.

In Fig. 4.12, we depict the evaluation of all the trackers under illumination variation (frame #4, frame #51, frame #87 and frame #177 in airplane_006). It is clear from the figure that CT, CN and

Figure 4.11: Tracking results of our approach and the output of 6 competing trackers during planar rotations in big_2. (Color code same as in Fig. 4.8.)

SAMF are incapable of keeping track of the object under such constraints. However, our method is not affected by such illumination variation and keeps good track of the object.

4.2.5.4 Camera Instability

A camera mouted on the flying UAV is always under a huge turbulence thus resulting into the instability of the camera. Also, several other environmental as well as mechanical factors might directly or indirectly affect the stability of the camera. Thus, it is very important to test the robustness of the tracking algorithms against such unstable scenarios.

Fig. 4.13 demonstrates competing tracker's performance over camera instability (frame #190 - #193 in airplane_001 and frame #37 - #40 in airplane_012). It can be clearly observed that almost all of the trackers, except ours, fail to correctly track the object when there is a significant jerk in the camera pose or sudden perturbations.

Figure 4.12: Tracking results of our approach and the output of 6 competing trackers during illumination variations in airplane_006. (Color code same as in Fig. 4.8.)

4.2.5.5 Roll, Pitch and Yaw

Finally, a robust object tracking algorithm should be confident about the object being tracked despite several complicated maneuvers of the object in the scene. We experimented the competing trackers with a similar dataset where an airplane undergoes roll, pitch and yaw movements.

Moreover, in Fig. 4.14 demonstrates the experimentation of the trackers on roll-pitch-yaw motion (frame #123, frame #147, frame #235 and frame #310 in planestv_9). It is evident that almost all of the trackers fail to keep track of the larger objects in such situations. However, our tracker is able to keep track of both the shape and pose of the object (small or large) being tracked.

4.2.6 Limitations

From several experimentations, we found that our approach, though fast and robust, has some limitations. Since our algorithm is designed to auto-initialize and utilizes salient object detec-

Figure 4.13: Tracking results of our approach and the output of 6 competing trackers during camera instabilities in airplane_012. (Color code same as in Fig. 4.8.)

tion scheme whenever necessary throughout the tracking process, it starts tracking all the objects (bounding box comprises maximum area occupancy of the objects in a given scene) in the presence of multiple objects in a scene as shown in Fig. 4.15. However, our approach effectively adapts its bounding box and narrows down to one object once the prime object to be tracked is segregated in a given scene as demonstrated in Fig. 4.15. It is also important to note that the above mentioned limitation has insignificant effect in sense and avoid UAVs as a single bounding box around multiple objects may suggest the detection of several obstacles within the region, thus need to be avoided during the UAV's trajectory. The other limitation of our approach is an inability to accurate auto-initialization provided a too complex background. For instance, in Fig. 4.16, the tracker's bounding box comprises of the background along with the object to be tracked. This limitation is only observed during the initialization phase. Once the tracker starts learning, the object is significantly tracked to a greater accuracy in the consecutive frames as shown in Fig. 4.16.

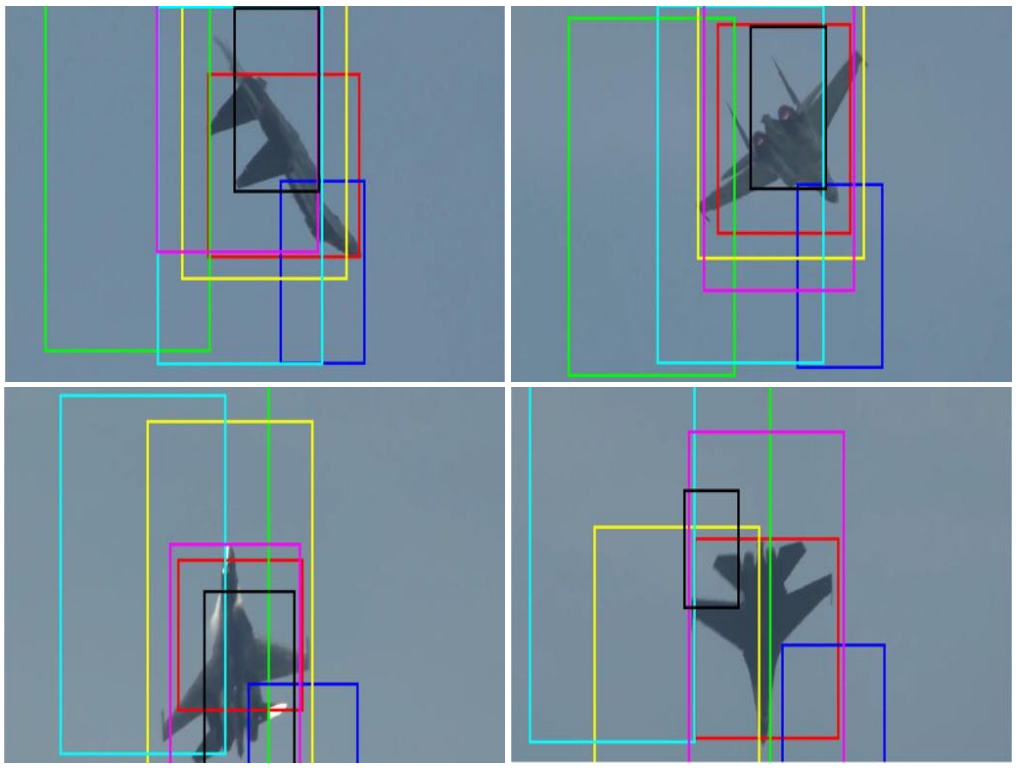


Figure 4.14: Tracking results of our approach and the output of 6 competing trackers during roll, pitch and yaw in planesv_9. (Color code same as in Fig. 4.8.)

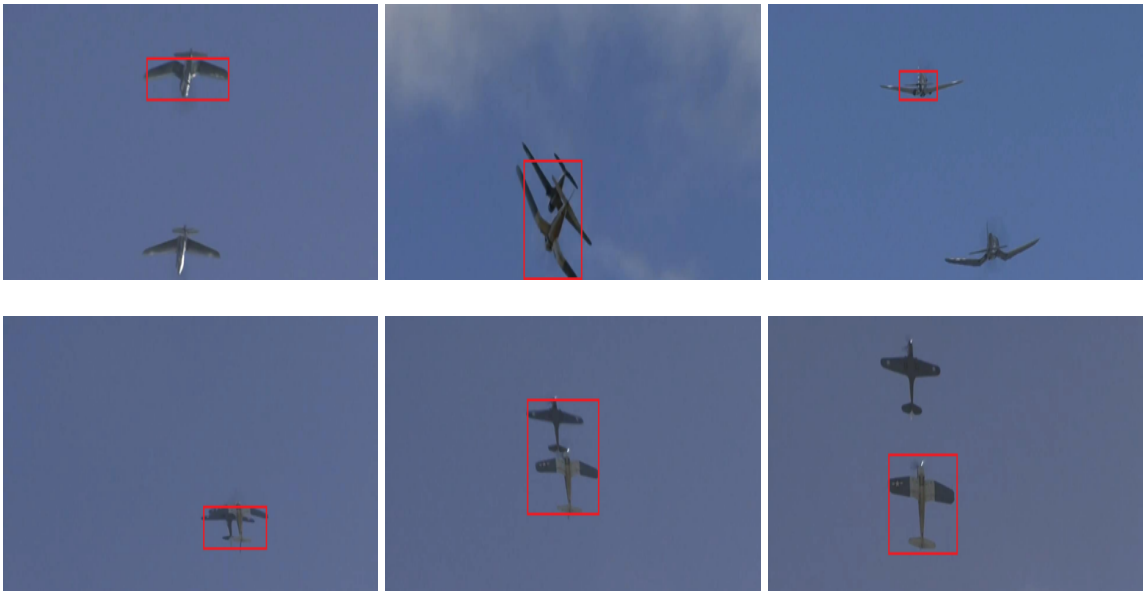


Figure 4.15: Frames showing our tracker narrows down to the prime object in a scene once the multiple objects are at a suitable distance apart or one of the objects move out of scene.

Figure 4.16: Limitation of our approach: Auto-initialization in too complex background is not accurate but our adaptive approach tracks smoothly in successive frames. (Best viewed in color)

Chapter 5

Fundamental Matrix Estimation

Epipolar geometry plays an important role in determining the fundamental matrix from the correspondence points in a stereo image pair for reconstruction of a scene for depth analysis in UAVs. Epipolar geometry holds the intrinsic projective geometry between two views. It does not depend on how the object in 3D space look like (scene structure) but only depends on the cameras' internal parameters and their relative pose.

This intrinsic geometry is well captured by fundamental matrix $\mathbf{F}$. It is a rank 2 matrix of size 3×3. Fundamenetal matrix can be solely computed using image point correspondences from the stereo image pair. But first, let us get well acquainted with the notations and definitions used throughout this chapter.

5.1 Notations and Definitions

Any point in 3D space is denoted by $\mathbf{X}$ and its projected point in the first image is denoted by $\mathbf{x}$ and in the second image is denoted by $\mathbf{x}'$. The images are taken using two stereo cameras with camera intrinsic parameters $\mathbf{K_1}$ along with a rotation $\mathbf{R_1}$ and a translation $\mathbf{T_1}$ for the first image and $\mathbf{K_2}$ along with a rotation $\mathbf{R_2}$ and a translation $\mathbf{T_2}$ for the second image. More suitably, camera parameters can be expressed as $\mathbf{P_1} = \mathbf{K_1}[\mathbf{R_1}|\mathbf{T_1}]$ for the first camera and $\mathbf{P_2} = \mathbf{K_2}[\mathbf{R_2}|\mathbf{T_2}]$ for the second camera. We can also use the same camera to take the stereo images. In this case, $\mathbf{K_1} = \mathbf{K_2} = \mathbf{K}$. Also, let the camera centers be denote by $\mathbf{C}$ and $\mathbf{C}'$ as shown in Fig. 5.1. The straight line that joins two camera centers is called baseline and the epipolar plane π is the plane containing $\mathbf{C}$, $\mathbf{x}$, $\mathbf{X}$, $\mathbf{x}'$ and $\mathbf{C}'$. The point of intersection of the baseline with the image plane is

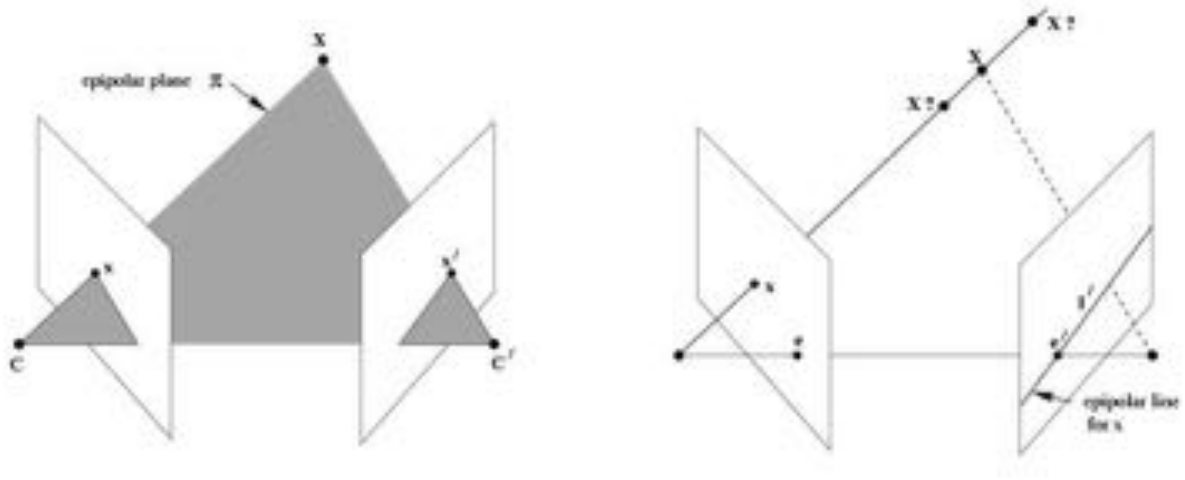


Figure 5.1: Point correspondence geometry. (Image source: [12])

called an epipole. Interestingly, epipole in an image plane is the image of the center of the other camera. In Fig. 5.1, $\mathbf{e}$ and $\mathbf{e}'$ are epipoles. The intersection of two planes - π and the image plane describes an epipolar line. $\mathbf{l}$ and $\mathbf{l}'$ are two epipolar lines in Fig. 5.1. All the epipolar lines intersect at the epipole. Epipolar line can be thought of as an image of back projected ray from the first image plane on the second image plane. For example, in Fig. 5.1, back projected ray $\mathbf{xX}$ from the first image casts an image $\mathbf{l}'$ on the second image plane. This understanding helps us to restrict the search of an image point correspondence for x in the second image plane to a linear search in $\mathbf{l}'$ rather than on an entire image and vice versa. Thus a map $\mathbf{x} \rightarrow \mathbf{l}'$ exists and the fundamental matrix captures this information which is discussed next.

5.2 The Fundamental Matrix

The fundamental matrix $\mathbf{F}$ can be best described as the algebraic representation of epipolar geometry. Geometrically, $\mathbf{F}$ represents a mapping from the 2-D projective plane of the first image to the pencil of epipolar lines (1-D) through the epipole $\mathbf{e}'$ and thus has rank 2 [12]. Fundamental matrix can be easily computed from the camera projection matrices $\mathbf{P}$ and $\mathbf{P}'$ provided non-coincident camera centers using $\mathbf{F} = [\mathbf{e}']_{\times}\mathbf{P}'\mathbf{P}^{+}$ where $[\mathbf{e}']_{\times}$ is a skew-symmetric matrix generated from $\mathbf{e}$ as

if $\mathbf{e} = (e_1, e_2, e_3)^T$ is a 3-vector, then its corresponding skew-symmetric matrix can be derived as

$$[\mathbf{b}]_\times = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \tag{5.1}$$

$\mathbf{P}^+$ is a pseudo inverse matrix of $\mathbf{P}$ such that $\mathbf{P}\mathbf{P}^+ = \mathbf{I}$, $\mathbf{I}$ being an identity matrix.

Similarly, for any correct corresponding match points in the stereo images, the fundamental matrix satisfies

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0 \tag{5.2}$$

This equation clearly demonstrates that the fundamental matrix is independent to camera parameters, pose or 3-D scene structure. This equation helps to calculate the fundamental matrix solely from the image point correspondences. Some of the few properties of the fundamental matrix is listed below.

1. Transpose: If $\mathbf{F}$ is the fundamental matrix for the camera parameters pair $(\mathbf{P}, \mathbf{P}')$, then $\mathbf{F}^T$ is the fundamental matrix for the opposite camera parameters pair $(\mathbf{P}', \mathbf{P})$.

2. Epipolar lines: The corresponding epipolar line for any point $\mathbf{x}$ in the first image plane is expressed as $\mathbf{l}' = \mathbf{F}\mathbf{x}$. Also, for any point $\mathbf{x}'$ in the second image plane, the corresponding epipolar line is given by $\mathbf{l} = \mathbf{F}^T\mathbf{x}$.

3. Epipole: For any point $\mathbf{x}$ other than epipole itself, the epipolar line $\mathbf{l}' = \mathbf{F}\mathbf{x}$ contains epipole $\mathbf{e}'$. Similar is for the points in the other image plane.

4. Degrees of freedom: The fundamental matrix is a 3×3 homogeneous matrix with seven degrees of freedom as it satisfies the contraint det($\mathbf{F}$) = 0 where det(.) is the determinant operator.

5. Fundamental matrix is a correlation that is it projects a point to a line.

5.3 Computation of the Fundamental Matrix

If we acquire sufficient matching point correspondences $\mathbf{x} \leftrightarrow \mathbf{x}'$, we can compute the unknown fundamental matrix $\mathbf{F}$ using equation 5.2. This is possible because each homogeneous matching point correspondences $(x, y, 1) \leftrightarrow (x', y', 1)$ gives one linear equation for solving $\mathbf{F}$. For instance, the equation can be expressed as

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{3,1} + yf_{32} + f_{33} = 0 \tag{5.3}$$

Let $\mathbf{f}$ denote 9-vector with the entries of $\mathbf{F}$ in row-major order, then equation 5.3 can be represented as

$$\big(x'x, x'y, x', y'x, y'y, y', x, y, 1\big)\mathbf{f} = 0 \tag{5.4}$$

Thus, if there are n corresponding matching points, we can set our linear functions as

$$\mathbf{Cf} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \mathbf{f} = \mathbf{0} \tag{5.5}$$

From the homogeneous equation 5.5, we can determine $\mathbf{F}$ upto a scale. Also, for solving the linear equations in equation 5.5, $\mathbf{C}$ must be a rank 8 matrix. If so, we can determine a unique solution up to scale using linear methods of equation solving to find a generator of the right null-space of $\mathbf{C}$.

However, if the image point correspondences are noisy, then the rank of $\mathbf{C}$ may be greater than 8 where least square solution can be used to solve the set of linear equations. The least square solution for $\mathbf{f}$ can be easily obtained using singular value decomposition technique where the solution is the smallest singular value of $\mathbf{C}$ that is the last column of V in singular value decomposed $\mathbf{C} = UDV^T$. The solution vector $\mathbf{f}$ determined minimizes $||\mathbf{Cf}||$ such that $||\mathbf{f}|| = 1$.

5.3.1 Seven point correspondences only

As it has been already discussed that if **C** has rank 8, **f** can be solved up to scale. But, there may be cases when only 7 point correspondences are available that is **C** has rank 7. In such case, **f** is solved using the singularity constraint.

The solution for the equation $\mathbf{Cf} = \mathbf{0}$ is in the form of $\beta F_1 + (1-\beta)F_2$ where β is any scalar value. The matrices F_1 and F_2 are evaluated as the matrices corresponding to the generators $\mathbf{f}_1$ and $\mathbf{f}_2$ of the right null space of **C**. Also, constraint $\det(\beta F_1 + (1-\beta)F_2) = 0$ is imposed which leads to cubic polynomial equation in β where det(.) is a determinant operator. The complex solutions are discarded among the three solutions of β to obtain the correct solution.

5.3.2 Normalized Eight Point Algorithm

This is one of the easiest and simplest method of computing the fundamental matrix **F**. This method solves the set of linear equations using least squares principle. One of the key steps in this algorithm is the normalization of the image points which involves translation and scaling so that the centroid is at the origin and the root mean square distance of the points from the origin is equal to $\sqrt{2}$. Also, a singularity constraint is applied before denormalization. The Normalized Eight Point algorithm [12] is presented below.

Algorithm 3 Normalized Eight Point Algorithm

Input: $n \geq 8$ image point correspondences $x_i \leftrightarrow x_i'$
Output: **F**

1: Normalize the image coordinates $\hat{x}_i = Tx_i$ and $\hat{x'}_i = T'x_i'$ with normalizing transformations T, T'
2: Determine $\hat{\mathbf{F}}$ from the singular vector corresponding to the smallest singular value of $\hat{\mathbf{C}}$ obtained from $\hat{x}_i \leftrightarrow \hat{x}_i'$
3: Apply the constraint to $\hat{\mathbf{F}}$ to obtain $\det(\hat{\mathbf{F}}') = 0$
4: Denormalize to obtain $\mathbf{F} = T'^T \hat{\mathbf{F}}' T$

5.4 Determining the Camera Parameters from F

To this point we have carefully examined the properties of the fundamental matrix $\mathbf{F}$ and the point correspondences $\mathbf{x}$ and $\mathbf{x}'$. It is clear from the previous section that the map $\mathbf{l}' = \mathbf{F}\mathbf{x}$ and the correspondence condition ${\mathbf{x}'}^T\mathbf{F}\mathbf{x} = 0$ are projectively related that is the derivation only involve projective geometric relationships like line or plane intersection.

Similarly, the fundamental matrix solely depends on the camera parameters $\mathbf{P}$ and $\mathbf{P}'$. Likewise the camera parameters depends on both the image coordinates and the world coordinate frame. There are several choices of camera parameters that can be derived from $\mathbf{F}$ in [12]. Three of them are presented below.

- If S be any skew-symmetric matrix, then the camera matrices can be defined as $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$ and $\mathbf{P}' = [S\mathbf{F}|\mathbf{e}']$.
- The camera matrices can also be derived from the fundamental matrix as $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$ and $\mathbf{P}' = [[\mathbf{e}']_\times\mathbf{F}|\mathbf{e}']$ where $[\mathbf{e}']_\times$ is defined in Appendix section 5.1.
- A pair of canonic camera matrices can also be expressed as $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$ and $\mathbf{P}' = [[\mathbf{e}']_\times\mathbf{F} + \mathbf{e}'\mathbf{v}^T|\lambda\mathbf{e}']$

5.5 Essential Matrix

A special case of the fundamental matrix is the essential matrix $\mathbf{E}$ where the normalized image coordinates are taken into consideration. Also, the essential matrix posses fewer degrees of freedom than the fundamental matrix.

5.5.1 Normalized Coordinates

We know that a point in a 3-D space $\mathbf{X}$ can be projected into an image plane via $\mathbf{x} = \mathbf{P_1}\mathbf{X}$ where $\mathbf{P} = \mathbf{K_1}[\mathbf{R_1}|\mathbf{T_1}]$. If calibration matrix $\mathbf{K_1}$ is known to us, we can multiply both the sides of the equation with its inverse to obtain $\hat{\mathbf{x}} = {\mathbf{K_1}}^{-1}\mathbf{x} = [\mathbf{R_1}|\mathbf{T_1}]\mathbf{x}$ where $\hat{\mathbf{x}}$ is the normalized image point

coordinate. It can also be expressed as the projected image point of $\mathbf{X}$ via $[\mathbf{R_1}|\mathbf{T_1}]$ with an identity camera calibration matrix.

5.5.2 Determining the Camera Parameters from E

The camera parameters from the fundamental matrix can be determined upto a projective ambiguity. However, the camera parameters from an essential matrix can be determined upto a scale and four-fold ambiguity [12]. Four-fold ambiguity means that there are four camera parameters that can be derived and only one of them is the required solution as described below.

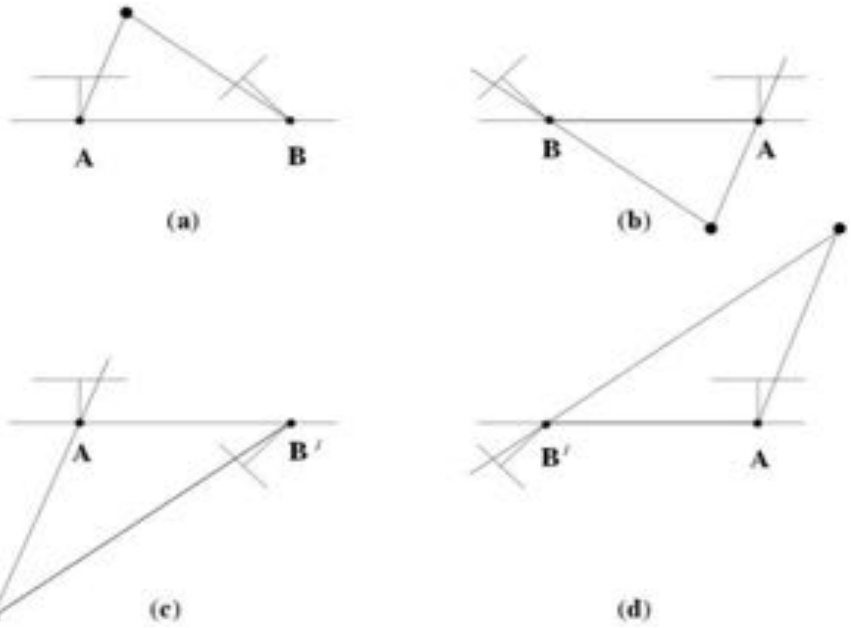


Figure 5.2: Possible solutions for calibrated reconstruction from $\mathbf{E}$. (Image source: [12])

Provided the essential matrix, $\mathbf{E} = U\ \text{diag}(1,1,0)\ V^T$ and the first camera matrix $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$, four possible camera parameters can be chosen from. They are defined as $\mathbf{P'} = [UWV^T|+\mathbf{u}_3]$ or $\mathbf{P'} = [UWV^T|-\mathbf{u}_3]$ or $\mathbf{P'} = [UW^TV^T|+\mathbf{u}_3]$ and $\mathbf{P'} = [UW^TV^T|-\mathbf{u}_3]$. The correct solution from these sets of equations are chosen such that the reconstructed 3-D point $\mathbf{X}$ should be in front of both the cameras as shown in Fig. 5.2. One can test this with a single point to deduce the correct solution.

Chapter 6

Robust Oultier Rejection Approach: RES-Q, Experiments and Results

In this chapter, the details of the proposed strategy for a robust fundamental matrix estimation technique RES-Q is discussed in detail. In the previous chapter, we have already discussed the importance of the fundamental matrix and the required mathematics to estimate the several parameters to obtain the fundamental matrix as well as the parameters that can be derived once the fundamental matrix is obtained. In this chapter we will focus on the proposed outlier detection and rejection methodologies for a symmetric gaussian noise model as well an asymmetric noise model. In the process, a robust outlier rejection approach RES-Q is discussed in detail and experimentation results on a synthetic as well as real datasets are shown. Further, the results are analysed to show that our proposed outlier rejection approach is more robust than the classical algorithms.

6.1 Proposed Approach

6.1.1 Fundamental Matrix Estimation and Algebraic Error

Given a stereo image pair, the intrinsic projective geometry between the pair of images is captured by its epipolar geometry. Epipolar geometry between two-view images is dependent only on the camera's intrinsic parameters and relative pose. Since it does not depend on the scene structure, the fundamental matrix $\mathbf{F}$ is able to encapsulate the epipolar geometry between the stereo image pair. The fundamental matrix is mathematically represented as a rank 2 matrix of size 3×3.

The fundamental matrix can be estimated from a set of point correspondences between a pair

of images. Provided a stereo image pair A and B such that $\mathbf{x}_i \in A$ and $\mathbf{x}'_i \in B$ be two i^{th} matching point correspondences represented using homogeneous coordinates, $\mathbf{F}$ should satisfy the following equation [12].

$$\mathbf{x}_i'^T \mathbf{F} \mathbf{x}_i = 0 \tag{6.1}$$

Each matching correspondence point pair contributes to a single linear constraint. Since $\mathbf{F}$ is a 3×3 rank 2 matrix defined up to a scale, eight pairs of matching point correspondences are sufficient to determine $\mathbf{F}$ using eight-point linear algorithm [12, 80]. However, least square estimation technique can also be utilized if there are more than eight point pair correspondences available.

In order to evaluate the correctness of the estimated fundamental matrix $\mathbf{F}$, we must determine an error measurement scheme that aids to further correction of the fundamental matrix estimation technique. Errors in the fundamental matrix estimation occur due to erroneous matching point pairs. One of the most widely used error measurement criteria is an algebraic error. An algebraic error e_{alg} can be calculated as

$$e_{alg}(i) = \mathbf{x}_i'^T \mathbf{F} \mathbf{x}_i \tag{6.2}$$

Since the matching point correspondences are faulty, they do not satisfy equation (6.1) and result in some error as calculated in equation (6.2).

6.1.2 Reprojection Residual Error

Though the calculation of algebraic error is rather simple, it does not contain any clear geometric meaning. Also, equation (1) provides a necessary but insufficient condition for correct matches as the point in $\mathbf{x}_i \in A$ may have the corresponding match points $\mathbf{x}'_i$ throughout the epipolar line defined by $\mathbf{F}\mathbf{x}_i$ in B. Therefore, in this paper, we propose to use the reprojection residual error to overcome such ambiguity.

First, we estimate the fundamental matrix $\mathbf{F}$ from available matching point correspondences between the stereo image pair. Next, a pair of camera matrices $\mathbf{P_1}$ and $\mathbf{P_2}$ are recovered from $\mathbf{F}$ for

both the cameras using the following equation as described in [12].

$$\mathbf{P_1} = \left[\mathbf{I} \,|\, \mathbf{0}\right] \tag{6.3}$$

$$\mathbf{P_2} = \left[\left[\mathbf{e}'\right]_{\times}\mathbf{F} + \mathbf{e}'\mathbf{q}^T \,|\, \beta\mathbf{e}'\right] \tag{6.4}$$

where $\mathbf{e}'$ is the epipole in the second image such that ${\mathbf{e}'}^T\mathbf{F} = 0$, $\mathbf{q}$ is an arbitrary three dimensional vector and β is any nonzero scalar ($\beta \neq 0$) .

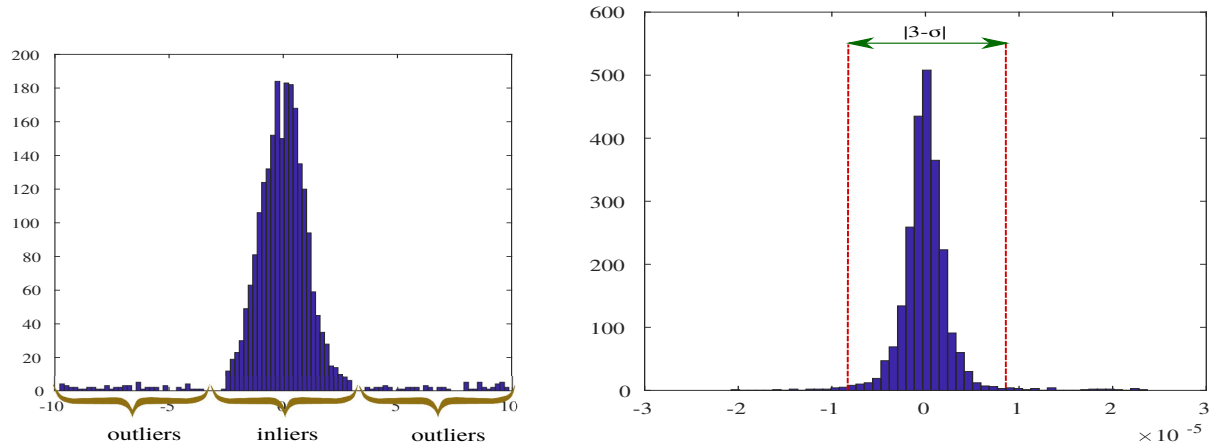
Based on the recovered projection matrices, a triangulation algorithm [81] is used to obtain a perspective 3D reconstruction of the matching correspondences. Thereafter, we use $\mathbf{P_1}$ and $\mathbf{P_2}$ to reproject the reconstructed 3D points back to the respective image planes. Let $\hat{\mathbf{x}}_i$ and $\hat{\mathbf{x}}'_i$ be the reprojected image point pairs for $\mathbf{x}_i$ and $\mathbf{x}'_i$ respectively, the reprojection residual for point i in the two images can be calculated as $\mathbf{r}_i = \begin{bmatrix} \Delta u_i \\ \Delta v_i \end{bmatrix}$ and $\mathbf{r}'_i = \begin{bmatrix} \Delta u'_i \\ \Delta v'_i \end{bmatrix}$, where Δu and Δv are the reprojection errors along the two coordinate axes, respectively. Then, the reprojection residual error can be defined as the following vector by collecting the residuals of all the corresponding matches and their reprojections in the two images.

$$e_{rr} = \left\{\mathbf{r}_i, \mathbf{r}'_i \mid i = 1, ..., N\right\} \tag{6.5}$$

where N is the total number of correspondences.

6.1.3 Outlier Detection and Removal Policy

Since erroneous matching point correspondences result in an inefficient fundamental matrix estimation, it is necessary to determine a suitable outlier detection policy to eliminate these faulty matches. In this section, we will discuss outlier detection and removal policies for a symmetric Gaussian noise model as well as a generalized random noise model to successfully eradicate the outliers present in the image pairs. Before moving on to these cases, it is vital to understand that the reprojection residual error undergoes similar noise distribution as present in the ordinary im-



(a) (b)

Figure 6.1: (a) Histogram of the added Gaussian noise (inliers) along with large standard deviation noise (outliers). (b) Histogram of the reprojection residual error that follows Gaussian distribution and 3-σ range for inliers classification.

age [17] as shown in the simulation result for a Gaussian inlier noise and random outlier noise in Fig. 6.1. The reprojection residual error plot is simply a histogram plot of the vector as defined in equation (6.5).

6.1.3.1 Gaussian Noise Model

One of the most widely adopted noise models in the field of computer vision and image processing is the Gaussian noise model. The credit can also be given to the central limit theorem. Gaussian model has a very simplistic representation as it can be analyzed using only two parameters: mean μ and variance σ^2. The probability density function of a Gaussian distribution is given by the following equation

$$f(y|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(y-\mu)^2}{2\sigma^2}} \tag{6.6}$$

Since the outliers always tend to have a larger standard deviation σ, through extensive experimentations we have found that the reprojection residual error of the outliers also tend to have a larger standard deviation in comparison to that of the inliers, as demonstrated in Fig. 6.1. This finding, supported by our earlier research work [17, 82], helps us to correctly estimate and remove the outliers using 3-σ principle. According to the Gaussian distribution, 99.7% of the inliers should

be within 3-σ of the mean. This principle explains that the matching points with the reprojection residual error larger than three times the variance also known as sigma scaling factor (σ_f) calculated from $e_{rr}(i)$ for all the corresponding point pairs can be classified as outliers. Thus, 3-σ can be considered as a suitable threshold to filter the inliers from the outliers effectively. Hence, the only policy needed to determine now is how to find the required standard deviation from the calculated residual errors.

We have based our approach on robust statistics [54] as they are proven to be the most reliable outlier detection method. Although the median absolute deviation (MAD) was extensively used in our previous work [17, 82], the study [16] suggests that the Q_n estimator is a more robust statistic that experimentally outperforms MAD. Thus, we decided to adopt the Q_n estimator in this paper which can be computed using the following equation.

$$Q_n = \sigma = c\{|x_a - x_b|; a < b\}_{(m)} \tag{6.7}$$

where c is a constant factor and $m = \binom{v}{2} \approx \binom{n}{2}/4$, $v = \binom{n}{2} + 1$. Thus, the formula suggests that we need a constant factor c to scale for the different noise distributions and m^{th} order statistic of the $\binom{n}{2}$ interpoint distances. The constant factor c for the Gaussian noise model is determined to be 2.2219.

Once the Q_n estimator is calculated, the outliers can now be easily classified using

$$\mathscr{O} = \{(x_i, x_i') : |e_{rr}(i)|_{std} > 3\sigma\} \tag{6.8}$$

where $\mathscr{O}$ is an outlier set and $|e_{rr}(i)|_{std}$ is a standardized (mean = 0) reprojection residual error for x_i and x_i'.

6.1.3.2 Generalized Noise Model

Since the noises in the stereo image cannot be always assumed to follow a symmetric Gaussian distribution in the real-world, it is essential to determine an outlier detection policy for asymmetric

noise distributions as well. Thus, we experimented our algorithm with the stereo images populated with random noises corresponding to a lower standard deviation (typically within ± 3) as inliers and a larger standard deviation (typically ± 4 to ± 10) as outliers. Now, the theoretical understanding follows the same as discussed earlier because the inlier distribution is bounded and the outliers can be successfully filtered using 3-σ principle.

6.1.4 Outline of RES-Q Algorithm

The implementation details of the above proposed robust algorithm RES-Q is summarized as below.

Algorithm 4 Robust Outlier Detection using RES-Q

Input: stereo images
Output: optimal fundamental matrix

1. Estimate an initial fundamental matrix $\mathbf{F_{int}}$ using all the points in the stereo image
2. Estimate the camera parameters $\mathbf{P_1}$ and $\mathbf{P_2}$ from $\mathbf{F_{int}}$
3. 3D reconstruct the point pairs using triangulation
4. Reproject the 3D cloud points using $\mathbf{P_1}$ and $\mathbf{P_2}$
5. Calculate the reprojection residual error e_{rr} from reprojection in step 4 with input image pairs
6. Use 3-σ principle to remove the outliers (use equation 8)
7. Re-estimate the fundamental matrix $\mathbf{F_{est}}$ using detected inliers and repeat steps 3 - 7 one more time to refine the inliers
8. Final $\mathbf{F_{est}}$ is the optimal fundamental matrix for the input stereo images

6.2 Evaluations on Synthetic Data

Our algorithm is evaluated on a computer simulated synthetic data and compared qualitatively as well as quantitatively against similar algorithms in the literature. During the simulation, we generated nearly 1200 space points to form a cube of $50 \times 50 \times 50$ as shown in Fig. 6.2. Then, an image pair of size 800×800 is produced using camera matrices $\mathbf{P_1}$ and $\mathbf{P_2}$. During the simulation, focal length of the camera is set to 800 and is kept 60 units apart from the cube and angular rotations of the camera are set to $[45, 30, 75]$ and $[45, 15, 0]$ to generate a pair of stereo images. Inliers were

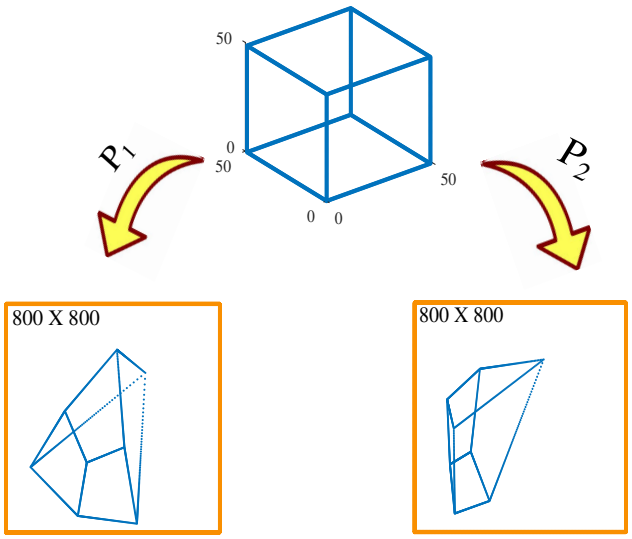


Figure 6.2: Simulated 3D cube and projected stereo image using camera parameters $\mathbf{P_1}$ and $\mathbf{P_2}$ respectively.

simulated by adding Gaussian noise of 0 mean and 3 pixels standard deviation whereas outliers were randomly introduced by adding noise with the larger standard deviation (up to ± 10). To best observe the performance of the algorithms on different amount of random noises, we varied the outlier percentage from 5% to 25% during the experiments. Every algorithm was run 500 trials for each outlier percentage value to best generalize the performance of the compared algorithms.

6.2.1 Mean Reprojection Residual Error

We analyzed our algorithm with the other state-of-the-art algorithms and plotted the mean reprojection residual error plot to determine the merits of the algorithms based on the different reprojection residual error at different outlier levels. In Fig. 6.3(a) and Fig. 6.4(a), we plot the mean reprojection residual error curve where the algorithm is run to determine the final set of inliers using final estimated fundamental matrix and the mean reprojection residual error is calculated for these inliers with their corresponding points in an input image over 500 trials. If the fundamental matrix determined by the algorithm is precise, the algorithm shall be robust to the outliers and effectively detect the inliers with less error. Thus, it can be verified that the robust algorithms should show less

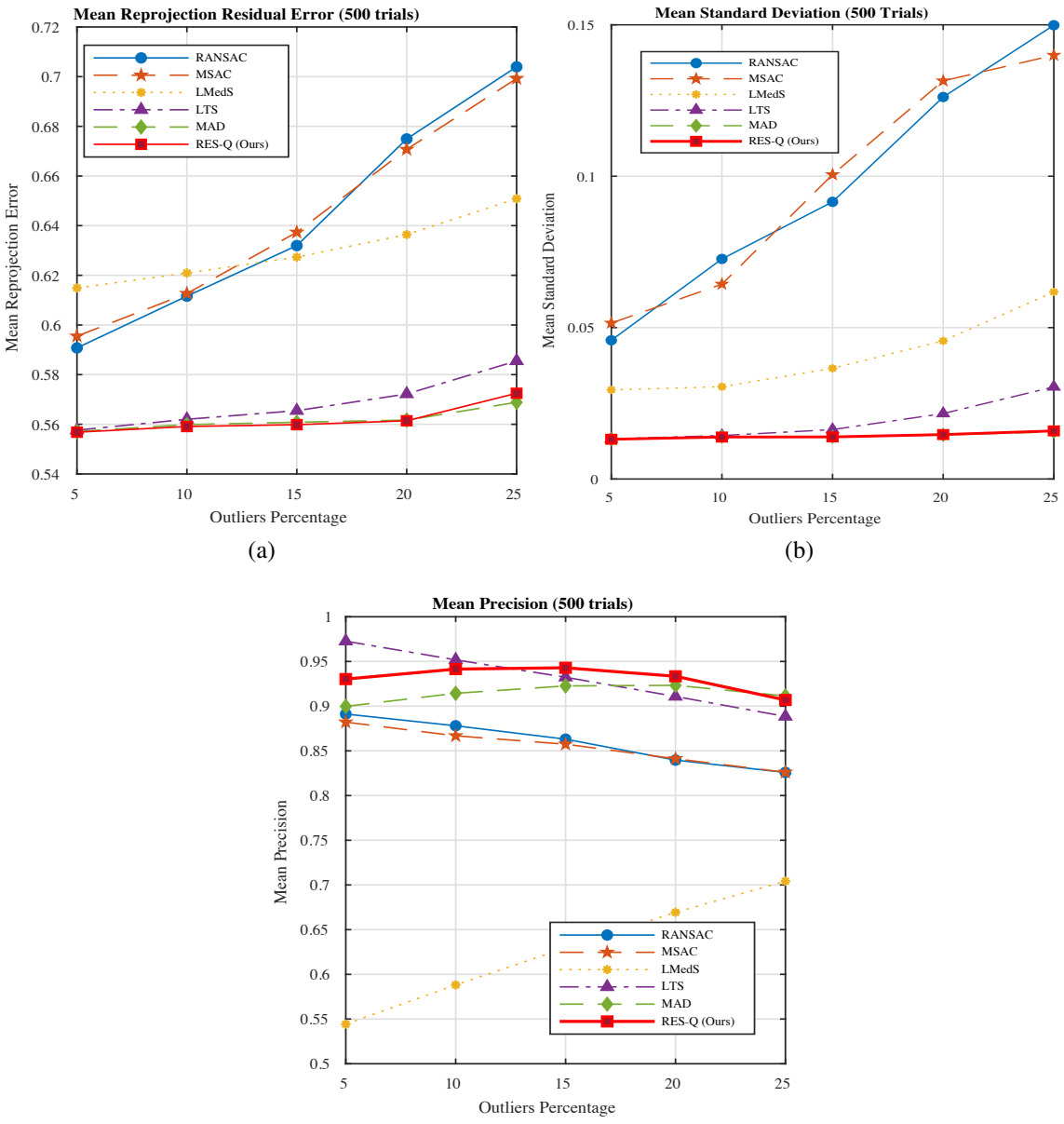


Figure 6.3: Synthetic experimentation plots (inliers modeled as a random symmetric noise) comparing algorithms for 500 independent trials. (a) Mean reprojection residual error. (b) Mean standard deviation of reprojection residual error. (c) Mean precision. (Best viewed in color)

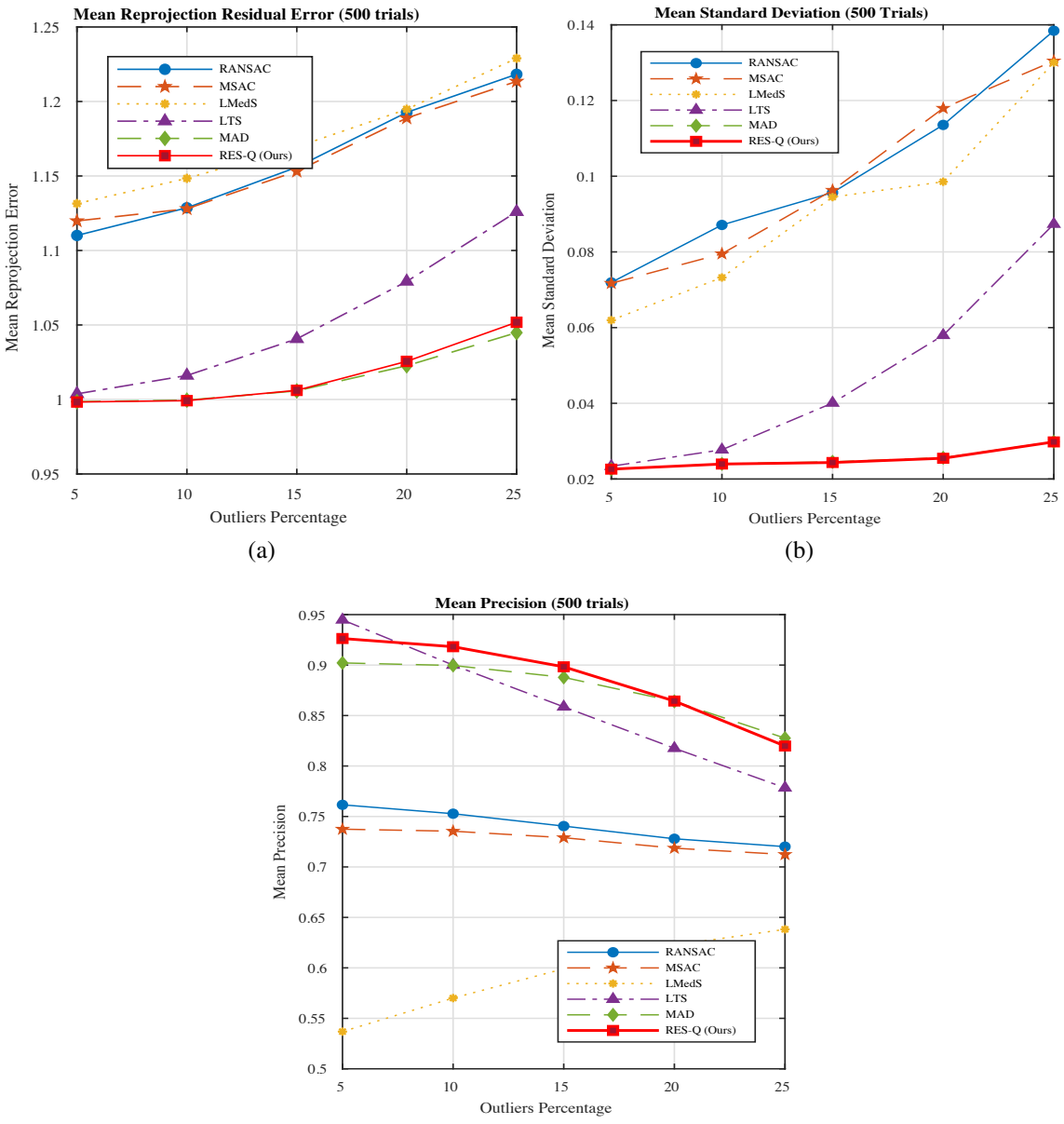


Figure 6.4: Synthetic experimentation plots (inliers modeled as a random asymmetric noise) comparing algorithms for 500 independent trials. (a) Mean reprojection residual error. (b) Mean standard deviation of reprojection residual error. (c) Mean precision. (Best viewed in color)

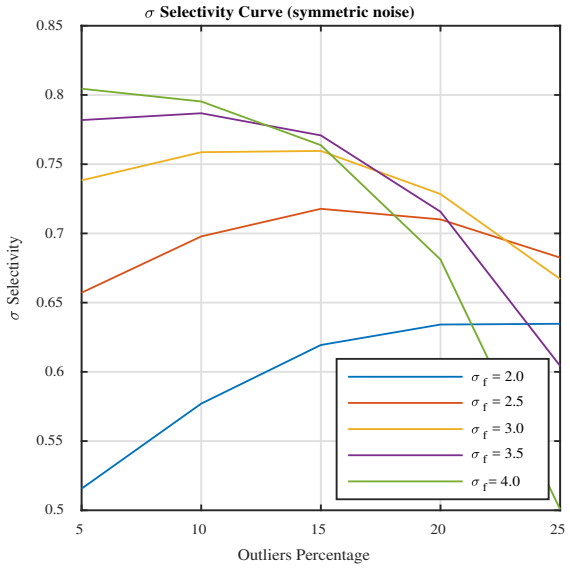


Figure 6.5: RES-Q sigma selectivity versus various outlier levels for a symmetric noise.

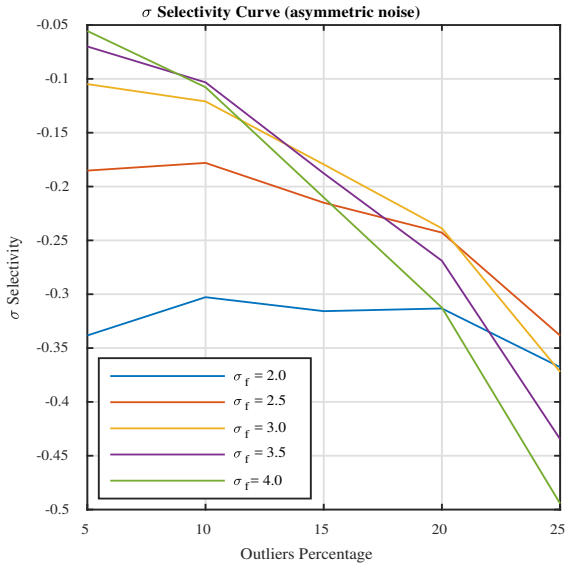


Figure 6.6: RES-Q sigma selectivity versus various outlier levels for an asymmetric noise.

mean reprojection residual error. In the curve, we can see that ours, as well as MAD [17], shows minimal reprojection residual error compared to the other algorithms. LTS [59] though starts with a lower mean reprojection residual error, further increases significantly when the percentage of the outliers in an input image is increased thus making it less robust to heavier noises. A similar pattern can be observed in RANSAC [48], MSAC [51], and LMedS [54] except for the fact that they are not as robust as LTS, MAD and our approach. The gradual increase in the curve can be accounted for the fact that as the outlier percentage increases, the algorithm finds harder to determine the accurate fundamental matrix which would result in erroneous inliers.

6.2.2 Mean Standard Deviation of Reprojection Residual Error

Mean standard deviation of the reprojection residual error is plotted by calculating the average standard deviation of the reprojection residual error for each percentage of the outliers added over 500 independent trials. Since the standard deviation measures the variation of the reprojection residual error from its mean over multiple trials, it determines the stability, reliability and reproducibility of the algorithm. Hence, if the algorithm tends to show the constant mean standard deviation of the reprojection residual error over different levels of outliers, we can conclude that the performance of the algorithm is noise independent and static. The experimentation plots in Fig. 6.3(b) and Fig. 6.4(b) clearly demonstrate that MAD and our approach have almost constant mean standard deviation for the different outlier percentages. LTS and LMedS tend to deviate from their previous mean reprojection residual errors when the outlier level increases, thus making them noise sensitive. Similarly, RANSAC and MSAC clearly are not able to cope-up with the changing outlier ratios in an input image. Ours and MAD show constant plot because these algorithms remove the outliers by adaptively choosing σ using 3-σ principle, unlike other methods which use a fixed threshold for every outlier levels, thus making them unable to choose the true inliers with the varying percentages of outliers.

6.2.3 Precision

To measure the relevancy of the inliers detected by the algorithms, we need to determine a measure that accurately scores the algorithm based on a number of true inliers/outliers detected. For example, if a particular algorithm classifies a handful amount of inliers, we cannot claim the algorithm to be a reliable algorithm merely based on the number of inliers detected unless we verify the authenticity of the detected inliers with the ground truth. This is where the precision of the algorithm comes to play which carefully scores the algorithm up if the detected inliers match the ground truth and thumbs down otherwise. Let TP denote true positives (detected inliers/outliers are true inliers/outliers) and FP denote false positives (detected outliers are true inliers), then the precision of the algorithm is calculated as $\frac{TP}{TP+FP}$. Thus, it can be deduced from the definition that the higher the precision, the better the algorithmic performance.

The precisions of different algorithms are plotted in Fig. 6.3(c) and Fig. 6.4(c). It is evident that our approach has higher precision compared to the other approaches for the most of the outlier percentages. LTS tend to show the highest precision for lower levels of outliers but the precision drastically decreases in a near linear fashion for higher outlier levels. This can be accounted for the fact that as the number of outliers increases, LTS finds it hard to segregate the true inliers from outliers. This can also be observed in Fig. 6.3(a) and Fig. 6.4(a) where mean reprojection residual error for LTS gradually increases due to the fact that the outliers being detected as the inliers. MAD shows considerable precision but does not perform better than our approach. This is one of the major advantages of using RES-Q over MAD algorithm. However, we observed that MAD performs slightly better than RES-Q at higher outlier level, i.e., 25%. RANSAC and MSAC perform very similar but have lower precisions. Again, this limitation is due to having a constant threshold for every outlier levels and no adaptive outlier rejection strategy. In addition, LMedS plot shows that LMedS is not a suitable algorithm for lower outlier percentage and is not able to outperform the compared algorithms for any higher outlier percentages. It can be concluded that for a varying outlier levels, our approach is more precise to use with the least concern of detecting false positives.

6.2.4 Computational Complexity Analysis

Let **N** be the number of iterations performed, **D** be the number of data points consisting of both inliers and outliers, **F** be the number of features (for example, 8 features for estimating the fundamental matrix using the normalized eight point algorithm [12, 80]), $\mathbf{C_{fit}}$ be the time complexity for determining the fundamental matrix, $\mathbf{M_L}$ be the time complexity for maximum likelihood operations to determine the most likely set of inliers in MSAC and **T** be the computational cost for triangulation, we can compare the total worst case scenario computational costs for different algorithms. Since RANSAC fits the detected inliers in every iteration to estimate the fundamental matrix to see the fit according to a preset threshold, its complexity is $\mathbf{O}(\mathbf{N}(\mathbf{C_{fit}}+\mathbf{D}))$. MSAC performs on a common ground with RANSAC except for calculating a maximum likelihood fit for the final estimation of the inlier set, its computational complexity is $\mathbf{O}(\mathbf{N}(\mathbf{C_{fit}}+\mathbf{D})+\mathbf{M_L})$, slightly higher than RANSAC. Both LMedS and LTS have the same order of complexity $\mathbf{O}(\mathbf{NF^2D})$ because of their iterative machine learning algorithmic approach. However, MAD and RES-Q, based on the previously proposed Algorithm 1, calculate the fundamental matrix twice, performs triangulation twice and computes the reprojection residual error twice which is dependent on **D**, it has a complexity of $\mathbf{O}(\mathbf{C_{fit}}+\mathbf{DlogD}+\mathbf{T})$. A key observation here is that MAD and RES-Q have their order of complexities independent to the number of iterations **N** thus making them computationally more efficient than the classical approaches.

6.2.5 Determination of Sigma Scaling Factor

We experimented with several sigma scaling factor (σ_f) versus outlier percentages to best determine the inliers with an apt precision. Since the precision of determining as well as eliminating the outliers should be higher for an algorithm and the reprojection residual error should be lower for better performance, we define sigma selectivity as a logarithmic ratio of precision p to reprojection residual error e_{rr}. The selectiveness of σ is mathematically calculated as $\sigma_{sel} = \frac{log(p)}{log(e_{rr})}$. Fig. 6.5 and Fig. 6.6 depicts sigma selectivity against different levels of outliers. It can be observed that lower sigma scaling factors (2.0 and 2.5) has less selectivity for the most of the outlier levels. Also,

higher scaling factors (4.0 and 3.5), though has higher selectivity at lower outlier levels, quickly drops down with the increasing outlier levels. Thus, it can be safely concluded that sigma scaling factor of 3.0 is the most appropriate value as it has steady selectivity across different outlier percentages and is chosen both for symmetric and asymmetric noise in our experiments for the RES-Q approach.

6.3 Evaluations on Real Data

We extensively evaluated the RES-Q algorithm against other competing algorithms in the real data sets. For this purpose, four datasets - Lion [83], Fountain [84], Merton College I and Merton College II from Oxford Multiple View VGG dataset[1] were chosen. Lion dataset consists of 3204 matching point correspondences and 5% random outliers were added for the experimentation. Fountain dataset consists of 4219 matching point correspondences and 10% outliers were added. Similarly, Merton College I and Merton College II consists of 383 and 344 matching point correspondences and 20% and 25% outliers were added respectively. All the stereo image pairs in the datasets have a resolution of 1024 x 768 and the calibrated camera parameters are provided along with the datasets. For the generalization purpose, the algorithms were run independently for 500 times on each dataset and the results are reported.

6.3.1 Mean Reprojection Residual Error

In Fig. 6.7(a) we plot the mean reprojection residual error curve. This curve determines the number of correct inliers detected as the reprojection residual error for the true positives contributes less to the reprojection residual error, whereas, that of the false positives contributes significantly. Thus, a good outlier rejection algorithm should have lower mean reprojection residual error. In the plot, we can observe that RES-Q, LTS and MAD have considerably lower mean reprojection residual error than that of RANSAC, MSAC or LMedS. Also, RES-Q and LTS outperform MAD in most of the

[1] http://www.robots.ox.ac.uk/vgg/data/data-mview.html

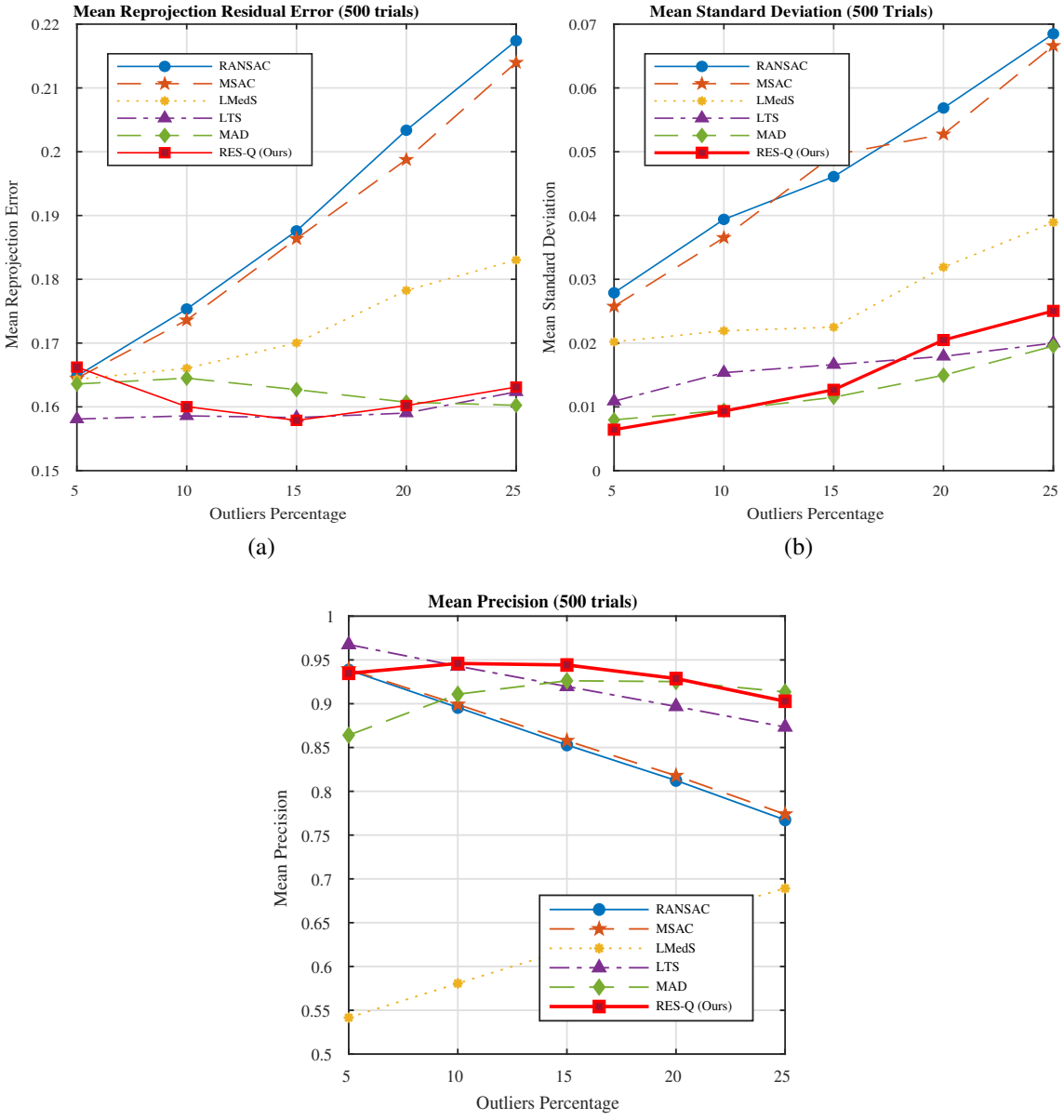


Figure 6.7: Real data experimentation plots comparing algorithms for 500 independent trials on Merton College II dataset. (a) Mean reprojection residual error. (b) Mean standard deviation of reprojection residual error. (c) Mean precision. (Best viewed in color)

outlier levels. Though LTS did not perform as good as MAD and RES-Q in the synthetic experiments, the real dataset experiments show that LTS could be a suitable choice whereas MAD may deviate from the synthetic experiment findings. Based on the experiments, we can conclude that RES-Q performs competitively better in both of the experiments, and thus, proving its robustness at several outlier levels.

6.3.2 Mean Standard Deviation of Reprojection Residual Error

To observe the noise stability, outlier level adaptability and reproducibility of the compared algorithms, the standard deviation of the reprojection residual errors over several outlier percentage values is evaluated over 500 independent trials to plot the mean standard deviation of the reprojection residual errors in Fig. 6.7(b) as explained in the synthetic experiments in the previous section. The plot clearly indicates that the mean standard deviation of the reprojection residual errors is almost constant for RES-Q, MAD and LTS compared to RANSAC, MSAC or LMedS. This observation also matches with the plots in the synthetic experiments. Though MAD and RES-Q performed almost similar in the synthetic experiments as their plots overlapped, real data experiments show that RES-Q could be unstable to noises compared to MAD and LTS for higher outlier levels.

6.3.3 Mean Precision

Precision plot over varying outlier percentages helps to determine the true analysis of the performance of the various competing algorithms. Since the precision plot carefully accounts for the ground truth match with the set of classified inliers by the various algorithms and penalizes the algorithm for every misclassification, the higher the precision curve, the more reliability of an algorithm. In the precision plot Fig. 6.7(c), RES-Q clearly outperforms all the competing algorithms. LTS, though has a slightly higher precision for the lowest outlier level (5%) (also similar findings in the synthetic experiments), the precision falls below RES-Q and MAD approaches for higher outlier levels. This can also be verified by the increasing mean reprojection residual errors for LTS

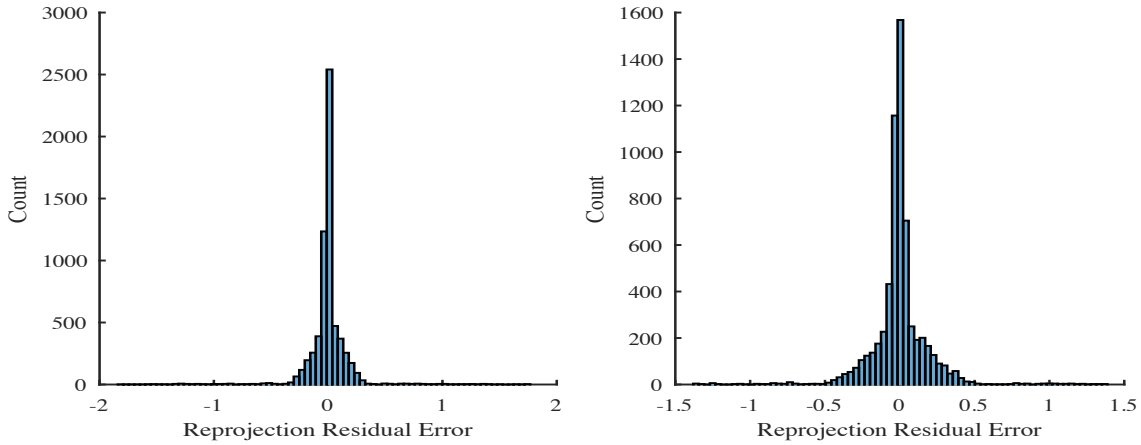


Figure 6.8: Histogram of reprojection residual error on Lion dataset with 10% outlier before (a) and after (b) RES-Q.

in Fig. 6.7(a) due to false classifications. MAD shows a good precision for higher outlier levels but is not able to perform with the utmost precision compared to LTS or RES-Q at lower outlier levels. This was also outlined in the precision plot for synthetic experiments and argued to be the key disadvantage of using MAD over RES-Q. RANSAC and MSAC due to their inadaptive outlier rejection strategy cannot outperform adaptive algorithms. LMedS does not seem to perform with much precision at any outlier levels. Thus, RES-Q is a better choice for precise outlier rejection scheme for a robust fundamental matrix estimation.

To further demonstrate the performance of RES-Q, the histogram distribution of the reprojection residual error in the Lion dataset before and after running RES-Q is plotted in Fig. 6.8. As discussed earlier, the reprojection residual errors follow Gaussian distribution and RES-Q is able to reject most of the outliers using 3-σ principle for the fundamental matrix estimation which can also be verified in Fig. 6.8.

6.3.4 Qualitative Evaluation Results

In order to further verify the robustness of our approach, different outlier levels were added to the different real datasets (5% to Lion, 10% to Fountain, 20% to Metron College I and 25% to

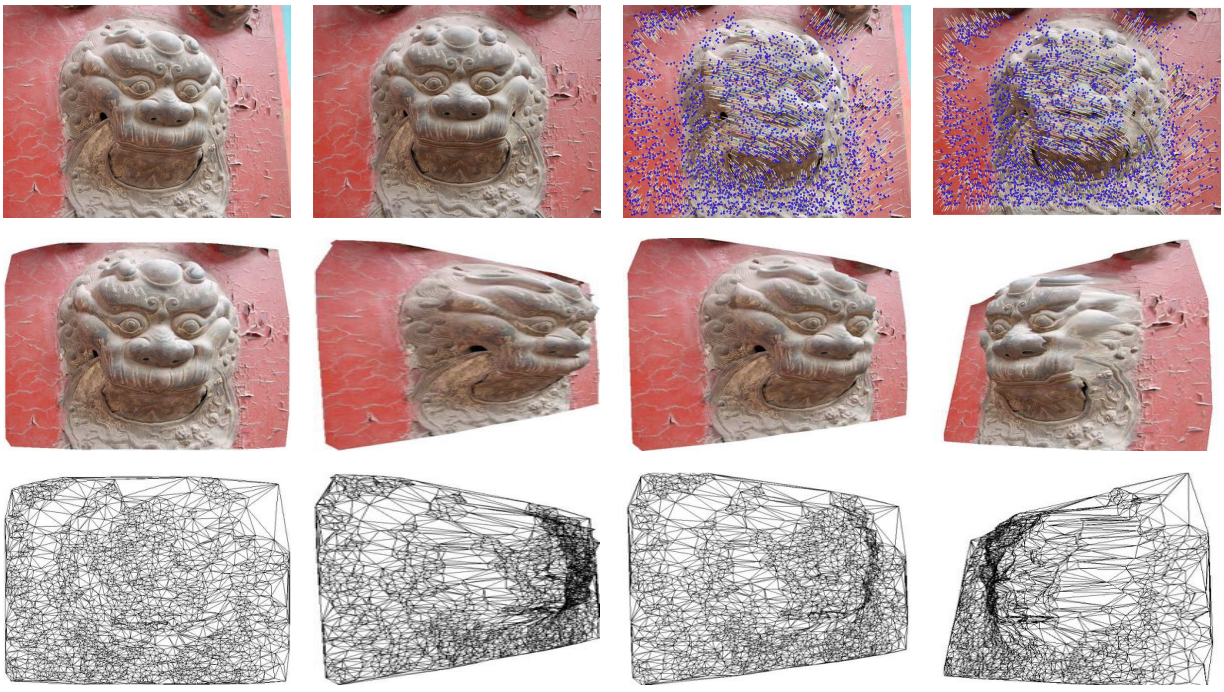


Figure 6.9: Reconstruction results of Lion dataset with 5% outlier. (First row) Left two are original stereo images and the right two are matching correspondences obtained after running through RES-Q. (Second row) the reconstructed VRML model of the scene shown from different viewpoints with texture mapping. (Last row) the corresponding triangulated wireframe of the VRML models.

Merton College II) and experimented with RES-Q to see the reconstructed VRML models and corresponding wireframes. Figs. 6.9, 6.10, 6.11 and 6.12 show that RES-Q successfully discards the outliers and recovers the Euclidean structure of the scene for different levels of outliers. As the levels of outlier increase, the VRML models show some instances of false reconstruction, however, most of the parts are accurately reconstructed and visually plausible.

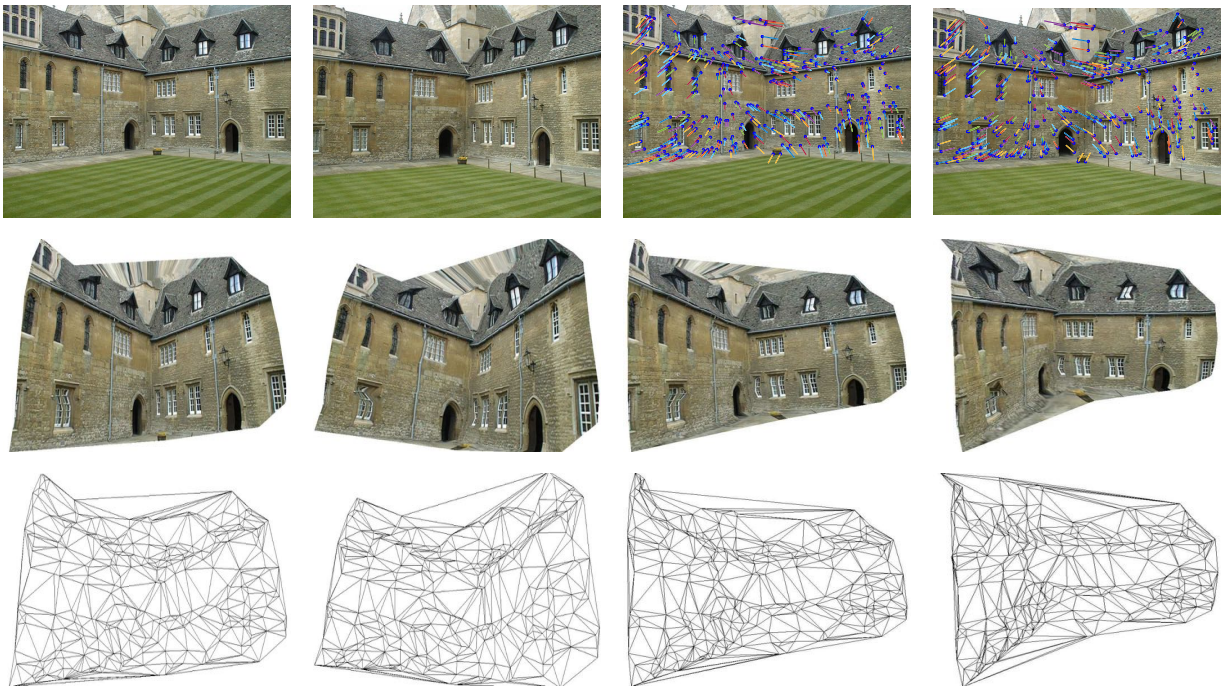


Figure 6.11: Reconstruction results of Merton Collge I dataset with 20% outlier. (First row) Left two are original stereo images and the right two are matching correspondences obtained after running through RES-Q. (Second row) the reconstructed VRML model of the scene shown from different viewpoints with texture mapping. (Last row) the corresponding triangulated wireframe of the VRML models.

Chapter 7

Conclusion and Future Work

7.1 Concludion

It is vital for an intelligent autonomous UAV to have an automatic, robust and real-time object tracking system built in it. To meet this, we discussed a tracking method that incorporates variations in shape, size, illumination as well as degenerate conditions like partial occlusion, planar/axial rotation and camera instability in it for better performance than the existing state-of-the-art trackers in this thesis. Most of the up-to-date trackers were found to fail in one or several such complex scenarios. However, our tracker is able to keep track of the object without any abrupt failures. Both qualitative and quantitative evaluation measures demonstrate that the proposed approach is more efficient than the competing trackers. Unlike other trackers, the proposed tracker is able to auto-initialize without any manual interference. Hence, our approach is found to be accurate and fast in terms of speed for realtime autonomous sense-and-avoid UAVs, drones or similar flying units. Nevertheless, some of the experiments show that our method may not perform as expected in the presence of several dubious salient objects in a scene or the object is too tiny to detect and auto-initialize the tracker. Source code and the dataset for the proposed approach can be downloaded for research purpose from the author's web page[1].

Similarly, we also emphasized on accurate fundamental matrix estimation technique for obstacle depth analysis in autonomous UAVs. We demonstrated that the robust statistics based method can be used to determine and remove the outliers present in the corresponding matching points between a pair of stereo images. Through extensive synthetic and real data experiments, we ver-

[1] http://www.ittc.ku.edu/cviu/tracking.html

ified that the reprojection residual error based technique is more robust than the algebraic error based approaches. Furthermore, we showed that Q_n estimator together with 3-σ principle in our RES-Q algorithm can efficiently detect several levels of outliers and accurately estimate the fundamental matrix for successful reconstruction of the given scene. In addition, several experiments were carried out to show that RES-Q performs equally well for both symmetric and asymmetric random noises. We also showed that one of the major advantages of using RES-Q over MAD is an improved precision which is vital in the autonomous vehicle applications like in UAVs. The greater precision of RES-Q over MAD was further verified by experimenting them with various synthetic as well as real datasets against several outlier levels.

7.2 Future Work

Though the proposed tracking by detection mechanism is robust than the state-of-the-art trackers for autonomous UAVs, there are certain limitations that could be a part of further research. For instance, the tracking by detection mechanism is unable to track multiple objects in a scene independently. Also, the tracker fails to auto-initialize in the presence of complex background as discussed in the limiation section in Chapter 4. These limitations can further be studied and explored to enhance the tracking mechanism in autonomous UAVs. Similarly, in this thesis, we presented a robust fundamental matrix estimation technique RES-Q which can be utilized for a scene reconstruction and depth analysis in autonomous UAVs. Thus, there is a possibility of integration of a robust object tracking by detection mechanism with RES-Q for depth analysis of the scene into a single pipeline which shall aid to the auto-navigation of the UAVs. This can also be extended to the object following mechanism if required.

References

[1] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm computing surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[2] Z. Kim, "Robust lane detection and tracking in challenging scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 16–26, March 2008.

[3] M. Shan, S. Worrall, and E. Nebot, "Probabilistic long-term vehicle motion prediction and tracking in large environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 539–552, June 2013.

[4] S. P. Bharati, S. Nandi, Y. Wu, Y. Sui, and G. Wang, "Fast and robust object tracking with adaptive detection," in *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*. IEEE, 2016, pp. 706–713.

[5] S. P. Bharati, Y. Wu, Y. Sui, , C. Padgett, and G. Wang, "Real-time obstacle detection and tracking for sense-and-avoid mechanism in uavs," *IEEE Transactions on Intelligent Vehicles*, Dec 2017.

[6] W. van der Mark and D. M. Gavrila, "Real-time dense stereo for intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 38–50, March 2006.

[7] H. Y. Cheng, B. S. Jeng, P. T. Tseng, and K. C. Fan, "Lane detection with moving vehicles in the traffic scenes," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 4, pp. 571–582, Dec 2006.

[8] A. De la Escalera, J. M. Armingol, and M. Mata, "Traffic sign recognition and analysis for intelligent vehicles," *Image and vision computing*, vol. 21, no. 3, pp. 247–258, 2003.

[9] F. Xu, X. Liu, and K. Fujimura, "Pedestrian detection and tracking with night vision," *IEEE Transactions on Intelligent Transportation Systems*, vol. 6, no. 1, pp. 63–71, March 2005.

[10] U. Handmann, T. Kalinke, C. Tzomakas, M. Werner, and W. Seelen, "An image processing system for driver assistance," *Image and Vision Computing*, vol. 18, no. 5, pp. 367–376, 2000.

[11] L. Itti and C. Koch, "Computational modelling of visual attention," *Nature reviews neuroscience*, vol. 2, no. 3, pp. 194–203, 2001.

[12] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[13] J. Zhang, S. Sclaroff, Z. Lin, X. Shen, B. Price, and R. Mech, "Minimum barrier salient object detection at 80 fps," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1404–1412.

[14] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[15] Y. Li and J. Zhu, "A scale adaptive kernel correlation filter tracker with feature integration," in *European Conference on Computer Vision*. Springer, 2014, pp. 254–265.

[16] P. J. Rousseeuw and C. Croux, "Alternatives to the median absolute deviation," *Journal of the American Statistical association*, vol. 88, no. 424, pp. 1273–1283, 1993.

[17] M. Zhang, G. Wang, H. Chao, and F. Wu, "Fast and robust algorithm for fundamental matrix estimation," in *International Conference Image Analysis and Recognition*. Springer, 2015, pp. 316–322.

[18] S. P. Bharati, A. Sharda, and G. Wang, "Res-q: Robust outlier detection algorithm for fundamental matrix estimation," *IEEE Transactions on Intelligent Vehicles*, 2018.

[19] H. Cholakkal, D. Rajan, and J. Johnson, "Top-down saliency with locality-constrained contextual sparse coding." BMVC, 2015.

[20] J. Yang and M.-H. Yang, "Top-down visual saliency via joint crf and dictionary learning," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 2296–2303.

[21] A. Kocak, K. Cizmeciler, A. Erdem, and E. Erdem, "Top down saliency estimation via superpixel-based discriminative dictionaries." in *BMVC*, 2014.

[22] Y. Wei, F. Wen, W. Zhu, and J. Sun, "Geodesic saliency using background priors," in *European Conference on Computer Vision*. Springer, 2012, pp. 29–42.

[23] S. He, R. W. Lau, W. Liu, Z. Huang, and Q. Yang, "Supercnn: A superpixelwise convolutional neural network for salient object detection," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 330–344, 2015.

[24] P. Wang, J. Wang, G. Zeng, J. Feng, H. Zha, and S. Li, "Salient object detection for searched web images via global saliency," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 3194–3201.

[25] R. Zhao, W. Ouyang, H. Li, and X. Wang, "Saliency detection by multi-context deep learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1265–1274.

[26] W. Zhu, S. Liang, Y. Wei, and J. Sun, "Saliency optimization from robust background detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2814–2821.

[27] J. Feng, Y. Wei, L. Tao, C. Zhang, and J. Sun, "Salient object detection by composition," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 1028–1035.

[28] P. Siva, C. Russell, T. Xiang, and L. Agapito, "Looking beyond the image: Unsupervised learning for object saliency and detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 3238–3245.

[29] S. Avidan, "Ensemble tracking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 2, pp. 261–271, 2007.

[30] B. Babenko, M.-H. Yang, and S. Belongie, "Visual tracking with online multiple instance learning," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 983–990.

[31] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *European Conference on Computer Vision*. Springer, 2012, pp. 864–877.

[32] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. Hicks, and P. Torr, "Struck: Structured output tracking with kernels," 2015.

[33] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[34] A. Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes," *Advances in neural information processing systems*, vol. 14, p. 841, 2002.

[35] Y. Wu, Y. Sui, and G. Wang, "Vision-based real-time aerial object localization and tracking for uav sensing system," *IEEE Access*, vol. 5, pp. 23 969–23 978, 2017.

[36] C. Ma, X. Yang, C. Zhang, and M.-H. Yang, "Long-term correlation tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5388–5396.

[37] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2544–2550.

[38] Y. Sui, G. Wang, and L. Zhang, "Correlation filter learning toward peak strength for visual tracking," *IEEE Transactions on Cybernetics*, vol. PP, no. 99, pp. 1–14, 2017.

[39] Y. Sui, Z. Zhang, G. Wang, Y. Tang, and L. Zhang, "Real-time visual tracking: Promoting the robustness of correlation filter learning," in *European Conference on Computer Vision*. Springer, 2016, pp. 662–678.

[40] Y. Sui, Y. Tang, L. Zhang, and G. Wang, "Visual tracking via subspace learning: A discriminative approach," *International Journal of Computer Vision*, pp. 1–22, 2017.

[41] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "Exploiting the circulant structure of tracking-by-detection with kernels," in *European conference on computer vision*. Springer, 2012, pp. 702–715.

[42] Y. Wu, J. Lim, and M.-H. Yang, "Online object tracking: A benchmark," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 2411–2418.

[43] M. Danelljan, G. Häger, F. Khan, and M. Felsberg, "Accurate scale estimation for robust visual tracking," in *British Machine Vision Conference, Nottingham, September 1-5, 2014*. BMVA Press, 2014.

[44] T. Liu, G. Wang, and Q. Yang, "Real-time part-based visual tracking via adaptive correlation filters," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4902–4912.

[45] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[46] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3074–3082.

[47] H. Li, Y. Li, and F. Porikli, "Deeptrack: Learning discriminative feature representations online for robust visual tracking," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1834–1848, 2016.

[48] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[49] O. Chum and J. Matas, "Matching with prosac-progressive sample consensus," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 220–226.

[50] P. H. Torr and A. Zisserman, "Mlesac: A new robust estimator with application to estimating image geometry," *Computer Vision and Image Understanding*, vol. 78, no. 1, pp. 138–156, 2000.

[51] P. H. Torr, "Bayesian model estimation and selection for epipolar geometry and generic manifold fitting," *International Journal of Computer Vision*, vol. 50, no. 1, pp. 35–61, 2002.

[52] B. Tordoff and D. Murray, "Guided sampling and consensus for motion estimation," *Computer Vision—ECCV 2002*, pp. 82–96, 2002.

[53] R. Raguram, J.-M. Frahm, and M. Pollefeys, "A comparative analysis of ransac techniques leading to adaptive real-time random sample consensus," *Computer Vision–ECCV 2008*, pp. 500–513, 2008.

[54] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. John wiley & sons, 2005, vol. 589.

[55] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial intelligence review*, vol. 22, no. 2, pp. 85–126, 2004.

[56] C. Leys, C. Ley, O. Klein, P. Bernard, and L. Licata, "Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median," *Journal of Experimental Social Psychology*, vol. 49, no. 4, pp. 764–766, 2013.

[57] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[58] C.-B. Xiao, D.-Z. Feng, and M.-D. Yuan, "An efficient fundamental matrix estimation method for wide baseline images," *Pattern Analysis and Applications*, pp. 1–10, 2016.

[59] F. Shen, W. Yang, H. Li, H. Zhang, and H. T. Shen, "Robust regression based face recognition with fast outlier removal," *Multimedia Tools and Applications*, vol. 75, no. 20, pp. 12 535–12 546, 2016.

[60] X. Wang, D. Shen, M. Bai, T. Nie, Y. Kou, and G. Yu, "Cluster-based outlier detection using unsupervised extreme learning machines," in *Proceedings of ELM-2015 Volume 1*. Springer, 2016, pp. 135–146.

[61] M. Salehi, X. Zhang, J. C. Bezdek, and C. Leckie, "Smart sampling: A novel unsupervised boosting approach for outlier detection," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2016, pp. 469–481.

[62] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *SIAM Conference on Data Mining*, 2017.

[63] P. H. Torr and D. W. Murray, "The development and comparison of robust methods for estimating the fundamental matrix," *International journal of computer vision*, vol. 24, no. 3, pp. 271–300, 1997.

[64] R. Maronna, R. D. Martin, and V. Yohai, *Robust statistics*. John Wiley & Sons, Chichester. ISBN, 2006.

[65] A. Rosenfeld and J. L. Pfaltz, "Distance functions on digital pictures," *Pattern recognition*, vol. 1, no. 1, pp. 33–61, 1968.

[66] K. C. Ciesielski, R. Strand, F. Malmberg, and P. K. Saha, "Efficient algorithm for finding the exact minimum barrier distance," *Computer Vision and Image Understanding*, vol. 123, pp. 53–64, 2014.

[67] R. Strand, K. C. Ciesielski, F. Malmberg, and P. K. Saha, "The minimum barrier distance," *Computer Vision and Image Understanding*, vol. 117, no. 4, pp. 429–437, 2013.

[68] Y. Zhai and M. Shah, "Visual attention detection in video sequences using spatiotemporal cues," in *Proceedings of the 14th ACM international conference on Multimedia*. ACM, 2006, pp. 815–824.

[69] R. Rifkin, G. Yeo, T. Poggio *et al.*, "Regularized least-squares classification," *Nato Science Series Sub Series III Computer and Systems Sciences*, vol. 190, pp. 131–154, 2003.

[70] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[71] R. Strand, K. C. Ciesielski, F. Malmberg, and P. K. Saha, "The minimum barrier distance," *Computer Vision and Image Understanding*, vol. 117, no. 4, pp. 429–437, 2013.

[72] N. Otsu, "A threshold selection method from gray-level histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975.

[73] R. M. Gray, *Toeplitz and circulant matrices: A review*. now publishers inc, 2006.

[74] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press, 2001.

[75] R. Rifkin, G. Yeo, and T. Poggio, "Regularized least-squares classification," *Nato Science Series Sub Series III Computer and Systems Sciences*, vol. 190, pp. 131–154, 2003.

[76] A. Li, M. Lin, Y. Wu, M. Yang, and S. Yan, "NUS-PRO: A New Visual Tracking Challenge," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 335–349, 2016.

[77] K. Zhang, L. Zhang, and M.-H. Yang, "Real-time compressive tracking," in *European Conference on Computer Vision*. Springer, 2012, pp. 864–877.

[78] K. Zhang, L. Zhang, Q. Liu, D. Zhang, and M.-H. Yang, "Fast visual tracking via dense spatio-temporal context learning," in *European Conference on Computer Vision*. Springer, 2014, pp. 127–141.

[79] M. Danelljan, F. Shahbaz Khan, M. Felsberg, and J. Van de Weijer, "Adaptive color attributes for real-time visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1090–1097.

[80] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 19, no. 6, pp. 580–593, 1997.

[81] R. I. Hartley and P. Sturm, "Triangulation," *Computer vision and image understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[82] G. Wang, J. S. Zelek, J. Wu, and R. Bajcsy, "Robust structure from motion in the presence of outliers and missing data," *arXiv preprint arXiv:1609.02638*, 2016.

[83] G. Wang and Q. J. Wu, *Guide to three dimensional structure and motion factorization*. Springer, 2011.

[84] G. Wang and Q. M. J. Wu, "Perspective 3-d euclidean reconstruction with varying camera parameters," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 19, no. 12, pp. 1793–1803, Dec 2009.