Towards efficient search methods in object tracking: An evaluation and application to precise tracking

by

Ankush Roy

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

Object tracking is a much researched subject in the computer vision community. With more and more tracking algorithms reported every year, standard benchmarking and evaluation methods are reported for long term tracking systems.

In this thesis we present a public dataset to evaluate trackers used for human and robot manipulation tasks. For these tasks high degrees of freedom (DOF) motion of the object is to be tracked with high accuracy. We describe in detail, both the process of recording the sequences and how ground truth data was generated for the videos. As an initial example, we evaluate the performance of seven published trackers [42, 67, 57, 23, 6, 9, 82] and analyze the results. We describe a new evaluation metric to test sensitivity of trackers to speed. A total of 100 annotated and tagged sequences are reported. All the videos, ground truth data, tagged image frames, original implementation of trackers and evaluation scripts are made publicly available.

We also introduce a new search method in tracking. Sequential Graph based Approximate Nearest Neighbour Search algorithm [70, 33] or SGANNS. It uses overlapping image features in videos to build a connected graph, offline. This graph is then searched efficiently during tracking to predict the best warp parameters. We test this algorithm on the dataset reported and further analyze the results.

Finally we show that using a detection module, registration based trackers can be made more robust. We address tracking challenges of occlussion and varying appearance where a regular registration based tracker fails to track.

Imagination is more important than knowledge. For knowledge is limited to all we now know and understand, while imagination embraces the entire world, and all there ever will be to know and understand.

– Albert Einstien

Acknowledgements

I would like to thank my supervisor Martin Jägersand, for guiding me throughout this journey and allowing flexibility to explore a variety of topics of my interest. His keen insights and suggestions helped me on numerous occasions, especially during the tough times. I would also like to thank my committe members Dana Cobzas and Nilanjan Ray for their suggesstions and insights to improve my thesis.

A special thanks to Xi Zhang, Abhineet Singh and Kiana Hajebi. I thoroughly enjoyed our discussions on research and life in general. Hope to have more in future. I would also like to thank my colleagues in the Computer Vision and Robotics Labs here in University of Alberta, for the various helpful discussions we had over the years.

I would also like to thank my friends - Sriram, Camilo, Ujjwal, Martha and Shiva for the fun and support. I hope it continues in future too. Finally I would like to thank my parents for their continuous love and support.

Contents

List of Tables

List of Figures

Nomenclature

$\mathbf{I}^*$	Template image
E_{AL}	Alignment error
X_{GT}	Four corner ground truth
A_{GT}	Area of ground truth
X_{GT}	Four corner tracked result
X_{GT}	Area of tracked result
OS	Overall Success
AD	Average Drift
$\mathbf{k}$	Degree of connectivity of graph
$\mathbf{H}$	Hessian
$\mathbf{J}$	Jacobian
$\mathbf{W}$	Warping function
$\mathbf{G}$	Connected graph
OLT	Online Learned Trackers
$\mathbf{p}$	State parameter
$\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))$	Warped image
$\mathcal{SSD}$	Sum of Squared Differences
$\mathcal{SCV}$	Sum of Conditional Variances
$\mathcal{IVA}$	Illumination Invariance Appearance
$\mathcal{NCC}$	Normalized Cross Correlation
t_p	Threshold for successful tracking

Chapter 1

Introduction

An important component of system(s) that rely on estimating the trajectory of an object in a continuous stream of images is *Object Tracking*. Applications like automated surveillance [39], targeting systems [60], robot manipulation [23], augmented reality [49], user interface design [15], image stabilization [80], medical analysis [72], video compression [22] and many others directly benefit from advances in tracking research.

2D Object tracking is challenging for several reasons. One may need to deal with complexities such as noise in images from camera sensors, varying illumination, low texture of the object, loss of information in two dimensional (2D) images of original three dimensional (3D) scenes, occlusion of the object, blur at high speed object motion and complex motion. An ideal tracker should be both robust to these challenges and precise in predicting the pose (position and orientation, as the case might be) of the object. Two distinct categories are observed in tracking research. (1) Trackers that are very precise [52, 32, 6, 23, 9], that are used in manipulation and servoing tasks, (2) trackers that adapt to the changing appearance of the object over time [42, 67, 57], thus focussing more on robustness, and are used in surveillance tasks. The former are referred to as registration based trackers as the core algorithm works around registering the image to the target template to find the best match. The latter is known as Online Learned Trackers (OLT), since the trackers learn and update the appearance of the object(s) online as tracking goes on.

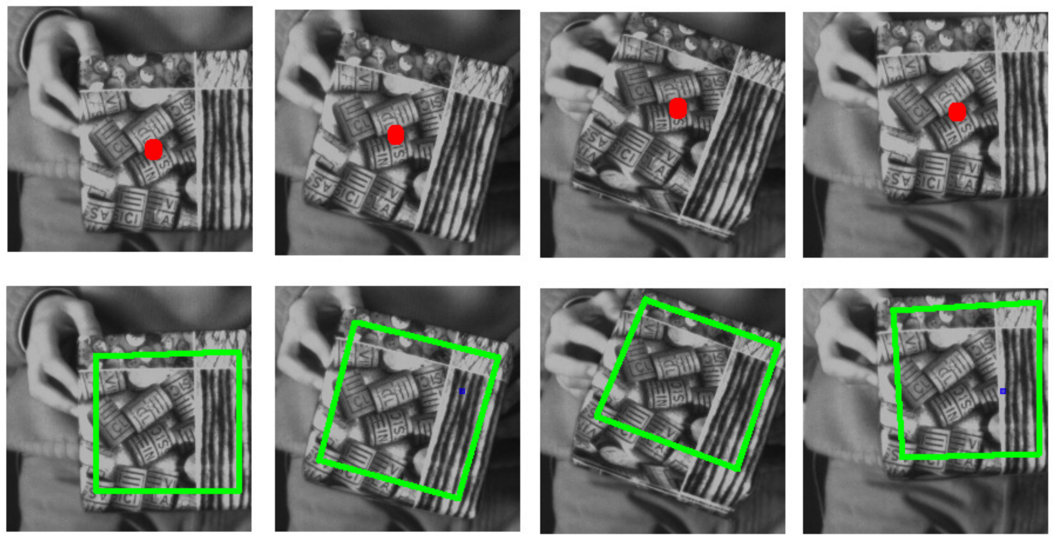


Figure 1.1: [TOP ROW] Example of an Online Learned Tracker, tracking (position, x,y centroid co-ordinates) in TLD [42] [BOTTOM ROW] Precise registration based tracker tracking full pose (4 corners) in IC [6]

Figure 1.1 shows the two categories discussed, on a video sequence from [9] [1]. Inverse Compositional tracker (IC) [6] tracks the full pose (precise bounding box represented by the green rectangle) of the object. Tracking Learning Detection (TLD) [42] tracker on the other hand roughly follows the centroid of the object shown by the red dot.

Object tracking is defined as a sequential state prediction problem. State of an object ($\mathbf{p_t}$) at time t is a vector that represents pose (location and orientation) of the object. Based on how precisely a tracker tracks, size of the state vector (DOF - Degrees Of Freedom) varies. Precise trackers in most cases track full pose (8 DOF, [52, 32, 6, 9, 23]) transformation of the object compared to OLT, 3 in [42, 5], 6 in [67, 57]. OLT update the appearance model of the object, taking into account newer templates, as more appearances are exposed. This process is computationally expensive, hence in-spite of tracking a low DOF state, they are mostly subpar with real time requirements.

In this thesis I focus on precise tracking, trackers that converge within sub-pixel accuracy and can be used in manipulation tasks.

[1] Images taken from [9] Available at: http://esm.gforge.inria.fr/ESMdownloads.html

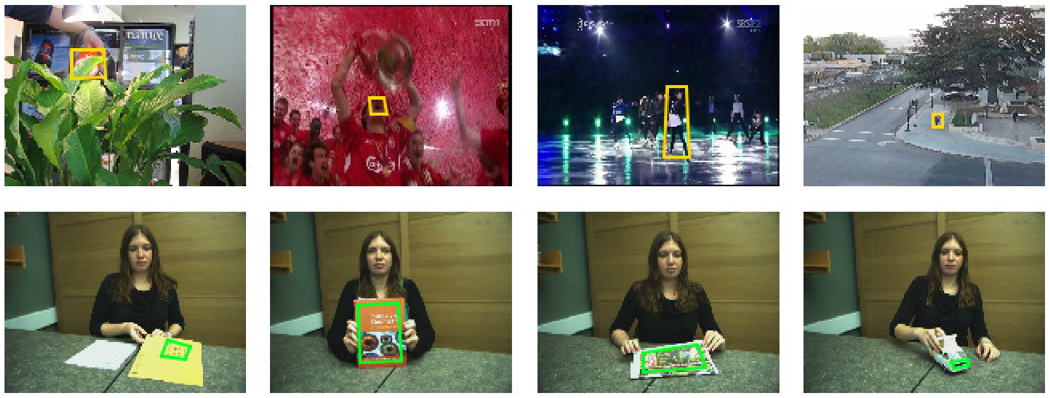


Figure 1.2: Example sequences show the two different sets of challenges that each of the tracking modalities (OLT and Registration based) test in a tracker. [TOP] IVT [67]) an OLT tracker is shown to track videos that has high occlusion, unstructured motion and rapid change of appearance, but they need not follow the object precisely. [BOTTOM] Registration tracker in ESM [9] precisely tracks the high DOF motion of the object.

1.1 Motivation

Any algorithm when reported, needs proper evaluation to bring out the full contribution and/or limits of the algorithm. Tracking research being divided, has different challenge sets that each of the sub groups (OLT and registration based) are interested in. This makes homogenous evaluation of tracking difficult. There are no common grounds to evaluate trackers from the two categories, in terms of the videos and evaluation metrics. Figure 1.2 show few example sequences used to evaluate OLT trackers [2] and registration based trackers [3] showing how they are different in their focus.

Lately, there has been much focus on evaluating OLT trackers [78, 46, 73], which over a period of time has evolved into making the evaluation process reach a set standard on a wide range of videos. This has been mostly missing in registration based trackers. Most published methods have been tested

[2]Available at: `http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html`

[3]Available at: http://webdocs.cs.ualberta.ca/vis/trackDB/

only on a few anecdotal videos. They are however limited in the number of challenges they present (Chapter 2). To the best of our knowledge Metaio [48] and UCSB [27] are the only two works that report a structured dataset and how to use it. We show in Chapter 2 how their setup is limited. This motivated us [4] to benchmark the evaluation process for registration based tracker by recording a diverse dataset and carefully choose the evaluation metrics. This would be relevant for all research groups working on registration based trackers.

One of the difficult problems in the domain of registration based is to track fast object motion maintaining the high precision required. Most search methods [52, 6] assume very little change in object pose, which is critical to their formulation. This restrict their application to a small set of object motion. Travis et al. in [23] showed that with Approximate Nearest Neighbour search using randomized KD-Trees this problem can be handled in a better way than existing techniques. We provide an improvement over this in terms of using the sequential property of video data with a Graph based Approximate Nearest Neighbour search method [33]. We evaluate this method on the proposed dataset, and compare it with existing trackers [6, 9, 23, 42, 67, 57, 82].

1.2 Contribution and Thesis Organization

We report a dataset to benchmark 2D object tracking algorithms. Furthermore, a new search method in tracking is reported with detailed results computed and analysed. The salient contributions in this thesis are:

- A publicly available dataset to evaluate registration based trackers. Evaluation scripts are made available. [5]
- A new evaluation metric, that tests a tracker's robustness against large object motions. [6]

[4]In collaboration with Xi Zhang, Nina Wolleb and Camilo Perez

[5]In collaboration with Nina Wolleb and Camilo Perez

[6]In collaboration with Xi Zhang

- A new search method in registration based tracking that uses the temporal coherency of video data.

Object tracking is broken into component sub-modules of appearance model, search method and state space. Each sub-module is further enumerated and a contribution is shown in search methods. Tracking is then evaluated on a dataset we propose. The dataset is carefully recorded to capture challenges of an object tracker used in a manipulation setup. Finally, an application of object detection and localization is shown in conjugation with object tracking to make it robust to challenges like occlusion and appearance variations.

- Chapter 2 lists existing tracking methods in literature. The second half of the Chapter provide a detailed summary of existing datasets and benchmarking methods.

- Chapter 3 discusses how a tracking algorithm can be broken down into subsequent components and individual components studied. Here we present a new search method that can be applied to registration based tracking and show how it's suited for tracking application, given the sequential nature of a video sequence.

- Chapter 4 describes in detail our setup for recording the videos and how we generate ground truth data for all videos. Furthermore, we describe our error metric and evaluation methods, highlighting their relevance in evaluating a registration tracker.

- Chapter 5 presents the evaluation results of the tracker we develop in light of the dataset reported. We present a global ranking of all trackers we study.

- Chapter 6 shows an application of using a simple detection technique on top of the registration trackers to make trackers robust to occlusion without compromising convergence.

- Chapter 7 concludes the work highlighting the significant contributions in the thesis and how it can be further extended to cover more challenges in the dataset.

Chapter 2

Background

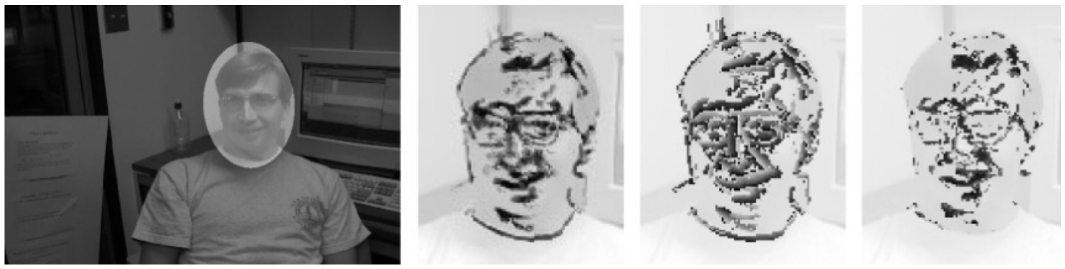
In this chapter we discuss different existing tracking algorithms. We summarize years of tracking literature and group them into different categories. The second half of this Chapter deals with details of existing benchmarking techniques for Object Trackers, their usability and limitations to evaluate registration based trackers.

2.1 Tracking Algorithms

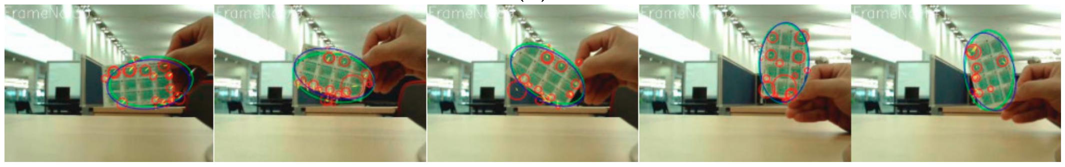
Visual tracking is the process of repeated estimation of the state of an object (position and orientation) in an image, given state(s) in previous frame(s). Approaches from different domains of computer vision have been adapted to track objects. Some of the existing works can be grouped into four major categories which are (i) **Feature Based**, (ii) **Segmentation Based**, (iii) **Detection Based** and (iv) **Registration Based**.

2.1.1 Feature Based

In feature based tracker, the object to be tracked is represented as a collection of features. Some of the popular features used in tracking are intensity based [52, 6, 9], color [56, 18, 56], steerable filter responses in [40] (Figure 2.1 (a)) and [45], optical flow features in [7], eigen features [77, 11], integral histogram in [2]. Scale invariant features like SIFT [51] and SURF [37] (Figure 2.1 (b)) have also been shown to work in tracking. Sometimes, a combination of these features is also utilized to improve tracking performance.



(a)

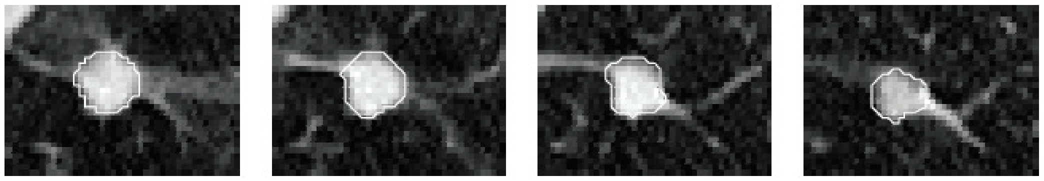


(b)

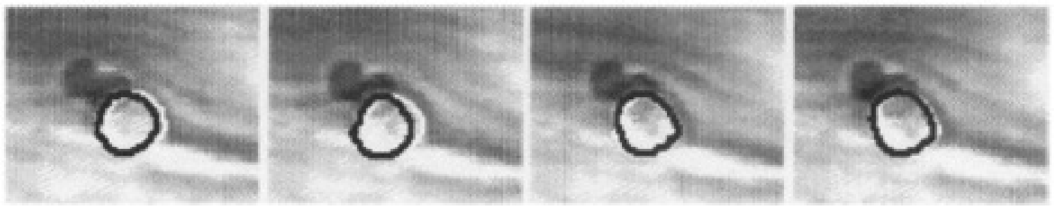
Figure 2.1: (a) Tracking is done using filter responses from a steerable pyramid using mixture model to fit data. Image taken from [40] show the tracked region and the model's mixing probability, mean, and ownership (left to right). (b) Tracking using SURF features. Images taken from [37] show the extracted features on the image frames and how the tracker follows the object in the video.

2.1.2 Segmentation Based

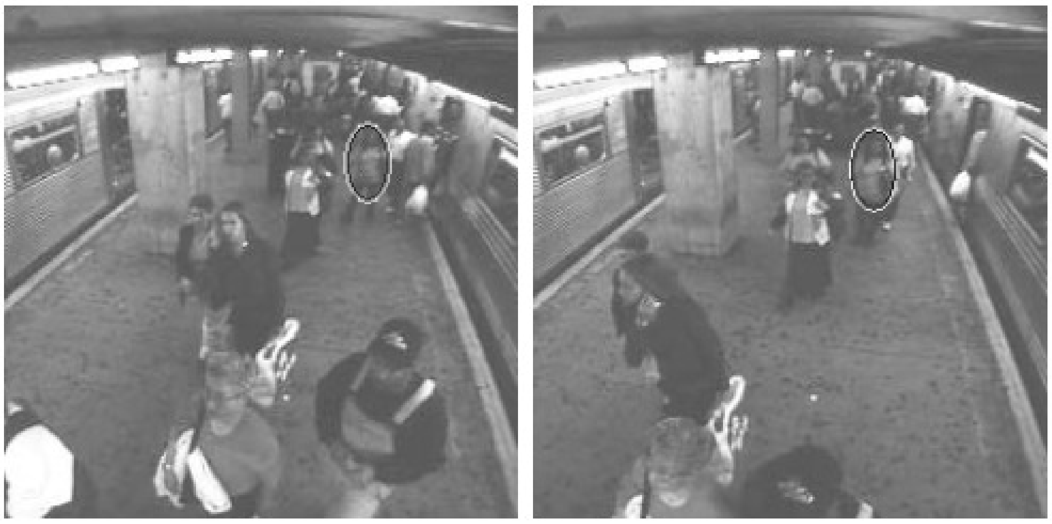
Segmentation algorithms partition the image into similar regions. This can be adapted to be used in tracking, by segmenting the object from the background. Naturally, object clustering approaches [83] translate well into video tracking. Approaches such as, mean shift [17] (Figure 2.2 (c)), normalised cross cut [79] (Figure 2.2 (a)) active contour [65] (Figure 2.2 (b)) and [81, 10] have been shown to work well in tracking. Segmentation based trackers in most cases provide pixel level accuracy [43] of the moving object by marking the exact pixel co-ordinates of the object boundary.



(a)



(b)

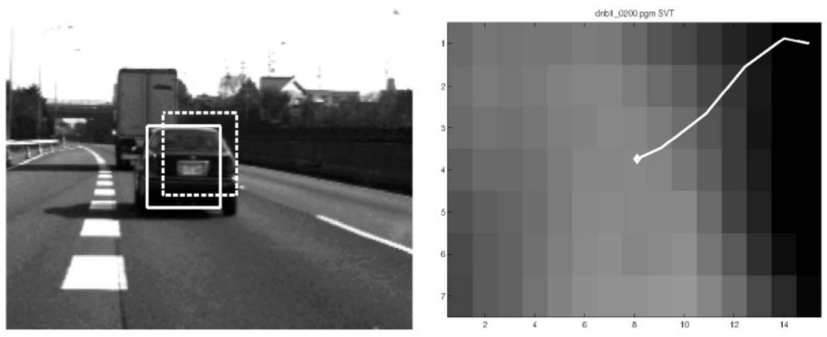


(c)

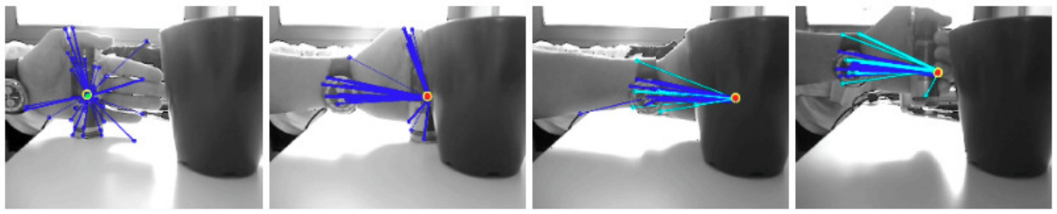
Figure 2.2: (a) Leukocytes are tracked in [79] (b) Snake based tracking as shown in [65]. The white boundary is the starting position of the snake where it evolves and the final black boundary is the new snake in the current frame (c) Mean Shift tracker in [17] is shown tracking pedestrians in a sub way system

2.1.3 Detection Based

Detection based trackers, use an object detection module together with a tracker to aid long term robust tracking. The detection module localises the object when the tracker fails to converge [5, 42] and subsequently reinitialises the tracker. Another way to use detection, is to use frame to frame detection [55]. Discriminative classifiers in SVM [4] (Figure 2.3 (a)), graph based pruning [34], kernelized structured output support vector machine [35], online boosting [29, 30] (Figure 2.3 (b)) is used to give a binary decision of object/background. Such systems rely on having object models that are either static or dynamic. Detection is done based on this model.



(a)

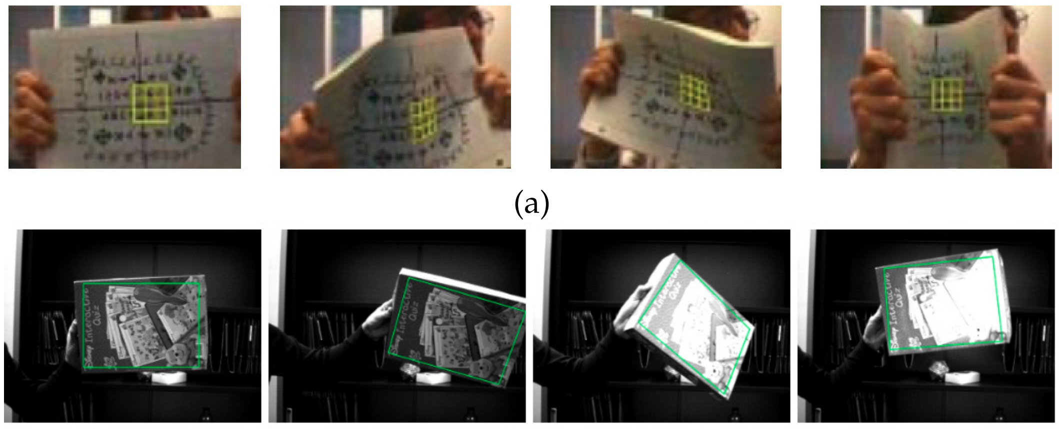


(b)

Figure 2.3: (a) Tracking using Support Vector Machine is done in [4]. The best fitting bounding box is searched close to the last known co-ordinates which eventually converge as shown in the right image. (b) Detected features are used to vote for the object position in [30]. The blue lines show the supporting features used to predict position.

2.1.4 Registration Based

First introduced by Lucas and Kanade in [52], registration based trackers use image alignment, to find the best possible warp parameters, that when applied to the object image ($\mathbf{I}$) closely resembles the template ($\mathbf{I}^*$). Gradient descent is the most commonly used method to acheive this among others like difference decomposition [28] (Figure 2.4 (a)) and linear regression [19]. Different image comparision metrics like sum of squared difference [52, 6], mutual information [21, 24], sum of conditional variance in [66] (Figure 2.4 (b)) and [23], normalised cross correlation [79] are used to compare the object image with the original template.



(a)

(b)

Figure 2.4: (a) Piecewise projective transformation is done in [28]. This allows the algorithm to track flexible objects as shown in the figure above (b) Sum of Conditional Variance (SCV) is used with ESM tracker in [66]. SCV corrects for illumination variation which allows it to track objects even in the presence of specular reflection.

In this thesis we focus mainly on registration based trackers. We show how they are more suitable for use in fine alignment tasks compared to some of the other trackers. To do this, we need to define an evaluation method and record videos that simulate fine alignment tasks.

2.2 Datasets and Evaluation

An important part of tracking research is to evaluate trackers using a standard set of videos that are carefully recorded to present a wide spectrum of challenges. Some of the existing datasets in literature are summarised in Table 2.1.

Table 2.1: Comparision of Different Object Tracking Datasets

	Geometry				**Appearance**				
Dataset	**TR**	**RO**	**PR**	**SC**	**SR**	**IL**	**TX**	**OC**	**DOF**
Metaio [48]	✓	✓	✓	✓	✗	✓	✓	✓	8
Static [23, 6, 23]	✓	✓	✓	✓	✗	✗	✗	✗	8
Zimmermann [84]	✓	✓	✓	✓	✗	✗	✗	✓	6
OLT Benchmark [78]	✓	✓	✗	✓	✓	✓	✓	✓	2
Gauglitz [27]	✓	✓	✓	✓	✓	✓	✓	✗	8
ESM [9]	✓	✓	✓	✓	✗	✗	✗	✗	8
BoBot [44]	✓	✓	✗	✓	✓	✓	✓	✓	2
VOT Challenge [46]	✓	✓	✗	✓	✓	✓	✓	✓	2
MOT Challenge [47]	✓	✓	✗	✓	✓	✓	✓	✓	2
Smuelders et al. [73]	✓	✓	✗	✓	✓	✓	✓	✓	2
KTH Dataset [3]	✓	✓	✓	✓	✓	✓	✓	✓	6
TMT [68]	✓	✓	✓	✓	✓	✓	✓	✓	8

TR = Translation; RO = Rotation; IL = Illumination; PR = Perspective; SR = Specular; SC = Scale; OC = Occlusion; TX = Texture; DOF = Degrees of Freedom of ground truth data

2.2.1 Anecdotal Video Datasets

Previous works either use synthetically generated data by applying random warps [41, 6] to a template followed by tracking it, or test algorithms on a small set of videos that are either recorded or pooled from the Internet. Zimmerman et al. [84] tested their algorithm on 3 grayscale videos (linTrack) that covers 6 DOF transformation of the object. They manually annotate the bounding box using either a special marker or visual texture based cues. [50] and [63] showed application of template tracking in augmented reality, but restricted themselves to a small test set when evaluating their tracking system.

Other popular sources of videos in tracking literature are by Babenko et al. in [5] [1], Ross et al. in [67] [2], Kalal et al. [42] [3], Collins et al. in PETS2005 [16] [4], Klein et al. in [44] [5] and Fisher et al. in CAVIAR [26] [6]. These datasets have sequences to test tracking algorithms that are mainly developed for surveillance applications where loosely following ($> 50\%$ area overlap of the target with ground truth) the object is good enough for tracking to succeed. The main focus is on testing trackers for occlusion, out of frame motion and large appearance variations.

2.2.2 Full Video Datasets

PETS2005 [16] made initial efforts of creating a comprehensive dataset having wide range of challenges, but as tracking research evolved it needed more diverse set of videos to test trackers. In a recent paper Wei et al. 2013 [78] [7] categorized some of the above mentioned video sequences to publish

[1]Available at: `http://vision.ucsd.edu/~bbabenko/project_miltrack.html`

[2]Available at: `http://www.cs.toronto.edu/~dross/ivt/`

[3]Available at: `http://personal.ee.surrey.ac.uk/Personal/Z.Kalal/TLD/TLD_dataset.ZIP`

[4]Available at: `http://vision.cse.psu.edu/data/vividEval/main.html`

[5]Available at: `http://www.iai.uni-bonn.de/~kleind/tracking/`

[6]Availabe at: `http://homepages.inf.ed.ac.uk/rbf/CAVIAR/`

[7]Available at: `https://sites.google.com/site/trackerbenchmark/benchmarks/v10`

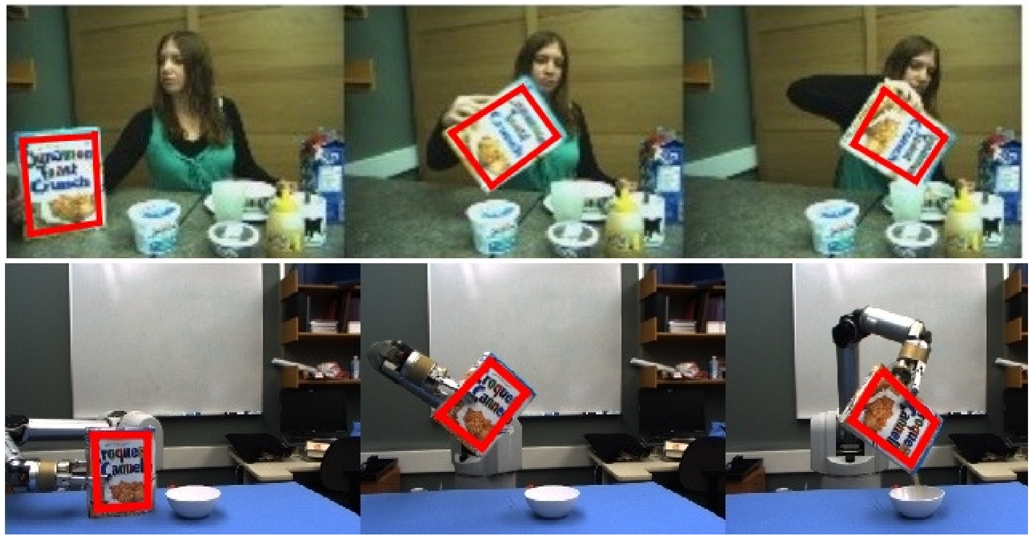


Figure 2.5: Sequence showing both the robot and human user performing identical tasks of pouring cereal in a bowl under normal light settings. The red rectangle shows the tracking result on the sequence using ESM [9]

a common benchmark [8]. Any new tracker can be evaluated and compared with other state of the art trackers in the literature using this benchmark. VOT [9] challenge [46] has a running catalog of videos and evaluation metrics by which they benchmark a new tracker. They reported an unique way to globally rank trackers based on how each tracker perform on individual metrices of rubustness, accuracy etc. Each frame in VOT is labelled with the challenges it presents. This makes it easy for a user to asses performance of the tracker, which might be suseptible to some specific challenges. MOT [47] host a similar competition for multiple object tracking in association with Winter conference on Application of Computer Vision (WACV). [46] proposes an unique way of globally ranking trackers. However, these video sequences are more suited to surveillance tracking. Furthermore, we elaborate why the error metrics used and the evaluation thresholds set [78, 46] are too coarse and ill suited for manipulation tasks.

Table 2.1 shows how diverse the popular datasets are in the set of chal-

[8]Available at: `http://cvlab.hanyang.ac.kr/tracker_benchmark/datasets.html`

[9]Available at: `http://www.votchallenge.net`

lenges they provide. It also brings out how little attention has been paid to benchmark registration based trackers with recent works mostly focussing on evaluating Online Learned Trackers (OLT).

Closest to our aim use to be Metaio dataset [10] meant to evaluate planar homography tracking by Lieberknecht et al. [48]. They collected videos under various motions, illuminations and textures. They used a camera mounted on a Faro arm (a passive robot arm with very high precision joint encoders) that was calibrated and all transformations of pose were stored. The stored values were used to calculate ground truth data. Instead of a 3D scene the Metaio benchmark like [58], records a printed (2D planar) poster, making the benchmark somewhat artificial. Furthermore, the setup with a moving camera on an arm means that the motions are not from natural tasks and were restricted by the arm workspace and mass. Metaio, haven't released the ground truth data which makes the dataset un-useable now that it's offline. An extensive dataset for 3D tracking is reported in [3]. [11] It reports 6 DOF ground turth, 3 degrees in rotation and 3 in translation.

To this effect, we report a public video dataset [12] to evaluate $2D$ markerless single object tracking algorithms. The tasks are natural table top manipulations, frequently performed in our daily life. We provide videos and annotated ground truth of both a human and a robot arm performing the same tasks. All videos are tagged with the task performed and the challenges that a tracker would face while tracking. Using these tags, susceptibility of the tracking algorithm can be properly narrowed down and subsequently improved. Each task is repeated with different speeds and under two different lighting conditions, of normal office light and diffused light.

[10]Available at: `http://www.metaio.com/research/`
[11]Available at: `http://robocoffee.org/datasets/`
[12]Available at: `http://webdocs.cs.ualberta.ca/~vis/trackDB/`

Chapter 3

Modular Decomposition of Tracking

A video stream is modeled as a temporal sequence of images, $\mathbf{F} = \{\mathbf{I_0}, \mathbf{I_1}, ..., \mathbf{I_n} : \mathbb{R}^2\}$. The pixel locations in an image patch with N pixels are denoted by $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_N})^T$ where $\mathbf{x_k} = (x_k, y_k)$ are the corresponding pixel intensities in image, denoted by $\mathbf{I}(\mathbf{x})$. When a geometric transform $\mathbf{W}$ with parameters $\mathbf{p} = (p_1, p_2, ..., p_l)$ is applied to an image patch $\mathbf{x}$, the transformed patch is denoted by $\mathbf{x}' = \mathbf{W}(\mathbf{x}; \mathbf{p})$ and the corresponding pixel values transform to $\mathbf{I}(\mathbf{W}(\mathbf{x}; \mathbf{p}))$. Object tracking is formulated (Eq 3.1) as a state prediction problem, where optimal transform parameters $\mathbf{p_t^*}$ for an image $\mathbf{I_t}$ that minimizes the difference, measured by a suitable distance metric $\mathbf{d}$, between the target patch $\mathbf{I}^* = \mathbf{I}(\mathbf{W}(\mathbf{x}; \mathbf{p_0}))$ and the warped image template $\mathbf{I}(\mathbf{W}(\mathbf{x}; \mathbf{p}))$. Equation 3.1 shows the formulation. t is dropped as both the image and optimal parameters are for the same time instance t. Common choices of distance functions are Eucledian distance (l_2) [52] and Manhattan distance (l_1) [13].

$$\mathbf{p_t} = \underset{\mathbf{p}}{\operatorname{argmin}}\, \mathbf{d}(I(\mathbf{W}(\mathbf{x}; \mathbf{p})), \mathbf{I}^*) \tag{3.1}$$

In registration based tracking, the three submodules (i) Appearance ($\mathbf{d}$), (ii) Search Method and (iii) State Space ($\mathbf{p}$) combine together as shown in Figure 3.1. This process is repeated over several iterations till the warped image ($\mathbf{I}(\mathbf{W}(\mathbf{x}; \mathbf{p}))$) is similar to the target ($\mathbf{I}^*$). This assumes static appearance model wherein no updates are done to the model with newer templates.

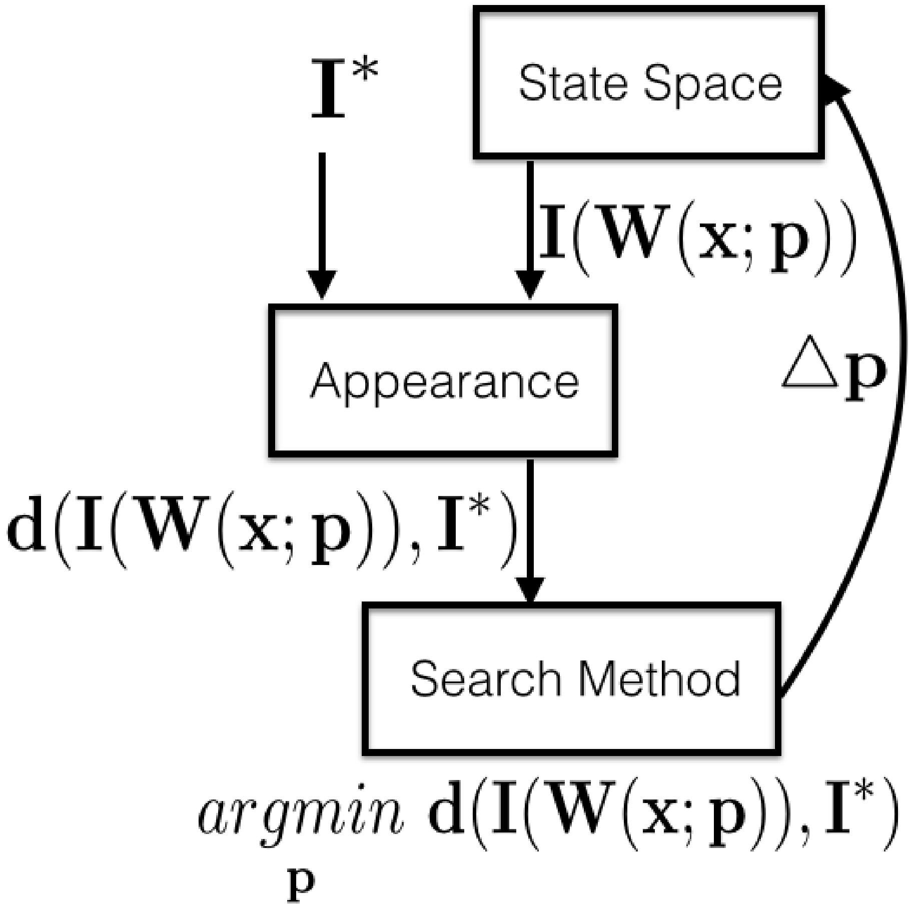

$\mathbf{I}^*$
State Space
$\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))$
Appearance
$\triangle\mathbf{p}$
$\mathbf{d}(\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})),\mathbf{I}^*)$
Search Method
$\underset{\mathbf{p}}{argmin}\ \mathbf{d}(\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})),\mathbf{I}^*)$

Figure 3.1: Breakdown of registration tracking into three modules: (i) Appearance model that compares two image patches, (ii) State space that parametrizes the object motion and (iii) Search method that searches for the best alignment to minimize the dissimilarity of the template with the target image.

3.1 Appearance Model

Appearance model is the transform space wherein the search method compares different warped patches from the current image to get the closest match with the original target $\mathbf{I}^*$.

3.1.1 Sum of Squared Difference (SSD)

Sum of squared difference ($\mathcal{SSD}$) is a common measure [52, 6, 9, 23] where raw pixel intensities are compared. $\mathcal{SSD}$ is expressed as an Eucleadian (l_2) norm in the image space.

$$\mathcal{SSD} = ||\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})) - \mathbf{I}^*||_2 \tag{3.2}$$

$\mathcal{SSD}$ is susceptible to lighting variations, as it does not correct for illumination bias. Sometimes smoothing of the image is done prior to tracking to reduce the effect of noise.

3.1.2 Sum of Conditional Variance (SCV)

Sum of Conditional Variance ($\mathcal{SCV}$) was originally used in medical imaging [64]. The authors show how $\mathcal{SCV}$ is invariant to non-linear illumination variations. This motivated its use in object tracking [23, 66], where changing appearance of the object is a common challenge. $\mathcal{SCV}$ corrects for illumination changes by taking into account $\mathbf{I}^*$ as reference.

$$\mathcal{SCV} = \sum_{\mathbf{x}} (\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})) - \mathbb{E}[\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))|\mathbf{I}^*])^2 \tag{3.3}$$

Where $\mathbb{E}$ is the Expectation Operator. We use the same formulation of as done in [23]. $\mathbb{E}[\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))|\mathbf{I}^*]$ is computed from the joint intensity distribution between $\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))$ and $\mathbf{I}^*$.

3.1.3 Normalised Cross Correlation (NCC)

Normalized Cross Correlation ($\mathcal{NCC}$) is another measure that accounts for illumination changes by centering and normalising the image data before comparing the two images.

$$\mathcal{NCC} = \frac{\sum_{\mathbf{x}} (\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})) - \bar{\mathbf{I}})(\mathbf{I}^*(\mathbf{x}) - \bar{\mathbf{I}}^*)}{\sqrt{\sum_{\mathbf{x}} (\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})) - \bar{\mathbf{I}})^2}\sqrt{\sum_{\mathbf{x}} (\mathbf{I}^*(\mathbf{x}) - \bar{\mathbf{I}}^*)^2}} \tag{3.4}$$

It has further been shown in [69] that maximizing $\mathcal{NCC}$ between two images $\boldsymbol{I}^*$ and $\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))$ is equivalent to minimizing the $\mathcal{SSD}$ score between the corresponding Z-score [38] normalized images.

3.1.4 Illumination Invariant Appearance (IVA)

Finally, we use another Illumination Invariant Appearance ($\mathcal{IVA}$) model in [61]. Colour image is decomposed into the consitituent channels of Red ($\mathbf{R}$), Green ($\mathbf{G}$) and Blue ($\mathbf{B}$). These independent channels are then used to recompute the illumination invariant image using two coefficients, α for Blue and β for Red channel. For a 2D image the transformed $\mathcal{IVA}$ image is computed as shown in Equation 3.5.

$$\mathcal{IVA} = \ln(\mathbf{G}[i,j]) - \alpha * \ln(\mathbf{B}[i,j]) - \beta * \ln(\mathbf{R}[i,j]) \tag{3.5}$$

The parameters α and β are subjected to $\frac{1}{\lambda_1} = \frac{\alpha}{\lambda_2} + \frac{\beta}{\lambda_3}$, where $\lambda_1, \lambda_2, \lambda_3$ are the peak sensitivity (wavelength) for each sensor. Parameters α and β for the camera sensors we use (SONY ICX274), are 0.482 and 0.518 respectively.

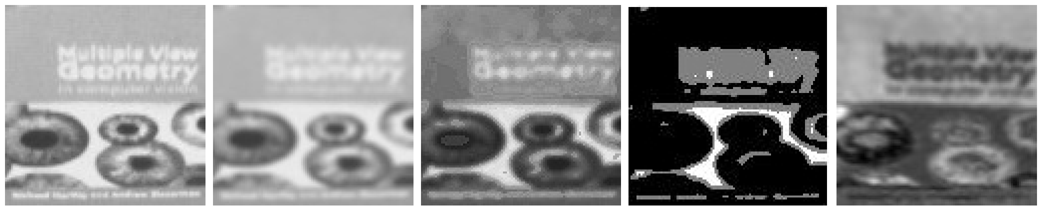

(a) (b) (c) (d) (e)

Figure 3.2: Example of different transforms, (a) Original Image (b) Original Image smoothed using a Gaussian Filter of window 5×5 and $\sigma = 3$ (c) Reverse Sum of Conditional Variance (d) Normalised Cross Correlation and (e) Illumination Invariant Appearance

3.2 Search Method

Search step finds the best transform parameters (state) $\mathbf{p}$ at time t, that minimize the difference between the target image ($\boldsymbol{I}^* = I(\mathbf{W}(\mathbf{x};\mathbf{p}_0))$) and the transformed template $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$. $\mathbf{W}$ is the transformation function and $\mathbf{p}$ the transform parameters. The difference is expressed as a distance score, $\mathbf{d}$ as shown in Eq 3.6.

$$\mathbf{p_t} = \underset{\mathbf{p}}{\operatorname{argmin}}\ \mathbf{d}(\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})), \mathbf{I}^*) \tag{3.6}$$

Euclidean distance or l_2 norm is a popular distance function [52, 6, 12]. The corresponding optimization is formulated as Eq 3.7.

$$\mathbf{p_t} = \underset{\mathbf{p}}{\operatorname{argmin}}\ ||\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})) - \mathbf{I}^*||^2 \tag{3.7}$$

Gradient based search is the most popular [52, 6, 9, 23] method used for optimization. Recent works include use of supervised learning [23]. In the following sub-sections, I elaborate these methods and describe how improvements can be done.

3.2.1 Gradient based search

One way to solve the above non-linear optimization problem is to iteratively update $\mathbf{p}$ till it converges to $\mathbf{p}^*$. Typically, iterations continue till some norm of the update ($\triangle\mathbf{p}$) is below a constant ϵ ($\triangle\mathbf{p} \leq \epsilon$). Lucas and Kanade [52] approximated the objective function ($||\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})) - \mathbf{I}^*||^2$), using a first order expansion of taylor series. Eq 3.8 is solved for the $\triangle\mathbf{p}$. This gives a closed form solution of the update, $\triangle\mathbf{p} = \mathbf{H}^{-1}\sum_{\mathbf{x}}[\nabla\mathbf{I}\frac{\delta\mathbf{W}}{\delta\mathbf{p}}]^T[I^* - \mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))]$. $\mathbf{H}$, the Hessian matrix, is computed as $\mathbf{H} = \sum_{\mathbf{x}}[\nabla\mathbf{I}\frac{\delta\mathbf{W}}{\delta\mathbf{p}}]^T[\nabla\mathbf{I}\frac{\delta\mathbf{W}}{\delta\mathbf{p}}] + \sum_{\mathbf{x}} r_x R_x$. Here r_x is the residual and R_x the second order differential of the residual. Dame showed in [20] (Figure 4.11) that using an approximate hessian $\mathbf{H} \approx \sum_{\mathbf{x}}[\nabla\mathbf{I}\frac{\delta\mathbf{W}}{\delta\mathbf{p}}]^T[\nabla\mathbf{I}\frac{\delta\mathbf{W}}{\delta\mathbf{p}}]$ the Gauss Newton type optimization has a larger area of convergence compared to using Newton type optimization. The authors, Lucas and Kanade [52] used an additive update, $\mathbf{p}^{k+1} \leftarrow \mathbf{p}^k + \triangle\mathbf{p}^k$, for $k+1^{th}$ iteration.

With an assumption that on each iteration the template image ($\mathbf{I}^*$) can be approximated very closely by the warped image $I(\mathbf{W}(\mathbf{x};\mathbf{p}))$ Hager et al.

[32], avoided the computationally expensive step of calculating $\mathbf{H}$ in every iteration. Their warp update was $\mathbf{p}^{k+1} \leftarrow \mathbf{p}^k - \triangle\mathbf{p}^k$.

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} ||\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}+\triangle\mathbf{p})) - \mathbf{I}^*||^2 \tag{3.8}$$

The formulations above assumes the image intensities are linearly dependent on the motion vector, i.e. they are consistent, to solve the over determined system. This assumption doesn't always hold true for real images. To avoid this, Jurie and Dhome [41] learnt the jacobian by fitting hyperplanes. This was done by randomly sampling data in the neighbourhood of $\mathbf{p}$.

Instead of using an additive update, Baker et. al [6] used an inverse compositional update, $\mathbf{W}(\mathbf{x};\mathbf{p}^{k+1}) \leftarrow \mathbf{W}(\mathbf{x};\mathbf{p}^k) \circ \mathbf{W}(\mathbf{x};\triangle\mathbf{p}^k)^{-1}$. The resultant minimization is as shown in Eq 3.9.

$$\mathbf{p}^* = \underset{\mathbf{p}}{\operatorname{argmin}} ||\mathbf{I}^*(\mathbf{W}(\mathbf{x};\triangle\mathbf{p})) - \mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p}))||^2 \tag{3.9}$$

The parameter update ($\triangle\mathbf{p}$) is expressed as, ($\triangle\mathbf{p} = \mathbf{H}^{-1}\sum_{\mathbf{x}}[\nabla\mathbf{I}^*\frac{\delta\mathbf{W}}{\delta p}]^T[\mathbf{I}(\mathbf{W}(\mathbf{x};\mathbf{p})) - \mathbf{I}^*]$. Hessian matrix ($\mathbf{H}$) is calculated only once, before tracking commences. $\mathbf{H} = \sum_{\mathbf{x}}[\nabla\mathbf{I}^*\frac{\delta\mathbf{W}}{\delta\mathbf{p}}]^T[\nabla\mathbf{I}^*\frac{\delta\mathbf{W}}{\delta\mathbf{p}}]$. This speeds up the optimization since both the gradient ($\nabla\mathbf{I}^*$) and the Jacobian ($\frac{\delta\mathbf{W}}{\delta\mathbf{p}}$) can be pre computed.

Benhimane and Mallis [9] presented a more robust compositional update scheme for the warp parameter $\mathbf{p}$, by using an Efficient Second order Minimization (ESM) [54] approach. The Hessian matrix ($\mathbf{H}$) is approximated using two Jacobians ($\mathbf{H} \approx \frac{1}{2}(\mathbf{J}(\mathrm{e}) + \mathbf{J}(\mathbf{x_c}))$), one of which is computed once, when tracking starts (on the template, $\mathbf{J}(\mathrm{e})$) and the other, ($\mathbf{J}(\mathbf{x_c})$), is updated on every iteration. This, though comparitively slow, on account of computing the Jacobian on every iteration, has it's own benefits. The authors [9] show that the algorithm converges using lesser number of iterations and also avoids the problem of not converging unless search starts close to $\mathbf{p}^*$.

3.2.2 Nearest neighbour based search

As an alternative to gradient descent search, Dick et. al [23] in a recent study (NNIC) used approximate nearest neighbor (ANN) search to find the best matching warp from a large list of pre-computed warps. They used randomized KD-trees [8] in Flann [1]. KD-Trees were used previously in [31] for tracking, however it was used as an object detector.

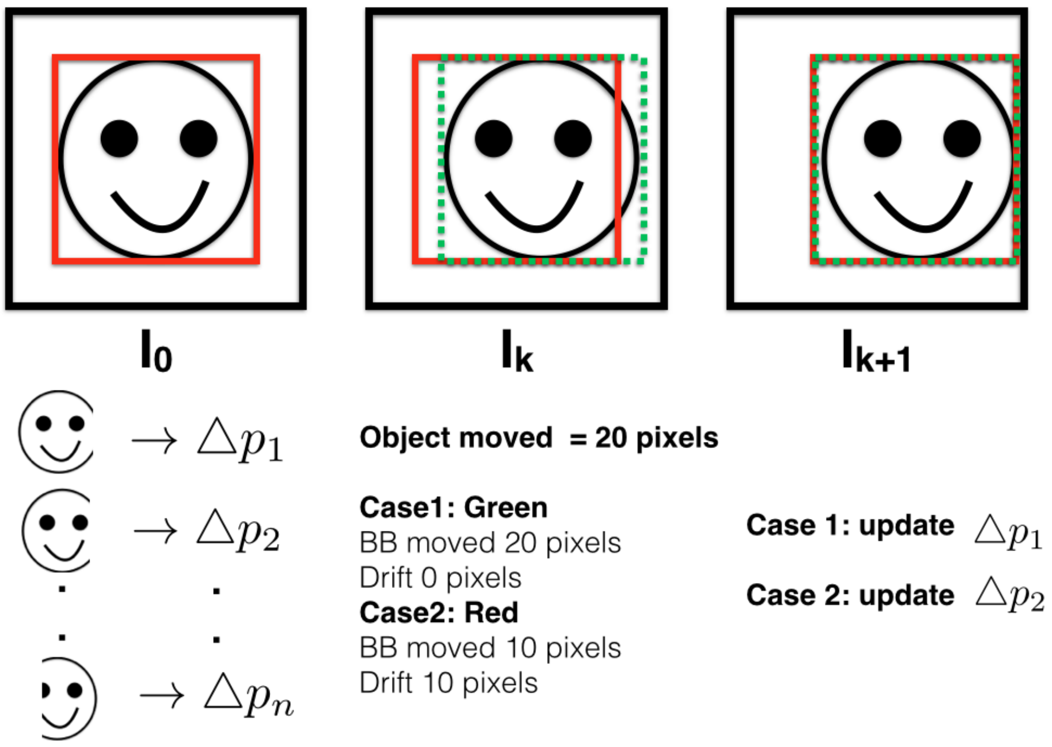
Randomised KD-Tree Based

Nearest Neighbour search by itself gives coarse alignment which needs further refinement using a precise tracker in IC [6]. Hence, NNIC [23] uses a cascade tracking system, with a coarse tracker using ANN (randomised KD-Tree) and fine alignment done using IC [6]. The authors randomly sample warps from a Gaussian distribution $\mathbf{p_i}$. Warped template images are generated as shown in Algorithm 1. The warped images are then used to build KD-Tree offline. During tracking, the best warp ($\mathbf{p}^*$) is searched using randomised KD-Tree search [8]. This warp $\mathbf{p}^*$ is the corresponding warp for which $\mathbf{I}(\mathbf{W}(x; \mathbf{p}))$ is the most similar to the target ($\mathbf{I}^*$). Updates to the parameters are done as, $\mathbf{p} \leftarrow \mathbf{p} \circ \mathbf{p}^{*-1}$.

The newton type trackers in Lucas Kanade tracker [52], IC [6], ESM [9] match the template with $\mathbf{I}^*$ which prevents the tracker from accumulating drift. Figure 3.3 show an example of how Nearest Neighbour search corrects for drift. Say we are tracking translational motion along X-axis. The tracker is initialised on the object on the first frame $\mathbf{I_0}$. Warped images are generated prior to tracking. Now, since we are modelling translational motion along X-axis only, those images would look like shifted versions of the template as shown below. An intermediate frame is shown in $\mathbf{I_k}$. Two cases are shown, Case 1: tracker is tracking, shown in green, and Case 2: tracker has drifted slightly, shown in red. For the next frame, the image region in $\mathbf{I_{k+1}}$ within the bounding box co-ordinates of the previous frame is matched from the stored set of images, for pose update ($\triangle\mathbf{p}$). This would move the bounding

[1] Available at: `https://github.com/mariusmuja/flann`

box right. For Case 1, the update would be $\triangle \mathbf{p_1}$, whereas for Case 2, the update would be $\triangle \mathbf{p_2}$. $\triangle \mathbf{p_1}$ has a smaller magnitude compared to $\triangle \mathbf{p_2}$. It can be seen from the corresponding warped images. This explains how Nearest Neighbour search corrects for drift and doesn't suffer from drift getting added. Following this the IC part of NNIC further refines the search for better alignment.



$\mathbf{I_0}$
$\mathbf{I_k}$
$\mathbf{I_{k+1}}$
$\rightarrow \triangle p_1$
$\rightarrow \triangle p_2$
$\rightarrow \triangle p_n$
Object moved = 20 pixels
Case1: Green
BB moved 20 pixels
Drift 0 pixels
Case2: Red
BB moved 10 pixels
Drift 10 pixels
Case 1: update $\triangle p_1$
Case 2: update $\triangle p_2$

Figure 3.3: Three frames of a sample tracking problem is shown. First frame $\mathbf{I_0}$, with the object initialised, k^{th} frame $\mathbf{I_k}$, where two cases are shown, Case 1: tracker is tracking, shown in green, and Case 2: tracker has drifted a bit, shown in red. Finally, the next frame $\mathbf{I_{k+1}}$ is shown where both the Cases converge to the same bounding box.

Motivated by how intuitive this algorithm is, we adapt a Sequential Graph Based Approximate Nearest Neighbour Search (SGANNS) [33] in tracking in GNNIC.

Algorithm 1 Pre-Computation of Sampled Images

```
1: procedure WARPIMAGE($I^*, \sigma$)                    ▷ $I^*$ Template Image
2:     $warpedData = dict\{\}$
3:     $N$ = No of Samples
4:     while $I < N$ do
5:         $\mathbf{p} = \ sample\, \mathcal{N}(0, \sigma)$
6:         $\hat{I} = \mathbf{I}(\mathbf{W}(\mathbf{x}; \mathbf{p}))$
7:         $warpedData[\mathbf{p}] = \hat{I}$
8:         $I = I + 1$
9:     end while
10:    return $warpedData$
11: end procedure
```

Sequential Graph based Approximate Nearest Neighbour Based

Randomised KD-Tree based approach doesn't take into account the temporal coherency of video data, an assumption that is implicit in videos. It also is restricted to multidimensional metric spaces as search is done by splitting the feature space (image space in this case) along different dimension. The original graph based approximate nearest neighbour algorithm proposed in [70] was not restricted to this assumption of multi dimensional metric spaces but didn't use the sequential property of data. Hajebi et al. in [33] showed how the temporal coherency of data can be used to speed up search by having an intelligent guess of the start node. We adapt it in tracking.

Building Graph G The same pre-computed sampled images from Algorithm 1 is used to construct the graph **G** needed in GNNIC. Each warped image template corresponds to a node in the graph.

Algorithm 2 Construction of the graph **G**

```
1: procedure BUILD GRAPH(warpedData, k)          ▷ k = Number of Neighbours
2:     G = {}
3:     while $I < len$(warpedData) do
4:         $Q$ = warpedData[$I$]
5:         $(indexes) = knnsearch(Q,$ warpedData, k)
6:         G[$Q$] = $indexes$
7:         $I = I + 1$
8:     end while
9:     return G
10: end procedure
```

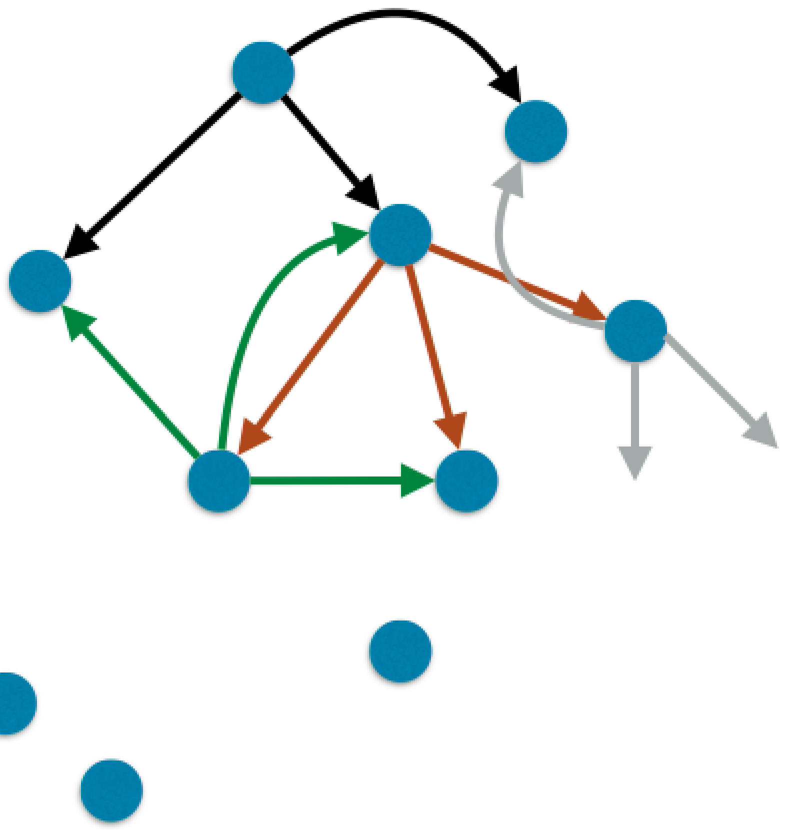


Figure 3.4: Graph is built by associating each node with **k** nearest nodes in image space. Example shows a graph constructed with degree of connectivity **k** = 3. A fully connected and directed graph is built. Complexity for exact graph construction is $O(n^2d)$, where n is the number of nodes (warped template image), d is the dimension of the flattened image patch.

The graph $\mathbf{G}$ is built (shown in Fig 3.4 for a connectivity of 3) by connecting each node with $\mathbf{k}$ nearest nodes in the image space. We explore both l_2 and l_1 distances to find nearest nodes. The parameter $\mathbf{k}$ decides sparsity of the resultant graph $\mathbf{G}$. Higher the value of $\mathbf{k}$ the more connected the graph is. We set $\mathbf{k}$ to 200 to have a fairly connected graph. *knnsearch* which calculates the k nearest neighbours, is implemented using quick-select [53], that has a computational complexity of $O(n)$, where n is the number of samples. The cost of constructing an exact graph $\mathbf{G}$ is $O(dn^2)$, where d is the resolution of the template $(d = (m \times n))$. Search during tracking is performed as explained in Algortihm 3.

Algorithm 3 SGANNS Search

```
 1: procedure SGANNS($\mathbf{G}, Q, T, E$)    ▷ T= No of greedy steps, E = No of possibilities
 2:     $\mathbf{S} = \{\}$
 3:     $\mathbf{U} = \{\}$
 4:     $Y_0$: a point drawn randomly from $\mathbf{G}$.
 5:     while $I < T$ do
 6:         $Y_t = \text{argmin}_{Y \in N(Y_{t\ 1}, E, \mathbf{S})}(Y, Q)$
 7:         $\mathbf{S} = \mathbf{S} \cup N(Y_{t-1}, E, \mathbf{S})$
 8:         $\mathbf{U} = \mathbf{U} \cup \{\mathbf{d}(Y, Q) : Y \in N(Y_{t-1}, E, \mathbf{S})\}$
 9:         $I = I + 1$
10:     end while
11: Sort U, pick the first K elements, and return the corresponding elements in S
12: end procedure
```

Sequential Graph Search When tracking starts, at every iteration, the algorithm searches for the best $\mathbf{E}$ possible matches within the $\mathbf{k}$ connected nodes. $N(\mathbf{Y}, \mathbf{E}, \mathbf{G})$ searches for $\mathbf{E}$ nearest neighbour of $\mathbf{Y}$ in the connected graph $\mathbf{G}$ where $\mathbf{E} \leq \mathbf{k}$. This is similar to *knnsearch*, but within the $\mathbf{k}$ connected child nodes of a node, $\mathbf{Y}$. The current node is replaced by the neighbor that is closest to the query and each of these $\mathbf{E}$ child nodes are then further explored. This exploration continues for T iterations to find the best

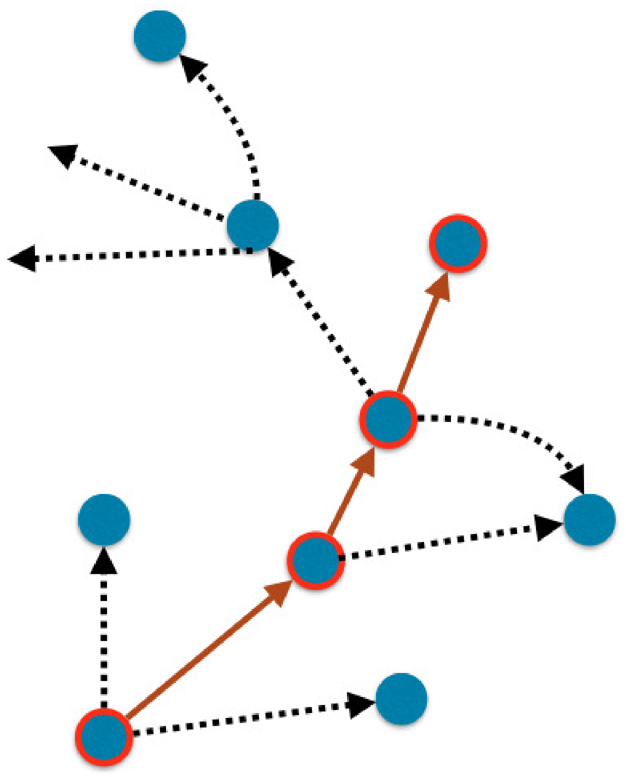


Figure 3.5: Greedy search is done by replacing the current node with one of the child nodes that has the least distance compared to the query template untill some stopping criteria is met. Nodes highlighted in red shows the path of search.

match. Termination can also be done when the residual of the best match with the template is less than a set thershold. Note the difference between $\mathbf{K}$ - the number of nearest neighbours to be returned and $\mathbf{k}$- degree of connectivity of the graph. In tracking applications, we use $\mathbf{K} = 1$, that returns the closest match. For the first frame, the algorithm initialises from a random node. From there on, the best node of the last search is used as the starting point. Hence the name Sequential Graph based Approximate Nearest Neighbour Search (SGANNS).

This allows us to remove the random restart loop in TABLE I [33]. The assumption here is that features are repeatable in consequtive frames, since video being a temporal sequence, lot of overlap exists between two frames.

We use the same setup, by using SGANNS with a cascade of IC [6] to further improve alignment. Detailed analysis and results of this tracker GNNIC is presented in Chapter 5.

3.3 State Space Model

The state space model parametize the motion of the object. It embodies any constraints that are placed on the search space of warping parameters to make the search process more efficient. This includes both the DOF of allowed motion, as well as the actual parameterization of the warping function. For instance, the ESM tracker [9] can be considered to search over a different parameterization of the state space than the conventional Lucas Kanade type trackers [52, 6] since it uses an $\mathcal{SL}3$ parameterization of homography rather than the actual entries of the corresponding matrix as used by the others. Different DOF common in tracking are 2, 3, 4, 6 and 8 as shown in Figure 3.6.

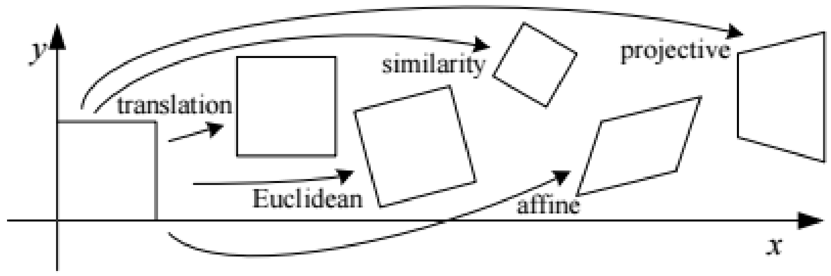


Figure 3.6: Example of different 2D transform is shown. Image used from [76].

2 DOF : 2D translation is modelled by following the object co-ordinates (X and Y positions of centroid) in the image plane.

3 DOF : In this case translation and rotation are both modelled as part of the object motion. This is also called 2D rigid body motion.

4 DOF : Translation, rotation and scale parameters are estimated for 4 DOF motion. This is also known as similarity transform. VOT challenge [46] evaluates trackers for 4 DOF motion.

6 DOF : Object motion is approximated using 6 parameters of the affine matrix. Parallel lines remain parallel under affine transformation. IVT [67] and L1-APG [57] are affine trackers.

8 DOF : Projective transform or homography transform accounts for the full pose of the planar object . It preserves straight lines. Most registration based trackers [52, 32, 9, 23, 82] track 8 DOF pose of the object.

The advantage of allowing higher DOF in the warping function to achieve greater precision in the aligned warp since transforms that are higher up in the hierarchy [36, sec. 2.4] can better approximate the projective transformation process that captures the relative motion between the camera and the object in the 3D world into the 2D images. However, there are two issues with having to estimate more free parameters. Firstly, the iterative search takes longer to converge, making the tracker slower. Secondly, the search process becomes more likely to either diverge or end up in a local minimum thus causing the tracker to be less stable and more likely to lose track. The latter is a well known phenomenon with Lucas Kanade type trackers [14] whose higher DOF (e.g. affine) variants are much less robust than simple 2 DOF versions. In the thesis I focus on 8 DOF trackers.

Chapter 4

Tracking for Manipulation Tasks Dataset and Evaluation

Designing tracking systems raises the question of how to best evaluate them. As tracking research is divided into two focus groups, it is (i) too varied to claim generality, and (ii) each group is interesting enough to have its own set of evaluation standards. A longstanding problem of evaluating registration based trackers is the unavailability of a standard dataset and evaluation methodology. We report a **Tracking for Manipulation Tasks (TMT)** [68] dataset to address this.

TMT is both a video dataset and an evaluation benchmark. The dataset consists of more than **100** videos of manipulation tasks carefully recorded by a human user and a robot arm. The tasks are chosen to mimic routine tasks one would perform on a daily basis. The videos are structured to systematically evaluate a tracker with evaluation scripts, challenge tags, sample results and ground truth data are made public. [1]

4.1 Video Capture Set Up

All videos (Fig 4.3) were recorded using a GRAS-20S4C-C fire-wire camera equipped with a Kowa LM6NCM F1.2/6mm lens. For the human recorded videos, the camera stood on a fixed position on a tabletop 90 cm from the edge of the table. The human user had no involvement in designing track-

[1]Available at: `http://webdocs.cs.ualberta.ca/~vis/trackDB/`

Table 4.1: Parameter Setting for Video Capture

Parameters	**Diffuse Light**	**Normal Light**
Exposure (in IL)	0.73	1.06
Gain (in dB)	6.02	0
Shutter (in s)	0.04	0.07
White Balance	Blue/U927 Red/V493	Blue/U757 Red/V490
Saturation (in %)	95.22	119.04

ing algorithm. This removes bias in the way the motion was planned while manipulating the objects. Camera settings were implemented through coriander 2.0.1 [62]. Lighting was varied between normal office lighting and diffused lighting by placing an umbrella to avoid direct source. The videos were recorded at 30 frames per second (F.P.S.) at a resolution of 600x800 in YUV colour space. Exact parameters of recording can be found in Table 4.1.

For the robot recorded videos, a 7 DOF WAM arm [1] with a Barrett hand was used. Fig. 4.1 shows the set up. The camera was fixed on a tripod stand at a distance of 120 cm from the WAM arm. Other parameters of exposure, gain, shutter, white balance and saturation were kept the same as in Table 4.1.

4.1.1 Description of Videos

A tracking algorithm is considered to be robust if it successfully tracks in the presence of *"Translation (TR)"*, *"Rotation (RO)"*, *"Illumination (IL)"*, *"Perspective (PR)"*, *"Specularity (SR)"*, *"Occlusion (OC)"*, *"Texture (TX)"* and *"Speed (SP)"* variations.

We structure the dataset into two categories of videos. *Oriented Motion* tasks (upper block in Table 4.2) and *Composite Motion* Tasks (lower block in Table 4.2). Each video consits of one or more of the challenges mentioned above.

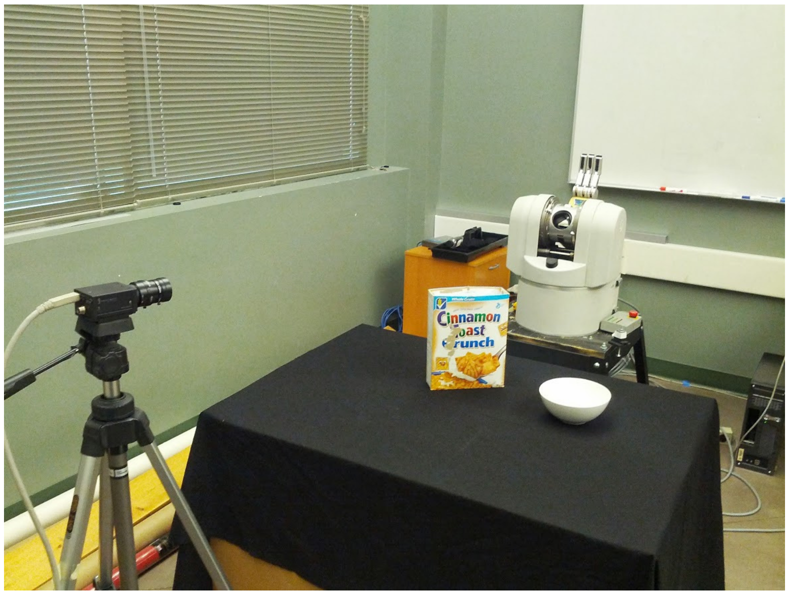


Figure 4.1: Set up for recording videos under normal light using a WAM arm [1] and a Barret hand. The camera is set 120 cm from the base of the robot arm.

Oriented Motion Tasks

Oriented or single motion tasks refer to highly structured motion of the object, where the human user or the robot hand performs one simple action in manipulating an object. Details of the tasks are described below.

Juice Juice from a juice box is poured onto a container. The goal is to track the juice box in the presence of both translational (TR) and rotational (RO) motion of the object.

Cereal This task is similar to *Juice*, with an added challenge. There is large jerky motion when the cereal is poured onto the bowl. Also, the cereal box has higher texture compared to the juice box.

Book I The object (book) is tilted from a vertical upright position to finally lie flat on the table and back up again. The challenge is to handle perspective (PR) transformation of the object in the image plane. During

Table 4.2: Description of Videos in TMT Dataset

Seqs	Object	Challenges	# of Seqs
Juice	Juice Box	TR,RO,SP,IL	6 NL 6 DL
Cereal	Cereal Box	TR,RO,SP,IL	6 NL 6 DL
Book I	Hardcover Book	PR,SP,IL,SR	6 NL 5 DL
Book II	Hardcover Book	SC,SP,IL	6 NL 6 DL
Book III	Hardcover Book	TR,OC,SP,IL	5 NL 5 DL
Mug I	Coffee Mug	TR,SP,IL,SR	6 NL 6 DL
Mug II	Coffee Mug	TR,PR,SP,IL,SR	6 NL 6 DL
Mug III	Coffee Mug	RO,SP,IL,SR	6 NL 6 DL
Bus	Toy Bus	TR,SC,PR,SP,IL	1 NL 1 DL
Highlighting	Newspaper	OC,SP,IL,TR	1 NL 1 DL
Letter	Envelope	PR,SP,IL	1 NL 1 DL
Newspaper	Newspaper Page	PR,SC,SP,IL	1 NL 1DL

TR = Translation; RO = Rotation; IL = Illumination; PR = Perspective; SR = Specular; OC = Occlusion; TX = Texture; SP = Speed; NL = Normal Light; DL = Diffuse Light

this motion there is Specular Reflection (SR) from the cover of the book.

Book II Here a book is brought near the camera, parallel to the camera axis and away. The goal is to capture scale (SC) variation of the object.

Book III The object (book) is placed inside a book holder. The challenge in this video is to track the translational (TR) and Rotational (RO) motion of the object, despite varying occlusion (OC) from the book holder.

Mug I Translational (TR) motion of a coffee mug is recorded when it's lifted up. The cup being a small shiny object, specular reflection (SR) and low texture (TX) of the mug present additional challenges in this case.

Mug II This sequence, although similar to *Mug I*, is more difficult to track because of high perspective (PR) deformation of the mug when it is tilted to drink the contents.

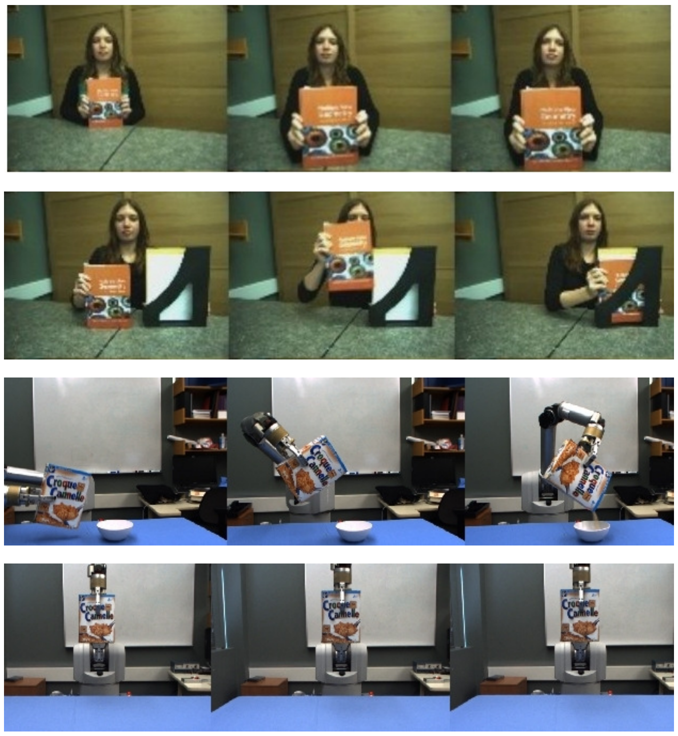


Figure 4.2: Image frames of the video sequences show sample tasks performed by both a human user and a robot hand with two different objects.

Mug III A low texture (TX) coffee mug is rotated (RO) from its initial position and back along the axis perpendicular to the principal camera axis.

Composite Motion Tasks

Composite motion tasks consists of videos that can be decomposed into simpler *Oriented Motion* tasks. Some tasks involve the use of both the hands as against the simple motions in *Oriented Motion* tasks. For the composite motion taks only human recorded videos are reported.

Bus The planar surface in front of a toy bus, that undergoes scale change (SC) and perspective deformation (PR) at varying speed (SP), is to be tracked.

Highlighting A portion of a newspaper is highlighted with a yellow marker pen. The challenge is to track the object in the presence of changing texture (TX) caused by highlighting and partial occlusion (OC) by the hand of the

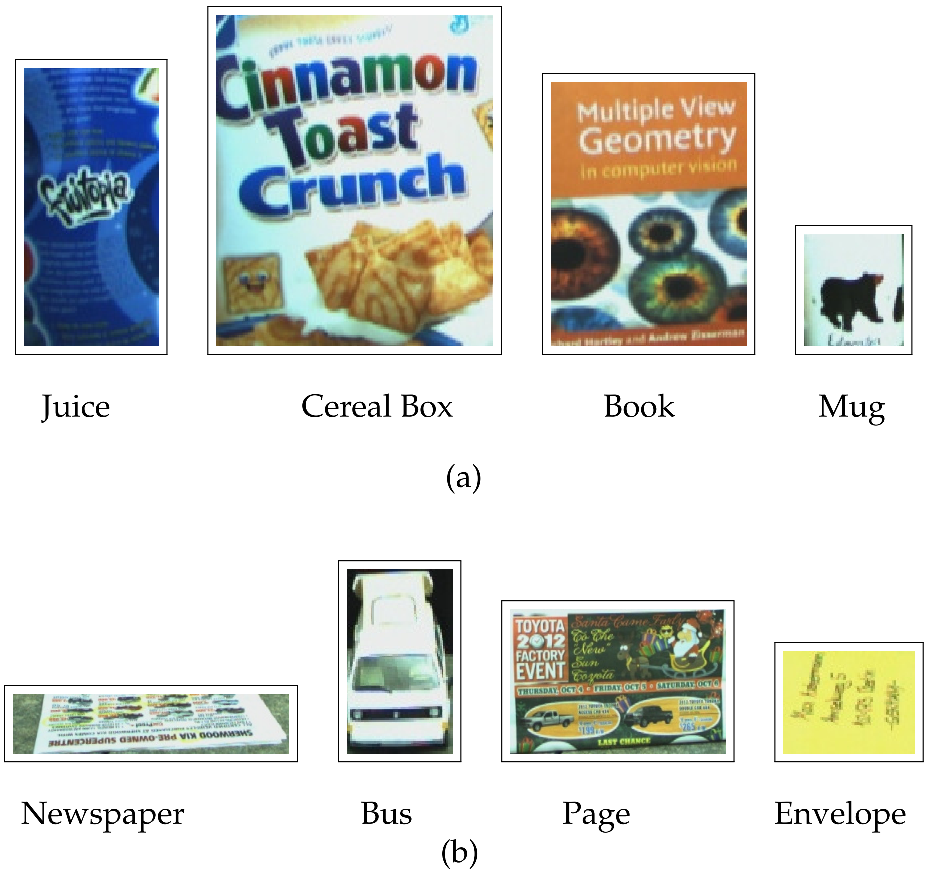


Figure 4.3: Planar projections of the objects used in the dataset are shown, both planar and curvilinear objects are chosen with varying texture, lambertian/specular to have a full spectrum of challenges. (a) Objects used to record *Oriented Motion Tasks*, (b) Objects used to record *Composite Motion Tasks*

user and the pen she uses for highlighting.

Letter The sequence records a letter being put inside an envelope and out again. The challenge in largely to tackle the perspective deformation (PR) of the object. This is a complex motion where both hands are used, one manipulating the envelope, the other taking out the letter from the envelope.

Newspaper A textured portion of the newspaper is to be tracked in the presence of perspective (PR) and scale (SC) changes under varying speed.

Most of the *Oriented Motion* tasks have 6 videos recorded in both Natural Light (NL) and Diffused Light (DL) settings. The six videos are of the same task performed at varying speeds. Speed vary from very slow to very fast with an added option of "increasing speed", in which the speed is varied while performing the task. *Complex Motion* tasks, on the other hand report one video with varying speed. Having videos of different speeds, correctly captures blurr as a challenge at higher speeds. A total of 8 different objects (Fig 4.3) were used to record the videos. To have a diverse selection, we choose objects that are planar (book, cereal box, juice box), curvillinear (mug, yogurt can), lambertian (newspaper), specular (book cover, mug, envelope), large (book, cereal box, juice box) and small (bus, mug).

Each frame is tagged with the different challenges it presents. The distribution of frames over the challenges is shown in Figure 4.4. Sometimes more than one challenge is present simultaneously in one frame. All such challenges are tagged. The number on top of each bar in Figure 4.4 shows the number of frames having the corresponding challenge. TR or translation is the most common challenge since in most videos it is part of the initial motion that the object goes through. Only the highlighting sequence presents change in Texture (TX). However, we do use different objects to cover a varied set of textures.

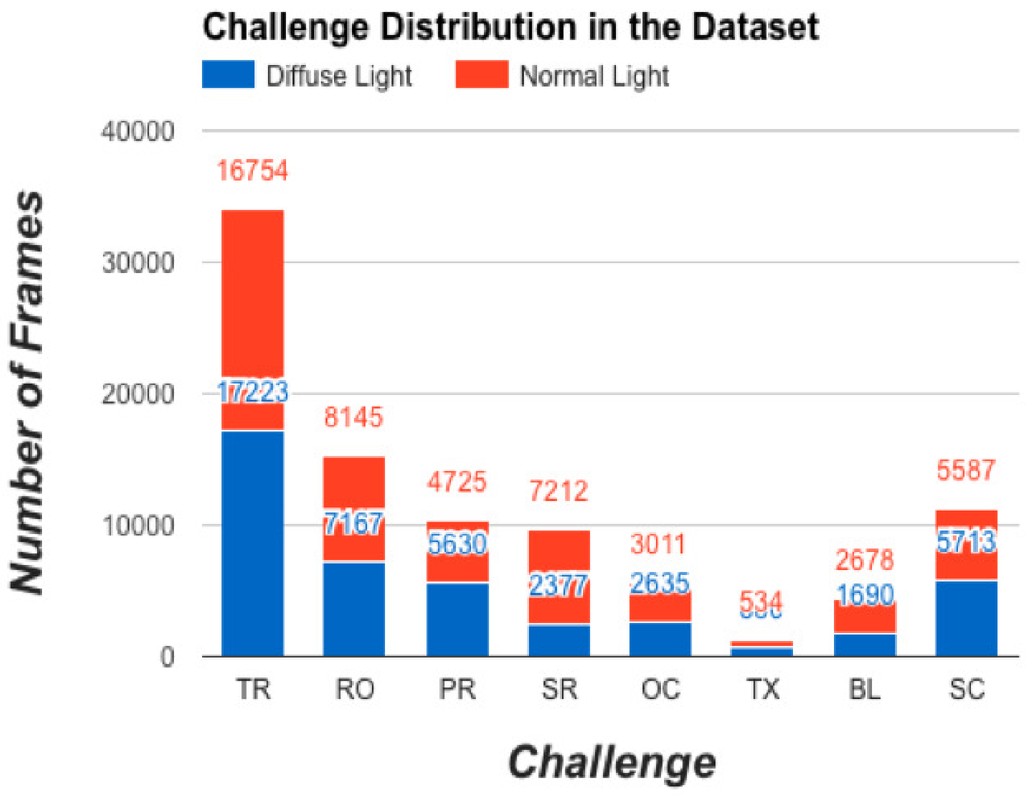


Figure 4.4: The distribution of number of frames over the different challenges is shown. Blue bar shows the number of frames under diffused light while the red bar shows the number of frames under normal light. The number on top of each bar represent the number of frames with the corresponding challenge. Abbreviations for challenges are, TR = Translation; RO = Rotation; PR = Perspective; SR = Specular; OC = Occlusion; TX = Varying Texture, BL = Blurr casued at high speeds, SC = Scale change.

4.2 Ground Truth

Ground Truth (GT) refers to the correct position and orientation of the object in the image frame. A tracker's performance is evaluated against this data. Video datasets that evaluate surveillance trackers manually annotate a loose bounding box around the object of either 3 DOF (x, y, scale) [78] or 4 DOF (x, y, scale, rotation) [46]. They also benefit from the fact that the trackers are required to roughly follow the object with an allowable error limit of 50% overlap. Frame to frame correspondence of the target object is not very rigid in this case.

With registration based trackers the thresholds are much more stringent. Metaio [48] uses an absolute error threshold of 10 pixels to decide if a tracker has failed on not. The dataset was generated using a camera mounted on a Faro arm that was calibrated and all transformations of pose were stored. The stored values were used to calculate ground truth data. Fiducial markers were also used to double check the ground truth thus generated. This setup is restricted by the workspace and mass of the robot arm.

We report sub-pixel level co-ordinate positions based on tracking data. Three trackers [23, 6, 9] are initiated on the first frame. Ground truth is registered only when bounding box co-ordinates reported by all three of them lie within ± 1 pixel. These trackers were chosen because of their high convergence which is further shown in Chapter 5.

The stringent convergence criterion (± 1 pixel) sometimes results in the trackers failing to converge to a common bounding box. We reinitialize the three trackers using positional information from previous frames every time they fail to converge. Number of reinitializations varied from 0 to 12, for more difficult sequences. As a final step, we also verify all ground truth data manually.

4.3 Error Metric

Error in single object tracking is a quantitative estimate of how closely the tracker follows the object. Tracked output (X_T) is compared with Ground Truth (X_{GT}) to calculate this error. Any further analysis of a tracker first needs an error measure. Some of the common error measures in literature are, *Center Distance* [78] and *Area Overlap* [46].

4.3.1 Center Distance

Center distance (E_C) is defined as the Eucledian distance between the centroid of the tracked data and ground truth data. Mathematically, it is expressed as shown in Eq. 4.1

$$E_C = \sqrt{(X_t - X_{GT})^2} \tag{4.1}$$

X_t is the centroid of a tracked object, X_{GT} the corresponding ground truth. This measure is more suitable for 2 DOF trackers that track translational motion of the object.

4.3.2 Area Overlap

Area overlap (E_A), a measure inspired from segmentation scores [71], has been used in tracking in [78, 46]. Area overlap is defined as the ratio of the area of intersection of the target region A_T and ground truth region A_{GT} with that of their union. Mathematically it is expressed as Eq. 4.2.

$$E_A = \frac{|A_T \cap A_{GT}|}{|A_T \cup A_{GT}|} \tag{4.2}$$

Both the above mentioned errors fail to properly capture the misalignment of pose. Fig 4.5 shows a case where the Ground Truth (X_{GT}) and Tracked result (X_T) are $180°$ out of phase. *Center Distance (E_C)* and *Area Overlap (E_A)* fail to account for this out of phase alignment, and have very low error scores. To address this, we describe a measure called Alignment Error, Sec. 4.3.3. This error measure is similar to the one used in [48].

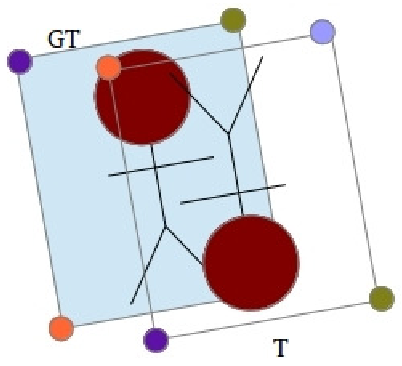


Figure 4.5: Alignment error E_{AL} accounts for the displacement of each corner (colour coded) from the corresponding GT data. It is expressed as a root mean square (RMSE) score to account for the misalignment of pose.

4.3.3 Alignment Error

Alignment error quantifies the change in pose between the Target image (X_T) and Ground Truth image (X_{GT}). Alignment Error (E_{AL}) is expressed as a root mean square distance of misalignment of the target image (X_T) with ground truth (X_{GT}). Mathematically it is expressed as shown in Eq. 4.3.

$$Error(E_{AL}) = \sqrt{\frac{\sum_{i=1}^{4}(X_T - X_{GT})^2}{4}} \tag{4.3}$$

The Root Mean Square Error (RMSE) is calculated between the corresponding four corners of the bounding box. For all of our experiments, we use Alignment Error (E_{AL}).

4.4 Evaluation Measures

Using Alignment Error, Sec. 4.3.3, we evaluate trackers based on four different measures. (i) **Overall Success** (Robustness), (ii) **Average Drift** (Convergence), a measure that we define (iii) **Speed Sensitivity** and (iv) **Overall Rank** similar to [46].

4.4.1 Overall Success

Overall Success (OS) is a commonly used measure [46, 78]. It is defined as the fraction of total frames that a tracker tracks within an error (E_{AL}) threshold of t_p pixels. It is expressed as

$$\begin{aligned} \mathbf{OS} &= \frac{|\mathbf{S}|}{|\mathbf{F}|} \\ \mathbf{S} &= \{f^i \in \mathbf{F} : E^i_{AL} < t_p\} \end{aligned} \tag{4.4}$$

$\mathbf{S}$ is the set of successfully tracked frames, $\mathbf{F}$ set of all frames, E_{AL} error of frame (f^i). Manipulation tasks calls for high precision in tracking. We set threhold t_p to be 5 pixels in our experiments.

4.4.2 Average Drift

Average Drift (AD) computes the expected error of the tracker when it operates within an allowed drift of t_p pixels. This is similar to the one proposed in [75].

$$\begin{aligned} \mathbf{AD} = E[e|e < t_p] &= \frac{\sum\limits_{f_i \in \mathbf{F}} E_{AL}}{|\mathbf{S}|} \\ &\text{subject to } e^i_{AL} < t_p \end{aligned} \tag{4.5}$$

Average Drift checks as to how precisely a tracker aligns itself with the target. Yang et al. [78] used a pixel threshold t_p of 20 pixels. This is too high for manipulation tasks. As used in OS above, we set t_p to be 5 pixels. It is important to calculate AD when the tracker is within 5 pixel drift or else it would be biased by large drifts when the tracker no longer tracks the object.

4.4.3 Speed Sensitivity

A common way [6, 23, 9] to quantify tracking convergence in dealing with large motion is to synthetically warp the standard "Lena" image and allow the tracker to recover this warp. The random warps are sampled from a Normal distribution having zero mean and standard deviation σ, ($\mathcal{N}(0, \sigma)$). However, this experimental methodology overestimates convergence. Warps sampled using a large sigma, with a certain probability still have smaller warps in them.

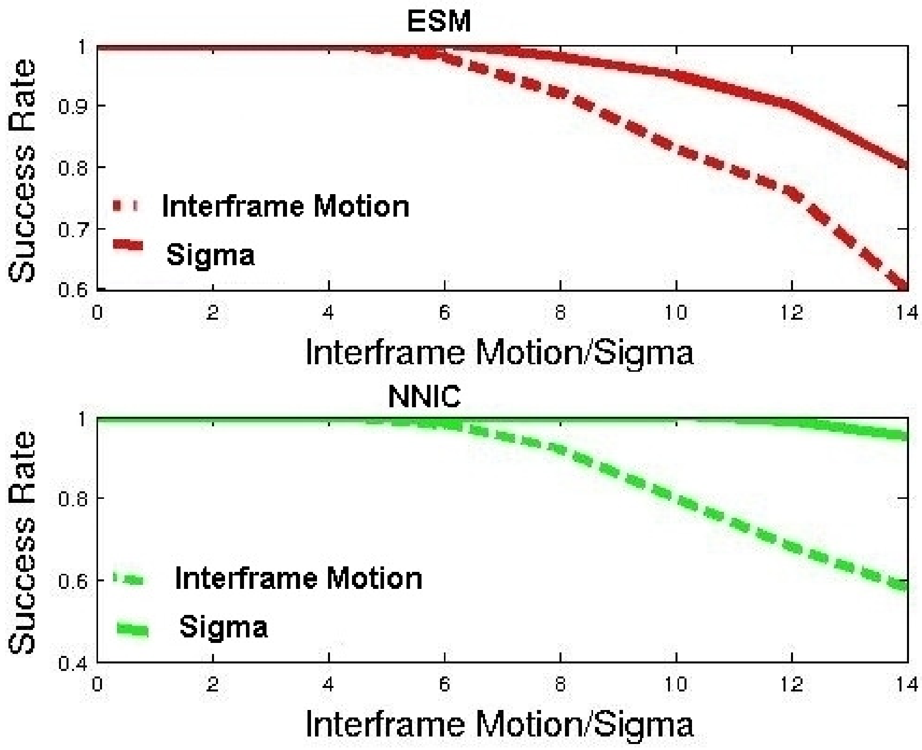


Figure 4.6: Success rate ($t_p = 2$) plotted against varying inter frame motion and sigma. It shows that even though the success rate for higher sigma is high it's lower for the corresponding inter frame motion.

This is further illustrated in Figure 4.6. Overall Success (OS) is plotted against both interframe object motion m_i as shown in Eq. 4.6 and sigma (σ). Trackers perform poorly on large motions of 12 pixels (0.6 Success Rate) compared to the corresponding sigma (0.9 Success Rate), for NNIC tracker.

$$m_i = rmse(gt^{i+1}, gt^{i}) = \sqrt{\frac{\sum_{k=1}^{4}(gt_k^{(i+1)} - gt_k^{(i)})^2}{4}} \tag{4.6}$$

Where $gt^{(i)}$ is the ground truth data for i^{th} frame. Algorithm 4 shows how *Speed Sensitivity* is computed. Since focus in given to precise trackers, threshold t_p (TH here) is 5 pixels. $\#tf$ is the number of frames that have error threshold below t_p with inter frame motion m in Algorithm 4. $\#f$ is the total number of frames having inter frame motion m.

Algorithm 4 Speed Sensitivity of Trackers

```
1:  procedure SPEEDSENSITIVITY($GT, T$)                ▷ T, tracked Result
2:      $r = dict\{\}$                                 ▷ {[m]:(#tf , #f)}
3:      $TH = 5$                 ▷ TH = Threshold for successful tracking
4:      while $I < len(GT) - 1$ do
5:          $m = rmse(GT[I+1], GT[I])$
6:          $err = rmse(GT[I+1], T[I+1])$
7:          if err < TH then
8:              dict[m] = (#tf+= 1, #f+= 1)
9:          else
10:             dict[m] = ( #tf, #f+= 1)
11:         end if
12:         $I = I + 1$
13:     end while
14:     return $dict.items()$
15: end procedure
```

When used with real videos from the TMT dataset, the number of samples having fast object motion is less. We skip frames (track every second, third and fourth frame) to simulate fast motion. We also track both forward and backward, initialise tracker on k^{th} frame and track $k+1^{th}$ frame, initialise on $k+1^{th}$ and track k^{th} frame.

4.4.4 Overall Rank

Measures discussed above rank trackers on isolated attributes of success and robustness. They fail to provide a global view of how the trackers compare. We use an idea similar to AR plots in [46]. Trackers are ranked based on Overall Success (OS) and Robustness. Finally a 2D plot of Overall Rank shows their relative performance taking into account both attributes of OS (Y-axis) and Robustness (X-axis). Since we are interested in evaluating precise trackers, we use Alignment Error (E_{AL}) instead of E_A used in [46].

Robustness here is quantified by the number of times a tracker fails, and needs to be re-initialised. A tracker is considered to have failed when E_{AL} $> t_p = 5$ pixels. Once the tracker fails, the tracker is re-initialised 10 frames from the frame it has failed to remove any bias and account for the same challenge twice. Total number of re-initialisations give an indication of how robust a tracker is in a manipulation setup. Two graphs are plotted. A rank plot graph where X-axis has the robustness rank and Y-axis the success rank, and un-normalised graph that shows the actual number of re-initializations on the X-axis and the overall success on the Y-axis.Trackers on the top right are the better performing trackers.

Chapter 5

Results and Evaluation

In this Chapter we evaluate and analyse both registration based trackers and OLT. Since we are interested in precise tracking, we use videos from TMT dataset (Chapter 4) [68] which is a publicly available dataset and reports 8 DOF ground truth. Static experiments using the standard "Lena Image" as done in [6, 9, 23, 82] are also performed with results reported and analysed.

5.1 Baseline Trackers

All trackers were initialised on the first frame of the video with bounding box co-ordinates of the object. Original implementations of RKLT [82], NNIC [23], IVT [67], TLD [42] and L1-APG [57] were used. One pass evaluation is done on the videos, with the tracker running the full length of the video.

Registration Based Among registration based trackers, we test ESM [9], NNIC [23], GNNIC (Chapter 3), IC [6] and RKLT [82]. 30 iterations with a resolution of 100×100 were used in IC for convergence. NNIC used a look up table of (0.06×0.04), (0.03×0.02) and (0.015×0.01) sigmas, each using 4000 samples and a resolution of 50×50. The corresponding IC used in the cascade setup, used 10 iterations with a resolution of 50×50. ESM used 30 iterations with a resolution of 50×50. RKLT used a template size of 40×40, with 10 iterations in IC for refinement. The parameters were chosen to give each tracker approximately the same execution time. Finally, we test SGANNS agorithm in GNNIC, using the same parameters as for NNIC, with degree of connectivity ($\mathbf{k}$) for graph construction set to 200. This gives a common baseline to compare the two related search methods of randomized kD-Trees and SGANNS. Two distance measures are used with GNNIC, Eucleadian distance (l_2) and Manhattan distance (l_1). Additionally, all four apperance models discussed in Chapter 3 are studied in conjugation with the registration based trackers.

Online Learning Based Three online learning based trackers (OLT) were studied: IVT [67] , L1-APG [57] and TLD [42]. IVT and L1-APG use 6 DOF particle filter as the search method. TLD uses median flow motion model. Yang et al. in [78] evaluated some of the state of the art trackers that use online learning to adapt the appearance model. We choose three of the trackers that performed well on most of the videos in their tests. Both IVT [67] and L1-APG [57] in our experiments used 600 particles for the particle filter search. Additionally, L1-APG [57] used a template subspace of 10 templates of the target image. Original parameters as specified in [42] were used for TLD.

A diverse selection, such as this, would show the usability of some of the popular trackers in application requiring precise tracking like manipulation and visual servoing. This is missing in the literature, since OLT benchmarks does not impose high convergence.

5.2 Result and Analysis

Overall Success Overall Success, averaged over three speeds (very slow, slow, normal) is reported in Table 5.1. It uses SCV as the appearance model. Later in Chapter, we study the effect of appearance variation on tracking performance. The best performing tracker for each sequence is shown in bold.

Table 5.1: Overall Success Expressed as Fraction of Frames Tracked

Video	**TLD**	**L1APG**	**IVT**	**ESM**	**NNIC**	**IC**	**RKLT**	**GNNIC**
Cereal	0.00	0.24	0.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Book I	0.00	0.10	0.48	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Book II	0.00	0.79	0.30	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Book III	0.00	0.42	**0.72**	0.34	0.32	0.32	0.38	0.34
Juice	0.00	0.16	0.98	**1.00**	0.41	**1.00**	**1.00**	**1.00**
Mug I	0.84	0.10	0.91	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
Mug II	0.41	0.30	0.72	0.89	0.89	0.89	0.31	**0.92**
Mug III	0.51	0.54	0.68	**1.00**	0.59	0.65	**1.00**	0.68
Bus	0.37	0.57	0.94	**1.00**	0.99	0.96	0.81	**1.00**
Highlighting	0.00	0.67	**0.95**	0.76	0.70	0.33	0.72	0.60
Letter	0.11	0.19	0.25	**1.00**	**1.00**	**1.00**	**1.00**	0.89
Newspaper	0.00	0.61	0.92	**1.00**	0.51	0.43	**1.00**	0.24

Table 5.1 clearly shows that registration based trackers have better overall performance compared to OLT. OLT are ill suited for manipulation tasks. They build an appearance model of the object (lower dimensional subspace of the initial template(s)) to begin with. This model is then updated as tracking progresses. Newer templates are accounted for in the appearance of the object. Though this makes the trackers more robust, this comes at the expense of convergence. Newer templates that are shifted, scaled from the original, are not exactly the same as the target image.

We observe that L1-APG tracker fails to track in-plane rotation (Fig 5.1). Although Mei et. al [57] use a 6 d.o.f motion model, the state space update mostly accounts for two DOF. The motion model uses a velocity component $\tilde{\mathbf{v}} = (v_1, v_2)$ (horizontal and vertical velocities), which are the last two terms of the state vector $\mathbf{p}$. Particles are updated based on average of the last few frames tracked. This accounts for the transitional motion of the object but fails to capture the full pose. It is not reported in any existing work that we know of. TLD, with it's 3 d.o.f. parametrization, understandably fails to precisely estimate the object pose.

We further analyse registration based trackers. We test them on higher speed videos (fast and very fast from TMT [68] Dataset). RKLT [82] and

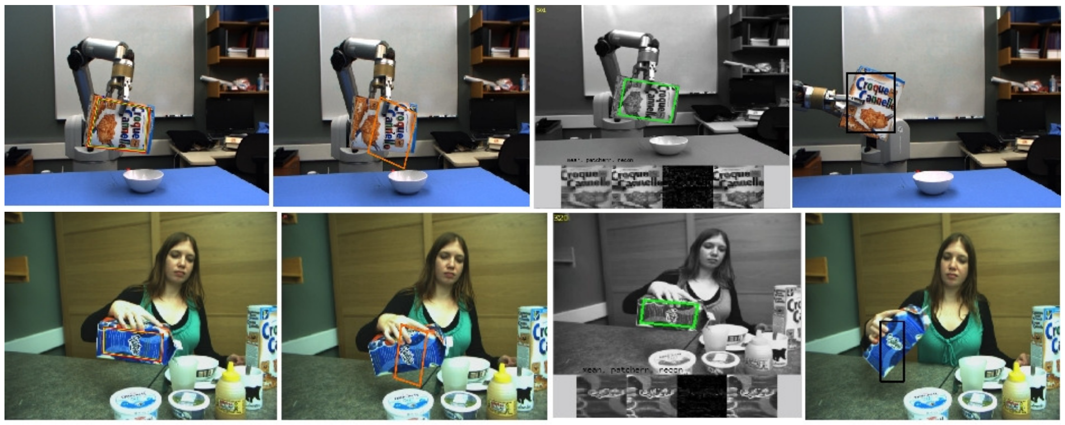


Figure 5.1: L1-APG (orange) [57] and TLD (black) [42] fails to track in plane rotation. Registration based trackers, ESM (red) [9], IC (yellow) [6] and NNIC (cyan) [23] and IVT (green) [67] tracks without considerable drift

GNNIC are the two best performing trackers on higher speed object motion. RKLT [82] tracker uses several point trackers (KLT trackers), that it initialises on each frame. This is then followed by RANSAC to estimate pose. This allows the tracker to handle large pose variation as long as at leastl some inliers are detected by RANSAC to estimate homography (Algorithm I in [82]). NNIC [23] and GNNIC use a two step search process. They model the possible pose variations prior to tracking. With sufficient number of samples, large motion variations can be modelled. Fine alignment following the Approximate Nearest Neighbour step is done using Inverse Compositional [6] search in NNIC [23] and GNNIC. The two search methods are complementary to each other, thus improving robustness maintaining high convergence.

Both NNIC [23] and GNNIC use Approximate Nearest Neighbour, which are computed using two different algorithms. We compare these two algorithms on high speed videos since for slower speeds, both track the full sequence, which makes it hard to distinguish. Figure 5.3 shows the result. The red boxes indicate those sequences where GNNIC is better than or equal to NNIC [23]. Apart from the one exception of “mugI” on very fast speed (“mugI-s5”), GNNIC is better than NNIC in terms of overall success. Fur-

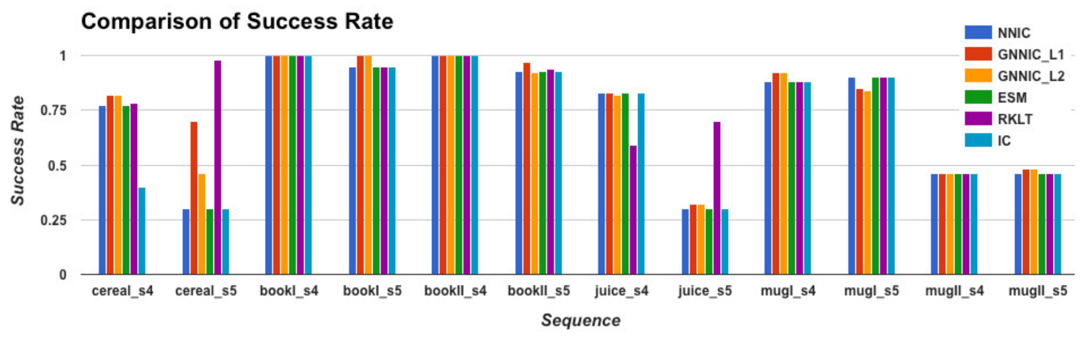


Figure 5.2: Overall Success of six registration based tracker are plotted. The six trackers being ESM [9], IC [6], RKLT [82], NNIC [23] and two variants of GNNIC one using Eucleadian distance (GNNIC-L2) and the other using Manhattan distance (GNNIC-L1) on 12 video sequences from TMT Dataset [68].

ther experiments on high speed object motions are shown and analysed in the next Section 5.2.

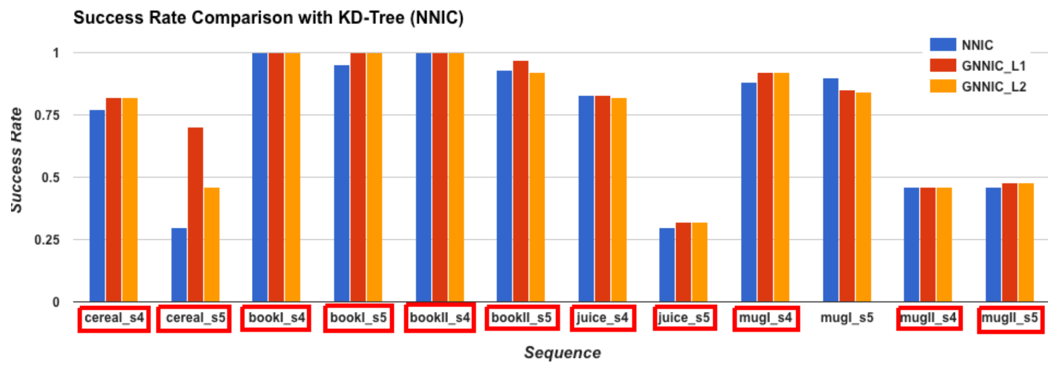


Figure 5.3: Overall Succes (OC) of NNIC [23] and the two variants of GNNIC on the higher speed videos from TMT Dataset [68].

Both the algorithms (NNIC and GNNIC) map warps to sampled images to model possible transformations before tracking starts. The efficiency of the system depends on how many samples are needed to properly model the object. We further test the two algorithms to verify this. KD-Tree [8] based NNIC and SGANNS based GNNIC are compared without using the cascade framework IC [6], to test the contribution of the approximate nearest neighbour part of the algorithm. Overall success is plotted against varying threshold t_p. SGANNS using 5000 samples perform comparable to ANN (for low pixel thresholds used in manipulation) using much higher particles ($>$ 10,000 particles). Figure 5.4 shows the comparison. Number of samples needed to appropiately represent data is much less in SGANNS which in turn makes the system more efficient. The result validates that keeping the same setting (number of particles and resolution of templates) SGANNS has better overall accuracy for most videos compared to KD-Tree in Figure 5.3.

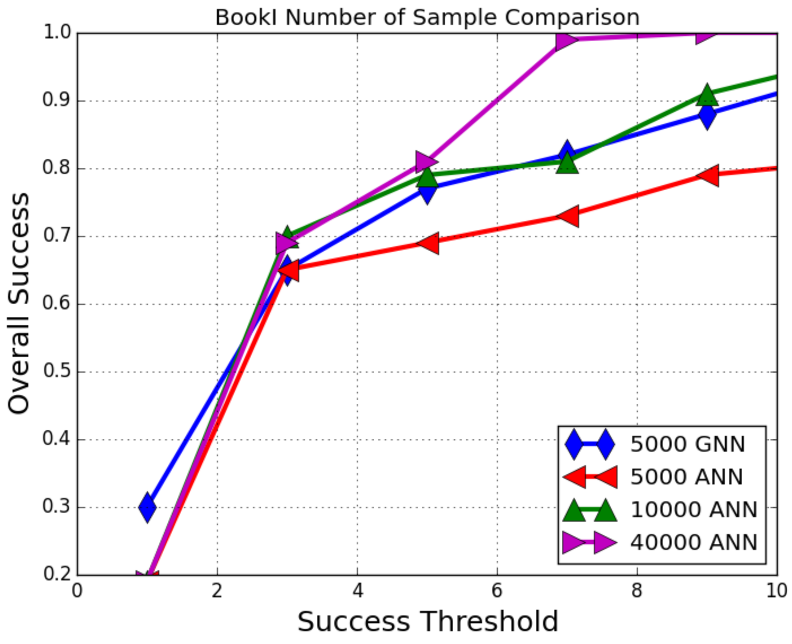


Figure 5.4: Overall Success is plotted with varying threhold t_p for different number of samples on "BookI" sequence. Results show that GNNIC performs well with much lesser samples compared to NNIC [23].

Not only does GNNIC require less number of samples to model the object, it speeds up search as pointed out in [33]. The speed up using SGANNS is measured based on the number of distance computations during search, to find the optimal parameters.

The reason for using such a method to compare speed is because KD-Tree is implemented using Flann [1], which is a more optimised version. The number of distance computations are shown in the Table 5.2 on six videos.

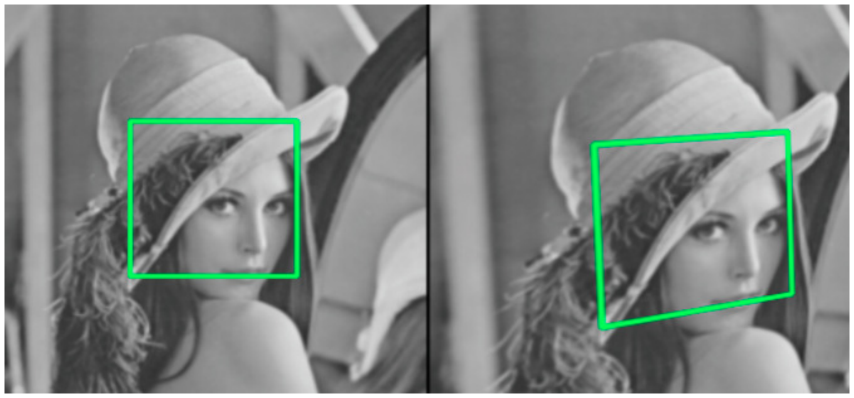
[1] Available at: https://github.com/mariusmuja/flann

Table 5.2: Speed Comparison expressed as number of distance computations.

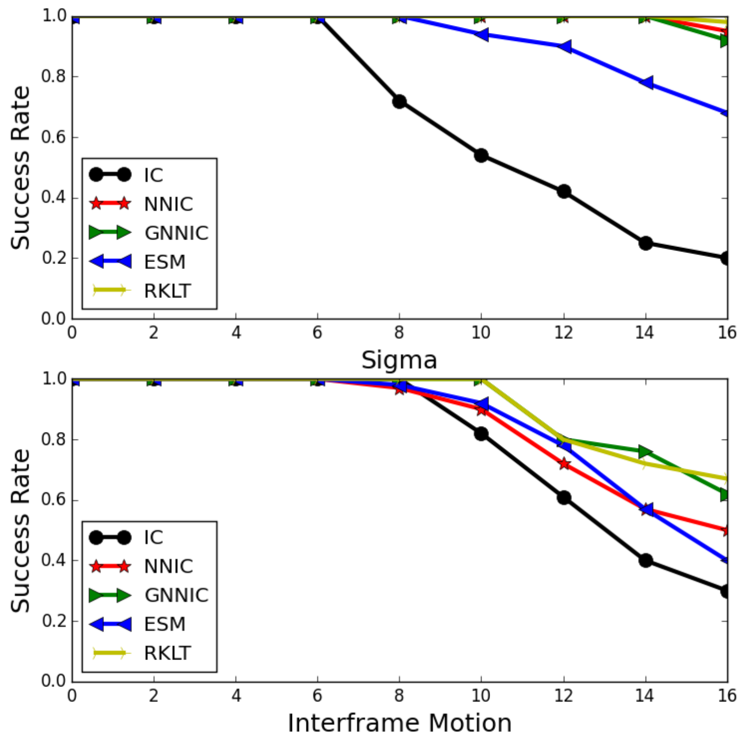
Video	**BookI**	**BookII**	**MugI**	**MugII**	**Juice**	**Cereal**
NNIC	1412	1216	811	821	2107	1781
GNNIC	922	651	408	411	732	691
Speed up	1.53	1.86	1.98	1.99	2.87	2.57

All results are reported on medium speed. **Speed up** is the ratio of the number of distance computations needed in NNIC compared to that of GNNIC. The speed is almost double in most cases. This is explained by the fact that having a smart criteria to initialise search (start traversing the graph **G**, from the best node in the previous search) reduces the number of distance computations.

Speed Sensitivity Evaluation measure Speed Sensitivity, introduced in Section 4.4.3, studies tracking performance on fast object motion. This is important for registration based trackers, which are often used in manipulation tasks, that have a wide range of motion. Originally, fast object motion was simulated by warping the standard "Lena" image (Figure 5.5) using a random warp, followed by the tracker retrieving this warp. Success Rate (OC) over a large number of trials (5000) for each sigma is plotted with increasing sigma.



(a)

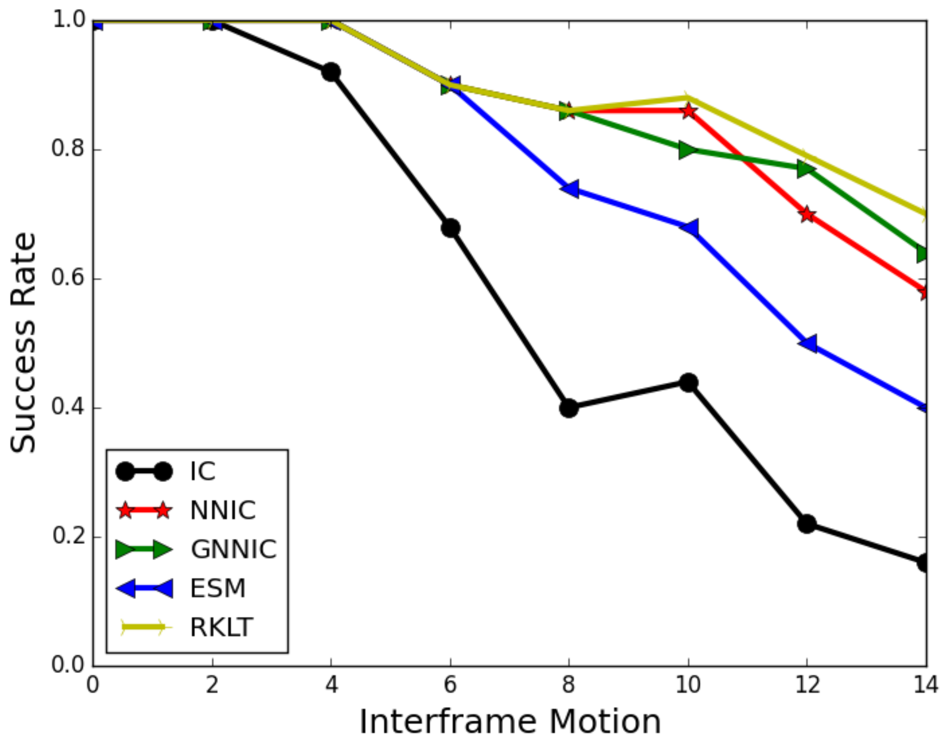


(b)

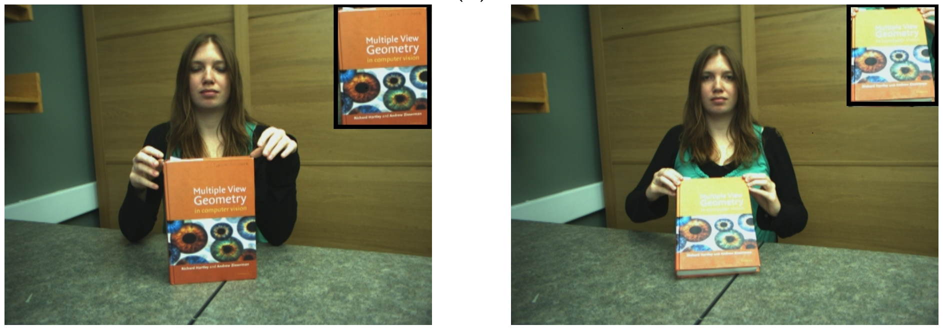
Figure 5.5: (a) Original "Lena" image that is morphed and tracked and (b) Speed Sensitivity experiments on the same image.

As reasoned in Section 4.4.3 and shown in Figure 4.6 , this is not an accurate estimate of a tracker's ability to handle large motion. This is further shown in Figure 5.5 (b). All five registraton trackers are tested using the original Static Experimental setup in [6, 9, 23] and the reported metric *Speed Sensitivity* in Chapter 4. Registration based trackers are in general are susceptible to large motion however speed sensitivity show their relative performance on large motion, with RKLT [82] and GNNIC the two best trackers. Note that the success plots are very similar when plotted against sigma but show a high degree of variability when plotted against interframe motion. RKLT [82] has the best performance with GNNIC coming close second. The trackers are tested further on real motion sequences. Speed sensitivity is reported taking into account all five speeds. We have made ground truth data publicly available so users can define their own evaluation criteria. *Speed Sensitivity* results are grouped under two object categories i.e. *Planar Object* and *Curvillinear Object*.

Planar Objects: "BookII" sequence is the easiest to track. The only challenge is scale (SC) variation. Compared to "BookII", "BookI" (Fig 5.6) has both perspective transformation (PR) and specularity (SR).



(a)



(b)

Figure 5.6: Speed Sensitivity result on "bookI" sequence. Frame 1 and Frame 190, in the "bookI" sequence, recorded at medium speed. The considerable change in appearance due to specular reflection is visually shown in top right corner

In this sequence, when the book is tilted, (Fig 5.6) the appearance changes considerably because of specular reflection from the surface of the book as shown in Figure 5.6 (b). This makes it hard to track. Another interesting observation can be made (Fig: 5.7), though “Cereal” and “Juice” sequences have same sets of challenges, the “Juice” sequence is more sensitive to large motion.

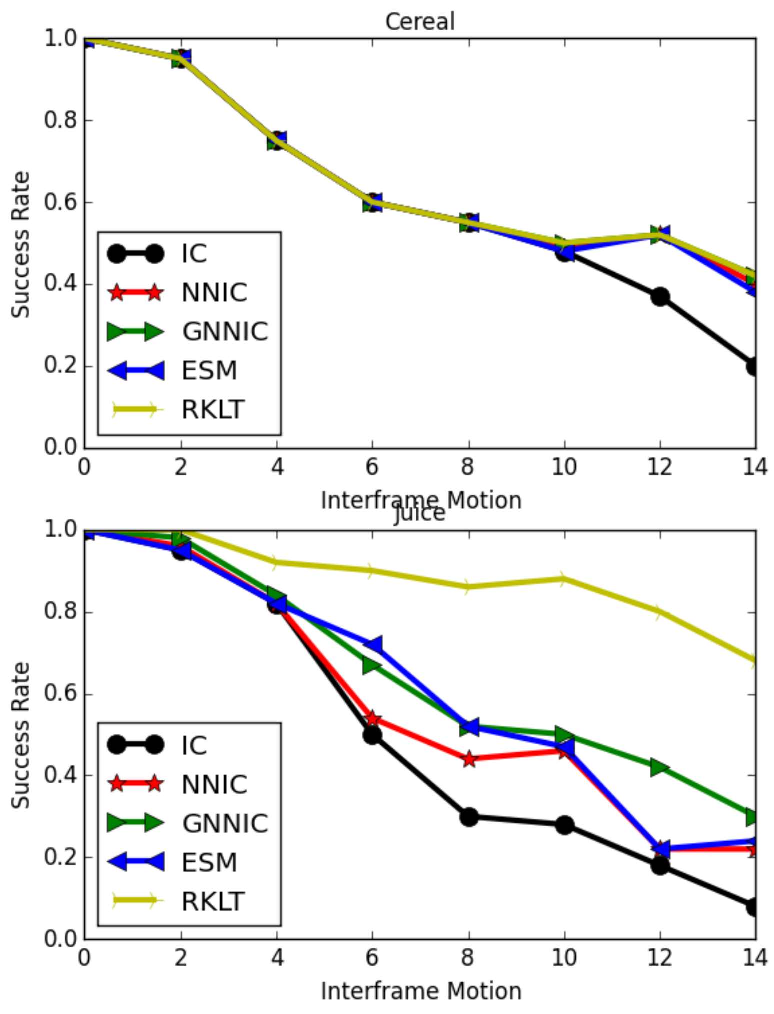


Figure 5.7: Speed sensitivity is plotted against inter frame motion for “Cereal” (top) and “Juice” (bottom) sequences

The only difference between them being texture, we investigate further.

Static image experiments with optimal parameters are done. The results in Fig 5.8 show the effect of texture playing a crucial role, making the two sequences distinct in their own way. Cereal box, having richer texture, is easier to track.

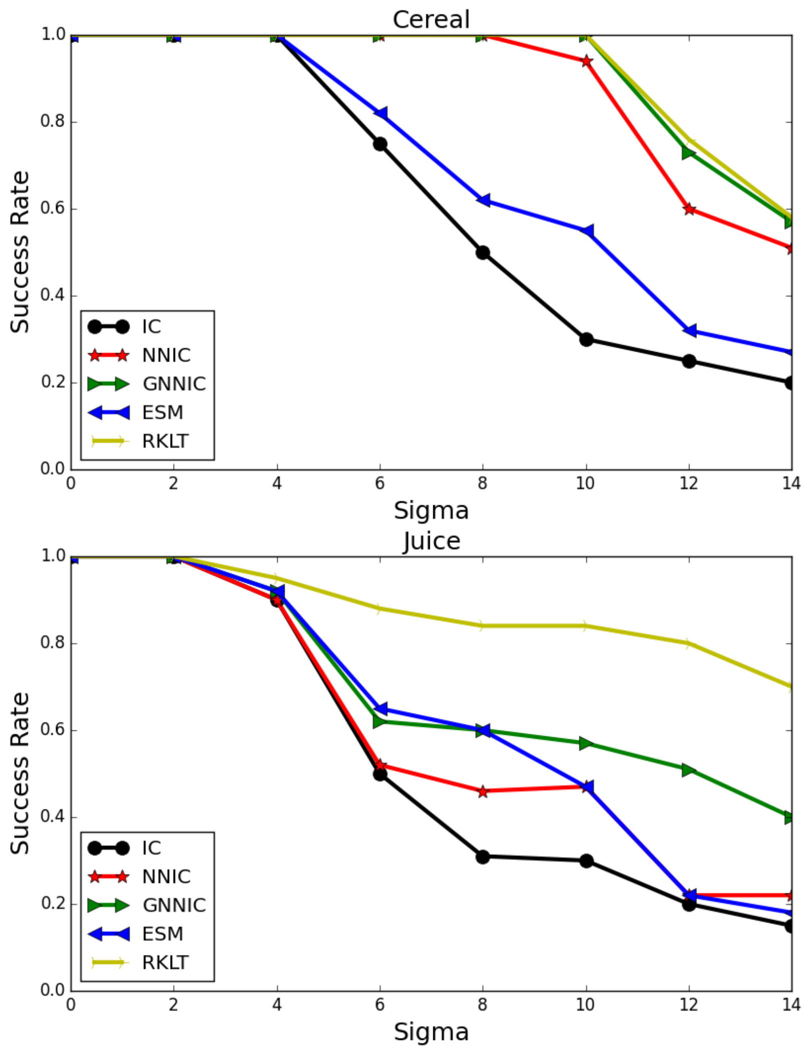


Figure 5.8: Static experiment with 5000 trials for each sigma ($\sigma \in \{1, ..14\}$), is done on the cereal box (top) and juice box (bottom), cereal box, having a rich texture, is easier to track.

Curvilinear Object: The mug sequences are on average more difficult to track. With higher inter frame motion the success rate falls steeply. The specularity (SR) on the surface makes it even hard to track. "MugII" is the most difficult sequence in the entire set, with the highest number of chal-

Table 5.3: Average Drift Expressed as an Expectation Value

Video	TLD	L1-APG	IVT	ESM	NNIC	IC	RKLT	GNNIC
Juice	N/A	2.39	3.27	0.79	0.89	0.94	**0.73**	1.57
Cereal	N/A	2.67	2.80	0.61	0.61	**0.15**	0.63	0.64
Book I	N/A	2.87	2.96	0.73	0.73	**0.70**	0.92	1.57
Book II	N/A	3.32	3.37	**0.70**	**0.70**	**0.73**	0.82	1.67
Book III	4.97	1.21	1.14	**1.38**	1.41	1.49	1.45	1.41
Mug I	3.48	3.3	1.34	0.49	0.49	**0.46**	0.50	0.90
Mug II	3.64	2.47	3.09	1.87	1.88	1.870	1.87	**0.74**
Mug III	3.20	2.51	2.28	0.74	0.97	0.75	0.75	**0.42**
Bus	3.25	0.68	2.18	0.63	**0.59**	0.63	1.20	1.20
Highlighting	N/A	3.71	1.61	1.23	1.21	**0.47**	1.06	1.21
Letter	4.53	1.79	1.68	**0.36**	0.505	**0.36**	1.1	0.89
Newspaper	N/A	2.49	3.18	0.42	0.53	**0.31**	1.63	0.92

N/A = There were no frames that had an error (E_{AL})less than 5 pixels

lenges (Table 4.2).

Average Drift Average Drift as described in Chapter 4 is calculated on all the sequences and reported in Table 5.3. Registration based trackers have low average drift compared to the online learned trackers for most of the sequences. The reason being, particle filter uses random samples to get an initial state estimate. This is unlikely to hit the best alignment when iterations are limited, while the Gauss Newton method of the original registration trackers provides fast and higher convergence when it does converge. IC tracker has the least drift in most of the sequences. This prompted us to use IC as the precise tracker in conjugation with the Approximate Nearest Neighbour based trackers which in itself does coarse search that often does not converge well.

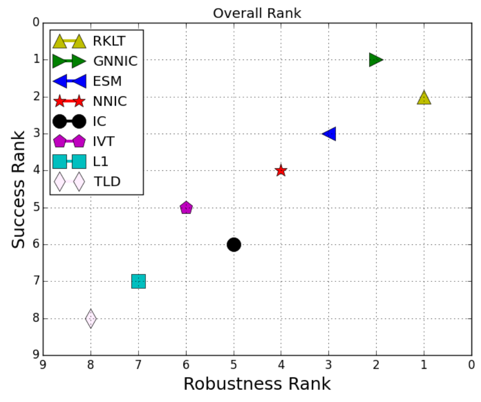
Overall Rank: We rank the eight trackers in Figure 5.9, similar to AR plots in [46] . This gives a global overview of how the trackers compare. The methodology is described in Chapter 4. RKLT [82] ESM [9] and GNNIC are the 3 top performing trackers. Approximate Nearest Neighbour based trackers have higher success rates compared to ESM due of their ability to track larger object motion (Fig 4.6). The actual ranks of trackers on two attributes are shown in Table 5.4 and 5.5.

Table 5.4: Robustness Rank of Trackers

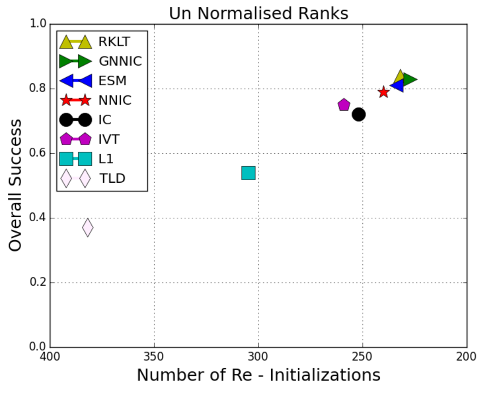
Robustness Ranks	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Trackers	RKLT	GNNIC	ESM	NNIC	IC	IVT	L1	TLD

Table 5.5: Success Rank of Trackers

Success Ranks	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
Trackers	GNNIC	RKLT	ESM	NNIC	IVT	IC	L1	TLD



(a)



(b)

Figure 5.9: Rank plots are shown. Trackers that are closer to the top right corner perform better in both metrices of robustness and overall success. The absolute rankings are shown in tables 5.4 and 5.5. The two plots are, a) Absolute rank plots, and (b) Un-Normalised rank plot that shows the actual values of re-initializations and overall sucess.

Figure 5.9 shows the rank plot and the un-normalised rank plots. The un-normalised plot (Figure 5.9 (b)) indicates that the top performing trackers (upper right corner of graph Figure 5.9 (a)) are similar in performance (Figure 5.9 (b) top left corner).

RKLT [82] initialises KLT trackers on every frame, which makes it robust to fast motion and thus it has the best overall success rate. TLD is the worst among the eight trackers, when used for precise tracking. It tracks 3 DOF state space and re-initialises often. IVT is the best among the Online Learned Trackers because of the 6 DOF motion model.

An important reason for the trackers to perform robustly is attributed to the appearance model of the object. Previously we tested using SCV as the appearance model. It corrects for illumination and hence tracks in the presence of specular reflection. In the next subsection we study the effect of appearance models and how it affects tracking performance.

5.2.1 Effect of Appearance Models in Tracking

Trackers are often subjected to changing appearance of the object over the course of a video. Hence different appearance models are used in registration based trackers, to correct for illumination variation. Here we study four illumination correcting appearance models described in Chapter 3. We show average success rate on six videos, averaged over all the different speeds and search methods in Figure 5.10. IVA [61] is only slightly better than SSD. IVA [61] was originally designed to eliminate the effect of shadows in autonomous navigation. A downside of this is it removes considerable texture information from the object to be tracked, which is important in gradient based search. This makes tracking difficult, particularly in the case of low textured objects like “Juice” and “Mug” sequences. Figure 5.11 show sample transformed images. This is not the case for “Book” sequences. Image of the book still retain lot of the original texture after the transform.

NCC and SCV have similar performances in terms of Overall Success. However NCC is faster than SCV. Using NCC as the appearance model is equivalent to using SSD on normalised image [69] as opposed to calculating the expected intensity value in SCV. The expectation operator in SCV has a complexity of $O(n^2m^2)$, where $(n\times m)$ is the dimension of the template. SSD, NCC and IVA, all three have a complexity of $O(nm)$. Table 5.6 presents a

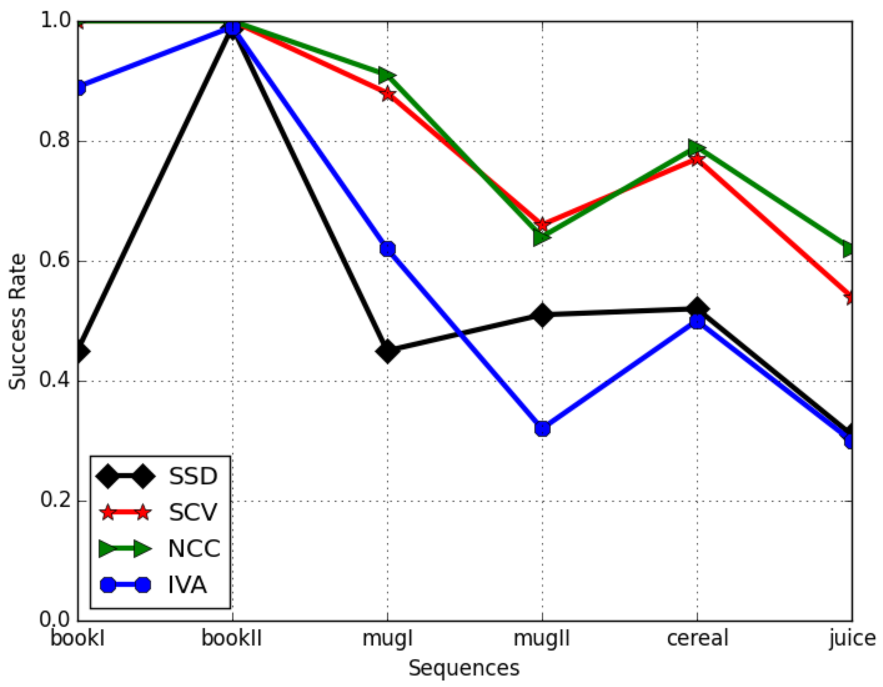


Figure 5.10: Average Success Rate plotted for the four appearance models discussed in Chapter 3 on six sequences from TMT dataset [68]

list of challenges and the variants of each sub modules that can handle the challenge.

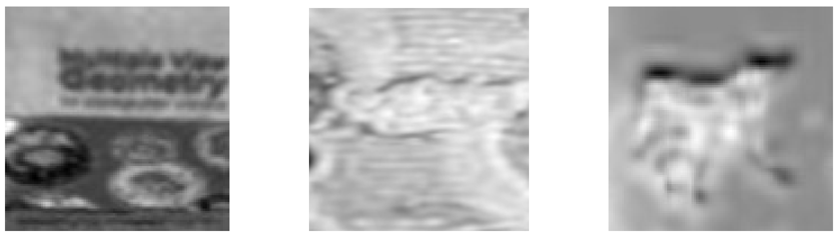


Figure 5.11: Sample images of tranformed template image using IVA [61]. Very little texture is retained in most cases which makes tracking using this appearance model difficult.

NCC being both fast and accurate is the preferred choice of appearance model.

Table 5.6: Summary of usefullness of variants of the submodules

	Appearance Model				**Search Method**			**State Space**				
Challenge	**NCC**	**SSD**	**SCV**	**IVA**	**GN**	**ESM**	**GNNIC**	**2**	**3**	**4**	**6**	**8**
Translation	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Rotation	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✓
Scale	✓	✓	✓	✓	✓	✓	✓	✗	✗	✓	✓	✓
Affine	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✓	✓
Perspective	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗	✓
Perspective + Specularity	✓	✗	✓	✗	✓	✓	✓	✗	✗	✗	✗	✓
Perspective + Large Motion	✓	✓	✓	✗	✗	✓	✓	✗	✗	✗	✗	✓
Perspective + Large Motion + Specularity +	✓	✗	✓	✗	✗	✓	✓	✗	✗	✗	✗	✓

(✓) shows the ability of the variant of each submodule to track a challenge and (✗) shows if it fails to do so.

Chapter 6

An application to track in the presence of Occlusion

Unlike TLD [42], the trackers discussed in this thesis do not have any detection module. Which means, once a tracker looses track, it never re-initailises. This is a bottleneck when faced with occlussion or objects that have very little texture exposed at the beginning of the sequence. Note that, even though OLT adapts for changing appearances, they rarely re-initialize once tracking fails. To counter this, we propose a detection system that works together with the tracker and re-initialises when it has drifted substancially. Workflow of the full system with the tracker is shown in Figure 6.1. Alternatively the detection module can also be used to initialize tracking, if the object to be tracked is known ahead of time.

Object template ($\mathbf{I}^*$) initialised by the user is used to build the initial model of the object $\mathcal{O}$. `keypoints` (positional information) and the corresponding `feature descriptors` are extracted from $\mathbf{I}^*$. We use SIFT [51] features. This forms the initial object model, $\mathcal{O}$. As shown in Figure 6.1 we check for tracker failure (blue box) at every image frame. The tracker is considered to have failed when residual error ($||\mathbf{I}^* - \mathbf{I}(\mathbf{W}(\mathbf{x}; \mathbf{p}*))||_2 > TH$) is above a certain threshold TH. We choose TH to be 20. If the tracker is judged to have failed, the detection system (green box) detects the object and re-initialises the tracker.

Feature descriptors from the new image (frame where the tracker drifted)

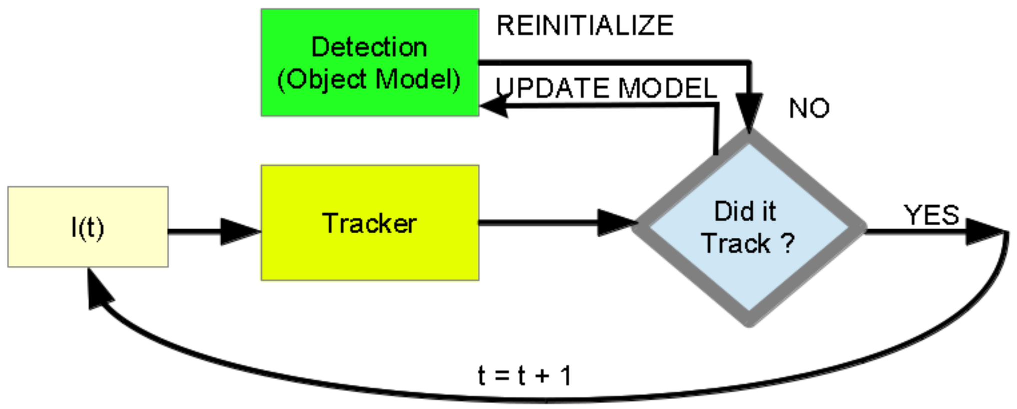


Figure 6.1: Worflow of the detection system together with the tracking module. When the residual is higher than a threshold, the object is detected and tracker re-initialised. The object module is updated with every re-initialization taking into account he new object template.

are matched with the existing object model using Approximate Nearest Neighbour [8]. The matched keypoints are further filtered using RANSAC [25]. RANSAC checks every combination for the best fit with the existing object model, $\mathcal{O}$. Since RANSAC is computaionally expensive, Approximate Nearest Neighbour matching is done prior to RANSAC, narrows down the number of potential points to be checked. Finally, we compute homography between the keypoints in the object model and the matched keypoints from the image frame. Homography parameters give the new bounding box co-ordinates of the object.

This is used to re-initialise the tracker. Note that if the object is out of focus, there are no matched keypoints, the detection system checks the next frame. With every re-initialization, the object model is updated with the matched keypoints ($goodKeypoints$ in Algorithm 5). Figure 6.2 shows the typical steps when the tracker fails. In Figure 6.2 (a) SIFT keypoints are generated, (b) matched keypoints, within a certain thereshold t (distance ratio) are filtered out, finally (c) shows RANSAC selecting a handful of keypoints (purple) to estimate homography parameters to give the new bounding box shown in red.

Algorithm 5 Detection Module

```
1:  procedure DETECTION($\mathcal{O}, I$)          ▷ $\mathcal{O}$=(reference keypoints, feature descriptors), I = Image
2:      $goodKeypoints = \{\}$
3:      $(keypoints, descriptors) = SIFT(I)$
4:      for keypoint in keypoints do
5:          $dist = nnsearch(\mathcal{O}, keypoint)$ ▷ Approximate Nearest Neighbour
6:          if $dist < t$ then
7:              $goodKeypoints.append($reference keypoint, keypoint$)$
8:          end if
9:      end for
10:     $RANSAC(goodKeypoints)$                       ▷ Filters $goodKeypoints$
11:     $POSE = HOMOGRAPHY(goodKeypoints)$
12:     $UPDATE(\mathcal{O})$
13:     return $POSE$
14: end procedure
```

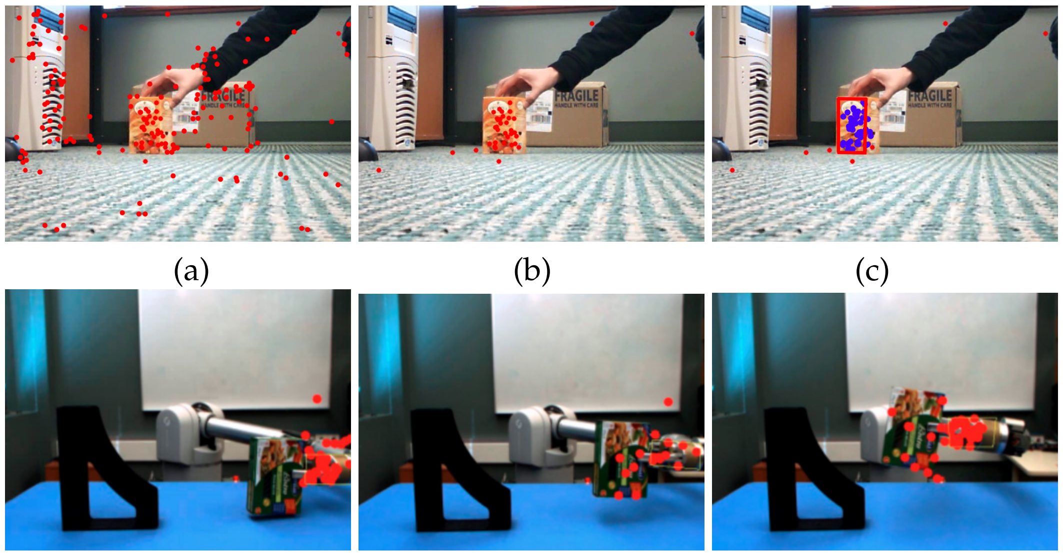


Figure 6.2: [TOP IMAGE] (a) Detected keypoints on an Image Frame (red points), (b) Keypoints that match closely with the object model $\mathcal{O}$, (c) Filtered keypoints after RANSAC in purple, used to calculate homography parameters. The resultant bounding box is shown in red. [BOTTOM IMAGE] End effector of the robot arm is tracked with the filtered keypoints.

Two examples are shown. The first example in Figure 6.3 shows occlussion from the book holder. Registration based trackers fail to track, but NNIC [23] coupled with the detection system re-initialises the tracker and it successfully tracks the object. The second example shows an instance of tracking very small objects. In this case a fish head. The position of the fish head is used as a reference to find the hook which is threaded. Being a very small object, it lacks significant texture. Other trackers fail very fast. The updating model of the object enables the tracker to track the fish head for a long period of time.

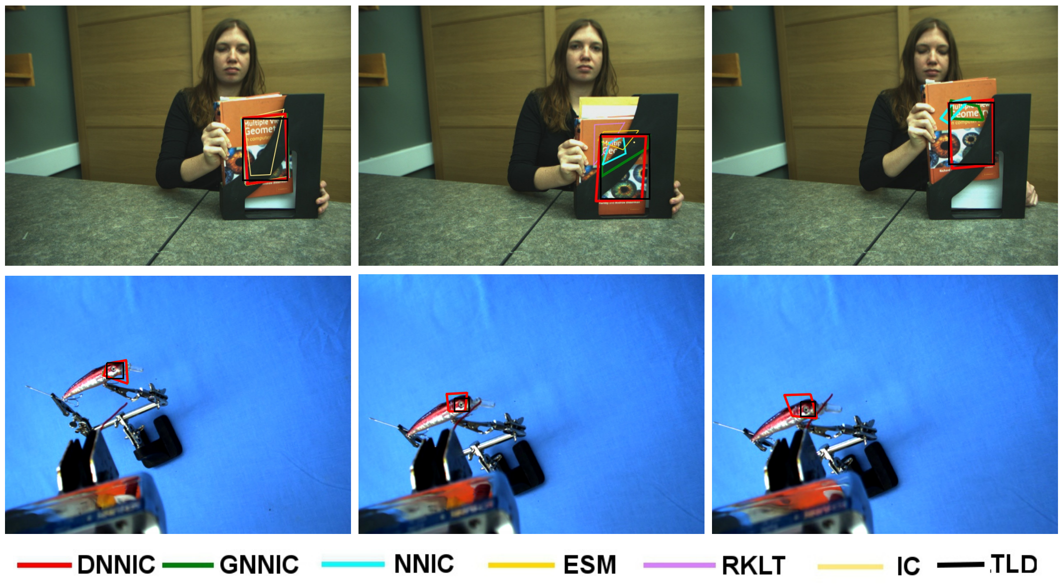


Figure 6.3: Examples frames where the tracker was re-initialized using the detection module. Seven trackers are shown, DNNIC refers to NNIC with the detection module (red), GNNIC, NNIC [23], ESM [9], RKLT [82], IC [6] and TLD [42]

TLD [42] is shown to track both the “Book” sequence and the small fish head in Figure 6.3. However, in the “Book” sequence, it fails to track full pose. This is becasue of the 3 DOF motion model used in TLD. The proposed detection system tracks full pose yet maintaining high robustness by re-initializing the tracker when it fails, hence is better suited in robotic applications.

Chapter 7

Conclusion and Discussion

7.1 Contributions

In this thesis we address registration based tracking. While tracking evaluation methodologies exist in literature, little attention has been paid to evaluate registration based trackers. We report an extensive dataset for that purpose on which we evaluate a diverse set of trackers and finally rank them. A new evaluation measure, Speed Sensitivity, is introduced that tests a tracker on large object motion.

We adapt a new search method, SGANNS [33], in tracking. It uses the sequential property of video data to efficiently search for the next best warp parameters, using the current warp as the starting node. The algorithm is explained in Chapter 3. Detailed results and analysis of the algorithm is presented in Chapter 5. Here we also study four appearance models in relation to registration based trackers. This is useful in making the tracker robust to illumination changes.

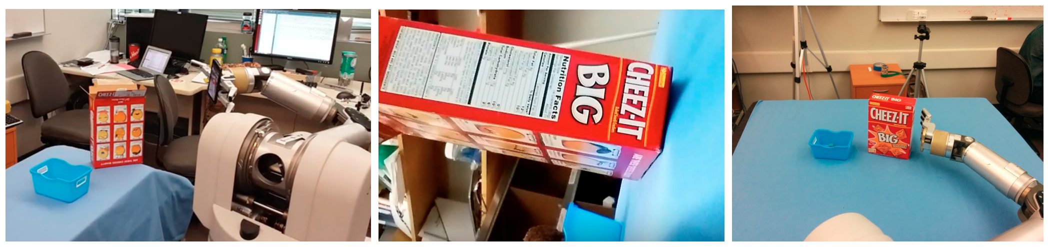
Interesting observations are revealed in our study. Firstly, results show L1-APG [57] tracker fails to track in plane rotation. This is attributed to the motion model used that primarily updates the translational parameters. Secondly, we observe that registration based trackers overall perform better than OLT. This explains the popular choice of registration based trackers in manipulation and visual servoing tasks. Thirdly, we show SGANNS to be more efficient compared to KD-Tree in tracking, with acheiving better accuracy with less number of samples to model motion. Fourthly, we study different appearance models. NCC is shown to be have the best trade off in terms of speed and accuracy. It corrects for illumination by centering the data.

Finally we describe a detection system in Chapter 6. This, coupled with a registration based tracker, re-initialises the tracker when it drifts. We show applications in tracking occlussion, tracking small objects and objects that change appearance.

7.2 Future Work

7.2.1 Dataset Augmentation and Tracker Evaluation

One key area, is to augment the dataset with more videos. This can be done by including eye in hand camera or first person view camera [74] videos, which would include large scale and perspective changes in appearance of the object. Figure 7.1 shows a typical setup from three view points. The camera is attached to the palm of the robot hand as it grabs the box and puts its content in to the blue container. Tracking in such videos is more challenging due to considerable change in appearance of the object.



(a) (b) (c)

Figure 7.1: Three view points of the setup is shown. (a) An external camera shows the overall setup used for recording. (b) A camera attached to the palm of the robot hand provides egocentric view. (c) First person view is shown from a camera placed on top of the base of the robot.

Also, videos in the dataset were recorded using a fixed camera and moving object. In future we would like to record videos using moving camera and moving object at the same time. We evaluate eight trackers and analyze performance. However the dataset being publicly downloadable with ground truth data, others are encouraged to use it to evaluate their trackers.

7.2.2 Approximate Nearest Neighbour Search using Graph Structure

In this thesis we show SGANNS to work well with sequential data in tracking videos. Application in SLAM is shown in [33]. Future research on this is to work on ways to efficiently add and delete nodes to the connected graph which forms the model prior to tracking. This is important, as it would allow for the model to adapt to newer appearances of the object (newer observations added) without substancial memory usage (prune the directed graph and remove old observations) which are no longer useful.

Bibliography

[1] Wam arm by barett technologies.

[2] Amit Adam, Ehud Rivlin, and Ilan Shimshoni. Robust fragments-based tracking using the integral histogram. In *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 798–805. IEEE, 2006.

[3] K.Pauwels A.Pieropan, G.Salvi and H.KjellstrÃ¶m. A dataset of human manipulation actions, 2014.

[4] Shai Avidan. Support vector tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(8):1064–1072, 2004.

[5] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 983–990. IEEE, 2009.

[6] Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[7] John L Barron, David J Fleet, and Steven S Beauchemin. Performance of optical flow techniques. *International journal of computer vision*, 12(1):43–77, 1994.

[8] Jeffrey S Beis and David G Lowe. Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 1000–1006. IEEE, 1997.

[9] Selim Benhimane and Ezio Malis. Real-time image-based tracking of planes using efficient second-order minimization. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 943–948. IEEE, 2004.

[10] Marcelo Bertalmio, Guillermo Sapiro, and Gregory Randall. Morphing active contours. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(7):733–737, 2000.

[11] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[12] Michael J Black and Allan D Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, 1998.

[13] Gunilla Borgefors. Distance transformations in digital images. *Computer vision, graphics, and image processing*, 34(3):344–371, 1986.

[14] Jean-Yves Bouguet. Pyramidal Implementation of the Affine Lucas Kanade Feature Tracker: Description of the algorithm. Technical report, Intel Corporation Microprocessor Research Labs, 2001.

[15] Gary R Bradski. Computer vision face tracking for use in a perceptual user interface. 1998.

[16] Robert Collins, Xuhui Zhou, and Seng Keat Teh. An open source tracking testbed and evaluation web site. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 17–24, 2005.

[17] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Real-time tracking of non-rigid objects using mean shift. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 142–149. IEEE, 2000.

[18] Dorin Comaniciu, Visvanathan Ramesh, and Peter Meer. Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577, 2003.

[19] Timothy F Cootes, Gareth J Edwards, and Christopher J Taylor. Active appearance models. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (6):681–685, 2001.

[20] Amaury Dame. *A unified direct approach for visual servoing and visual tracking using mutual information*. PhD thesis, Université Rennes 1, 2010.

[21] Amaury Dame and Eric Marchand. Accurate real-time tracking using mutual information. In *Mixed and Augmented Reality (ISMAR), 2010 9th IEEE International Symposium on*, pages 47–56. IEEE, 2010.

[22] Alessio Del Bue, Dorin Comaniciu, Visvanathan Ramesh, and Carlo Regazzoni. Smart cameras with real-time video object generation. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 3, pages III–429. IEEE, 2002.

[23] Travis Dick, Camilo Perez Quintero, Martin Jägersand, and Azad Shademan. Realtime registration-based tracking via approximate nearest neighbour search. In *Robotics: Science and Systems*. Citeseer, 2013.

[24] Nicholas Dowson and Richard Bowden. Mutual information for lucas-kanade tracking (milk): An inverse compositional formulation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (1):180–185, 2007.

[25] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[26] Robert Fisher, Jose Victor, and James Crowley. Caviar: Context aware vision using image-based active recognition dataset.

[27] Steffen Gauglitz, Tobias Höllerer, and Matthew Turk. Evaluation of interest point detectors and feature descriptors for visual tracking. *International journal of computer vision*, 94(3):335–360, 2011.

[28] Michael Gleicher. Projective registration with difference decomposition. In *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pages 331–337. IEEE, 1997.

[29] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via on-line boosting. In *BMVC*, volume 1, page 6, 2006.

[30] Helmut Grabner, Jiri Matas, Luc Van Gool, and Philippe Cattin. Tracking the invisible: Learning where the object might be. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1285–1292. IEEE, 2010.

[31] Steve Gu, Ying Zheng, and Carlo Tomasi. Efficient visual object tracking with online nearest neighbor classifier. In *Computer vision–ACCV 2010*, pages 271–282. Springer, 2011.

[32] Gregory D Hager and Peter N Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(10):1025–1039, 1998.

[33] Kiana Hajebi and Hong Zhang. An efficient index for visual search in appearance-based slam. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 353–358. IEEE, 2014.

[34] Mei Han, Amit Sethi, Wei Hua, and Yihong Gong. A detection-based multiple object tracking method. In *Image Processing, 2004. ICIP'04. 2004 International Conference on*, volume 5, pages 3065–3068. IEEE, 2004.

[35] Sam Hare, Amir Saffari, and Philip HS Torr. Struck: Structured output tracking with kernels. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 263–270. IEEE, 2011.

[36] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[37] Wei He, Takayoshi Yamashita, Hongtao Lu, and Shihong Lao. Surf tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1586–1592. IEEE, 2009.

[38] Anil Jain, Karthik Nandakumar, and Arun Ross. Score normalization in multimodal biometric systems. *Pattern recognition*, 38(12):2270–2285, 2005.

[39] Omar Javed and Mubarak Shah. Tracking and object classification for automated surveillance. In *Computer Vision?ECCV 2002*, pages 343–357. Springer, 2002.

[40] Allan D Jepson, David J Fleet, and Thomas F El-Maraghi. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1296–1311, 2003.

[41] Frédéric Jurie and Michel Dhome. Hyperplane approximation for template matching. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(7):996–1000, 2002.

[42] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(7):1409–1422, 2012.

[43] Changick Kim and Jenq-Neng Hwang. Fast and automatic video object segmentation and tracking for content-based applications. *Circuits and Systems for Video Technology, IEEE Transactions on*, 12(2):122–129, 2002.

[44] Dominik Alexander Klein, Dirk Schulz, Simone Frintrop, and Armin B Cremers. Adaptive real-time video-tracking for arbitrary objects. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 772–777. IEEE, 2010.

[45] Bianca Kochner, Dietrich Schuhmann, Markus Michaelis, Gerd Mann, and Karl-Hans Englmeier. Course tracking and contour extraction of retinal vessels from color fundus photographs: Most efficient use of steerable filters for model-based image analysis. In *Medical Imaging'98*, pages 755–761. International Society for Optics and Photonics, 1998.

[46] Matej Kristan, Roman Pflugfelder, Ale Leonardis, Jiri Matas, Fatih Porikli, Luka Cehovin, Georg Nebehay, Gustavo Fernandez, Toma Vojir, Adam Gatt, et al. The visual object tracking vot2013 challenge results. In *Computer Vision Workshops (ICCVW), 2013 IEEE International Conference on*, pages 98–111. IEEE, 2013.

[47] Laura Leal-Taixé, Anton Milan, Ian D. Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *CoRR*, abs/1504.01942, 2015.

[48] Sebastian Lieberknecht, Selim Benhimane, Peter Meier, and Nassir Navab. A dataset and evaluation methodology for template-based tracking algorithms. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*, pages 145–151. IEEE, 2009.

[49] Liang Lin, Yongtian Wang, Yue Liu, Caiming Xiong, and Kun Zeng. Marker-less registration based on template tracking for augmented reality. *Multimedia Tools and Applications*, 41(2):235–252, 2009.

[50] Liang Lin, Yongtian Wang, Yue Liu, Caiming Xiong, and Kun Zeng. Marker-less registration based on template tracking for augmented reality. *Multimedia Tools and Applications*, 41(2):235–252, 2009.

[51] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[52] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[53] Hosam M Mahmoud, Reza Modarres, and Robert T Smythe. Analysis of quickselect: An algorithm for order statistics. *Informatique théorique et applications*, 29(4):255–276, 1995.

[54] Ezio Malis. Improving vision-based control using efficient second-order minimization techniques. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1843–1848. IEEE, 2004.

[55] Mario Edoardo Maresca and Alfredo Petrosino. Matrioska: A multi-level approach to fast tracking by learning. In *Image Analysis and Processing–ICIAP 2013*, pages 419–428. Springer, 2013.

[56] Stephen J McKenna, Yogesh Raja, and Shaogang Gong. Tracking colour objects using adaptive mixture models. *Image and vision computing*, 17(3):225–231, 1999.

[57] Xue Mei and Haibin Ling. Robust visual tracking using l1 minimization. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1436–1443. IEEE, 2009.

[58] Xue Mei and Haibin Ling. Robust visual tracking and vehicle classification via sparse representation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(11):2259–2272, 2011.

[59] A. Zisserman Michaelmas. Optimization lectures. In *B1 Optimization*, volume 2.

[60] Rainer Mobus and Uli Kolbe. Multi-target multi-object tracking, sensor fusion of radar and infrared. In *Intelligent Vehicles Symposium, 2004 IEEE*, pages 732–737. IEEE, 2004.

[61] Michael Paton, Kirk MacTavish, Chris J Ostafew, and Timothy D Barfoot. It's not easy seeing green: Lighting-resistant stereo visual teach & repeat using color-constant images. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 1519–1526. IEEE, 2015.

[62] Gord Peters. CoraindER: Ieee1394 camera gui. *Linux Magazine*, 69, 1999.

[63] Antoine Petit, Guillaume Caron, Hideaki Uchiyama, Eric Marchand, et al. Evaluation of model based tracking with trakmark dataset. In *2nd Int. Workshop on AR/MR Registration, Tracking and Benchmarking*, 2011.

[64] Mark R Pickering, Abdullah Muhit, Jennie M Scarvell, Paul N Smith, et al. A new multi-modal similarity measure for fast gradient-based 2d-3d image registration. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 5821–5824. IEEE, 2009.

[65] Nilanjan Ray, Scott T Acton, and Klaus Ley. Tracking leukocytes in vivo with shape and size constrained active contours. *Medical Imaging, IEEE Transactions on*, 21(10):1222–1235, 2002.

[66] Rogério Richa, Raphael Sznitman, Russell Taylor, and Gregory Hager. Visual tracking using the sum of conditional variance. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2953–2958. IEEE, 2011.

[67] David A Ross, Jongwoo Lim, Ruei-Sung Lin, and Ming-Hsuan Yang. Incremental learning for robust visual tracking. *International Journal of Computer Vision*, 77(1-3):125–141, 2008.

[68] Ankush Roy, Xi Zhang, Nina Wolleb, Camilo Perez, Quenterio, and Martin Jagersand. Tracking benchmark and evaluation for manipulation tasks. In *International Conference on Robotics and Automation*. IEEE, 2015.

[69] Lars Ruthotto. Mass-preserving registration of medical images. *German Diploma Thesis (Mathematics), Institute for Computational and Applied Mathematics, University of Münster*, 2010.

[70] Thomas B Sebastian and Benjamin B Kimia. Metric-based shape retrieval in large databases. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 291–296. IEEE, 2002.

[71] David W Shattuck, Gautam Prasad, Mubeena Mirza, Katherine L Narr, and Arthur W Toga. Online resource for validation of brain segmentation methods. *Neuroimage*, 45(2):431–439, 2009.

[72] Ihor Smal, Katharina Draegestein, Niels Galjart, Wiro Niessen, and Erik Meijering. Particle filtering for multiple object tracking in dynamic fluorescence microscopy images: Application to microtubule growth analysis. *Medical Imaging, IEEE Transactions on*, 27(6):789–804, 2008.

[73] Arnold WM Smeulders, Dung M Chu, Rita Cucchiara, Simone Calderara, Afshin Dehghan, and Mubarak Shah. Visual tracking: An experimental survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(7):1442–1468, 2014.

[74] Siddhartha S Srinivasa, Dave Ferguson, Casey J Helfrich, Dmitry Berenson, Alvaro Collet, Rosen Diankov, Garratt Gallagher, Geoffrey Hollinger, James Kuffner, and Michael Vande Weghe. Herb: a home exploring robotic butler. *Autonomous Robots*, 28(1):5–20, 2010.

[75] Björn Stenger, Thomas Woodley, and Roberto Cipolla. Learning to track with multiple observers. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 2647–2654. IEEE, 2009.

[76] Richard Szeliski. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2006.

[77] Tiesheng Wang, Irene YH Gu, and Pengfei Shi. Object tracking using incremental 2d-pca learning and ml estimation. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 1, pages I–933. IEEE, 2007.

[78] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. Online object tracking: A benchmark. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2411–2418. IEEE, 2013.

[79] Ning Xu, Narendra Ahuja, and Ravi Bansal. Automated lung nodule segmentation using dynamic programming and em-based classification. In *Medical Imaging 2002*, pages 666–676. International Society for Optics and Photonics, 2002.

[80] Junlan Yang, Dan Schonfeld, and Magdi Mohamed. Robust video stabilization based on particle filter tracking of projected camera motion. *Circuits and Systems for Video Technology, IEEE Transactions on*, 19(7):945–954, 2009.

[81] Alper Yilmaz, Xin Li, and Mubarak Shah. Contour-based object tracking with occlusion handling in video acquired using mobile cameras. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(11):1531–1536, 2004.

[82] Xi Zhang, Abhineet Singh, and Martin Jagersand. Rklt: 8 dof real-time robust video tracking combining coarse ransac features and accurate fast template registration. In *Canadian Conference on Robot Vision, 2015. CRV 2015*, 2015.

[83] Tao Zhao and Ram Nevatia. Tracking multiple humans in crowded environment. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–406. IEEE, 2004.

[84] Karel Zimmermann, Jiri Matas, and Tomas Svoboda. Tracking by an optimal sequence of linear predictors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 31(4):677–692, 2009.