A NEW VISUAL TRACKING ALGORITHM BASED ON TEMPLATE REGISTRATION FOR ACCURATE OBJECT TRACKING

by

Xi Zhang

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

Master of Science

Department of Computing Science

University of Alberta

Abstract

Visual tracking serves an important role in a wide variety of applications like video surveillance, robotic manipulation and augmented reality. The goal of tracking in the last two cases here is to efficiently and accurately locate the object in each frame of an image sequence/stream, with the target selected in the first frame. Various visual tracking algorithms have been proposed in literature recently, but many of them fall in the category of long-term object tracking algorithms that are more focused on the tracking robustness and are not accurate enough for such applications. On the other hand, registration based tracking algorithms can achieve greater accuracy in tracking high degree-of-freedom (DOF) object motion, but are likely to fail when fast object motion or noise in the image space is present. In this thesis, we focus on improving the robustness of registration based tracking algorithms towards fast motion and noise, while retaining good accuracy and efficiency. Concretely, we propose a novel tracking algorithm called RKLT that takes advantage of both 2D KLT trackers and the RANSAC algorithm for robust 8 DOF inter-frame target motion estimation. Inlier pixels selected by RANSAC are used to perform global registration using the efficient Inverse Compositional (IC) tracker to avoid tracking drift.

In addition, we also explore the different parameterizations on the state space model of a registration based tracker which characterizes the object state in 2D image space during tracking. In particular, we show how the corner based parameterization can be applied to the 8 DOF tracker using efficient second-order minimization (ESM). The impact of different parameterizations on the performance of IC and ESM trackers is also investigated in the experiments.

Finally, we introduce a new tracking dataset, Tracking for Manipulation Tasks (TMT) dataset with over 100 image sequences. New evaluation methods are also designed for better evaluation of high DOF trackers with greater accuracy. A tracking testbed is also provided for more convenient comparison among different tracking algorithms. In the experiments, the proposed RKLT algorithm performs better than three other registration based trackers, especially in the faster sequences of TMT dataset. In the public Metaio benchmark too, RKLT achieves better results than the ESM tracker which is considered the state-of-the-art.

Acknowledgements

First of all, I would like to express my sincere gratitude to my supervisor, Prof. Martin Jagersand, for his continuous support during my master's studies.

I would like to thank my committee members, Hong Zhang and Nilanjan Ray for spending time on reading and commenting my thesis.

In addition, I would like to thank my colleagues and friends in University of Alberta. Thanks Ankush Roy, Abhineet Singh, Vincent Zhang, Camilo Parez, Mona Gridseth, Oscar Ramirez and many others for their help and support.

Last but not least, I would like to thank my family for the love and support they provided me through my entire life.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Visual Tracking Algorithms

Visual tracking is one of the important topics in computer vision since it's wide use in different applications. Visual tracking algorithm aims to automatically estimate the position/pose of the selected target in image sequences/videos based on visual information. Based on the requirements of different applications, we can categorize different visual tracking algorithms into two categories. Trackers design for applications such as surveillance, human computer interaction, where challenges like object occlusion, lighting variation are common, can be categorized as robustness-oriented tracking algorithms. While for applications in vision based robotic manipulations, virtual reality, medical imaging where fine and accurate tracking results are required, trackers that fit these applications can be categorized as accuracy-oriented tracking algorithms.

Numerous visual tracking algorithms have been proposed in literature in the past decade. Many of them fall into the category of robustness-oriented tracking algorithms. Several of these algorithms are able to pre-train a robust appearance model [1, 40] and/or learn adaptive appearance models for the target during tracking [32, 4, 26]. Although these algorithms can achieve very robust tracking performance, in most cases they will fail to track complex object motion with high accuracy. Two reasons may explain this behavior: (1) The learning stage may bring in noise which fuzz the original target information initialized at the beginning of the tracking process. (2) There is a computational constraint that limit these tracking algorithms to adopt higher Degree-Of-Freedom (DOF) tracking state space models for accuracy purpose. For example, the MIL tracking algorithm [1] uses a greedy strategy to update the location of the target by searching an area that is within a certain radius of the location of the target at previous frame. This will be efficient if only x-y translational motion is considered but the computation may increase immediately when more complex motions need to be covered.

While there is still in high demand of accuracy-oriented tracking algorithms in scenarios such as vision aided robotic manipulations, virtual reality, medical image segmentation, etc. For example, by tracking the state of the target in real-time, the robot will be able to estimate the location and

Figure 1.1: Motion involved from left column to right column: Out-of-plane rotation, Scaling, In-plane rotation, Out-of-plane rotation. Purple tracker is a robustness-oriented tracker [32]. Cyan tracker is an accuracy-oriented tracker [41]

.

pose of the end-effector in image space, thus is able to perform visually guided control on robot manipulators. In Fig 1.1, a human is imitating the motion of a robot manipulator. Two different trackers are applied and the two sequences that containing out-of-plane rotation are hard for the robustness-oriented tracker [32] to track accurately. This kind of application requires the tracking algorithm to be highly efficient because the tasks performed by the robot are real time. In addition, trackers are also required to be able to track highly complex object motions well which may involves more than one type of motions simultaneously, e.g., translation, in-plane and out-of-plane rotations.

Our research focuses on tracking algorithms that can achieve high efficiency and high accuracy towards complex object motions. To be more specific, we focuses on the registration based tracking algorithms, which nicely meet the two requirements. The registration based algorithms always assume that there exists a warped version of the template/target in every frame and the goal is to extract the warping parameters (See Chapter 2.3). Instead of using local features of target such as edges or contours, which may fail easily when the appearance of target lack such features. In order to estimate the motion of the object, we will directly make use of image intensity information which contains complete target information. In most cases, this strategy can achieve more accurate results.

1.2 Thesis Motivation

Many applications of computer vision require high accuracy trackers as mentioned early. But there are few tracking algorithms that are qualified and can be directly used in such applications. For

example, the tracking algorithm based on brute-force approach directly search the warp parameter space to find the optimal target. However it's limited to use due to high computation requirement to track high DOF object motion. Some registration based algorithms like [29, 2], although are able to track high DOF object motions efficiently, will drift and fail easily. The reason is that they are sensitive to noise and can't handle cases like large inter-frame object motion, environmental interference due to occlusion, lighting variation, etc. This motivates us to do careful research on the existing tracking algorithms and design new algorithms that are able to track fast complex object motion efficiently with better robustness.

During the research on various tracking algorithms, we realize the importance of different evaluation methods in comparing among different tracking algorithms. Good evaluation methods should be able to clearly indicate the pros and cons of different tracking algorithms under different criteria. Thus, based on the evaluation results, we will be able to analyze on the weaknesses of these tracking algorithms for further improvement. It is also very beneficial to have well designed tracking datasets along with ground truths for us to evaluate tracking algorithms. Unfortunately, there are not so many datasets and evaluation methods that are designed for evaluating high accuracy tracking performance, which motivates us to design our own.

Finally, we would like to have a tracking system that is integrated with different tracking algorithms that are commonly seen in literature. There are two main reasons: Firstly, a integrated system makes it more convenient to compare new tracking algorithms with the ones in literature. The system can serve as a testbed for new-designed tracking algorithms and different evaluation methods can be applied easily. Secondly, the system can be easily adopted in the real applications in for example vision based robotic manipulations. Users are free to compare and choose algorithms that may fit in any particular application. This also inspire us to build a testbed-like tracking system.

1.3 Thesis Contribution

In this thesis, we focus our research on the registration based tracking algorithms that are able to track fast and complex object motions with high accuracy. Three main contributions are introduced.

- We propose a novel registration based tracking algorithm Ransac KLT (RKLT) that is able to track fast and complex object motion which may simultaneously involve translation, scaling, in-plane and out-of-plane rotations. Unlike many registration based trackers which directly estimate the object motion based on the whole target region, sub-region tracking of the target is firstly performed using multiple 2D KLT trackers. Tracking results are then used to estimate 8 DOF inter-frame motion via the RANSAC algorithm. Finally Inverse Compositional algorithm with the static target template is used to perform global registration in order to avoid tracking drift. The RKLT algorithm is able to filter the regions of the target template such that only reliable and easy-to-track regions will be used to extract object motions, which improves

the tracking robustness. In the experiments we show that the RKLT algorithm compares favorably with 3 other tracker including the state-of-the-art on two datasets, in both accuracy and robustness.

- We study the state space model of the registration based tracking algorithms and how it should contribute to the tracking performance. In particular, we study the parameterization of the state space model that is adopted by the ESM tracking algorithm [6], which is generally considered as one of the state-of-the-art in registration based trackers. In addition, we show how the corner based parameterization [19] can be applied to the ESM tracker. The new parameterization can also be adopted by other registration based tracking algorithms, and is also straightforward to be extended to other DOFs. In the experiment we investigate the impact of three different parameterizations on the performance of IC and ESM trackers. An integrated tracking system based on the original tracking system in [16] will also be introduced and discussed.

- We introduce a new dataset called Tracking on arm and hand Manipulation Task dataset (TMT), which involves a variety of object motions and also challenges like object motion blur, partial occlusion that tracking algorithms may face in real applications. We also propose two new tracking experimental settings, one for the modified static image experiment and the other for the speed sensitivity experiment. These two settings are able to indicate more accurate tracking convergence in both synthetic and real scenarios.[1]

1.4 Thesis Outline

The reminder of this thesis is organized as follows: Chapter 2 will cover the background and related work on 2D visual tracking algorithms, especially registration based algorithms in literature. Chapter 3 will introduce the new proposed RKLT algorithm. Chapter 4 will show a new type of parameterization for ESM tracker on modeling the target state in 2D image space and will also briefly introduce the integrated tracking system. Chapter 5 will describe the new dataset and evaluation methods we proposed and used in our experiments. Chapter 6 presents the experimental results and analysis. The conclusion and future work will be given in Chapter 7.

[1] In collaboration with Ankush Roy, Nina Wolleb and Camilo Perez

Chapter 2

Background and Related Work

Over the past few decades, research on visual tracking algorithms has been widely studied in computer vision area. This is mainly because of the increasing demand of different vision based applications and also the breakthroughs in new technologies like machine learning. Thus a variety of visual tracking algorithms have been proposed aimed for different applications. Among the tracking algorithms, many of them differentiate from each other in three main factors: the appearance model for tracking target modeling, the object state space model for modeling the target state (mainly location and pose) in 2D images and the object state estimation/search method [16]. In Section 2.1, we will first give an overview on some of the well known 2D visual tracking algorithms.

In order to meet the high accuracy requirements in tracking applications like vision aided robotic manipulation, we will then focus more in the registration based tracking algorithms. Following Section 2.1, two main factors that contribute to the accuracy of registration based trackers will be discussed: the object state space model and the search method. In Section 2.2, we will first introduce different state space models that are commonly used by a variety of tracking algorithms. In Section 2.3, several registration based tracking algorithms will be discussed, with a focus on the search methods. Finally several publicly available tracking systems that are integrated with tracking algorithms will also be reviewed and discussed.

2.1 Background

Shape/contour based tracking algorithm is very useful in tracking targets that may have less texture and/or is non-rigid, as long as there is significant contour. For example, in the application of human body tracking in gesture recognition and medical segmentation, this type of tracking algorithm is commonly used.

The contour based tracking algorithm always tries to track the curve that adheres to the outline of the target. In 1996 Isard et al. [23] adopted the contour information of the target as the appearance model. The appearance model is combined with the condensation algorithm to inference the new location and shape of the target. The contour information they used is basically curve fragments

Figure 2.1: Contour based tracking results [39] on Walking Person sequence

obtained by edge detection. Good performance could be achieved for tracking both rigid and non-rigid object under clustered background in real time. Yilmaz et al. [39] proposed another contour based object tracking algorithm in 2004. The tracking algorithm is formulated as the contour energy minimization problem. Two energy terms are taken into consideration: the image energy and the shape energy. The former energy is based on color and texture observations around the contour, the latter is based on the past contour observations and is able to preserve the shape during partial occlusions [39]. The algorithm achieves very robust tracking performance on sequences involving partial occlusion. Fig. 2.1 shows some tracking results of the algorithm.

Feature based tracking algorithms always extract specific features from the target, in order to build an appearance model of the target. Although only part of the target information will be used, a highly efficient and robust tracking algorithm can be built on top of these features.

Babenko et al. [1] took advantage of Haar-like features in their feature extraction step in their Multiple Instance Learning (MIL) tracker. Instead of focusing on target template information only, in their work they extract the target features as positive samples and background features as negative samples. When a new frame coming, potential target patches that are close to the previous location are sampled in a greedy way. Then classifiers are used to determine which patch should be closest to the target. Since background is also taken into consideration, the algorithm can reach very stable performance in a varieties of scenarios. Trackers like [40] share a similar structure with the MIL.

Region based tracking algorithms always define a target region in the first frame and try to estimate the location based on the whole target region. Since the target region has complete information of the target including shape and texture, this type of algorithm should always be able to track different types of objects, thus can be used in many applications.

A bunch of region based tracking algorithms that can achieve robust tracking performance have been proposed recently. These tracking algorithms are always building/learning robust appearance models along with relatively simple space state models (low DOF). Thus can always achieve very robust tracking performance. Ross et al. [32] proposed an incremental robust visual tracking (IVT) algorithm based on the incremental subspace learning method. A low-dimensional subspace representation of the target is adopted and is incrementally learning the appearance of the target during tracking. As a result the algorithm is very robust to interferences like lighting variation, partial occlusion, etc. Particle filter is used to interference the target state in each new frame. Fig. 2.2 shows an example of how IVT tracks a human face. The IVT algorithm has been recognized as an important work in the area of robust template based visual tracking. Many recently proposed algorithms

Figure 2.2: IVT tracking results on David sequence [32]. The last two rows of each figure are the eigenfaces learned by the incremental subspace learning stage.

like [4, 27, 24] share a similar structure.

Although the tracking algorithms stated in the previous paragraph are very robust towards different types of interferences, they in many cases lack the ability to achieve very accurate tracking results. One reason is that when learning the appearance model during tracking, the original target information that the appearance model contains could be affected by the new tracking results which always contain some noise. While in the case of registration based algorithms, many of them are simply using grayscale image template and always trying to retain the original template and keeping searching the target when new frames come. Thus more accurate tracking results can be achieved.

In the following sections, we will first introduce some state space models that are commonly used by the tracking algorithms, especially by the registration based algorithms. Then several popular registration based tracking algorithms are discussed, with a focus on the search methods.

2.2 State Space Model

State space model is used to model the actual location and pose of the target in 2D image space. In order to model the target state in an efficient way, we made two assumptions:

- The surface of the target is or can be approximated as a planar surface.
- The surface of the target is on a rigid object.

These two assumptions make it easier for us to model the target state by simply adopting a 2D geometric transformation. For example, the coordinates of each pixels of the target in a new image can be determined by applying Eq. 2.1. Here $\mathbf{x} = \{x, y, 1\}$ is one of the reference points belonging to the target. $\mathbf{x}^{'} = \{x^{'}, y^{'}, 1\}$ is the corresponding point location in the new image frame and s is the scale factor. $p = \{p_1, p_2, ..., p_8\}$ is the parameter set for the transformation matrix. Note in many cases $\mathbf{x}$ is defined during tracking initialization and fixed during tracking.

$$s \begin{bmatrix} x^{'} \\ y^{'} \\ 1 \end{bmatrix} = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.1}$$

The standard hierarchy of geometric transformations has 2, 3, 4, 6 and 8 degrees of freedom (DOF) [22]. A simple 2 DOF transformation may only contain the x-y location information of the target. While for an 8 DOF transformation, also referred as projective transformation, additional rotation and deformation information caused by camera perspective projection will be included.

$$H = \begin{bmatrix} p_1 & p_2 & p_3 \\ p_4 & p_5 & p_6 \\ p_7 & p_8 & 1 \end{bmatrix} \tag{2.2}$$

Since the geometric transformation can model the target state well under the two assumptions, we always use the parameters p from the transformation matrix H as the target state parameters. For example, if p has length 8, then the H matrix defined above is a homography matrix. The state space model is an 8 DOF model based on projective transformation. Many registration based trackers like [6, 16] are using the 8 DOF model. If p_7 and p_8 are 0, then H is an affine transformation matrix. The state space model is then a 6 DOF model based on affine transformation. Trackers like [32, 4] are using the 6 DOF model.

More complex state space models may not require the two assumptions on the target we made early, such as the piecewise affine warps model used in Active Appearance Models (AAM) [12]. But due to the huge computational requirements, they are not practical in real time tracking. Therefore we will not use and discuss them in this thesis. Note for many tracking algorithms, different state space models can be plugged in and used. A higher DOF model in general can characterize more complex object motion but may require more computation. So the choice of state space models really depends on the scenarios where these tracking algorithms are applied.

2.2.1 Lie Parameterization

In the case of 8 DOF state space model, we can use the homography matrix H to represent tracking results. While Benhimane et al. [6] pointed out that H_{33} of Eq. 2.2 may be 0 instead of 1, which may happen when big camera displacement occurs, in that case $det(H) = 0$. This corresponding to the case when the observed plane passes through the optical center of the camera. To avoid the problem they apply an additional constraint $H \in SL(3)$ where $SL(3)$(the Special Lie group of dimension 3) is the group of 3×3 matrices that have determinant equal to 1. The $sl3$ will be the Lie algebra that associates with the group. As long as we know $\{x_1, x_2, ..., x_8\} \in sl3$, we can directly recover H by applying Eq. 2.3.

$$H = exp\left(\sum_{i=1}^{8} x_i \underset{SL3}{gen_i}\right), x \in sl_3 \tag{2.3}$$

where exp is the matrix exponential function, $\underset{SL3}{gen_i}, i \in \{1, ..., 8\}$ are the Lie group generators for $SL3$ group.

$$
\underset{SL(3)}{gen_1} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \underset{SL(3)}{gen_2} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad \underset{SL(3)}{gen_3} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}
$$

$$
\underset{SL(3)}{gen_4} = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \underset{SL(3)}{gen_5} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \underset{SL(3)}{gen_6} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}
$$

$$
\underset{SL(3)}{gen_7} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \quad \underset{SL(3)}{gen_8} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
$$

In [6] Benhimane et al. introduced additional benefits in formulating the ESM tracking algorithm with the lie based parameterization on the state space model. This parameterization will simplify the computation of the Jacobians required by the algorithm and can achieve second-order convergence rate based only on the Jacobians. In Chapter 4 we will further discuss the benefits in detail and also introduce an alternative approach to the parameterization method.

2.3 Registration based Tracking Algorithms

The registration based tracking algorithms always assume that in each frame, the object surface will appear fully or partially (due to occlusion) and the unoccluded part can always be regard as a warped version of the target template. This constraint is also known as the image constancy constraint [16].

Let $F = \{I_0, I_1, ..., I_t\}$ be a sequence of grayscale images. Given the position of a pixel $x = (u, v)$ in frame I_t, the intensity value is denoted as $I_t(x)$. Let set $\mathbf{X} = \{x_1, x_2, ..., x_n\} \subset \mathbb{R}^2$ contain all the sampled pixel positions, then the corresponding image patch is denoted as $I_t(\mathbf{X})$. In registration based trackers, $\mathbf{X}$ at frame I_t is computed by transforming the reference points set $\mathbf{R}$ which is in the reference coordinate to the new image coordinate $\mathbf{X} = W(\mathbf{R}; p(t))$ via a warping function. $\mathbf{R}$ is initialized at the first frame and fixed during tracking. The corresponding new image patch is denoted as $I_t(W(\mathbf{R}; p(t)))$. The image constancy can be formulated as Eq. 2.4.

$$
I_0(W(\mathbf{R}; p(0))) = I_t(W(\mathbf{R}; p(t)^*)) \tag{2.4}
$$

$p(0)$ is first initialized by the target region selected on the first frame I_0. This target region is always referred to the template and is always denoted as $T = I_0(W(\mathbf{R}; p(0)))$ for convenience. The tracking problem is then to estimate the warping parameters p^* for each frame that satisfy Eq. 2.4. Due to the reason, we often refer to the $p(t)$ as the object state parameters and in many cases the warping function is simply a geometric transformation function similar to Eq. 2.2.

Given the image constancy assumption, in registration based tracking algorithms the problem can be simply reformulated as Eq. 2.5. Here d is the distance measure and an Euclidean distance is commonly used [29, 2, 16]. The problem is always solved in an iterative way. The state parameters

get simultaneously updated towards the optimum in each iteration. In the following subsections several algorithms aiming for efficiently solving this problem will be discussed.

$$p(t) = \arg\min_{p(t)} d(I_t(W(\mathbf{R}; p(t))), T) \tag{2.5}$$

2.3.1 Gradient based Approach

Since the pixel intensity value $I(\mathbf{X})$ is in general nonlinear in $\mathbf{X}$, Eq. 2.5 is a nonlinear optimization problem [3]. The common idea is to iteratively solve for increments to the parameters Δp while assuming a current estimate $\hat{p}$ is known. This assumption always holds in the registration based tracking scenario because we are always given $p(t-1)$ when trying to estimate $p(t)$. When $t = 1$, $p(0)$ is often manually set on the first frame. In the following sections, we will mainly discuss several gradient based algorithms to search for the optimal warping parameters.

Classical Lucas Kanade Algorithm

The classical Lucas Kanade (LK) algorithm is first proposed by Lucas et al. [29] in 1981 as an image alignment algorithm. The idea is to use the spatial intensity gradient information to direct the search for the position of the target. The algorithm starts with an initial guess of the target $\hat{p}$ at frame I_t, where $\hat{p}$ can simply be the target position $p(t-1)$ from previous tracking results. $\hat{p}$ will be updated iteratively by firstly solving Eq. 2.6 for Δp.

$$\Delta p = \arg\min_{\Delta p} ||I_t(W(\mathbf{R}; \hat{p} + \Delta p)) - T||_2^2 \tag{2.6}$$

The warp parameters can be updated additively by applying Eq. 2.7.

$$\hat{p} \leftarrow \hat{p} + \Delta p \tag{2.7}$$

The updating process continues until the algorithm is converged or the number of iterations has reached a pre-defined threshold. Eq. 2.6 can be efficiently solved by first performing the Taylor expansion on $I_t(W((\mathbf{R}; \hat{p} + \Delta p))$ followed with a gauss newton optimization method [3].

Although the algorithm is much more efficient compared to the simple greedy approach, it needs to calculate the Jacobian and Hessian matrix for every iteration in order to get the Δp, which reduces the tracking efficiency. The Taylor approximation makes the algorithms work well for small inter-frame object motion, but will fail to track if the inter-frame motion became relatively large. Also there are some constraints on the selection of the warping functions $W(\mathbf{R}; p)$. The warp $W(\mathbf{R}; p)$ should be differentiable with respect to the warp parameters p [3].

Inverse Compositional Algorithm

The classical Lucas Kanade algorithm needs to calculate the Jacobian and Hessian matrices for every iteration, which makes the algorithm less efficient. In contrast, the Inverse Compositional (IC)

algorithm, first proposed by Baker et al. [2] in 2001 tried to avoid the redundant computation. The idea is to change the role between the template and the new coming image as been done by the LK algorithm. The template is first warped to match the target patch warped by $W(\mathbf{R};\hat{p})$ and then the inverse of the warping on template will be used to update the $\hat{p}$.

Given the starting $\hat{p}$ for frame I_t, a warped template is estimated to match the warped image, see Eq. 2.8.

$$\Delta p = \arg\min_{\Delta p} ||T(W(\mathbf{R};\Delta p)) - I_t(W(\mathbf{R};\hat{p}))||_2^2 \tag{2.8}$$

The $\hat{p}$ is later updated in a compositional approach [2] see below:

$$\begin{aligned} W(\mathbf{R};\hat{p}) &\leftarrow W(\mathbf{R};\hat{p}) \circ W(\mathbf{R};\Delta p)^{-1} \\ &= W(W(\mathbf{R};\Delta p)^{-1};\hat{p}) \end{aligned}$$

We can simplify the notation to be Eq. 2.9:

$$\hat{p} \leftarrow \hat{p} \circ \Delta p^{-1} \tag{2.9}$$

The gauss newton method which used to solve Eq. 2.6 can also be adopted to solve Eq. 2.8 with the advantage that the Jacobian and Hessian (purely depend on the template T which is static) can be pre-computed during tracking initialization, which largely improved the computational efficiency. Thus we always treat the IC algorithm as a variant of the LK algorithm.

Although the IC algorithm has high computational efficiency, it does suffer from a similar problem to the LK algorithm, which is that it's fragile towards large inter-frame object motion. Also there is a constraint on the selection of warps: the $W(\mathbf{R};\Delta p)$ should also be invertible [3].

Efficient Second-order Minimization Algorithm

Lucas Kanade type algorithm using Newton-type optimization method would always require the calculation of Hessian. However, the Hessian calculation may be time consuming (classical LK approach) and convergence problems may occur if the Hessian is not positive definite [6]. To avoid the Hessian calculation while retaining high convergence rate, Benhimane et al. [6] in 2004 proposed a new optimization approach. The idea is to use Efficient Second-order Minimization (ESM) optimization in planar object tracking.

The ESM algorithm is proposed based on the 8 DOF state space model, which basically depends on the projective transformation to represent the target tracking results, as we have introduced in the previous section. The algorithm is going to solve Eq. 2.10 for Δp. Where $G(\Delta p)$ is a updating homography matrix with vector Δp containing a local parameterization of $SL(3)$ and will be used to update the starting homography matrix $\hat{G}$. The operation of G on Δp is same as Eq. 2.3.

$$\Delta p = \arg\min_{\Delta p} ||I_t(W(\mathbf{R};\hat{G} \circ G(\Delta p))) - T||_2^2 \tag{2.10}$$

Let $y(x) = I_t(W(\mathbf{R}; \hat{G} \circ G(x))) - T$. By applying the ESM optimization as proposed in [7], the Δp can be estimated by Eq. 2.11 and the homography can be updated by applying Eq. 2.12.

$$\Delta p(t) = -2(J_e + J_c)^{+} y(0) \tag{2.11}$$

$$\hat{G} \leftarrow \hat{G} \circ G(\Delta p) \tag{2.12}$$

Where $J_c = \nabla_x y(x)|_{x=0}$ and $J_e = \nabla_x y(x)|_{x=x_0}$ where x_0 is the optimal Δp of Eq. 2.10. The Jacobian J_e can be reformed as the Jacobian with respect to the reference template thus can be pre-computed, while J_c is computed on the new coming frame thus needs to be updated in every iteration (See Chapter 4 for details). But since it can achieve high convergence rate, the ESM algorithm is regarded as one of the state-of-the-art among the registration based tracking algorithms.

2.3.2 Matching based Approach

Although the gradient based approach is efficient in estimating the warp parameters, the main problem it faces is its weakness towards large inter-frame motion. The main reason is that the first/second order taylor expansion need to be applied to linearize the optimization problem, which will not hold when the inter-frame motion is large. In 2012, an alternative approach was proposed by Dick et al. [16] to overcome this issue while retaining high efficiency. The idea is similar to the IC algorithm. But instead of depending on gradient information to find the warp update, an image patch matching approach based on nearest neighbor search is used in order to efficiently find the update of the warp parameters.

In the algorithm, Dick et al. first build a dictionary/map between the randomly warped templates and the corresponding warping update parameters, i.e. $map\{T(W(\mathbf{R}, \Delta p)) : \Delta p\}$. Given starting point $\hat{p}$, the warp parameters are updated by first retrieving the warp update by Eq. 2.13 via a nearest neighbor search on the warped templates, given current warped patch $I_t(W(\mathbf{R}, \hat{p}))$.

$$\Delta p = map(I_t(W(\mathbf{R}, \hat{p}))) \tag{2.13}$$

Then the warp is updated similar to Eq. 2.9. The iteration will continue until the stop criteria is reached. In their implementation the FLANN library [30] released by Muja et al. is used to increase the search efficiency. The robustness of Nearest Neighbor based tracking algorithm can be improved by increasing the number of warped templates in the map.

The Nearest Neighbor based tracking algorithm surpasses the gradient based algorithms mainly in the tracking ability towards large inter-frame motion. But it may require an initial training stage which limits it's applicability to scenarios that require fast initialization.

2.3.3 Other Approaches

We have discussed several common registration based tracking algorithms. There still exists some tracking algorithms which may not be categorized into registration based tracking group but share

some similarity. Those algorithms are template/region based tracking algorithms and always look for warping parameters of the object in new frames that are the best match to the template under some criteria. A few tracking algorithms (mainly focus on search methods) will be discussed in this subsection.

Greedy Approach

The approach has been used in some popular tracking algorithms like MIL tracker [1] and Compressive tracker [40]. The assumption is that the object location l_t^* at frame t, will always be close to the object location at frame $t-1$. In other words, the new location is equally likely to appear within a radius r of l_{t-1}^*. So for each new frame, a set of image patches $S = \{s|r > ||l(s) - l_{t-1}^*||\}$ are cropped out and the probability $p(y|s)$ is computed for all $s \in S$, where y is a boolean value indicating whether the patch is the target or not according to the appearance model. The optimal l_t^* will be from Eq. 2.14.

$$l_t^* = l\left(\arg\max_{s \in S}(p(y|s))\right) \tag{2.14}$$

Note that only simple state space models can be used in this approach in order to achieve high efficiency. In [1, 40] only 2 DOF translational model is used. A higher state space model will dramatically increase the size of image patch set S that need to be cropped out and to compute the corresponding $p(y|s)$ in order to achieve good tracking results. Briefly, the greedy approach is a simple strategy to search for object motion state but trend to be less efficient on complex state space models.

Particle Filter Approach

The particle filter approach is more sophisticated compared to the greedy approach. This approach has been used in many visual tracking algorithms including [32, 4]. An estimated posterior distribution of motion state at frame t is obtained $p(m_t|s_{1:t})$, where m_t is the vector that model the motion of the target and $s_{1:t} = \{s_1, s_2, ..., s_t\}$ denotes the observation of the target from frame 1 to frame t.

The posterior distribution is estimated via two steps: prediction step Eq.2.15 and update step Eq. 2.16.

$$p(m_t|s_{1:t-1}) = \int p(m_t|m_{t-1})p(m_{t-1}|s_{1:t-1})dm_{t-1} \tag{2.15}$$

$$p(m_t|s_{1:t}) = \frac{p(s_t|m_t)p(m_t|s_{1:t-1})}{p(s_t|s_{1:t-1})} \tag{2.16}$$

The optimal motion state of the target will be $m_t^* = \arg\max_m p(m_t|s_{1:t})$. Both [32] and [4] use 6 DOF affine model as their state space model, which enables the tracking algorithms to track motions like translation, scaling and in-plane rotation. In general, the tracking results can't reach sub-pixel level accuracy unless sufficient number of particles are used, which in turn bring down the tracking efficiency.

2.4 Tracking Systems

A large number of 2D visual tracking algorithms have been proposed in literature as we introduced earlier. It would be nice to have an integrated tracking system that researchers/users can use to compare, test and even adopt different visual tracking algorithms in real applications. In the section, tracking systems that have been integrated with more than one tracking algorithms are briefly reviewed.

In 1993 Hager, et al. [37] wrote the first version of Xvision framework which only contains an edge tracker. After that more trackers like contour tracker [37], region trackers [29] were integrated to the framework while revamping the framework. These trackers are highly efficient since the whole framework is built in C++. In addition, since Xvision consists of a small set of image-level tracking primitives, various efficient trackers can be created by combining these tracking primitives. In [21] Hager et al. pointed out several different potential applications where Xvision can be used, such as robotic hand-eye application, modeled objects tracking.

Although the Xvision framework provides a high efficiency visual tracking repository, these tracking algorithms are relatively old. Some new tracking libraries that have been created recently. In [38] Wu et al. provided a tracking library with about 29 recently proposed trackers. The input and output format of all the trackers is unified which makes it more convenient for users to do large scale performance evaluation. Although a large set of tracking algorithms have been collected, the implementations of different algorithms are independent from each other and may be implemented in different programming languages. Also most of the trackers are robustness-oriented algorithms which are only able to track relatively low DOF target motions. Therefore this library is mainly designed for the evaluation and comparison of robustness-oriented algorithms.

In 2012, Dick et al. released a new tracking system along with the nearest neighbor based tracking algorithm [16]. This system integrates the relatively new registration based algorithms such as the ESM tracker [6] and the nearest neighbor based tracker. An efficient Inverse Compositional tracker [2] is also implemented in the system. The full system is mainly implemented in Python and the core algorithm is optimized in Cython in order to achieve real-time speed. The usage of Python makes the system easy for prototyping, testing and evaluating new tracking algorithms. Our integrated tracking system is based on it. More details will be introduced in Chapter 4.

2.5 Summary

In this chapter, we first reviewed some general tracking algorithms with different appearance models. By constantly updating the appearance model of the target, more robust tracking performance can be achieved at the cost of lower accuracy. Then we introduced several state space models from the literature and also briefly mentioned the Lie based parameterization for homography matrix. With a higher DOF state space model, more complex target motion could be captured. In addition,

different search methods that have been used by the registration based tracking algorithms were also discussed. The iterative search of the target state leads to a relatively high tracking accuracy. Finally, several tracking systems integrated with different tracking algorithms were discussed.

Chapter 3

The RKLT Algorithm

Most gradient based tracking algorithms that we introduced in Chapter 2 can not track large inter-frame object motion. Although the nearest neighbor based algorithm can handle it to some degree, it requires a relatively slow training step in order to achieve good tracking performance. In this chapter we are going to introduce a novel tracking algorithm, the RKLT algorithm that is able to track complex motion with larger inter-frame motion without the pre-training step. The algorithm is partially inspired by the observation that only a subset of regions on the template is sufficient to be used to extract accurate object motion, with the additional benefit of better robustness towards noise. In addition, the pyramid implementation of the KLT algorithm [8] contributes to the success of the RKLT algorithm towards large inter-frame motion.

3.1 Related Work

The benefits of pixel selection in the application of template based tracking algorithms have been pointed out in several publications. In 1999, Dellaert et al. [14] pointed out that by selecting a best pixel subset from template pixels, the Lucas-Kanade type algorithms are able to achieve almost same accuracy as using the complete template with orders of magnitude faster speed. The method they used is to randomly select M pixels from the top 20 percent of the "best" pixels of the template. All the pixels of the template are sorted according to the object information content. By selecting the subset of pixels off-line before tracking starts, the computation can be reduced dramatically while retaining same level of accuracy.

Benhimane et al. [5] also indicated that by selecting a subset of the template, it can dramatically improve the performance of template-based tracking in both robustness and accuracy. The subset of pixels are also generated off-line before tracking starts. But unlike [14], they selected the so-called *linear subset* for the IC algorithm [2] and the *quadratic subset* for the ESM algorithm [6], which are two commonly seen gradient based algorithms. For example, when only the pixels in *linear subset* were used, the IC algorithm will converge towards the minimum ideally in only one iteration.

Although the two methods mentioned above can improve the tracking performance in some as-

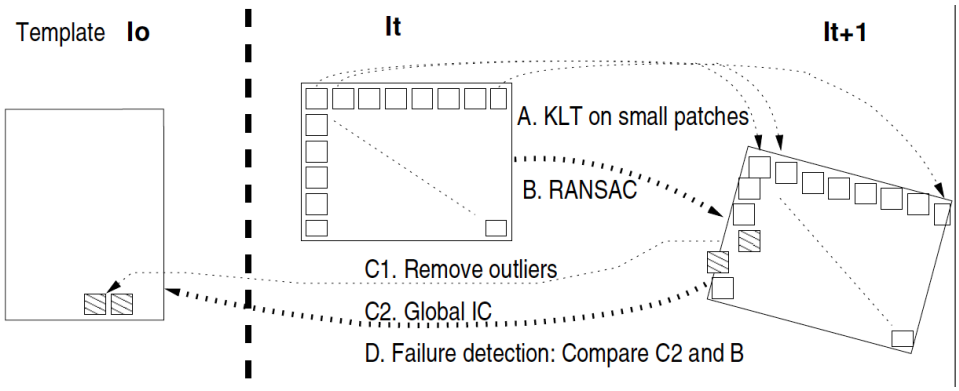


Figure 3.1: RKLT tracker overview. Tracking image template is selected in I_0. (A.) Several KLT trackers compute 2D image motion. (B.) RANSAC estimates a frame-to-frame homography. (C1.) RANSAC outlier regions in the template are removed (shaded patches). (C2.) Inverse Compositional (IC) global homography registration in the original template coordinate frame. (D.) If the IC incremental update computation diverges (significantly different from RANSAC H), then global warp H gets updated by RANSAC, otherwise the more precise IC H is returned.

pects, they cannot keep updating the "best" pixels or subset during tracking since the subset is always built off-line. The selected subset of pixels may be easily affected by camera noise or environmental interference during tracking, thus causing tracking failure. In contrast, our algorithm tries to keep selecting/updating the reliable target regions for online tracking.

The RKLT algorithm is also closely related to the median flow tracker [25], which was later adapted for the Tracking-Learning-Detection framework (TLD) [26]. Instead of selecting the subset pixels off-line, a set of evenly sampled KLT trackers [8] are initialized within the target region in the previous frame and then track the current frame. KLT trackers are used as 2D point trackers. Lost trackers are rejected using the Forward-Backward (FB) error criteria. The remaining points which can be regarded as the reliable points are used to estimate 3 DOF target object state, i.e., translation and scaling. Since it can only handle 3 DOF motion, the method can be most likely applied in surveillance application. Though our algorithm is indeed inspired by this method, we have improved upon it by introducing the RANSAC algorithm, which enable the algorithm to handle 8 DOF object motion and meanwhile achieving a more robust rejection criterion.

The idea of using RANSAC to estimate the homography between two frames from set of matching points has also been used by [11] but in the context of panoramic image stitching. Although feature detection and matching works well in the scenarios like image stitching, it is not able to meet the stringent time constraints for real-time object tracking. Also, [11] uses feature detection to find good features to match, which is not always suitable in our tracking framework. Concretely the objects being tracked are sometimes small and/or have low textured appearance, which makes it difficult to extract enough feature points to get a good estimate. It's also not likely to work well with a cluttered background.

Figure 3.2: Intermediate results of applying RANSAC algorithm to the point tracking results on the *bookIII* sequence from the TMT dataset (introduced in Chapter 5). The point trackers are initialized in the previous frame. The red points are outliers detected by RANSAC which correspond to the regions that are partially occluded or affected by other noise, while the green points are inliers which will be passed to the refinement step.

3.2 RKLT Algorithm

The RKLT algorithm consists of 4 main steps: Sampled Points Tracking, Homography Estimation, Tracking Refinement and Failure Detection. The first two steps are focused on selecting the reliable regions of the template that are suitable for registration based tracking algorithms to track and meanwhile generating the initial motion estimation for each new frame. The tracking refinement step will take advantage of these reliable regions as well as using the initial motion estimation as a starting point to generate the refined motion. Any tracking failure caused by refinement step will be detected by the Failure Detection. Fig. 3.1 shows an overview of the algorithm. In the following subsections, we are going to discuss the four steps in details.

3.2.1 Sampled Points Tracking

As mentioned earlier, the detection and matching based frame work in image stitching [11] can not be directly applied to tracking. Instead, we have used a similar idea as what Kalal did in his median flow tracker [25]. A sufficient number of evenly sampled point trackers were initialized in each frame and tracked in the next frame in 2D image space. Instead of applying any further filtering step as in [25], all the point pairs between the adjacent frames are fed into the RANSAC algorithm. Though any reasonable fast tracker can be used for this step, we have used the pyramid KLT point trackers [8] in our work for convenience since an efficient and bug free implementation is available in OpenCV [10]. In Fig. 3.2, each point is in the centroid of a patch where a 2D KLT tracker is initialized. A window size of 10×10 pixels are used for each KLT tracker in our implementation.

3.2.2 Homography Estimation

Since we are using simple coarse trackers in the step above, it is likely that some of the trackers we initialized in this step will not track well due to factors like lighting variations, partial occlusions. Trackers like [2] and [6] that track the object as a single template may also face this problem. In fact, since they formulate the search as an L2 minimization problem (introduced in Chapter 2), they may be even more sensitive to these factors. In our case, we consider the initial and final positions of these point trackers as pairs of matching points that are used as input to the robust RANSAC algorithm [17] to estimate the 8 DOF homography that best describes their relative positions. Since RANSAC detects outliers as part of the estimation process, the lost trackers are implicitly discarded in this step. The inliers are corresponding to the reliable pixels of the global template for tracking and will be used in the refinement step. Fig. 3.2 shows in red the tracking outliers found by the RANSAC algorithm. Note that we slightly tuned the threshold of RANSAC for outlier detection in order to handle occlusion.

Let I_t denote the frame at time t and H_t denote the homography that warps the initial template in I_0 into the object patch in I_t. Then, if H_t^{t+1} is the true homography between I_t and I_{t+1}, we can get H_{t+1} by simply applying Eq. 3.1.

$$H_{t+1} = H_t^{t+1} H_t \tag{3.1}$$

Therefore, if $\hat{H}_t^{t+1}$ is the estimate of the homography between I_t and I_{t+1} produced by the RANSAC algorithm, we can get a coarse estimate of H_{t+1} by Eq. 3.2.

$$\hat{H}_{t+1} = \hat{H}_t^{t+1} H_t \tag{3.2}$$

In general, $\hat{H}_t^{t+1}$ will be close but not equal to H_t^{t+1} unless all the points are tracked perfectly in Step A. Hence, without the following refinement step, the accumulated error when estimating H_{t+1} will keep increasing and will eventually cause tracking drift. This is why an global registration step is needed to refine $\hat{H}_{t+1}$ and get it closer to H_{t+1}.

3.2.3 Tracking Refinement

The goal of this step is to find a better estimate for H_{t+1} given the coarse estimate $\hat{H}_{t+1}$ from Step B (Fig. 3.1). This is achieved by using $\hat{H}_{t+1}$ as the initial guess or starting point for the search process of an efficient gradient based tracking algorithm, which is initialized with the original template in I_0. We have used the IC tracker with gauss newton method [2] for this purpose since this is one of the efficient and most commonly used variants of the Lucas Kanade tracker. Since we are assuming that our initial guess $\hat{H}_t^{t+1}$ is a good one, far fewer iterations should be required for convergence than if we were to use H_t as the starting point for the search.

A further optimization in this step is achieved by considering only a part of the full template, determined by the inliers from Step B, while solving the optimization equations of the IC tracker. In

frame# 65 frame# 110 frame# 206 frame# 250 frame# 480

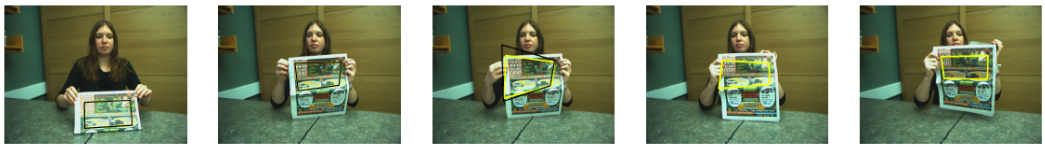
Figure 3.3: Tracking results on the BookI sequence from the TMT dataset. Yellow: RKLT with refinement; Purple: RKLT without refinement. Once tracking error occurs at frame# 110, the purple bounding box couldn't track the target as accurately as the yelow bounding box.

other words, the IC tracker will only make use of the pixels which have been recognized as inliers by Step B (green points in Fig. 3.2). These pixels are always regarded as the most reliable pixels for the refinement step. This is based on the assumption that the points where the trackers in Step A failed are probably the ones that are most difficult to track. Thus is least likely to contribute to the convergence of the IC tracker since both trackers use different variants of gradient based searching method. This is why we have used gradient based trackers for both steps A and C, even though, in principle, any two trackers can be used.

An important point to note here is that the intensity image template for the IC tracker, unlike the KLT point trackers in Step A, is *not* reinitialized at each frame, but the original template is used throughout the tracking. Therefore, this step registers the current image with the first template, and removes the drift common in trackers that update/on-line learn the target appearance during tracking. Fig. 3.3 shows the tracking results of two RKLT trackers with/without refinement step on the BookI sequence from TMT dataset. The one without refinement couldn't track target accurately starting at frame#110 and couldn't recover, which indicates the importance of the refinement step in avoiding the accumulating error.

3.2.4 Failure Detection

Since the tracker chosen in Step C has to be fast, the choice of this tracker is determined more by considerations of time efficiency and tracking accuracy than robustness. Therefore, even if the initial guess from Step B is good, this tracker may still be affected by noise and fail. For this reason, we use a simple way to detect failure in Step C by measuring the mean corner distance err (defined in Eq. 5.1) of the two bounding boxes resulting from steps B and C respectively. We empirically selected a threshold th such that, when $err >= th$ we assume that Step C failed and the final tracking results will be the one given by Step B. Although this simple strategy seems ad-hoc we do found it's benefits especially when the target involves very fast motion or heavy interference that making the refinement step fail. It's worth mentioning that we didn't use the simple image residual between the template and the tracked patch in I_{t+1} to detect the tracking failure since image intensity varies, e.g., due to light and view angel. A simple example is shown in Fig. 3.4.



frame# 188 frame# 320 frame# 352 frame# 405 frame# 420

Figure 3.4: Tracking results on the Newspaper sequence from the TMT dataset. Yellow: RKLT with failure detection; Black: RKLT without failure detection. With failure detected at frame# 320, the wrong result from the refinement step is ignored and target is later reacquired accurately at frame# 420 and tracks till the end of the frame.

Algorithm 1 RKLT tracking algorithm

$H_1 \leftarrow$ *Identity Matrix*
$X_1 \leftarrow$ *Evenly sampled points locations on template*
for *each new frame* I_i **do**
  for each x_j in X_i **do**
    Initialize a point tracker K_j with x_j on I_i
    $\hat{X}_{i+1}[j] \leftarrow K_j$ *result on* I_{i+1}
  end for
  $\hat{H}_i^{i+1} \leftarrow$ *RANSAC* $(X_i, \hat{X}_{i+1})$
  $\hat{H}_{i+1} \leftarrow \hat{H}_i^{i+1} H_i$
  $H_{i+1} \leftarrow$ *Tracking refinement given* $\hat{H}_{i+1}$
  $H_{i+1} \leftarrow$ *Failure detection based on* $\hat{H}_{i+1}$ *and* H_{i+1}
  $X_{i+1} \leftarrow$ *New target points sampled on* I_{t+1} *based on* H_{i+1}
end for

3.3 Discussion

3.3.1 Density of Sampling

There is a trade off between tracking performance and tracking efficiency for RKLT algorithm, just like general tracking algorithms. With a higher density of sampling on the target, the tracker can estimate the motion based on a higher number of KLT trackers and is expected to achieve better performance, with a cost of more computation. Thus it is important to sample the template with a resolution that is able to achieve real-time speed while retaining good tracking performance. In Fig. 3.5 we did the new static experiment as described in Chapter 4, with template resolutions varying from 20×20 to 50×50. Note that the original target is the same for all resolutions, only the number of points on the target which are sampled for initializing KLT trackers varies. It is expected that with a higher resolution, the convergence should be higher. In Fig. 3.5, after the resolution is above 30×30, the improvement of the convergence becomes relatively small. In our experiments, we will use RKLT tracker with template resolution of 40×40 in order to achieve relatively high convergence with fast tracking speed (See Fig. 6.8 for speed test results).

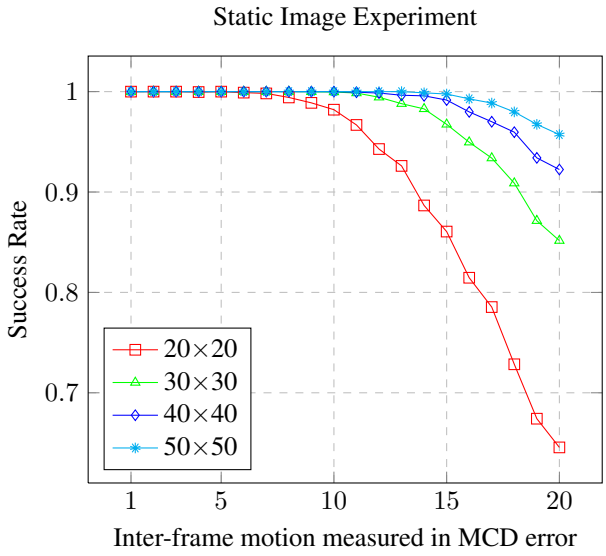


Figure 3.5: Convergence Comparison among four different template resolutions.

3.3.2 Robust Tracking Refinement

Although the refinement step largely increase the accuracy of the tracking results by performing global registration on the template using IC tracker, it decreases the robustness of the tracker when the "bad" points can not be filtered by the RANSAC algorithm. For example, when the size of occluded area is larger than that of unoccluded one, the RANSAC can't figure out which area is actually belong to the target.

Alternatively, the refinement step can be improved in robustness towards occlusion by adopting a robust error function for the Lucas kanade type tracker. Instead of minimizing the Euclidean L2 norm, the goal is to iteratively minimize a robust error function Eq. 3.3.

$$\sum_{\mathbf{R}} \varrho([I_0(W(\mathbf{R};\Delta p(t))) - I_t(W(\mathbf{R};\hat{p}(t)))]^2) \tag{3.3}$$

A variety of robust functions have been used in literature. $\varrho(x)$ can be defined, for example in a similar form as the Geman-McLure function[34] see Eq. 3.4 or the Huber function [20], see Eq. 3.5.

$$\varrho(x) = \frac{x}{\delta_1^2 + x} \tag{3.4}$$

$$\varrho(x) = \begin{cases} \frac{1}{2}x & if\ 0 \leq x \leq \delta_1^2 \\ \delta_1\sqrt{x} - \frac{1}{2}\delta_1^2 & if\ x > \delta_1^2 \end{cases} \tag{3.5}$$

The full derivation of the Lucas kanade type algorithm with a robust function can be found in the TR version of [3]. Fig. 3.6 shows the tracking results of five registration based trackers. The first three are IC tracker, ESM tracker and NNIC tracker [16], which we will use in our experiments. The new RKLT tracker and also a variant of RKLT (RRKLT) which uses a robust error function in Eq.

Figure 3.6: Tracking results on the Highlighting sequence from the TMT dataset. IC: red; ESM: green; NNIC: blue; Original RKLT: yellow; RRKLT: black

3.4 in the refinement step are tested. For the RRKLT tracker, the parameter δ_1 is manually picked and kept the same during testing.

In Fig. 3.6 we can find that all the trackers track well when only small portion of the target is occluded, see the tracking results of the first two columns. However, when the occluded area became larger, only the RRKLT that uses a robust error function, will succeed till the end.

3.4 Summary

In this chapter, we have introduced the new proposed RKLT algorithm. The RKLT algorithm is based on the observation that different regions may contribute differently towards indicating the motion of the target. We apply RANSAC algorithm to filter the regions of the template so that only those inliers will be passed to the refinement step to estimate the final object motion. We also give a brief discussion on the density of sampling and it's effect on tracking performance. The RKLT algorithm can also be simply extended/improved by adopting other robust refining trackers. More detailed experimental results and analysis will be presented in Chapter 6. The RKLT algorithm is shown to have comparable performance and in many cases surpass the state-of-the-arts on two public datasets.

Chapter 4

State Space Parameterization and Tracking System

In the chapter we will introduce a new parameterization method on the ESM algorithm and a new integrated tracking system. First we will briefly discuss the necessity for the ESM algorithm to obtain the $SL3$ parameterization on the 8 DOF state space model in order to achieve quadratic convergence. Then we will provide an alternative to the $SL3$ parameterization, which is based on the corners of bounding box of the target region. Finally, we will introduce the integrated tracking system that we built on top of the work by Dick, et al. [16].

4.1 State Space Parameterization

As presented in Chapter 2, the Inverse Compositional (IC) algorithm [2] with the gauss newton approximation couldn't achieve quadratic convergence nearing the target when the residual between the current target patch and the template is large. The reason is that the approximation of the Hessian matrix used by the gauss newton method only holds when the residual is sufficiently small [3]. The approximation of the Hessian matrix could improve the tracking efficiency since the Hessian can be pre-computed. But the tracker may fail when large inter-frame motion is involved. In contrast, the ESM algorithm [7] is able to reach a high convergence without the use of the Hessian. However, unlike IC, the ESM algorithm may not directly use the 8 DOF Homography matrix elements as the state parameters. Instead, the corresponding parameters in $sl3$ algebra that associated to the homography matrix will be used. In the following section, we will first introduce the benefits of adopting the $SL3$ parameterization to the ESM algorithm. Then another corner based parameterization will be discussed.

4.1.1 Related Work

Benhimane et al. [7] defined the tracking problem as to minimize Eq. 4.1. The notations are similar to the ones in Chapter 2. Where T is the target template and $\mathbf{R}$ is the reference points set. $\hat{G}$ is

a homography matrix served as the starting point and $G(x)$ is an function to find the homography matrix given x. Where x is the updating parameter vector (in $sl3$ for the original ESM algorithm).

$$f(x) = ||I_t(W(\mathbf{R}; \hat{G} \circ G(x))) - T||_2^2 \tag{4.1}$$

Let $y(x) = I_t(W(\mathbf{R}; \hat{G} \circ G(x))) - T$. In order to minimize Eq. 4.1, we first perform second order Taylor serious approximation on $y(x)$ about $x = 0$.

$$y(x) = y(0) + J(0)x + \frac{1}{2}M(0, x)x + O(||x||^3) \tag{4.2}$$

Where the Jacobian is $J(x) = \bigtriangledown_x y(x)$. $M(x_1, x_2) = \bigtriangledown_{x_1}(J(x_1)x_2)$. By performing first order Taylor expansion on $J(x)$ we have:

$$J(x) = J(0) + M(0, x) + O(||x||^2) \tag{4.3}$$

Substitute $M(0, x)$ in Eq. 4.2 with the one in Eq. 4.3, the former can be written as follow:

$$y(x) = y(0) + \frac{1}{2}(J(0) + J(x))x + O(||x||^3) \tag{4.4}$$

Now we get the second-order approximation of y(x) about $x = 0$ [7]. Thus the cost function Eq. 4.1 can be written as follows by ignoring the high order term:

$$f(x) = ||y(0) + \frac{1}{2}(J(0) + J(x))x||_2^2 \tag{4.5}$$

We assume $x = x_0$ is the minimum of $f(x)$, which can be found below:

$$x_0 = -2(J(0) + J(x_0))^+(I_T(\mathbf{R}; \hat{G}) - T) \tag{4.6}$$

As long as we have the two jacobians $J(0)$ and $J(x_0)$, we should be able to compute the incremental update x_0, thus update the $\hat{G}$. Note $J(0)$ is easy to get, it is the jacobian on I_t given $\hat{G}$ from the previous iteration. While $J(x_0)$ is hard to find because x_0 is unknown. We can write the Jacobian as follows [7]:

$$\bigtriangledown_x y(x)|_{x=x_0} = \bigtriangledown_x I_0(W(\mathbf{R}; G(x_0)^{-1}G(x)))|_{x=x_0} \tag{4.7}$$

Note $J(x_0)$ is based on the first frame I_0. By applying chain rule Eq. 4.7 can be written as the product of three Jacobians:

$$\bigtriangledown_x y(x)|_{x=x_0} = J_{I_0} J_{w_0} J_{G_0} \tag{4.8}$$

Where J_{I_0} is actually the spatial derivative of I_0, J_{w_0} is the jacobian of the warp function, in our case a projective transformation function. The first two jacobians does not depend on x_0, thus are always constant during tracking [7]. While J_{G_0} does depend on the unknown x_0 see below:

$$J_{G_0} = \bigtriangledown_x G(x_0)^{-1}G(x)|_{x=x_0} \tag{4.9}$$

If we use pure homography parameterization, Eq. 4.9 can be very complicated and it depends on the unknown x_0. Instead [7] using $SL3$ parameterization bypasses the problem by applying the exponential map properties as shown below.

$$J_{G_0} x_0 = J_G x_0 \tag{4.10}$$

Where J_G is the jacobian of exponential map function (see Eq. 2.3) with respect to $SL3$ parameters. Note this J_G is constant which doesn't depends on x_0. As a result, by using $SL3$ we can successfully get $J(0)$ and $J(x_0)$. Once we get the two Jacobians, x_0 can be found by using Eq. 4.6.

4.1.2 Corner based Parameterization on ESM Algorithm

Although the original ESM algorithm could achieve quadratic convergence rate without the necessity of computing the Hessian matrix, it does have a strict constraint on the parameterization of the state space model. For 8 DOF ESM, $SL3$ based parameterization need to be adopted to ensure theoretically correctness.

We observed that the object state in 2D image space can also be parameterized by the corners of the bounding box of the target. For example, for trackers using 8 DOF state space model, the state parameters are basically the eight efficients of a homography matrix as we defined in Chapter 2, which can equivalently be replaced by the four corners of the target. Note that the conversion from the corners to the homography efficients can be done using the Direct Linear Transformation (DLT) algorithm [22].

This observation provides an alternative for updating the tracking state. Instead of directly estimating the update Δp of the homography matrix (IC tracker) or the update of corresponding parameters in $sl3$algebra (ESM tracker), we can update the tracking state by continuously computing the displacement of the corners of the target. The idea of corner based representation on 8 DOF state space model has also been discussed in [19], which is used to model non-planar object motions and track using the difference decomposition approach. This is a more straightforward way of parameterizing a homography matrix than the Lie based parameterization. In addition, different DOFs of ESM algorithms can be derived by using different number of corners. In this subsection, we introduce the corner based parameterization on the 8 DOF ESM algorithm.

The original ESM uses the $SL3$ parameterization on the state space model, here we reformulate it by replacing the $SL3$ parameterization with the corner based one. Thus $y(x)$ can be written as follows

$$y(x) = I_t(W(\mathbf{R}; \hat{H} \circ DLT(c^*, c^* + x))) - T \tag{4.11}$$

Where c^* is a vector of four corners of the target in the reference coordinate (always fixed during tracking) for the 8 DOF ESM tracker. H is the homography matrix since 8 DOF state space model is used. x is a vector containing the displacement of the four corners of the target. $DLT(c^*, c^* + x)$

is used to find the transformation matrix H' which transforms the four corners from the original c^* to the new position $c^* + x$.

The cost function which we want to minimize is $f(x) = ||y(x)||_2^2$. Following the same idea as the original ESM algorithm, x_0 can be found below which is very similar to Eq. 4.6.

$$x_0 = -2(J(0) + J(x_0))^+(I_T(\mathbf{R}; \hat{H}) - T) \tag{4.12}$$

The problem is how to get the new Jacobians $J(0)$ and $J(x_0)$. Although it is nontrivial to find the explicit math expression of the two Jacobians, we can still estimate them in a numerical way. Since we always use simple geometry transformation as the warping function (Section 2.2), the expression of $y(x)$ can be rewritten as

$$y(x) = I_t(W(W(\mathbf{R}; DLT(c^*, c^* + x)); \hat{H})) - T$$

Then the Jacobian $J(0)$ can be written as the product of three Jacobians:

$$\bigtriangledown_x y(x)|_{x=0} = J_I J_W J_T \tag{4.13}$$

where the three Jacobians are listed below:

$$\begin{aligned} J_I &= \bigtriangledown_{\mathbf{X}} I(W(\mathbf{X}; \hat{H}))|_{\mathbf{X}=\mathbf{R}} \\ J_W &= \bigtriangledown_H W(\mathbf{R}; H)|_{H=E} \\ J_T &= \bigtriangledown_c DLT(c^*, c)|_{c=c^*} \end{aligned}$$

Where E is the identity matrix. J_I and J_W are easy to get because they all depend on the known parameters. J_T can be estimated numerically and only once in the whole tracking process.

$$J_{Ti} = \frac{DLT(c^*, c + \epsilon_i) - DLT(c^*, c - \epsilon_i)}{2||\epsilon_i||} \tag{4.14}$$

Where J_{Ti} is the i^{th} column of J_T. ϵ_i is a vector of length 8, with all of the entities of the vector are 0 except for the i^{th} element, of which a sufficiently small positive value is assigned. In order to get $J(x_0)$ we first connect I_t with I_0:

$$\begin{aligned} I_t(W(\mathbf{R}; \hat{H})) &\approx I_0(W(\mathbf{R}; H_0 \circ DLT(c^*, c^* - x_0))) \\ I_t(W(\mathbf{R}; \hat{H} \circ DLT(c^*, c^* + x))) &\approx I_0(W(\mathbf{R}; H_0 \circ DLT(c^*, c^* - x_0 + x))) \end{aligned}$$

The idea is similar to the Inverse compositional algorithm as we introduced in Chapter 2. In order to warp the current patch to match the template, we need to update the corners to be $c^* + x_0$ in the reference coordinate. But alternatively, we could keep the current patch and transform the corners of the template to be $c^* - x_0$ at frame I_0. Thus the $J(x_0)$ can be solved by Eq. 4.15. Briefly, $J(x_0)$ can be estimated by finding the jacobian with respect to the four corners on the reference image I_0.

$$\begin{aligned} J(x_0) &= \bigtriangledown_x I_t(W(\mathbf{R}; \hat{H} \circ DLT(c^*, c^* + x)))|_{x=x_0} \\ &\approx \bigtriangledown_x I_0(W(\mathbf{R}; H_0 \circ DLT(c^*, c^* + x)))|_{x=0} \end{aligned} \tag{4.15}$$

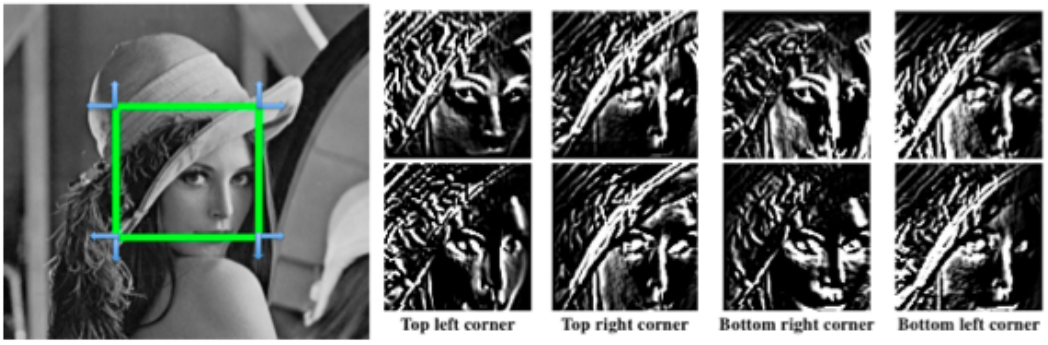


Figure 4.1: Left image: select target on Lena's image; Right images: Jacobian w.r.t patch corners on Lena's Image. The first row are jacobians w.r.t x axis, the second tow are jacobians w.r.t y axis

Now the $J(x_0)$ can be computed similar to $J(0)$ using Eq. 4.13. Once we got the $J(0)$ and $J(x_0)$, the x_0 can be found by simply applying Eq. 4.12. The new tracking result is updated to be $H = \hat{H} \circ DLT(c^*, c^* + x_0)$. Fig. 4.1 shows how the Jacobian with respect to corners looks like on the Lena's image.

The idea of corner based parameterization can be easily applied to other gradient based algorithms. For example, in order to have corner based parameterization on the IC algorithm, the only two modifications need to be done are the jacobians calculation and the tracking state updating process during tracking.

4.2 Integrated Tracking System

We have reviewed several visual tracking systems that are publicly available in Chapter 2. We showed that some systems are integrated with registration based trackers that are relatively old. While for libraries like the one proposed in [38], the library is just a collection of different trackers, with each one independently implemented. In this section, we will introduce our tracking system that is based on the system proposed by Dick et al. [16].

The original tracking system contains three different registration based trackers, i.e. IC tracker, ESM tracker and the nearest neighbor based tracker proposed in [16]. All the three trackers used an 8 DOF projective model. We add two extension to the original tracking system by first adding lower DOF state space models, including 2, 4, 6 DOF models. Since a lower DOF models will improve the tracking efficiency, especially for the gradient based algorithms. Because the number of columns of the Jacobian that needs to be computed depends on the DOF of the state space model that is used by the gradient based algorithms. With a lower DOF, the size of the Jacobian is reduced. Also fewer iterations are needed for the tracker to converge. As a result, less computation is required. It has also been observed by [9] that higher DOF variants of Lucas Kanade type trackers are more likely to drift than simple 2 DOF versions. The low DOF trackers can be used in the scenarios like high efficient point tracking where small image patches are used, which only contain information of

simple motion. In addition, two more trackers, the template based tracker based on Particle Filter (PF) and the new proposed RKLT tracker are also integrated into the tracking system. Three main benefits from the integrated tracking systems are listed below:

- The system can be used as a testbed for the comparison of different registration based tracking algorithms. Trackers including the state-of-the-art ESM algorithm, NNIC algorithm [16] are all integrated into the system. All the scripts for the evaluation methods introduced in Chapter 5 are made available in the system.
- The trackers in the system can be directly deployed into applications that require a single tracker with real time performance. Part of the code has been optimized in cython.
- New tracking algorithms can be designed and prototyped based on the system. Part of the code can be reused by the new trackers. For example, if a new search method is proposed, part of the code for appearance model and state space model can still be reused. Different trackers can also be organized in a specific way. For example, the NNIC tracker proposed in [16] is built in a cascade approach, which follows a coarse-to-fine tracking strategy.

In order to make the system more convenient for users, we introduce a simple user interface[1] for the system, see Fig. 4.2. The Basic Parameters menu will first pop up when running the user interface. There are three main settings need to be configured: the image/video source, the image pre-processing and the tracker's setting. The image/video source can be selected from the source drop-down list. The source can be from a physical camera or image files stored on the hard disk. If the user selected the 'jpeg' in the source drop-down list, then probably the source images could be linked to the TMT dataset [33] stored on the machine. The user can select a specific sequence in the TMT dataset in the Dataset sub-menu. The image pre-processing settings include the RGB to Grayscale image conversion, image smoothing and the image pre-processing using Sum of Conditional Variance (SCV) [31].

Different trackers can be selected from the tracker source drop-down list. When a specific tracker is selected from the list, a new Tracking Param menu will pop up for detailed tracker parameter settings. Some of the common settings for different tracking algorithms are the maximum number of iterations, threshold for stopping the iteration if the algorithm is in an iterative approach and the density of sampling for the target template. After all the settings are done, the user can click the Start button to initialize the tracker and a new window called Tracked Images will be created and tracking results will be shown in the window.

4.3 Summary

In this chapter, we introduced the new corner based parameterization on the ESM tracking algorithm. Then based on the work in [16], we presented an integrated tracking system consisting of 5 different

[1] The user interface was implemented by Abhineet Singh

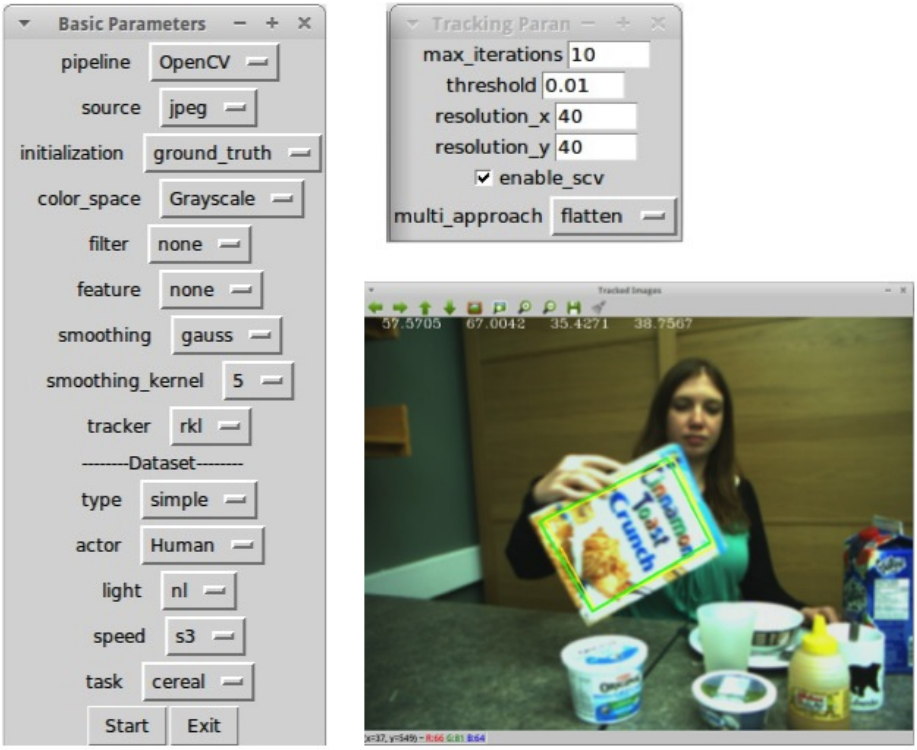


Figure 4.2: User Interface for tracker's setting: Left: Basic Parameters Menu, top right: Tracking Parameters Menu, bottom right: Tracked Image Window

tracking algorithms. New state space models and new tracking algorithms are added to the system, which can not only be used as a tracking testbed, but can also be directly applied in real applications such as robotics manipulation applications. A simple user interface is also discussed in order to simplify the usage of the tracking system.

Chapter 5

Datasets and Evaluation Methods

In this chapter, we will first discuss several existing image datasets made for tracking evaluation. Then we will introduce a new dataset called Tracking on arm and hand Manipulation Task (TMT) dataset. In addition, two evaluation metrics will be introduced in order to measure the accuracy and robustness of different registration based trackers respectively. Finally we will show two new tracking experimental settings, the first one is a modified static image experiment that is based on synthetic image dataset. The second one is the speed sensitivity experiment that is based on real image dataset.

5.1 Datasets

Recently, different types of datasets have been released and have played an important role in the development of research in computer vision community. For example, the popular ImageNet [15] dataset provides a uniform image benchmarking data for algorithm evaluation in the area of image based object classification and recognition. In visual tracking, although there are some tracking videos/datasets available, most of them are designed for low DOF tracking evaluation.

In 2013 Wu et al. [38] collected and categorized a large tracking dataset with 50 fully annotated sequences to facilitate tracking evaluation. About 11 different challenge attributes are concluded based on which sequences are annotated. The ground truth is represented by the position and size of a rectangle which contains the target. As a result, the dataset is more suitable for surveillance type tracking evaluation.

Another famous dataset resource is from the VOT (Visual Object Tracking) challenge [1]. This challenge is held annually from 2013 and aiming for building up a repository of benchmarks for the comparison of short-term trackers. In the newest 2015 VOT challenge, the dataset has been enriched to 60 sequences and labelled with 3 DOF (translation and rotation) bounding boxes. The advantage of the benchmark compared to [38] is that the attributes are per-frame labeled and also a cross-platform evaluation kit has been provided for convenience. Although they have provided a

[1] http://www.votchallenge.net/

very concrete work in evaluating and presenting the results of tracking algorithms, it has the same problem as [38]. The dataset is designed for robustness-oriented tracking evaluation.

There are only a few datasets that are made specially for high DOF tracking evaluation. Gauglitz et al. [18] provided a dataset consisting of 96 video streams which contains object motion in 3D space (3D translation and rotation). Every object is first printed as a poster, then videos are captured from moving camera with the static poster. The ground truths are provided as 3×3 homography matrix that warps each video frame to the canonical reference frame based on the markers. Although this project has provided a large dataset with accurate ground truths, there are still some concerns as the 2D poster may not be able to fully represent the 3D object . The surface property of the poster and the real object may be different. Another concern is that since this dataset is mainly designed for evaluation of feature based visual tracking algorithms, the textureless objects are ignored in the dataset.

Another popular dataset for template based tracking evaluation is the Metaio dataset released by Lieberknecht et al. [28] in 2009. This dataset is mainly designed for template based trackers and has been adopted to evaluate many template based tracking algorithms such as [16, 13]. Since it has been recognized as one of the most commonly used benchmark in registration based tracking evaluation, we will adopt the metaio dataset as one of our datasets in our evaluation experiments. More details about the Metaio dataset will be discussed in the following section.

5.1.1 Metaio Dataset

The metaio dataset [28] is carefully designed for template based tracking evaluation. In the dataset 8 different targets are included as shown in Fig. 5.1, which ranges from lower texturedness to high texturedness. For each target, five different types of dynamic behaviors of object are considered, the first four types are: ”Angle”, ”Range”, ”Fast Far”, ”Fast Close”, which are mainly categorized by the motion of the camera when recording sequences. The last one is the ”Illumination” which corresponds to the challenge caused by lighting variation. In total there are 40 different sequences in the Metaio benchmark. The high accuracy ground truths are generated for each sequence with an industrial camera, which is rigidly mounted on a high-precision FaroArm robotic arm with the arm and camera fully calibrated. When recording the videos, only the camera will be moving and camera pose information will be stored, based on which high accuracy ground truths of target locations can be generated.

Figure 5.1: The reference targets in Metaio dataset. Targers from left: Bump, Stop, Lucent, Board, Isetta, Phila, Grass, Wall. Challenges from left: Low, Repetitive, Normal, High Texturedness [28].

Although the setup for dataset generation is carefully designed and seems promising, there are

Table 5.1: Parameter Setting for Image Acquisition

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

still a few flaws that can be eliminated. Firstly printed posters instead of real objects are used which is kind of synthetic, as mentioned previously. Secondly, users don't have the flexibility to use their own tracking evaluation methods since Metaio withhold full tracking ground truths.

5.1.2 TMT Dataset

In order to perform a better evaluation in high accuracy tracking, the new TMT dataset[2] [33] is proposed. In contrast to some datasets like [18, 28] that use posters as targets to record sequences, we directly make use of real objects such as cereal box or mug to record our sequences. A GRAS-20S4C-C firewire camera equipped with a Kowa LM6NCM F1.2/6mm lens was used and all the videos/sequences were recorded at 30 Frames Per Second (FPS). Each image frame in the sequences has a resolution of 600×800 in YUV color space. All the sequences are recorded under two lighting conditions: normal lighting and diffused lighting. More detailed parameter settings for image acquisition can be found in table 5.1 [33].

The TMT dataset contains over 100 sequences, which are mainly recorded and categorized based on different types of object motions involved. The object motions are performed in different manipulation tasks conducted by human or robot arms. Sequences can be roughly categorized into two parts: oriented motion tasks and composite motion tasks. For oriented motion task sequences, only one main type of motion is involved in each sequence, such as x-y translation, scaling caused by 3D object translational motion, in-plane rotation and image deformations caused by 3D object rotations. Five different motion speeds varying from very slow (s1) to very fast (s5) and one increasing speed (si) are performed for each motion type by human users. For composite motion task sequences, only one speed is used and generally each sequence consists of more than one main motion types. All the sequences of the two tasks are recorded in two different lighting conditions: normal lighting (nl) and diffusion lighting (dl). More detailed information about different challenges the TMT dataset involved can be found in Table. 5.2.

We have generated and also released all the ground truths for all the TMT sequences. The ground truths are formed as sub-pixel level coordinates positions of four corners of the target in each frame. As long as the target is planar, this can accurately models the image projection of objects (under perspective camera model). The ground truths are solely based on the tracking results

[2]This work is in collaboration with Ankush Roy

Table 5.2: Tracking Challenges of Sequences

Category	Video	Object	Tracking Challenges
Oriented Motion Tasks	Juice	Juice Box	RO, SR
	Cereal	Cereal Box	RO, SR
	BookI	Book	PR, SR
	BookII	Book	SC
	BookIII	Book	OC, RO
	MugI	Coffee Mug	SR, TX
	MugII	Coffee Mug	PR, SR, TX
	MugIII	Coffee Mug	PR, SR, TX
Composite Motion Tasks	Bus	Toy Bus	TX, SC, PR
	Highlighting	Newspaper	OC
	Letter	Envelope	PR, SR
	Newspaper	Newspaper	PR, SC, SR

RO:Rotation; SR:Specular Reflection; PR:Perspective Deformation;
SC:Scale; OC:Occlusion; TX:Low Texture.

of three trackers [2, 6, 16]. For each sequence, three trackers are initialized at the first frame and start tracking the remaining frames until one of the tracking results of the three trackers doesn't agree with the other two. This is done by simply measuring the bounding box alignment errors, see next section for more details. Reinitialization will be performed in case failure of any tracker is detected. Trackers will be reinitialized in the previous several frames instead of the frame that cause failure. The final ground truths can be generated by averaging the new tracking results of the three trackers. We also verify all ground truth data manually. More detailed information can be found in the TMT webpage[3].

5.2 Evaluation Methods

We have introduced two datasets which are designed for registration based tracking evaluation in the previous section. In this section, we will first introduce the error metrics we used to evaluate the high accuracy registration based trackers. Then several evaluation metrics for registration based trackers will be introduced, including Average Drift and Overall Success Rate. The former is mainly used to measure tracking accuracy and the latter is used to measure tracking robustness. Meanwhile, two new experimental setups are discussed. The first one is to evaluate the algorithm based on synthetic images generated from a single Lena image and the second one is based on real image sequences.

5.2.1 Evaluation Metrics

Wu, et al. [38] introduced two error metrics for tracking evaluation: the center location distance and the normalized bounding box overlap between tracked results and the ground truth. However, both of them are designed for measuring the robust-oriented tracking algorithms. For example, even

[3] `http://webdocs.cs.ualberta.ca/~vis/trackDB`

the ground truth bounding box is rotated $180°$, the error between the new tracking result and the ground truth will still be relatively small. But in applications like robot manipulation, a large error should be computed, which has been pointed out in [33]. Instead, we will use the corner based error metric which has been used in [28, 33]. The error metric for registration based tracking evaluation is introduced below:

$$MCD(c, gt) = \sqrt{\frac{1}{4}\sum_{j=1}^{4} ||c_j - gt_j||_2^2} \tag{5.1}$$

Where $c = \{c_1, c_2, ..., c_4\} \subset \mathbb{R}^2$ represents the four corners of the tracked object bounding box in the current frame. Similarly $gt = \{gt_1, gt_2, ..., gt_4\} \subset \mathbb{R}^2$ contains the corresponding corners of the ground truth bounding box. We have used this error metric as the basic performance measure in our experiments and refer to it as the Mean Corner Distance (MCD) error. In addition, we have adopted two evaluation measures which were introduced in [33]: average drift and overall success rate.

Average Drift: Given a sequence F with n frames, the average drift of a tracker is defined as the average MCD error of those frames that have been successfully tracked, i.e., whose MCD is below the threshold *TH* pixels (Eq. 5.2).

$$AD(C, GT, S) = \frac{\sum_{s \in S} MCD(C_s, GT_s)}{|S|}, \tag{5.2}$$

where the index set S is defined as $S = \{s : MCD(C_s, GT_s) \leq TH\}$ while C and GT are sets containing c and gt for all n frames respectively. $|S|$ is the number of elements in set S.

Success Rate: Given a set of frames F, the success rate of a tracker is defined as the ratio of the number of successfully tracked frames and the total number of frames. The definition of a successfully tracked frame is the same as in the definition of average drift. Given index set S

$$SR(S, F) = \frac{|S|}{|F|} \tag{5.3}$$

5.2.2 Static Experiment

The original static experiment proposed by [2] and later used by [6, 16] is designed like this: a subregion with corner coordinates $c = \{c_1, c_2, c_3, c_4\}$, is first selected in the original image I_0 (usually Lena's image) as region of interest. Next, a Gaussian noise with mean 0 and variance σ is added to each $c_i \in c$ and results in c'.

Given c and c', a homography with parameters p can be estimated by applying the Direct Linear Transformation (DLT) algorithm [22]. Given I_0, we can get a warped image $I_w = I_0(W(\mathbf{R}; p^{-1}))$ where W is the warp function and $\mathbf{R}$ is the set of reference points. The goal of this experiment is to initialize each tracker on I_0 with target defined by corners c, to track the warped image I_w, see Fig. 5.2. The tracking results $\hat{c}$ on I_w should be close enough to c' if successfully tracked. In the complete experiment, the σ of Gaussian noise varies from 1 to 20, and 5000 trials are conducted for each σ to generate 5000 warped images. Thus for each σ, the success rate can be computed using

Figure 5.2: Left: original lena image with initial bounding box; Right: Warped image with the bounding box that need to be found by the trackers [16]

Eq. 5.3 based on the tracking results of the 5000 warped images. The full algorithm of the original static experiment can be found in Algorithm 2.

Algorithm 2 Original Static Image Experiment

I_0 *is the Lena's Image*
$set = dict\{\}$
for $\sigma = 1 \rightarrow 20$ **do**
  for $i = 1 \rightarrow 5000$ **do**
    $c' = c + GaussianNoise(mean = 0, variance = \sigma)$
    $p = DLT(c, c')$
    $I_w = I_0(W(\mathbf{R}, p^{-1}))$
    $set[\sigma].append(I_w)$
  end for
end for

The standard test introduced above may not be able to give a precise indication of convergence. The issue is that, even for distributions with large σ, many samples are drawn with small inter-frame motion. Hence a tracker will look as if it converges for a large perturbation σ, while most of successful trials in the graph may actually correspond to small motions [33].

Instead, we propose to use MCD error to represent the motion displacement. We vary MCD values α from 1 to 20 and for each α , we randomly generate a warping set consisting of 5000 warping parameters $\mathbf{p}_\alpha$. The $MCD(c_i, c')$ corresponding to each parameter p in $\mathbf{p}_\alpha$ is between $\alpha - 1$ and α. Each c' can be generated by calling the $GenerateRandomCorners(c, \alpha)$. SR for each α is generated in the same way as Eq. 5.3. The advantage of this modification is that it ensures sufficient number of samples are generated for different magnitudes of motion displacement so that the result is statistically correct. The full algorithm for the modified static experiment that generating warped images can be found in Algorithm 3.

Algorithm 3 Modified Static Image Experiment

```
1: I_0 is the Lena's Image
2: set = dict{}
3: for mcd = 1 → 20 do
4:    for i = 1 → 5000 do
5:       c' = GenerateRandomCorners(c, mcd)
6:       p = DLT(c, c')
7:       I_w = I_0(W(R, p^{-1}))
8:       set[mcd].append(I_w)
9:    end for
10: end for
```

5.2.3 Speed Sensitivity Experiment

In the previous subsection, we have introduced the modified static experiment to evaluate the convergence of a tracking algorithm. Although the idea of synthetically generating warped images from a static image is good, there is still limitation in mimicking the real inter-frame object motion. Noises and influences from environment such as lighting variation, image blur may cause tracking algorithms fail while not the case in the static experiment. In order to overcome the shortage, we propose another experimental setting which referred to as the speed sensitivity experiment [33].

The main idea of the speed sensitivity experiment is to generate a set of image pairs (I_m, I'_m) from real image datasets, where m indicates the inter-frame motion in MCD error between I_m and I'_m. m varies from 1 to a specific value (15 in our experiment) and for each m value, we need to generate a sufficient number of image pairs. Note the m is analogous to the α in the static experiment. With a larger m value, the tracker should be harder to track the image pair (Initialized on I_m and try to track I'_m).

There are many different ways to generate such image pair sets, we introduce a very straightforward approach, by sampling image pairs from real image datasets in two different directions: forward and backward. Given I_i which is a frame in a image sequence, the second image to-be-tracked can be I_{i+s} where s is the step that can varies from negative $-j$ to positive j and $s \neq 0$. As long as there is challenges like image blur, partial occlusion in the real image dataset, these challenges should also be included in the new set of image pairs, thus covered in the speed sensitivity experiment. In case there is less number of image pairs contributing to a higher m, more than one sequences can be used to generate enough image pairs. The overall algorithm for generating the image pairs in our experiment can be found in Algorithm 4.

Once the image pair set is generated, we can simply apply the same method like the static experiment, by computing the the SR using Eq. 5.3 for each inter-frame motion m. In order to conduct the experiment, the ground truth of the dataset have to be known. This is the main reason we only use the TMT dataset in our experiment, since the ground truths of the Metaio dataset are not available.

Algorithm 4 Speed Sensitivity Experiment

```
1: I is an Image sequence
2: set = dict{}
3: for each new frame I_i do
4:    for s = −5 → 5 do
5:       if I_{i+s} is valid then
6:          mcd = floor(MCD(I_i, I_{i+s}))
7:          set[mcd].append((I_i, I_{i+s}))
8:       end if
9:    end for
10: end for
```

5.3 Summary

In this chapter we first reviewed several image datasets that are publicly available and can be used to evaluate registration based algorithms. We then introduced two dataset, the Metaio dataset which is already available and commonly used and the new TMT dataset with ground truth released. Meanwhile, several popular evaluation methods were also reviewed. We showed the shortage of the classical static image experiment and introduced the modified static experiment. Since the static experiment is based on synthetic images, in order to take the real noise and environmental interference into consideration, we introduced the new speed sensitivity experiment which is based on the real image dataset. Both the datasets and the evaluation methods will be used to evaluate our tracking algorithms. The results can be found in Chapter 6.

Chapter 6

Experimental Results

In this chapter, we will start with an introduction of the different trackers setup. Then the experimental results and analysis of the RKLT algorithm on TMT and Metaio datasets will be presented. Finally, we further investigate the influence of different parameterizations on the performance of the ESM algorithm.

6.1 Trackers Setup

In this chapter we will first compare our new algorithm RKLT with three registration based trackers: the Inverse Compositional (**IC**) tracker [2], the original ESM tracker (**ESM**) [6] and the Nearest Neighbor based tracker (**NNIC**) [16]. In order to show the influence of different parameterizations on the tracking performance, we further compare the performance of the corner based parameterizations on ESM with the original Lie based and also the pure homography based ESM. All the trackers will use an 8 DOF state space model in order to tracking complex object motions. All the experiments are done using our integrated system introduced in Chapter 4. A few common image processing steps are done before feeding images to trackers. Image preprocessing includes image conversion from RGB to Grayscale, image smoothing and image filtering using Sum of Conditional Variance (SCV) [31].

The parameter settings can have a significant impact on the performance of the tracking algorithms. Thus we retain the original parameter settings as mentioned in [16] for all the three trackers, in order to compare them with our proposed algorithms in a reasonable way. For the IC tracker, we used a template resolution of 100×100 and ran for a maximum of 30 iterations. For the ESM tracker, the resolution used is 50×50 and also ran for a maximum of 30 iterations. The NNIC tracker has the same configuration as in [16], with 3 layers of individual Nearest Neighbour trackers followed by an IC tracker. The template is sampled with a resolution of 50×50. Parameter settings for the individual NN tracker and IC tracker are the same as in [16].

6.2 RKLT Results

Here we present the robustness and accuracy evaluation of the RKLT algorithm. The accuracy is measured based on the simple average drift measure on the TMT dataset (The Metaio benchmark doesn't provide such evaluation). The robustness of the RKLT algorithm is evaluated based on three experiments: the modified static experiment, the simple success rate measure ran on two datasets (TMT and Metaio) and the speed sensitivity experiment ran on TMT dataset. All the experiments will be conducted in comparison with three other registration trackers described in the trackers setup section.

We followed the same convention used in [16] for setting the RKLT tracker parameters, which is to maximize the tracking performance while trying to give each algorithm approximately the same running time. For RKLT, we used a template resolution of 40×40 and a maximum of 10 iterations for the IC tracker in the refinement step.

6.2.1 Static Experiment

In the experiment, we conduct the modified static image experiment as we introduced in the previous chapter. For each inter-frame ranging from 1 to 20 which is measured in MCD error (Eq. 5.1), we generate 5000 warped images from the original Lena's image according to Algorithm 3. A region of 100×100 pixels located in the center of the original Lena's image is selected as the target, see Fig. 5.2. Trackers will be initialized with the original Lena's image and track the target in the warped images. Success rate is computed for each MCD value with a threshold of 2 pixels.

The results of the modified static experiment are plotted in Fig. 6.1. It is obvious from the figure that the IC algorithm performs the worst among the four algorithm. The success rate of IC drops steeply for MCD ≥ 1. The main reason is because of the simplicity of the IC algorithm with gauss newton method, which is based on the first order Taylor expansion [3].

The ESM algorithm is much better than IC, of which the success rate begins to decrease at around 10. This is expectable because the ESM algorithm is more complicate (based on second order approximation [6]) and can always achieve higher convergence than IC. The NNIC algorithm performs closely with the RKLT algorithm and both of them outperform the other two. The multi-layered structure of NNIC makes it suitable to track large inter-frame motion with good accuracy. Individual NN trackers that is tuned to track large motions will be first applied to track and intermediate results will be refined by the ones tuned for accurate tracking.

The RKLT algorithm is slightly better than the NNIC algorithm when approaching $MCD = 20$, where the inter-frame motion is very high. The success rate value of the RKLT (0.89) is more than twice than the one of the ESM (0.34). The main reason to it's high convergence is because the pyramid KLT trackers are used in our implementation [8].

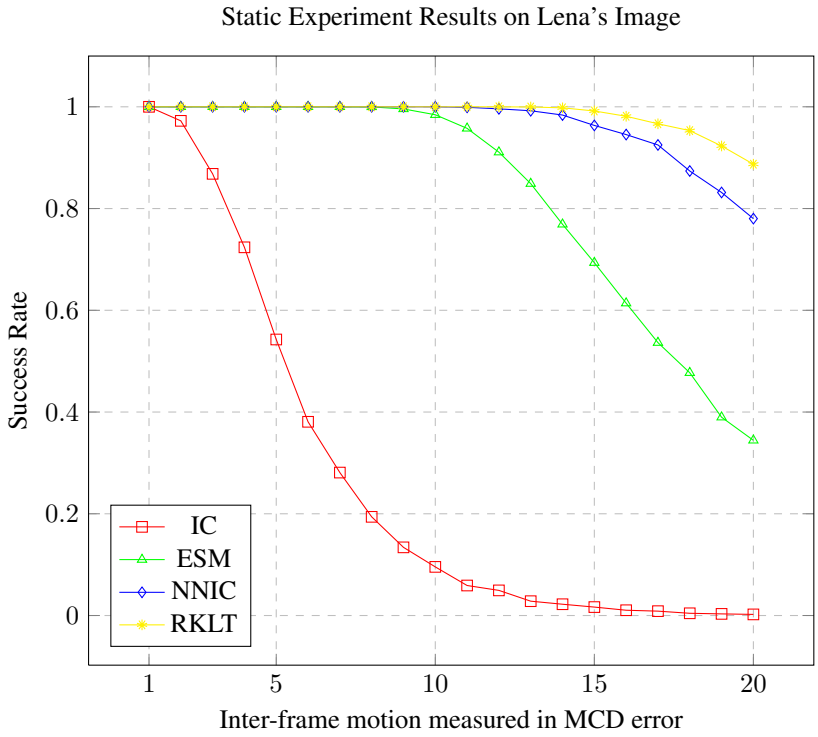


Figure 6.1: The modified static experiment is conducted (see Algorithm 3) with 5000 warped images generated for each type of inter-frame motion measured in MCD error. The threshold for calculating the success rate is 2 pixels.

6.2.2 Experiments on TMT Dataset

In this subsection, we will evaluate the four registration based trackers in addition with some learning based trackers. The dataset we used is the TMT dataset [33], which is designed for registration based trackers evaluation. We used both the evaluation metrics Average Drift (AD) and Success Rate (SR) introduced earlier for the experiments. The TMT dataset provides about 100 different image sequences. In order to evaluate these trackers with all motion types and with different target objects, we used the oriented motion sequences of TMT dataset, which includes *bookI*, *bookII*, *bookIII*, *mugI*, *mugII*, *mugIII*, *cereal* and *juice*. In order to better distinguish between different trackers, we focus on those with high motion speeds s4 and s5, of which sequences are harder to track. Challenges of each sequence can be found in Table 5.2.

Average Drift Results

The average drift (AD) metric is mainly used to measure the tracking accuracy defined in Eq. 5.2. The experiment is done on the TMT dataset in a straightforward way: for each sequence, the trackers are initialized in the very first frame and track the rest. The tracking results will be used to compare with TMT's ground truths and AD value for each sequence will be computed using Eq. 5.2. Note the AD values are averaged on two different speeds: fast (s4) and very fast (s5).

The average drift of each of the tested trackers can be found in Table 6.1. In this experiment, we not only test the four registration based trackers used in the static experiments, but also three learning based trackers, in order to compare the tracking accuracy between the two types of trackers. We used the IVT by Ross et. al [32], L1-APG [4] by Bao et. al and TLD [26] by Kalal et. al as representatives of learning based trackers. All the three trackers were running with the default parameter settings.

The first observation from Table 6.1 is that the learning based trackers always have relatively larger AD values compared to the registration based algorithms. Among the learning based algorithms, TLD tracker seems to perform the worst since for almost half of the sequences, there is no frame with tracking error MCD value below 4 pixels (N/A in the table). The main reason to the low accuracy of TLD is that a low DOF state space model is used. Since TLD tracker only has 3 DOF state space model, which can only track target translation and scaling. Thus it is not likely to perform as well as IVT tracker and the L1-APG tracker, both of which use 6 DOF affine model as their state space model. While for IVT and L1-APG, they still can't track as accurate as the registration based algorithms. Since the search method (particle filter) they used can't achieve similar accuracy as the iteratively methods used by the four registration based algorithms. Meanwhile, both of the two algorithms are continuously learning the appearance model from tracking results that are not so accurate, which in turn affects the accuracy of new tracking results.

While for the four registration based trackers, all of them can achieve sub-pixel level accuracy on more than half of the sequences. IC has the lowest AD values for 5 out of 8 sequences. The main reason is because it uses the highest resolution 100×100 for the target compared to other trackers. For RKLT algorithm, although not having the lowest MCD values for all the sequences, is very close to the lowest ones. Since we only use 10 iterations as the maximum number of iterations for the refinement step, the accuracy of RKLT can be improved by simply increasing the maximum number of iterations.

Table 6.1: Average Drift on TMT Dataset

	IVT	L1-APG	TLD	IC	ESM	NNIC	RKLT
BookI	1.29	2.94	N/A	0.62	0.62	0.68	0.70
BookII	3.47	2.30	N/A	0.58	0.45	0.46	0.50
BookIII	3.06	2.69	N/A	1.63	1.63	1.61	1.62
MugI	3.01	1.82	3.18	0.36	0.41	0.37	0.45
MugII	2.31	1.88	3.43	1.70	1.70	1.71	1.70
MugIII	2.87	1.70	2.97	0.60	0.65	0.71	0.77
Cereal	2.68	3.86	N/A	0.07	0.22	0.22	0.40
Juice	3.27	2.73	1.59	1.42	1.42	1.40	1.49

N/A: No frame has MCD error below *TH=4* pixels

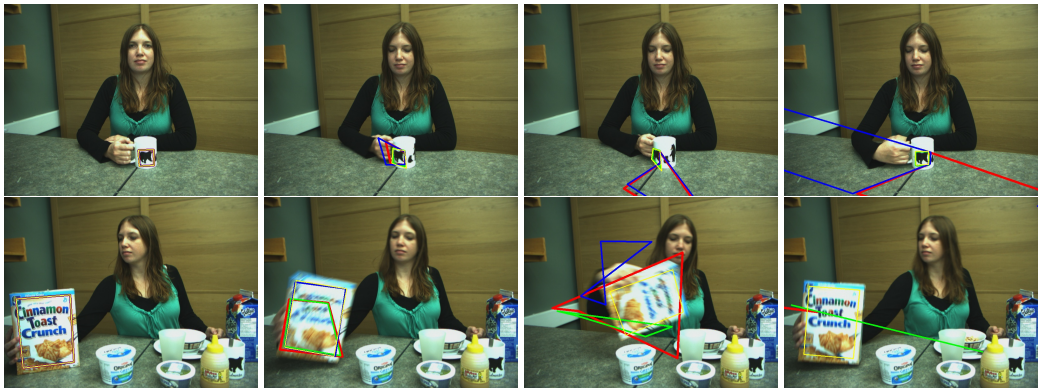


Figure 6.2: Tracking results of four trackers on two sequences. Red: IC; Green: ESM; Blue: NNIC; Yellow: RKLT. Top row: *mugIII*_s5. The target is non-planar and has little texture. IC and ESM failed when the mug is heavily rotated. Bottom row: *cereal*_s5. All the trackers failed except RKLT when motion blur occurred.

Success Rate Results

The experiment is done by first initializing trackers on the first frame of each sequence, then computing the success rate for each sequence according to Eq. 5.3. This is one of the experiments aimed for evaluating the robustness of trackers towards different challenges that the TMT sequences have included.

The success rates results of the four registration based trackers are shown in Fig. 6.3. In the figure, it is clear that all the four trackers perform equally well on sequences like *bookI*, *bookII* and *mugI*. The main reason is the book target has relatively high texture and no serious interferences, thus makes it easy for registration based trackers to track well. While for *mugI*, although the mug target is small and has low texture, it's still easy to track since it only involves y-axis translation. For *bookIII* sequences, large occlusion occurs which cause failure to all the trackers. As a result all four trackers have lower success rates on *bookIII*. *mugII* and *mugIII* are relatively harder to track since the two sequences involve large out-of-plane rotation with a low texture and non-planar target. All the trackers fail quickly on the *mugII* sequence. For the *mugIII*, especially for the fastest speed sequence s5, RKLT tracks pretty good and is closely following the ESM. Similar results can also be found in the $cereal\_s5$ sequence and $juice\_s5$ where in the case, the RKLT algorithm outperforms the other three with almost twice the success rate. Fig. 6.2 shows some tracked frames of two sequences *mugIII* and *cereal* with the fastest speed.

The RKLT algorithm tracks well especially for the fast motion sequences. This is mainly due to the advantage of the first two steps of the RKLT algorithm, where pyramid KLT trackers [8] are used and the global target motion is estimated based on those tracking results. ESM tracks the best on *mugIII*_s5 sequence. While for IC, again it's not performing as good as other trackers due to it's simplicity as we pointed out earlier.

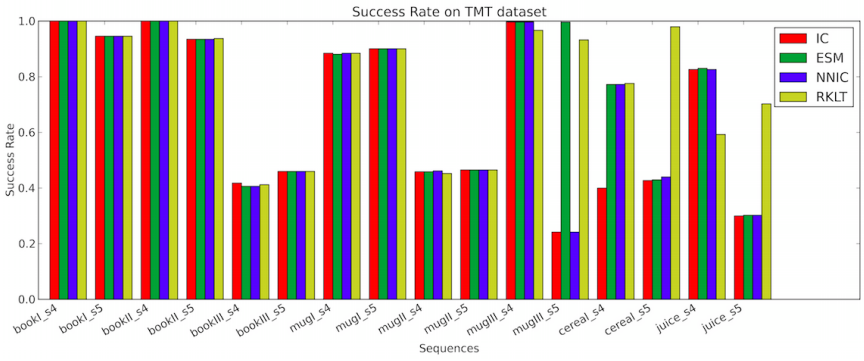


Figure 6.3: Success Rate Comparison among IC, ESM, NNIC and RKLT trackers. Each tracker is initialized on the first frame of each sequence from the TMT dataset and track the rest frames. Success rate for each sequence was computed with a threshold of 4 pixels. Only sequences with fast (s4) and very fast (s5) speeds are used in the figure.

In addition, we also conduct another type of comparison to show the significance of the four steps of the RKLT algorithm. Two more trackers were implemented by making slight modifications to the original RKLT algorithm. The first one selects points using the feature selection method described in [36] instead of evenly sampling points on the target. We refer to it as **RKLT+GoodFeat** tracker. The other one removes the refinement step while leaving other steps unchanged and thus we refer to it as **RKLT+NoRefine** tracker.

The results are shown in Fig. 6.4. RKLT+GoodFeat tracker can track equivalently well on most of the sequences as the original RKLT. But the good performance of RKLT+GoodFeat depends on a sufficient number of features detected and selected by using feature selection in [36]. In particular, we up-sampled the target region when the target is relatively small and used a low threshold for accepting more corners as good features. Only when the detected number of points is in the similar level as which used by RKLT (1600 for 40×40 template), will the RKLT+GoodFeat achieve similar performance. In practice, the strategy of evenly sampling used by RKLT is good enough for tracking regular size of target. RKLT+GoodFeat tracker may be more efficient when the target has a relatively large size and with low texture.

RKLT+NoRefine tracks well only on *mugI* which only has translational motion but is significantly worse for all the other sequences that have more complex motions. The main reason is that for the first two steps of RKLT algorithm, only coarse results are estimated, without refinement the error keep accumulating as tracking proceed.

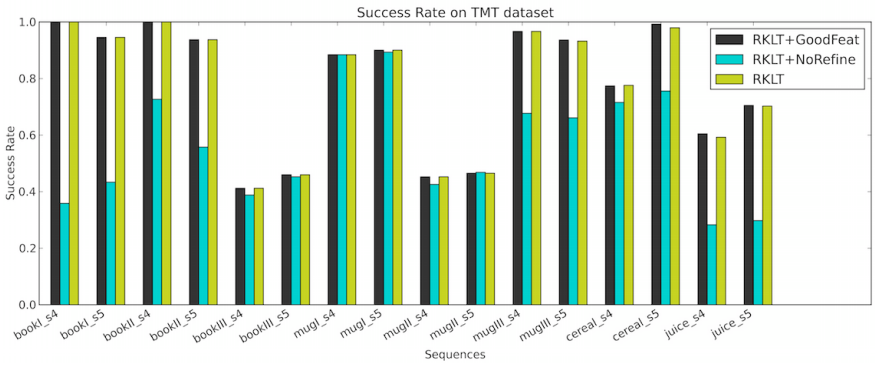


Figure 6.4: Success Rate Comparison among RKLT and its two variants. RKLT+Goodfeat selects positions for initializing KLT trackers using [36] instead of doing evenly sampling. RKLT+NoRefine removes the refinement step of RKLT tracker. Success rates are computed on TMT dataset with a threshold of 4 pixels

Speed Sensitivity Results

The speed sensitivity experiment is conducted by first generating sets of image pairs as indicated by Algorithm 4 on the TMT dataset, of which the ground truths are all released for all the sequences. Then compute the success rate for each set of image pairs. Instead of using s4 and s5 speed sequences only, in this experiment we use sequences with all five speeds from s1 to s5 in order to generate sufficient number of image pairs, especially for those with large inter-frame motions.

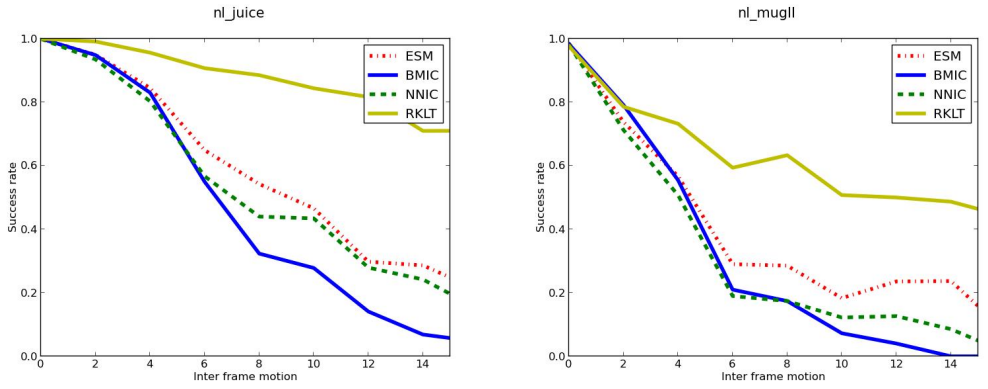
Fig. 6.5 shows the speed sensitivity results for *juice* and *mugII* sequences. It is clear that RKLT performs better in high inter-frame motion (with larger MCD values) than other trackers. Note that *juice* has challenges like in-plane rotation and specular reflection, while *mugII* has low texture object and also out-of-plane rotation. In both scenarios, RKLT can still achieve good performance, especially for large inter-frame motions.

Results for other sequences can be found from Fig. 6.11. One observation is that the success rate of the same algorithm varies among different figures in the speed sensitivity experiments. The main reason is that the image pairs are generated from real image sequences (TMT) with different target and including different types of challenges.

The performance of the ESM and NNIC in Fig. 6.5 and Fig. 6.11 are closer compared to the static experiment in Fig. 6.1. The NNIC seems better in *bookI*, *mugIII*, while in sequences like *juice*, *mugII*, ESM is slightly better. Overall, the two performs closely with each other. Since the IC algorithm is the simplest one as we mentioned before, it always cannot track as good as other three algorithms.

6.2.3 Experiment on Metaio Dataset

In the experiment we run the four registration based trackers on the full Metaio sequences, a very popular dataset designed for registration based tracking evaluation. For each sequence, each tracker will be initialized for the first and every 250th frame in case the trackers failed too early [28]. The



(a) Juice (b) MugII

Figure 6.5: Speed sensitivity results (see Algorithm 4) of four trackers on TMT dataset: IC, ESM, NNIC and RKLT. A threshold of 4 pixels is used for computing the success rates. Sequences of speeds from very slow (s1) to very fast (s5) are used for each figure.

results of the four trackers on this dataset are summarized in Table 6.3. The performance is measured in terms of success rate for each sequence in percentage. It is worth noting that the evaluation is performed at Metaio since detailed tracking ground truth coordinates are withheld from researchers.

Visual trackers proposed in [13, 35] have also been evaluated on Metaio dataset. But these trackers are tuned to achieve better performance during evaluation with a higher template resolution and a larger number of iterations, regardless of tracking speed. For example, the NCC based tracker introduced in [35] performs well in general on Metaio and achieves outstanding performance for sequences with illumination challenge, where more than 99.8% of the images are tracked. In their experiment a 320×240 template resolution is used by the tracker which is allowed to run at most 200 iterations for each frame. While in our experiments, we used relatively low template resolution and maximum number of iterations so that the trackers can achieve real-time speed. Since it's unfair to compare trackers with different parameter settings, we didn't compare these evaluation results.

We observe that RKLT achieves a good performance especially in *Angle* (Seq01) and *Illumination* (Seq05) sequences. The *Angle* sequences mainly involve a large amount of in-plane/out-of-plane rotations. The RKLT algorithm performs the best on 5 out of 8 sequences in the category, followed by the NNIC tracker who wins 3 out of 8. The results of *Angle* sequences are plotted separately in Fig. 6.6 for convenience. Although only translational KLT trackers are used, there is still a sufficient number of trackers succeed since the KLT trackers are initialized on every new-coming frame and track inter-frame motion. Thus the small template for each point tracker will be updated in order to handle the target rotation.

While in the *Illumination* sequences, the main challenge is the heavy illumination variation involved in the sequences. In Table 6.3, the RKLT algorithm also wins 5 out of 8 sequences in this category. The results indicate that the RKLT algorithm is robust towards illumination variance. Although part of the KLT trackers may fail due to the illumination variation, the RANSAC algorithm

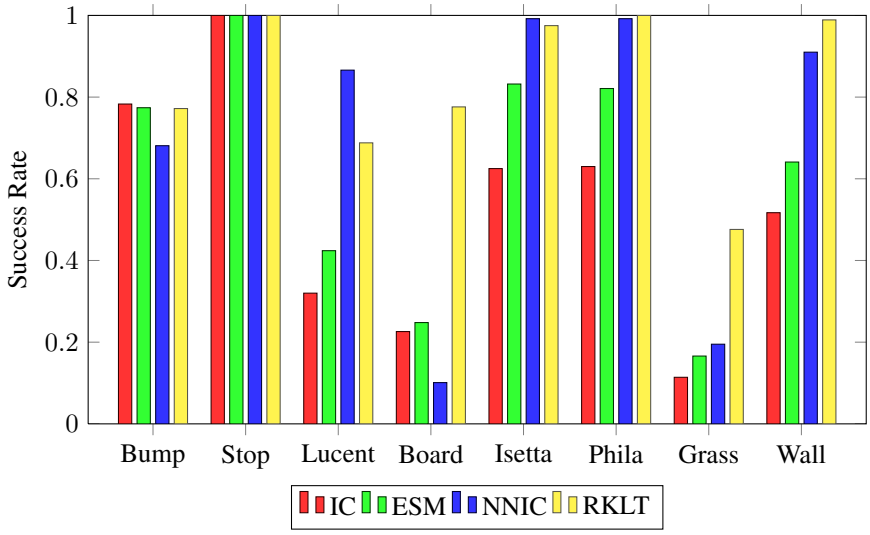


Figure 6.6: Success rate results of four registration based trackers on the *Angle* sequences of Metaio dataset.

can still get a good estimation of the target motion based the noisy tracking results. The ESM tracker wins 3 out of 8 sequences and achieve better performance than the NNIC tracker in term of illumination variance.

It is also obvious from Table 6.3 that the NNIC tracking results on *Fast Far* (Seq03) are the best among the four registration based trackers, who wins 5 out of 8 sequences. Since in *Fast Far* sequences, the target is relatively small and move in a very fast speed (mostly translational motion). The multi-layer structure and nearest-neighbor based image searching for tracking updates enable the NNIC tracker to track small objects with fast motion. Although the RKLT algorithm doesn't win as many sequences as NNIC, for all of the sequences in *Fast Far*, RKLT always ranks within first two places.

Another interesting observation from the results is that Board and Grass (See Fig. 6.7) are the two most challenging targets for the registration based trackers. The reason may be that, the two targets contain repetitive texturedness with small pieces of repeated objects, which may make the registration based trackers easy to fail. The Lucent target also has repeated objects but with a larger size.

Figure 6.7: Two most challenging targets in Metaio dataset for registration based trackers

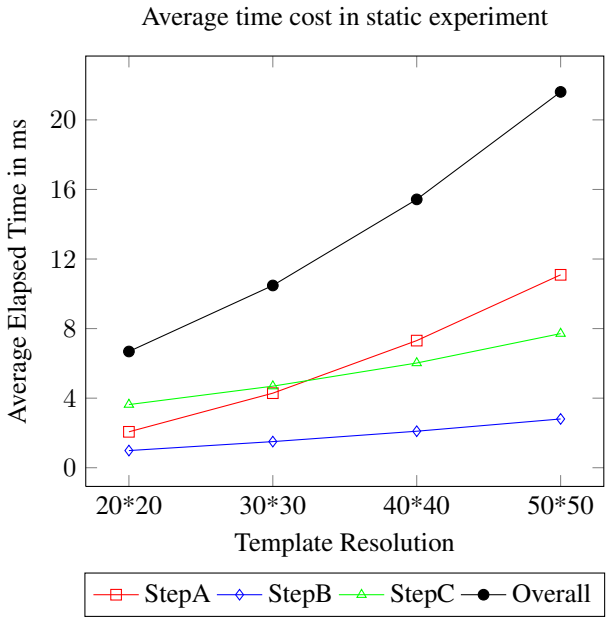


Figure 6.8: The time spent on first three steps of RKLT is measured when conducting the static experiment. The elapsed time is averaged for single frame tracking in milliseconds

In order to demonstrate the overall performance of the four trackers on Metaio dataset, we first counted the number of times each tracker got the highest number of frames tracked. The simple IC tracker was best 2 times, ESM tracker 10 times, NNIC tracker 16 times. Our proposed RKLT algorithm won 17 tests. The RKLT algorithm has highest number of sequences where the tracker is ranked in the first place and also within first two places, which indicates that the RKLT algorithm always achieve comparable tracking results with the state-of-art algorithms like ESM algorithm and NNIC algorithm.

6.2.4 Experiment on Time Efficiency

Tracking speed may also affect the performance of a tracker in real world applications. In real-time tracking, the tracker needs to process the image feed from the camera immediately because the object may keep moving while the processing is going on. A slower tracker will automatically down sample the image frames (i.e. skip intermediate frames) and the object motion displacement between two adjacent frames it processes will consequently be larger than for a tracker with higher tracking speed. Hence there is a tradeoff between tracking accuracy and speed in real applications. Since the frame rates of most cameras does not exceed 30 Hz, if a tracker can track at a speed above 30 fps (frames per second) in a single thread on a normal research machine, it then should be good enough to be used in real applications.

In order to analyze the tracking efficiency of the RKLT algorithm, we first measure the time spent on the first three steps of the algorithms separately when conducting the modified static experiment. Since the Failure Detection step is quite efficient, we didn't include it in our analysis. The elapsed

Table 6.2: Tracking efficiency on TMT Dataset (fps)

	IC	ESM	NNIC	RKLT
bookI	272.67	73.39	82.78	60.41
bookII	418.55	139.81	82.89	59.64
bookIII	72.47	44.80	80.52	72.07
mugI	180.18	51.74	81.57	61.47
mugII	69.04	43.78	74.85	56.06
mugIII	164.21	62.01	76.79	64.02
Cereal	122.49	67.08	78.64	63.08
Juice	101.48	37.66	81.59	65.63

time spent on each step is averaged for tracking a single frame during the static experiment. Note for all the experiments requiring time measurement, we only take into account the time spent running the tracking algorithm in a single processing thread. The measurements were done on a desktop computer with 3.5 GHz Intel processor and 32 GB RAM.

The results are plotted in Fig.6.8. The most time consuming part can be found from the figure is the sampled points tracking step (Step A) which is indicated in red line. In our implementation we use the high efficient KLT trackers provided in OpenCV [8]. Although it looks quite fast when the resolution is low (20×20), the computation increases linearly with the number of sampled points. The refinement step (Step C) increases, but not as fast as Step A. The main reason may be that the number of iterations allowed during searching is low (maximum of 10 iterations) and will stop as long as the tracking is converged. Although the overall time increases quickly, as long as we sample the template in a reasonable way, the algorithm is still efficient enough for real applications. For example, by tracking using 40×40 sampling resolution, the average cost for tracking a single frame takes about 15.5 milliseconds in the static experiments.

We also test the time cost for all the trackers running the success rate experiment on the TMT dataset. The results can be found in Table. 6.2. We averaged the time cost for different speeds (s4/s5) of the same type of sequence and represent the tracking efficiency in frames per second (fps).

With our implementation, the speed of RKLT (40×40 resolution) is stable among all the different sequences and is close to 60 fps on all the sequences. IC runs the fastest in most sequences which is expected since it is the simplest and most efficient algorithm, followed by NNIC which takes advantage of the efficient FLANN library. ESM has similar performance as ours but with a larger variance, which need to compute the Jacobian J_c for every iteration, as we pointed out in Chapter 2.

6.3 Corner based Parameterization Results

We have introduced a different parameterization of the 8 DOF state space model and have shown it's application to the ESM algorithm in Chapter 4. Here we present the experimental results of the new parameterized ESM (**Corner based ESM**) algorithm on the static experiment and also the overall success rate on the TMT dataset. The comparison is made with the original ESM algorithm (**SL3 based ESM**), the ESM algorithm without using the $SL3$ parameterization (**Homo based ESM**) and the three parameterizations on IC tracker. For the **Homo based ESM**, we approximately set J_{G_0} in Eq. 4.8 to be an identity matrix, thus can be ignored when calculating the Jacobian. Note for the different variants of ESM and IC, the parameter settings are exactly the same.

6.3.1 Experimental Results

The results for the static experiment can be found in Fig. 6.9. It is obvious from the figure that the three parameterizations on IC tracker have almost the same results. The results are expected because IC doesn't depend on any additional properties of the parameterization to find the Jacobians. Thus the three parameterizations are equivalent to each other. While for the ESM trackers, although the three may not be mathematically equivalent, the performance among the three is very close.

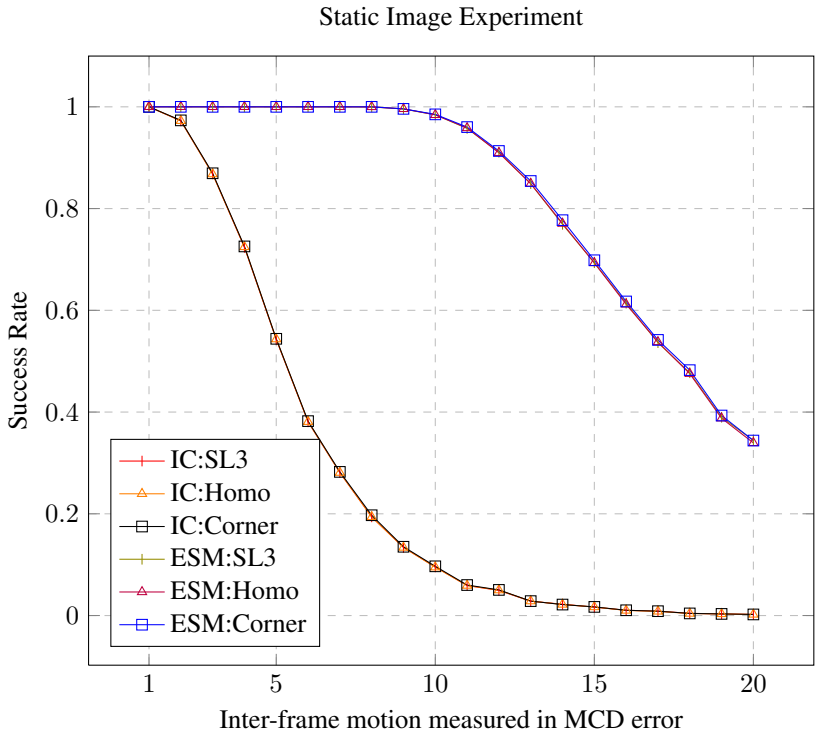


Figure 6.9: Three different parameterizations on both ESM and IC trackers are compared in the modified static experiment (see Algorithm 3). The threshold for calculating the success rate is 2 pixels.

Similarly, the overall success rates for oriented motion sequences on TMT dataset are computed.

the results can be found in Fig. 6.10. It is clear that for all the sequences, ESM using the three parameterizations gets almost the same success rates, so as the IC tracker. This agrees with the observation we found in Fig. 6.9. An interesting observation is that although the **Homo based ESM** couldn't achieve second-order convergence as the original **SL3 based ESM** (which has been pointed out in Chapter 4), the former performs surprisingly well and having results almost the same to the latter. One possible reason may be that it is the averaging of the two Jacobians (One is from the reference frame I_0, the other is from I_t) that mainly contributes to the good performance of the ESM tracker.

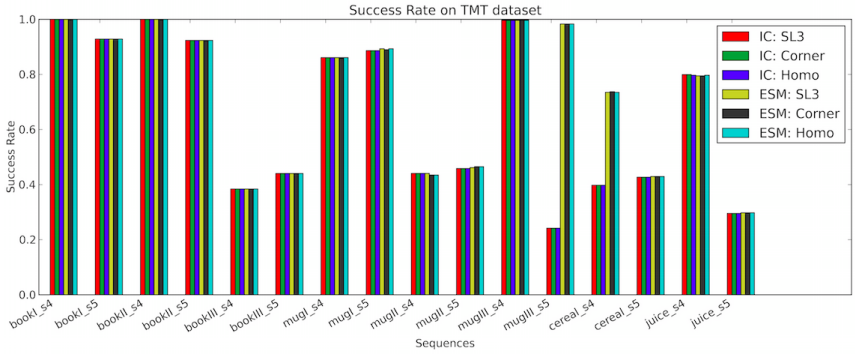
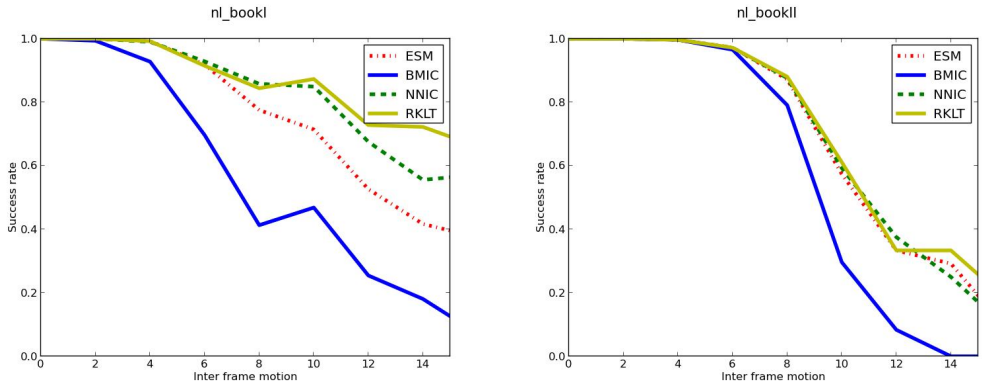


Figure 6.10: Success rate results computed on the oriented motion sequences of the TMT dataset. IC and ESM tracker with three different parameterizations are tested. A threshold of 4 pixels is used for calculating the success rate.

6.4 Summary

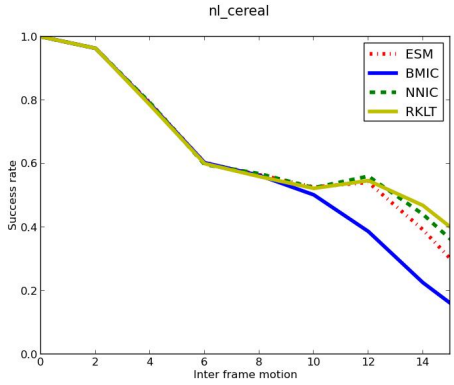
In this chapter, the experimental results of the new RKLT algorithm and three other registration based algorithms are first shown and compared. The static experiment on Lena's image is conducted, with the RKLT algorithm achieving the highest convergence. Then both average drift and success rate results are computed on the TMT dataset. The former is to evaluate the accuracy of tracking results and the latter is to indicate the tracking performance toward challenges included in the TMT dataset. Our RKLT algorithm achieves similar accuracy as other trackers, with good tracking results especially for fast motion sequences. We also conduct the speed sensitivity experiment on the TMT dataset, which is basically a variant of the static experiment, with new tracking images generated from the TMT dataset. The results agree with the static experiment. Since the Metaio benchmark is well accepted in registration based tracker's evaluation, we also did the test and find that RKLT tracks better than other three trackers especially when complex target motion were involved or illumination variance occurred. Time efficiency results are also presented. Although RKLT is not the fastest one, it is still very close to the tracking efficiency like ESM tracker with a more stable tracking speed on tested sequences. Finally, we also compared the corner based ESM with the original ESM and the approximated ESM with pure homography parameterization only. Two experiments are done, one for the static experiment and the other for overall success rate on TMT dataset. The experimental

results show that all the three parameterizations on ESM result in very similar tracking performance, although the ESM using pure homography is not equivalent mathematically to the ones with $SL3$ and corner based parameterizations.

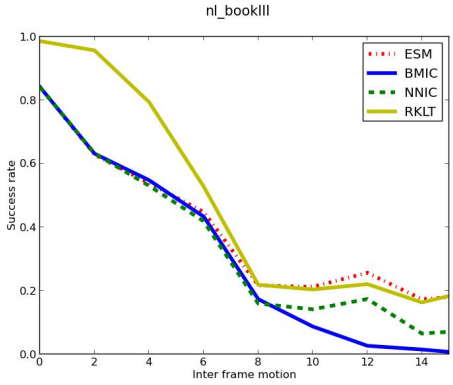


(a) bookI

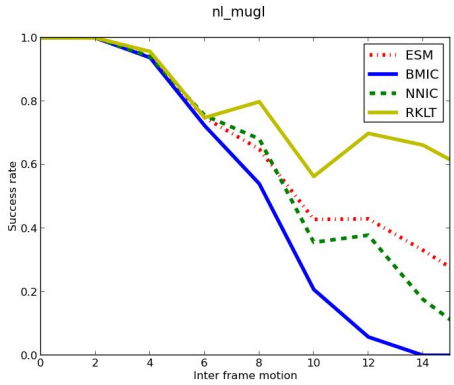
(b) bookII



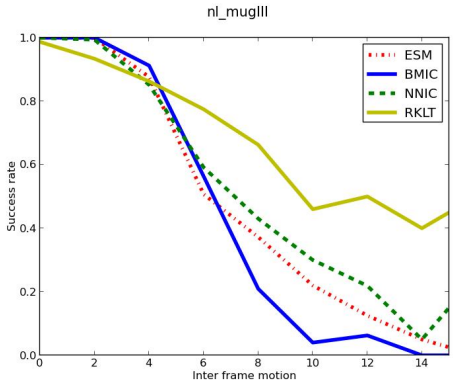
(c) cereal



(d) bookIII



(e) mugI



(f) mugIII

Figure 6.11: The speed sensitivity results (see Algorithm 4) on the oriented motion task sequences of the TMT dataset . Sequences of five different speeds from very slow (s1) to very fast (s5) are used for each figure. A threshold of 4 pixels is used for computing the success rates.

Table 6.3: Metaio Dataset Results (Success Rate in %)

Seq01	IC	ESM	NNIC	RKLT
Bump	78.3	*77.4*	68.1	77.2
Stop	100	100	100	100
Lucent	32	42.4	86.6	*68.8*
Board	22.6	*24.8*	10.1	77.6
Isetta	62.5	83.2	99.2	*97.5*
Philadelphia	63	82.1	*99.2*	100
Grass	11.4	16.6	*19.5*	47.6
Wall	51.7	64.1	*91*	98.9

Seq02	IC	ESM	NNIC	RKLT
Bump	70.9	91	*72.4*	*72.4*
Stop	72.8	96.2	*87.2*	83.8
Lucent	21.8	24.2	97.8	*51.7*
Board	7.9	*9.7*	2.8	20.2
Isetta	33.2	89	*83.1*	81.3
Philadelphia	53.2	89	99	*98.8*
Grass	8.6	*9.1*	25	7.2
Wall	26.1	39.3	98.2	*79.8*

Seq03	IC	ESM	NNIC	RKLT
Bump	20.6	50.8	56.5	*55.4*
Stop	12.2	24.9	*37.1*	68.2
Lucent	8.9	12.1	45.8	*21.9*
Board	4.6	*6.9*	5.7	10.4
Isetta	6.2	17.2	*74.8*	75.8
Philadelphia	6.7	14.6	38.2	*21.4*
Grass	5.7	7	7.8	*7.1*
Wall	8.1	8.4	70.9	*41.1*

Seq04	IC	ESM	NNIC	RKLT
Bump	30.6	40.9	33.8	*35.3*
Stop	30.4	54.1	*50.1*	26.9
Lucent	14.7	*17.6*	59.3	13.8
Board	23.5	*37.8*	3.9	38.1
Isetta	18	31.9	*21.4*	19.3
Philadelphia	50.5	*76.8*	78.2	58
Grass	3.7	*7.8*	5.4	8.4
Wall	12.4	*28.5*	24.2	50.4

Seq05	IC	ESM	NNIC	RKLT
Bump	*94.2*	95.9	78.8	48.2
Stop	62.7	62.9	75.8	*69.6*
Lucent	78.9	*89.6*	72.2	100
Board	11.9	*15.4*	0.8	39.4
Isetta	91.8	100	77.3	*99.5*
Philadelphia	99.4	100	100	100
Grass	14.2	*23.9*	8.9	31.7
Wall	73	*81.3*	72.9	82.2

Success rate of five challenges of sequences with different objects (in percentage). Seq01: Angle; Seq02:Range; Seq03:Fast Far; Seq04: Fast Close; Seq05:Illumination. Best tracker marked as red; Second best in *green*.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

In this thesis, a novel visual tracking algorithm called RKLT algorithm is proposed. The algorithm mainly takes advantage of the fact that different subregions of the target may have different effects on the estimation of the target motion. This is more obvious in real applications when images to-be-tracked contain camera noise, environmental interferences, etc. In contrast to the typical registration based trackers where full pixels in the target template are used, the RKLT algorithm can keep filtering out the reliable regions from the outliers and feed them into the refinement tracker, which results in a more accurate performance and also being less affected by the noise. By adopting an 8 DOF projective state space model, the RKLT tracker can track complex object motions with high accuracy and efficiency.

We also studied the different parameterizations on state space models for the gradient based tracking algorithms. We showed an example how corner based parameterization can be applied to ESM tracker. In the experiments, the performance of the three different parameterizations is presented. The results indicate that all the three parameterizations on ESM result in very similar tracking performance, although the ESM using pure homography should not be equivalent mathematically to the ones with $SL3$ and corner based parameterizations. To best of our knowledge, this observation hasn't been reported by other researchers.

In addition, the new TMT dataset and several evaluation methods including the new speed sensitivity experiment are also discussed in the thesis, which are all aimed for high accuracy trackers evaluation. The new dataset includes more than 100 image sequences with all the ground truths publicly available. In order to make it more convenient for people to use the dataset, we also provide several simple evaluation methods based on the dataset. Users are also encouraged to design their own since all the ground truths are released.

In the experiments, we first compared the registration based trackers with three other learning based trackers. The experiments demonstrate that the registration based trackers can achieve much better performance in terms of accuracy, compared to the learning based trackers. This is mainly

due to the high DOF state space model and also the iterative search methods used by the registration based trackers. A static appearance model with a simple template also contributes to tracker's accuracy, with an effect of reducing it's robustness. The RKLT algorithm is also compared with three other registration based algorithms on two datasets: the new proposed TMT dataset and the popular Metaio dataset. Our RKLT algorithm achieves at least comparable performance with ESM and NNIC algorithms, which are always regarded as the state-of-the-arts in registration based algorithms. The RKLT algorithm surpasses them in cases like fast and/or complex target motion involving in/out-of-plane rotations, lighting variation, etc.

7.2 Future Work

In the RKLT algorithm, we used a straight forward failure detection step to avoid drift caused by the refinement step. This could be further improved by introducing some similarity measure or a robust classifier to determine whether the tracking result is correct or not. Further improvements include adding strategies to raise the robustness towards large area of target occlusion or even full occlusion. Currently RKLT can only be able to handle partial occlusion when the occluded region is relatively small since the RANSAC step may not able to filter out all the outliers from the inliers when large occlusion involved. A detection augmented framework like [26] could meet the requirements.

We have both theoretically and experimentally shown the equivalence of the corner based parameterization with the original $SL3$ parameterization on the ESM algorithm and the IC algorithm. Further work could be done to apply this parameterization on the ESM tracker to multiple region tracking or mesh tracking for non-planar objects similar to [19]. Corners within the object region are shared by a number of subregions, which can be approximated as planes. The locations of these corners can be updated by the tracking algorithms. This enables us to track non-planar objects with algorithms like ESM which can achieve high convergence.

Bibliography

[1] 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.

[2] Simon Baker and Iain Matthews. Equivalence and efficiency of image alignment algorithms. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–1090. IEEE, 2001.

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

[4] Chenglong Bao, Yi Wu, Haibin Ling, and Hui Ji. Real time robust l1 tracker using accelerated proximal gradient approach. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 1830–1837. IEEE, 2012.

[5] Selim Benhimane, Alexander Ladikos, Vincent Lepetit, and Nassir Navab. Linear and quadratic subsets for template-based tracking. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–6. IEEE, 2007.

[6] 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.

[7] Selim Benhimane and Ezio Malis. Homography-based 2d visual tracking and servoing. *The International Journal of Robotics Research*, 26(7):661–676, 2007.

[8] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade feature tracker description of the algorithm.

[9] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5:1–10, 2001.

[10] Gary Bradski et al. The opencv library. *Doctor Dobbs Journal*, 25(11):120–126, 2000.

[11] Matthew Brown and David G Lowe. Automatic panoramic image stitching using invariant features. *International journal of computer vision*, 74(1):59–73, 2007.

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

[13] Amaury Dame and Eric Marchand. Second-order optimization of mutual information for real-time image registration. *Image Processing, IEEE Transactions on*, 21(9):4190–4203, 2012.

[14] Frank Dellaert and Robert Collins. Fast image-based tracking by selective pixel integration. In *Proceedings of the ICCV Workshop on Frame-Rate Vision*, pages 1–22, 1999.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] Gregory D Hager and Kentaro Toyama. X vision: A portable substrate for real-time vision applications. *Computer Vision and Image Understanding*, 69(1):23–37, 1998.

[22] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[23] Michael Isard and Andrew Blake. Contour tracking by stochastic propagation of conditional density. In *Computer VisionECCV'96*, pages 343–356. Springer, 1996.

[24] Xu Jia, Huchuan Lu, and Ming-Hsuan Yang. Visual tracking via adaptive structural local sparse appearance model. In *Computer vision and pattern recognition (CVPR), 2012 IEEE Conference on*, pages 1822–1829. IEEE, 2012.

[25] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2756–2759. IEEE, 2010.

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

[27] Junseok Kwon and Kyoung Mu Lee. Visual tracking decomposition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1269–1276. IEEE, 2010.

[28] 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.

[29] 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.

[30] Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. *VISAPP (1)*, 2, 2009.

[31] 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.

[32] 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.

[33] Ankush Roy, Xi Zhang, Wolleb Nina, Perez Camilo, and Jagersand Martin. Tracking benchmark and evaluation for manipulation tasks. In *Robotics and Automation, 2015 IEEE International Conference on*, pages 2448–2453. IEEE, 2015.

[34] Harpreet S Sawhney and Serge Ayer. Compact representations of videos through dominant and multiple motion estimation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(8):814–830, 1996.

[35] Glauco Garcia Scandaroli, Maxime Meilland, and Rogério Richa. Improving ncc-based direct visual tracking. In *Computer Vision–ECCV 2012*, pages 442–455. Springer, 2012.

[36] Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

[37] Kentaro Toyama and Gregory D Hager. Keeping your eye on the ball: Tracking occluding contours of unfamiliar objects without distraction. In *Intelligent Robots and Systems 95.'Human Robot Interaction and Cooperative Robots', Proceedings. 1995 IEEE/RSJ International Conference on*, volume 1, pages 354–359. IEEE, 1995.

[38] 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.

[39] 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.

[40] Kaihua Zhang, Lei Zhang, and Ming-Hsuan Yang. Real-time compressive tracking. In *Computer Vision–ECCV 2012*, pages 864–877. Springer, 2012.

[41] 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 *Computer and Robot Vision (CRV), 2015 12th Conference on*, pages 70–77. IEEE, 2015.