國立臺灣大學電機資訊學院電子工程學研究所

博士論文

Graduate Institute of Electronics Engineering
College of Electrical Engineering and Computer Science
National Taiwan University

Doctoral Dissertation

精準六自由度物體姿態之估測與追蹤

Accurate 6DoF Object Pose Estimation and Tracking

吳柏辰

Po-Chen Wu

指導教授：簡韶逸 博士

Advisor: Shao-Yi Chien, Ph.D.

中華民國 107 年 05 月

May 2018

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

國立臺灣大學博士學位論文
口試委員會審定書

精準六自由度物體姿態之估測與追蹤

Accurate 6DoF Object Pose Estimation and Tracking

本論文係吳柏辰君（F99943021）在國立臺灣大學電子工程學研究所完成之博士學位論文，於民國 107 年 05 月 22 日承下列考試委員審查通過及口試及格，特此證明

口試委員：　　（指導教授）

所長：

doi:10.6342/NTU201800854

致謝

從簽下逕讀博班申請書的那一天起，我就開始想像並期待著這一刻的到來。回首博班路，從最初的懵懂青澀，歷經途中的困惑茫然與奮力拼搏，到最後終於開花結果，或許旁人甚難領略整段過程的艱辛苦楚，但我深知這本論文確實得來不易。一路走來，除了需要倚賴自身的熱情與毅力，同時也承蒙了許多貴人的幫助與支持。

首先我想感謝我的指導教授簡韶逸老師。從大二那一年修了簡老師的交電課後，便深深被簡老師冷面笑匠的魅力所吸引，因此在升大三時義無反顧的選修了簡老師的專題研究，歷經十載直至今日。或許有些指導教授在面對大學生與研究生時會呈現截然不同的態度，但這十年來簡老師給我的感覺始終如一，除了保持既有的風趣節奏外，也總是盡心盡力幫助學生渡過每段困難與挫折。在前大半的博班生涯中，其實自己的研究與投稿結果往往不盡如人意，但簡老師自始自終沒有半句責備，反而四處尋找更多可用資源幫助自己渡過難關，我也才能得以在年限將至前順利完成博士論文與通過學位口試。我感佩簡老師的從一而終，也同時替十年前的自己感到驕傲，因為他沒有看錯人。

再來我想特別感謝我的共同指導教授楊明玄老師。明玄是簡老師在三年前某次我和泓諭合作的研究成果被審查者狠狠洗臉後請來幫忙指點迷津的高人，同時也是我生命中的貴人。在我們剛認識明玄時，明玄已是電腦視覺界名人堂等級的學者，原本以為平日事務如此繁忙的大師不太可能有暇顧及他校學生，萬萬沒想到明玄其實是個充滿熱情並且樂意幫助他人成長的親切前輩，不僅勞心費力和我們共同釐清研究主軸、合力完成論文投稿，甚至主動幫忙我們將履歷投至各大公司申請實習機會；即使當時自己對於國外的工作面試並不抱持著多大的自信，明玄始終是那個站在第一線鼓勵、並以行動支持我的人。在明玄的信心加持下，我才有幸在當年以實習生的身分加入臉書的混合實境研究部門，並得以在畢業前夕幸運拿到敝司全職研究員的聘書。我想我上輩子肯定扶了不少老太太過馬路，今生才能有此等福氣認識如此難得的生命貴人。

doi:10.6342/NTU201800854

接著我想好好地感謝我的兩位父母。爸爸是位偉大的男人，除了努力在外打拼支持家計，也總是以寬大度量包容家裡兩位不時無理取鬧的女人（笑）。媽媽同樣也是位偉大女性的代表，每天重複著上班工作、下班與爸爸一同整理家務的生活，至今已持續三十餘年。我們家雖不富有，但爸媽總是將最好的留給了他們的子女，並且打理好家庭與家族上的大小事宜，讓我和妹妹能夠無後顧之憂地擁抱自己的夢想。如果有人認為我做事仔細負責，那我肯定是從爸媽那裡學來了不少東西；生在一個充滿愛與幽默的家庭也讓我得以時常帶著微笑迎接生活上的種種挑戰。

最後，我想感謝從專題生時期便和我一起長期奮鬥至研究所畢業拿到最佳碩論的泓諭、幫助我共同完成 OPT Dataset 的岳穎及宣逸、帶我入門多媒體研究的 Larry 學長、實驗室所有曾經給予我研究上的建議與幫助的同學及學長姐學弟妹、臉書所有曾經參與 DodecaPen 專案的同事（尤其是 Rob 與 Kenrick，雖然我覺得你們大概看不懂我在寫什麼）、所有口委老師們的不吝指教與建議分享，以及所有和我合作過的專題生們。同時我也對專題生們感到頗為抱歉，因為許多時候我並不是很有把握自己的想法是可行的，實際上最後也證實許多的確不太可行，這主要是我的鍋，你們其實都非常優秀。能和大家合作是我的榮幸，也希望你們未來都能各自擁有自己的一片天。

謹以此論文獻予我所摯愛的家人與朋友。

2018.05.24 吳柏辰

doi:10.6342/NTU201800854

中文摘要

本篇論文主要探討的問題是如何從已校準過的相機影像中穩定且可靠地計算出目標物體相對於拍攝相機的六自由度姿態，而此六自由度姿態是由三自由度的旋轉變量與三自由度的移動變量所組成。

雖然目前文壇上已有許多研究人員提出不同種解決此類問題的演算法，但是由於各方在做演算法間的交互評比時所使用的測試影像序列資料往往不盡相同，甚至在大部分情況下接近真實情況的影像序列是缺乏的，導致目前無法得到不同類型的演算法在多種情況與條件下的客觀表現優劣分析。因此在本篇論文中，我們提出了一個同時具備彩色與深度影像序列的大規模之物體姿態追蹤基準資料庫，此資料庫不僅包含數種不同型態的平面與立體目標物外，也同時提供物體真實姿態值。此外，我們也針對目前現有的演算法在此基準資料庫上做了完整的表現評估，也分析了現有演算法效能提升手段的可能切入點。

即使多點投影演算法通常可以精準地計算出目標物體相對於拍攝相機的姿態，此種演算法本質上需要目標物本身有足夠的紋理表徵，並且其特徵點群能成功與相機影像中的特徵點群正確配對時才能順利計算出準確的物體姿態，這也是傳統特徵點演算法的主要缺陷之一。因此，我們設計了一個用來估測物體姿態的二步直接法，此演算法無論目標物的紋理表徵足與不足皆可穩定且精準地計算出其姿態。基於此演算法，我們發展了一套能即時以六自由度追蹤被動目標筆件的系統，此系統的準確度甚至可達亞毫米的水準，使其足以在混合實境的環境中書寫及作畫。我們透過一系列在合成資料庫與真實資料庫上運行分析的實驗結果展現此系統在各種狀況下所成就的高效準確的姿態追蹤結果，其精準程度甚至可比由多台相機所組成的工業級動作捕捉系統。

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

Accurate 6DoF Object Pose Estimation and Tracking

Po-Chen Wu

Advisor: Shao-Yi Chien

Co-Advisor: Ming-Hsuan Yang

Graduate Institute of Electronics Engineering

National Taiwan University

Taipei, Taiwan

May 2018

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

Accurate 6DoF Object Pose Estimation and Tracking

By

Po-Chen Wu

Dissertation

Submitted in partial fulfillment of the requirement
for the degree of Doctor of Philosophy
in Electronics Engineering
at National Taiwan University
Taipei, Taiwan, R.O.C.

May 2018

Approved by：

Advised by：

Approved by Director：

doi:10.6342/NTU201800854

Abstract

This dissertation is concerned with the problem of determining the six degrees of freedom (6DoF) object poses from a calibrated camera. Given camera images which contain the target object, we wish to estimate the position and orientation of the target with respect to the camera accurately and robustly.

Although a variety of algorithms for this task have been proposed, it remains difficult to evaluate existing methods in the literature as oftentimes different sequences are used, and no benchmark datasets close to real-world scenarios are available. In this dissertation, we present a large-scale object pose tracking benchmark dataset consisting of RGB-D video sequences of 2D and 3D targets with ground-truth information. In particular, we perform the extensive quantitative evaluation of the state-of-the-art methods on this benchmark dataset and discuss the potential research directions in this field.

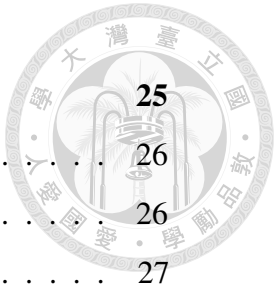
While advanced Perspective-n-Point algorithms perform well in pose estimation, the success hinges on whether feature points can be extracted and matched correctly on target objects with rich texture. Consequently, we develop a two-step robust direct method for 6DoF pose estimation that performs accurately on both textured and textureless planar target objects. Based on the proposed two-step direct approach, we present a system for real-time 6DoF tracking of a passive stylus that achieves submillimeter accuracy, which is suitable for writing or drawing in mixed reality applications. We demonstrate the system performance regarding speed and accuracy on a number of synthetic and real datasets, showing that it can be competitive with state-of-the-art multi-camera motion capture systems.

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

Contents

doi:10.6342/NTU201800854



doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

List of Figures

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

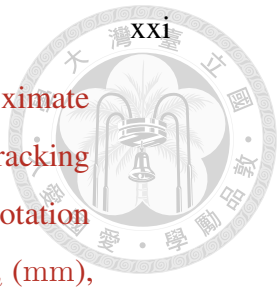
doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

List of Tables

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854



doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

Chapter 1

Introduction

Determining the pose of a target object from a calibrated camera is a classical problem in computer vision that finds numerous applications such as robotics, augmented reality (AR), and virtual reality (VR). For the case of a rigid body, its pose can be described by a six degrees of freedom (6DoF) transformation, consisting of three position parameters and three orientation parameters. While much progress has been made in the past decade, it remains a challenging task to develop a fast and accurate pose estimation algorithm.

In general, object pose estimation refers to computing the position and orientation of a target object given a single-view image. The target object is with prior knowledge (e.g., shape or texture) in most cases. On the contrary, object pose tracking indicates determining the poses of an object in an ordered sequence of camera frames. In this case, the object pose in a previous frame is already known, and thus one can exploit this information when computing the object pose in a current frame. Furthermore, it is also applicable to recover the object pose with multiple views, which may achieve superior pose estimation results.

In this dissertation, we primarily discuss how to compute the pose of a target object accurately and robustly with a single view. In particular, we propose a large-scale object pose tracking benchmark dataset consisting of RGB-D video sequences

doi:10.6342/NTU201800854

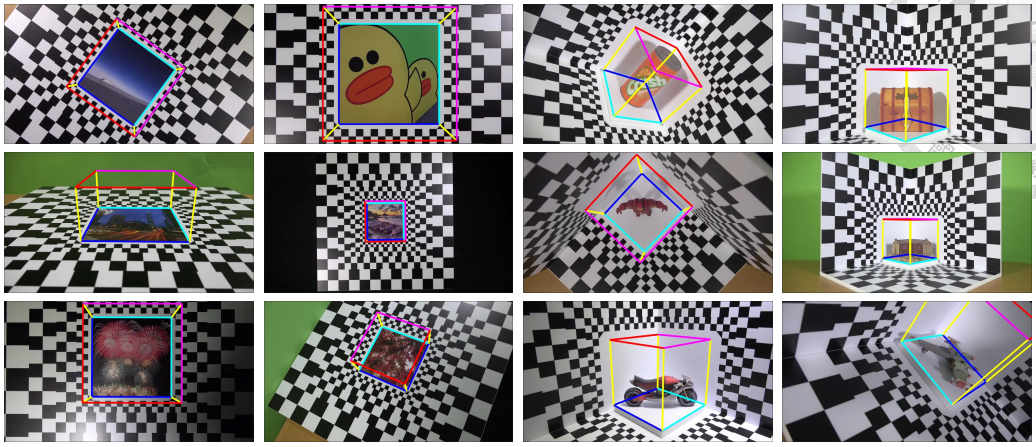


Figure 1.1: Images of 2D (left two columns) and 3D objects (right two columns) in our benchmark dataset with 6DoF pose ground-truth notation. The proposed benchmark dataset contains 690 color and depth videos of various textured and geometric objects with over 100,000 frames. The recorded sequences also include image distortions for performance evaluation in real-world scenarios.

of 2D and 3D targets with ground-truth information, as shown in Figure 1.1.[1] Furthermore, we perform the thorough quantitative evaluation of the state-of-the-art methods on this benchmark dataset. We observe that while advanced Perspective-n-Point (PnP) algorithms perform well in pose estimation, the success hinges on whether feature points can be extracted and matched correctly on target objects with rich texture. Consequently, we develop a two-step robust direct method for 6DoF pose estimation that performs accurately on both textured and textureless planar target objects. First, the pose of a planar target object with respect to a calibrated camera is approximately estimated by posing it as a template matching problem. Second, each object pose is refined and disambiguated using a dense alignment scheme, as illustrated in Figure 1.2. Based on the proposed two-step direct method, we present a system for real-time 6DoF tracking of a passive stylus that achieves submillimeter accuracy, which is suitable for writing or drawing in mixed reality applications, as demonstrated in Figure 1.3. We demonstrate

[1] All the images in the dissertation are used under Creative Commons license.

doi:10.6342/NTU201800854

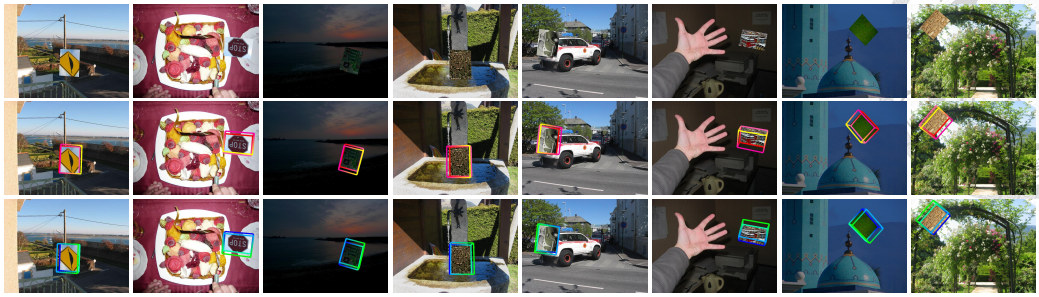
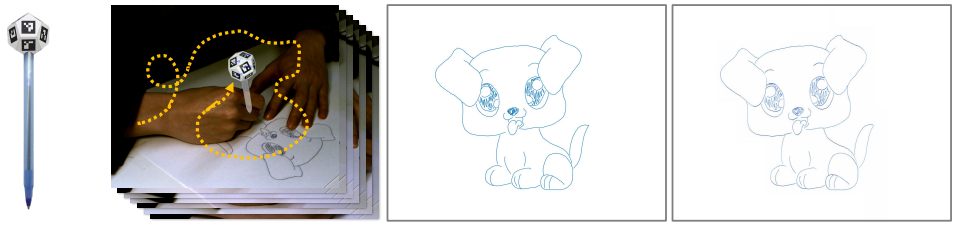


Figure 1.2: Direct pose estimation for planar targets. The pose ambiguity problem occurs when the objective function has several local minima for a given configuration, which is the primary cause of flipping estimated poses. First row: original images. Second row: images rendered with a box model according to the ambiguous pose obtained from proposed algorithm without refinement approach. Third row: pose estimation results from the proposed algorithm, which can disambiguate plausible poses effectively.



(a) DodecaPen (b) Monocular video (c) DodecaPen tracking (d) Ground-truth scan

Figure 1.3: Our proposed system can track the 6DoF pose of (a) a calibrated pen (the DodecaPen) from (b) a single camera with submillimeter accuracy. We show (c) a digital 2D drawing as the visualization of the tracking result, and compare with (d) a scan of the actual drawing.

the system performance regarding speed and accuracy on a number of synthetic and real datasets, showing that it can be competitive with state-of-the-art multi-camera motion capture systems. We also demonstrate several applications of the technology ranging from 2D and 3D drawing in VR to general object manipulation and board games.

doi:10.6342/NTU201800854

1.1 Object Pose Recovering

In computer vision and robotics, it is a typical task to identify some specific object in a camera image and estimate its position and orientation relative to the camera coordinate system. We regard this process as *object pose estimation* if the input data is only an image. When referring to *object pose tracking*, the input image data would be an ordered sequence of camera frames instead. This type of task can also be called *outside-in tracking*, where the target object is observed from outside by the camera system. And since the object model is known before computing its pose, it is also called *model-based tracking*.

An object is regarded as a *rigid body* if the distance between any two given points on it remains constant in time regardless of external forces exerted on it. For a rigid body, its position and orientation in space are defined by three components of translation and three components of rotation, which means that it has six degrees of freedom. In contrast, the freedom of movement of a *non-rigid body* (e.g., the human hand) may be more than six. In this dissertation, we focus on recovering the pose of an object which is a rigid body.

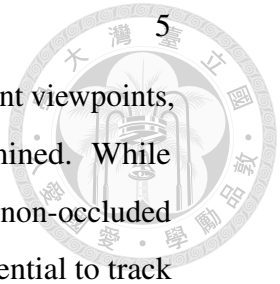
Existing algorithms for recovering the 6DoF pose of an object can be broadly categorized into three main approaches:

Direct approaches [5, 3]. These approaches address the problem by finding the best fit from numerous pre-determined candidates based on the holistic template or appearance matching. The corresponding pose of the best candidate is considered as the estimation result.

Feature-based approaches [6, 7, 8]. The core idea is to first establish a set of feature correspondences between the target object and projected camera frame [9, 10]. Outliers are then removed to obtain reliable feature pairs [11], and the final pose is computed with PnP algorithms [12, 13]. In contrast to direct methods, the performance of feature-based methods depends on whether both features can be extracted and matched well.

Learning-based approaches [14, 15, 16, 17]. These methods learn an abstract

doi:10.6342/NTU201800854



representation of an object from a set of images captured from different viewpoints, from which the pose of the target in a new input frame is determined. While feature-based and direct methods are more effective for textured and non-occluded objects respectively, learning-based approaches have shown the potential to track poses of objects with diverse textures under partial occlusion.

Real-time pose tracking can be accomplished by leveraging the information obtained from previous frames [18, 19, 20, 21]. In addition, the pose estimation task can be accelerated by exploiting a small search range within the camera viewpoint or reducing the number of pose candidates. To prevent pose jittering during the tracking process, which is indispensable especially in AR applications [22], further pose refinement should be performed. We refer the interested readers to [23] for more information on AR.

To evaluate existing pose estimation algorithms, many benchmark datasets have been proposed [5, 24, 14, 15, 25, 26]. However, there are two main issues that need to be addressed. First, while the datasets are mainly designed for single-frame based pose estimation, most images do not contain distortions (e.g., motion blur caused by different object or camera motions) that are crucial for performance evaluation for real-world scenarios. Second, the camera trajectories in most datasets are not carefully designed (i.e., freestyle motion), which do not allow detailed analysis for specific situations. Most importantly, it is of great interest for fields of computer vision to develop an extensive benchmark dataset for thorough performance evaluation of 6DoF pose tracking in real-world scenarios.

1.2 Camera Pose Recovering

In contrast to the 6DoF object pose, the 6DoF camera pose denotes the camera's position and orientation with respect to the object (or world) coordinate system. Though the object pose and the camera pose merely have the inverse transformation relationship to each other, the approaches of between object pose recovering,

doi:10.6342/NTU201800854

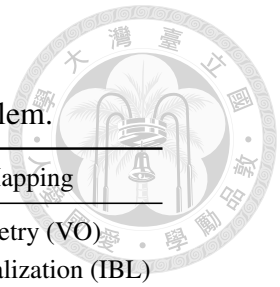


Table 1.1: Four categories of camera pose recovering problem.

	With Mapping	Without Mapping
Online	Simultaneous Localization and Mapping (SLAM)	Visual Odometry (VO)
Offline	Structure from Motion (SfM)	Image-based Localization (IBL)

and camera pose recovering are still quite different. In general, we exploit the information extracted from the entire camera image to compute the camera pose. But when determining the object pose, only a subregion of the image where the object locates contributes to the final pose estimation result.

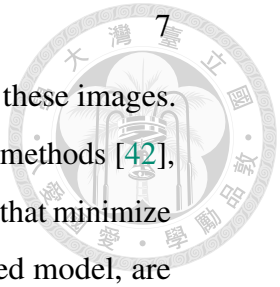
The task of camera pose recovering from visual data can be broadly subdivided into four categories, according to whether or not it is processed online, and if a map of the environment is built concurrently. We present the four types in Table 1.1 and provide the corresponding descriptions below.

Simultaneous Localization and Mapping (SLAM) [27, 28, 29]. SLAM techniques build a map of an unknown environment and localize the sensor in the map simultaneously in real-time. Among different sensor types, a camera is the most common one as it can provide rich information about the environment that allows robust and accurate localization and mapping. The SLAM approach which uses a camera as the primary sensor is denoted as *Visual SLAM*, and it has been a hot research topic in the last years [30, 31, 32, 33, 2].

Visual Odometry (VO) [34, 35, 36]. VO is the process of determining the location of a camera by analyzing the associated camera images in real-time. The main difference between VO and SLAM is that VO mainly addresses itself on *local consistency* and focuses on incrementally estimating the path of the camera pose after pose. Nevertheless, SLAM aims to achieve a *globally consistent* estimate of the camera trajectory and map by realizing that a previously mapped area has been re-visited and this information is used to reduce the drift in the estimates by *loop closure* techniques [37, 38].

Structure from Motion (SfM) [39, 40, 41]. SfM approaches recovers the map of an environment from a set of projective measurements, represented as a collection

doi:10.6342/NTU201800854



of 2D images, via estimation of the camera poses corresponding to these images. And the images can be either ordered or not. The *bundle adjustment* methods [42], which aim to determine the map and the camera poses simultaneously that minimize the discrepancy between image measurements and the reconstructed model, are usually used to solve the SfM problem [43, 44].

Image-Based Localization (IBL) [45, 46, 47]. IBL methods address the problem of finding the camera pose from which a camera image is taken. Traditionally, large-scale IBL has been treated as an image retrieval problem. After finding images in a database that are most similar to the query image, the location of the query image can be recovered with respect to them [48]. However, the localization accuracy obtained this way cannot be very satisfactory. Recently, IBL algorithms benefit from a 3D reconstruction of the scene produced by SfM or SLAM, which the query images can be accurately registered to [49, 46, 50].

1.3 Cameras

In vision-based applications, camera images are the necessary data when estimating either the object pose or the camera pose. Generally speaking, the cameras used in the pose recovering problem can be classified into three categories, namely monocular camera, stereo camera, and depth camera. And each pose recovering algorithm is designed according to a specific class of cameras.

Monocular Camera. It is the most common type of camera with one lens and a corresponding image sensor or film frame. The image sensor can be either monochrome (i.e., grayscale) one or color (i.e., RGB, which stands for red, green, and blue) one. Parameters of the lens and the images sensor of a camera can be estimated by performing *camera calibration* [51], which is also called *camera resectioning* or *geometric camera calibration*.[2] These parameters consist of camera *intrinsic parameters* and *distortion coefficients*. The intrinsic parameters

[2]The process is different from *photometric camera calibration* (i.e., color mapping).

doi:10.6342/NTU201800854

encompass focal length, image sensor format, and principal point. In addition, the distortion coefficients include radial distortion and tangential distortion coefficients. In most 3D computer vision tasks, camera calibration is the first step before applying further algorithms like pose estimation and tracking. Moreover, cameras equipped with global shutter are much more preferable than those provided with rolling shutter because the latter can cause undesirable effects such as wobble, skew, spatial aliasing, and temporal aliasing [52].

Stereo Camera. This type of camera has two or more lenses with a separate image sensor for each lens. This configuration allows the camera to simulate human binocular vision and therefore gives it the ability to capture depth information. Consequently, a stereo camera can also be regarded as a depth camera, which will be presented in the next paragraph. One well-known technique applied with the images captured by a stereo camera is *stereo matching* [53, 54], by which the depth information of a scene can be acquired. This is also called *stereoview* or *stereoscopic* in computer vision. Different from doing single camera calibration which is mentioned in the previous paragraph, we should perform stereo camera calibration to estimate not only the intrinsic parameters and distortion coefficients of each lens, but also the relative poses between lenses.

Depth Camera. A depth camera (or *range camera*) is a *range imaging* device that measures the distance from the camera to points in a scene, and as shown in Figure 1.4. The resulting image is called *range image* or *depth image*, which has pixel values corresponding to the measured distance. If the depth camera is accurately calibrated, then the pixel values can be given directly in physical units, such as millimeters applied by Microsoft Kinect V1 [55] and Microsoft Kinect V2 [56]. Depth cameras can operate according to numerous techniques, such as *stereo triangulation* [57], *structured light* [58] (e.g., Microsoft Kinect V1), and *time-of-flight* [59] (e.g., Microsoft Kinect V2). Furthermore, depth cameras using either of the latter two techniques should be equipped with an infrared (IR) projector and an IR camera.

doi:10.6342/NTU201800854

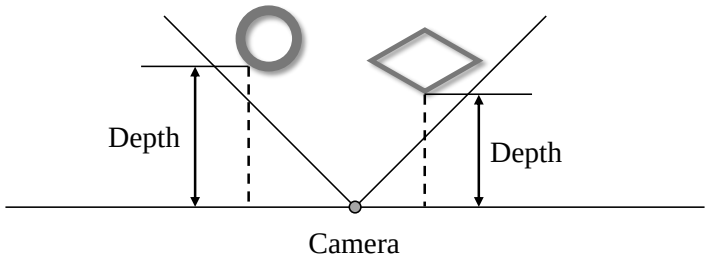


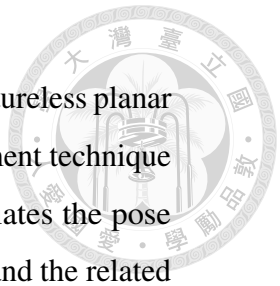
Figure 1.4: The depth data is measured from scene points to the camera plane on where the camera is (instead of from scene points to camera center).

1.4 Contributions

In this dissertation, we propose a benchmark dataset for 6DoF object pose tracking, which consists of 690 videos under seven varying conditions with five speeds. It is a large-scale dataset where images are acquired from a moving camera for performance evaluation of both 2D and 3D object pose tracking algorithms. The proposed dataset can be used in other computer vision tasks such as 3D feature tracking and matching as well. Furthermore, we extensively evaluate and analyze each pose tracking method employing more than 100,000 frames including both 2D and 3D objects. Since the advanced SLAM methods [2, 60] are able to track and relocalize camera pose in real time, we also evaluate these approaches by adapting them to object pose tracking scenarios. We present the extensive performance evaluation of the state-of-the-art methods using the proposed benchmark dataset and discuss the potential research directions in this field. The OPT datasets are available on our project website at media.ee.ntu.edu.tw/research/OPT.

From our observation, since the performance of feature-based methods hinges on whether or not point correspondences can be correctly established, these approaches are less effective when the target images contain less textured surfaces or motion blurs. Therefore, we propose an efficient direct pose estimation (DPE) algorithm for planar targets undergoing arbitrary 3D perspective transformations. The DPE algorithm performs favorably against the state-of-the-art feature-based ap-

doi:10.6342/NTU201800854



proaches in terms of robustness and accuracy on both textured and textureless planar target objects. In addition, we demonstrate the proposed pose refinement technique not only improves the accuracy of estimated results but also alleviates the pose ambiguity problem effectively. The source code of the DPE method and the related datasets are available on our project website at media.ee.ntu.edu.tw/research/DPE.

Based on the pose refinement technique proposed in the DPE method, we develop a 6DoF tracking system called DodecaPen that requires only a single off-the-shelf camera and a passive 3D-printed fiducial with several hand-glued binary square markers printed from a laser printer. We show that off-the-shelf fiducial tracking with markers is insufficient for achieving the accuracy necessary for digital 2D drawing. Instead, our system consists of the following components: (a) a 3D printed dodecahedron with hand-glued binary square markers mechanically designed for pose estimation, (b) a one-time calibration procedure for the (imprecise) model using bundle adjustment, (c) approximate pose estimation from fiducial corners, (d) inter-frame fiducial corner tracking, and (e) dense pose refinement by direct model-image alignment. We show that each step of the above system is essential to robust tracking and that the combined system allows us to achieve an absolute accuracy of 0.4 mm from a single camera, which is comparable to state-of-the-art professional motion capture (mocap) systems. We rigorously evaluate the performance of the proposed method when we degrade the camera (with shot noise, spatial blur, and reduced spatial resolution). Thorough evaluation results can be found on our project website at media.ee.ntu.edu.tw/research/DodecaPen. We conclude with demonstrations of this accurate and easy-to-setup 6DoF status tracking system for the application of drawing in 2D and 3D as well as object manipulation in a virtual reality (VR) environment.

1.5 Publications

The core of the dissertation relies on the following peer-reviewed publications:

doi:10.6342/NTU201800854

- Po-Chen Wu, Yueh-Ying Lee, Hung-Yu Tseng, Hsuan-I Ho, Ming-Hsuan Yang, and Shao-Yi Chien, "**A Benchmark Dataset for 6DoF Object Pose Tracking**." In *Proceedings of the IEEE International Symposium on Mixed and Augmented Reality (ISMAR Adjunct)*, 2017. [61]

- Po-Chen Wu, Hung-Yu Tseng, Ming-Hsuan Yang, and Shao-Yi Chien, "**Direct Pose Estimation for Planar Objects**." In *Computer Vision and Image Understanding*, 2018. [62]

- Po-Chen Wu, Robert Wang, Kenrick Kin, Christopher Twigg, Shangchen Han, Ming-Hsuan Yang, and Shao-Yi Chien, "**DodecaPen: Accurate 6DoF Tracking of a Passive Stylus**." In *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST)*, 2017. ***(ACM UIST Honorable Mention Award)*** [63]

1.6 Dissertation Organization

We formulate the pose recovering problem in the subsequent chapter. In particular, we introduce the vectorial parameterization of rotation and different evaluation metrics used for pose recovering problem. In Chapter 3, we give an overview of the related work which is relevant for the remainder of the dissertation. Furthermore, we discuss methods of object pose recovering, and existing benchmark datasets proposed in the literature. Afterwards, we present a large-scale object pose tracking benchmark dataset of RGB-D video sequences for both 2D and 3D objects, as well as extensive quantitative evaluation results of the state-of-the-art methods on this dataset in Chapter 4. In Chapter 5, we propose a two-step direct pose estimation algorithm for planar objects. Then, in Chapter 6, we present a system for real-time 6DoF tracking of a passive stylus that achieves submillimeter accuracy, which is suitable for writing or drawing in mixed reality applications. Finally, we conclude this dissertation with discussions on future work in Chapter 7.

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

Chapter 2

Problem Formulation

Given a target object $\mathcal{O}_t$, represented by either a plane or a dense surface model (i.e., triangle mesh), and an observed camera image $\mathcal{I}_c$, the task of object pose recovering is to determine the object pose of $\mathcal{O}_t$ in 6DoF parameterization based on the orientation and position of the object with respect to a calibrated camera. With a set of reference points $\mathbf{x}_i = [x_i, y_i, z_i]^\top, i = 1, \ldots, n, n \geq 3$ in the coordinate system of $\mathcal{O}_t$, and a set of camera-image coordinates $\mathbf{u}_i = [u_i, v_i]^\top$ in $\mathcal{I}_c$ depicted by Figure 2.1, the transformation between them can be formulated as:

$$\begin{bmatrix} hu_i \\ hv_i \\ h \end{bmatrix} = \mathbf{K} \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix}, \tag{2.1}$$

where

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \ \mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \in SO(3), \ \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \in \mathbb{R}^3, \tag{2.2}$$

are the intrinsic matrix of the camera, rotation matrix, and translation vector, respectively. In (2.1), h is the scale factor representing the depth value in the camera coordinate system. In (2.2), (f_x, f_y) and (x_0, y_0) are the focal length and the principal point of the camera, respectively.

doi:10.6342/NTU201800854

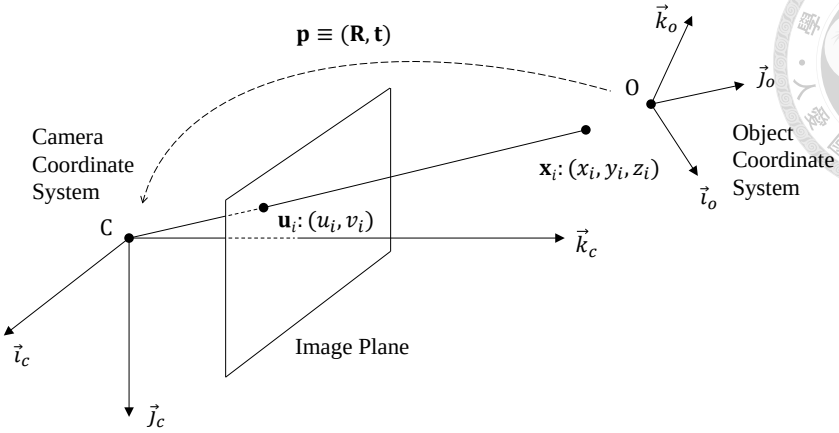


Figure 2.1: The perspective projection model. $(O, \vec{i}_o, \vec{j}_o, \vec{k}_o)$ is the object coordinate system, $(C, \vec{i}_c, \vec{j}_c, \vec{k}_c)$ is the camera coordinate system, $\mathbf{x}_i$ is a 3D point, and $\mathbf{u}_i$ is its projection onto the image plane.

Given the observed camera-image points $\hat{\mathbf{u}}_i = [\hat{u}_i, \hat{v}_i]^\top$, a pose estimation algorithm needs to determine values a for pose $\mathbf{p} \equiv (\mathbf{R}, \mathbf{t})$ that minimize an appropriate error function. The rotation of the pose $\mathbf{p}$ can be parameterized in numerous ways [64], and will be further discussed in Section 2.1.

There are two types of error functions commonly used for pose estimation. The first one is called *reprojection error* and is broadly used in the PnP algorithms:

$$E_r(\mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n}\left((\hat{u}_i - u_i)^2 + (\hat{v}_i - v_i)^2\right). \tag{2.3}$$

The second type of error function is based on *appearance distance* and is primarily used in direct methods:

$$E_{a_1}(\mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n}|\mathcal{I}_c(\mathbf{u}_i) - \mathcal{O}_t(\mathbf{x}_i)|, \tag{2.4}$$

or

$$E_{a_2}(\mathbf{p}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{I}_c(\mathbf{u}_i) - \mathcal{O}_t(\mathbf{x}_i)\right)^2, \tag{2.5}$$

where $\mathcal{I}_c(\mathbf{u}_i)$ is the image pixel value of $\mathcal{I}_c$ at $\mathbf{u}_i$, and $\mathcal{O}_t(\mathbf{x}_i)$ is the texture pixel value of $\mathcal{O}_t$ at $\mathbf{x}_i$. In most cases, the pixel values are normalized in the range $[0, 1]$. The error functions in (2.4) and (2.5) are the normalized Sum-of-Absolute-Differences (SAD) and Sum-of-Squared-Difference (SSD) errors, respectively.

doi:10.6342/NTU201800854

2.1 Parameterization of Rotation

The general form of a rotation in $\mathbb{R}^3$ is a 3×3 orthogonal matrix with determinant 1. However, the matrix representation with nine elements seems redundant since it only has a maximum of three degrees of freedom (3DoF). Actually, there are many ways to parameterize 3DoF rotations with fewer parameters, such as *Euler angles*, *axis-angle representation*, and *quaternions*.[1]

2.1.1 Euler Angles

The Euler angles are three angles describing a rotation in $\mathbb{R}^3$. Any rotation in $\mathbb{R}^3$ can be achieved by composing three *elemental rotations* (i.e., rotations about the axes of a coordinate system), and the Euler angles can be defined by three of these elemental rotations. Rotations about the three principle axes x-axis, y-axis, and z-axis by angles θ_x, θ_y, and θ_z are defined as follows:

$$\begin{gathered}\mathbf{R}_x(\theta_x)=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos\theta_x & -\sin\theta_x\\ 0 & \sin\theta_x & \cos\theta_x\end{bmatrix},\mathbf{R}_y(\theta_y)=\begin{bmatrix}\cos\theta_y & 0 & \sin\theta_y\\ 0 & 1 & 0\\ -\sin\theta_y & 0 & \cos\theta_y\end{bmatrix},\\ \mathbf{R}_z(\theta_z)=\begin{bmatrix}\cos\theta_z & -\sin\theta_z & 0\\ \sin\theta_z & \cos\theta_z & 0\\ 0 & 0 & 1\end{bmatrix}.\end{gathered} \tag{2.6}$$

Since matrix multiplication does not commute, the order of the elemental rotations will affect the result. For example, one might want to factor a rotation as $\mathbf{R}=\mathbf{R}_x(\theta_x)\mathbf{R}_y(\theta_y)\mathbf{R}_z(\theta_z)$, whose ordering is xyz. The rotation $\mathbf{R}$ first rotates about the z-axis, then the y-axis, and finally x-axis. Such a sequence of rotations can be

[1] There are still other ways to parameterize rotations, such as *Rodrigues parameters*, *Gibbs representation*, and *Cayley–Klein parameters*.

doi:10.6342/NTU201800854

represented as the matrix product:

$$\mathbf{R}(\theta_x, \theta_y, \theta_z) = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} = \begin{bmatrix} c_y c_z & -c_y s_z & s_y \\ c_z s_x s_y + c_x s_z & c_x c_z - s_x s_y s_z & -c_y s_x \\ -c_x c_z s_y + s_x s_z & c_z s_x + c_x s_y s_z & c_x c_y \end{bmatrix}, \tag{2.7}$$

where we use the notation $c_a = \cos(\theta_a)$ and $s_a = \sin(\theta_a)$ for $a = x, y, z$.

Starting with R_{13}, we find $R_{13} = s_y$, so $\theta_y = \text{asin}(R_{13})$. Then there are three cases to consider.

Case 1: If $\theta_y \in (-\pi/2, \pi/2)$, then $c_y \neq 0$. In this condition, $\theta_x = \text{atan2}(-R_{23}, R_{33})$ as $(-R_{23}, R_{33}) = (c_y s_x, c_y c_x)$, where atan2 is the *two-argument arctangent*. In addition, $\theta_z = \text{atan2}(-R_{12}, R_{11})$ as $(-R_{12}, R_{11}) = (c_y s_z, c_y c_z)$. In summary:

$$\theta_y = \text{asin}(R_{13}),\ \theta_x = \text{atan2}(-R_{23}, R_{33}),\ \theta_z = \text{atan2}(-R_{12}, R_{11}). \tag{2.8}$$

Case 2: If $\theta_y = \pi/2$, then $s_y = 1$ and $c_y = 0$. In this condition,

$$\begin{bmatrix} R_{21} & R_{22} \\ R_{31} & R_{32} \end{bmatrix} = \begin{bmatrix} c_z s_x + c_x s_z & c_x c_z - s_x s_z \\ -c_x c_z + s_x s_z & c_z s_x + c_x s_z \end{bmatrix} = \begin{bmatrix} \sin(\theta_z + \theta_x) & \cos(\theta_z + \theta_x) \\ -\cos(\theta_z + \theta_x) & \sin(\theta_z + \theta_x) \end{bmatrix}. \tag{2.9}$$

Therefore, $\theta_z + \theta_x = \text{atan2}(R_{21}, R_{22})$. Since there is only one degree of freedom, the factorization is not unique. This ambiguity is known as *gimbal lock* in applications. In summary:

$$\theta_y = \pi/2,\ \theta_z + \theta_x = \text{atan2}(R_{21}, R_{22}). \tag{2.10}$$

Case 3: If $\theta_y = -\pi/2$, then $-s_y = 1$ and $c_y = 0$. In this condition,

$$\begin{bmatrix} R_{21} & R_{22} \\ R_{31} & R_{32} \end{bmatrix} = \begin{bmatrix} -c_z s_x + c_x s_z & c_x c_z + s_x s_z \\ c_x c_z + s_x s_z & c_z s_x - c_x s_z \end{bmatrix} = \begin{bmatrix} \sin(\theta_z - \theta_x) & \cos(\theta_z - \theta_x) \\ \cos(\theta_z - \theta_x) & -\sin(\theta_z - \theta_x) \end{bmatrix}. \tag{2.11}$$

Therefore, $\theta_z - \theta_x = \text{atan2}(R_{21}, R_{22})$. It has lost one of the degrees of freedom (i.e., gimbal lock). In summary:

$$\theta_y = -\pi/2,\ \theta_z - \theta_x = \text{atan2}(R_{21}, R_{22}). \tag{2.12}$$

doi:10.6342/NTU201800854

We note that xyz is not the only ordering. In fact, there exist twelve possible orderings divided into two groups:

- Tait–Bryan angles $(xyz, yzx, zxy, xzy, zyx, yxz)$,
- Proper Euler angles $(zxz, xyx, yzy, zyz, xzx, yxy)$.

The methods of factoring a rotation according to different orderings are similar. We refer the interested readers to [65] for more detailed information.

If an angle θ is close to zero, i.e., $\theta \approx 0$, then we can use the approximations $\sin\theta \approx \theta$ and $\cos\theta \approx 1$. Therefore, when $\theta_x, \theta_y, \theta_z \approx 0$:

$$\begin{aligned}\mathbf{R}(\theta_x, \theta_y, \theta_z) &\approx \begin{bmatrix} 1 & -\theta_z & \theta_y \\ \theta_x\theta_y + \theta_z & 1 - \theta_x\theta_y\theta_z & -\theta_x \\ -\theta_y + \theta_x\theta_z & \theta_x + \theta_y\theta_z & 1 \end{bmatrix} \\ &\approx \begin{bmatrix} 1 & -\theta_z & \theta_y \\ \theta_z & 1 & -\theta_x \\ -\theta_y & \theta_x & 1 \end{bmatrix} = \hat{\mathbf{R}}(\theta_x, \theta_y, \theta_z).\end{aligned} \tag{2.13}$$

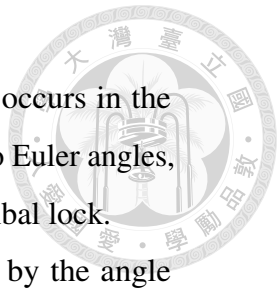
The small angle approximation $\hat{\mathbf{R}}$ can be used in many applications requiring linear equations. However, this approximation is no longer a rotation since $\hat{\mathbf{R}}^{-1}\mathbf{R} \neq 1$.

2.1.2 Axis–Angle Representation

The axis–angle representation of a rotation parameterizes a rotation $\mathbb{R}^3$ by two quantities: a 3D unit vector $\mathbf{a}$ which describes the direction of an axis of rotation, and an angle θ which indicates the magnitude of the rotation about the axis. This axis is also called *Euler axis*, which comes from *Euler's rotation theorem* stating that any rotation or sequence of rotations of a rigid body in a 3D space is equivalent to a pure rotation about a single rotation axis. The angle θ scalar multiplied by the unit vector $\mathbf{a}$ is the axis-angle vector:

$$\mathbf{r} = \theta\mathbf{a}, \tag{2.14}$$

doi:10.6342/NTU201800854



which is also called *rotation vector* or *Euler vector*. The rotation occurs in the sense prescribed by the *right-hand rule* (anticlockwise). Compared to Euler angles, an Euler vector is simpler to compose and avoid the problem of gimbal lock.

Let $\mathbf{v}$ be a vector in $\mathbb{R}^3$. After begin rotated about the axis $\mathbf{a}$ by the angle θ, the rotated vector $\hat{\mathbf{v}}$ can be computed according to the *Rodrigues' rotation formula* [66]:

$$\hat{\mathbf{v}} = \mathbf{v}\cos\theta + (\mathbf{a}\times\mathbf{v})\sin\theta + \mathbf{a}(\mathbf{a}\cdot\mathbf{v})(1-\cos\theta). \tag{2.15}$$

If we represent $\hat{\mathbf{v}}$ as a matrix product of a rotation matrix $\mathbf{R}$ and the original vector $\mathbf{v}$, then $\mathbf{R}$ can be expressed as follows [66]:

$$\mathbf{R}(\theta,\mathbf{a}) = \mathbf{I} + \sin\theta\,[\mathbf{a}]_\times + (1-\cos\theta)\,[\mathbf{a}]_\times^2, \tag{2.16}$$

where $\mathbf{I}$ is the 3×3 identity matrix, and $[\mathbf{a}]_\times$ denotes the *cross-product matrix* (which is also a *skew-symmetric matrix*) for the vector $\mathbf{a} = [a_x, a_y, a_z]^\top$:

$$[\mathbf{a}]_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}. \tag{2.17}$$

The matrix equation $[\mathbf{a}]_\times\,\mathbf{v} = \mathbf{a}\times\mathbf{v}$ holds for any vector $\mathbf{v}$. In (2.16), $[\mathbf{a}]_\times^2$ stands for the matrix product $[\mathbf{a}]_\times \cdot [\mathbf{a}]_\times$. In addition, the rotation matrix can also be expressed in terms of the rotation vector $\mathbf{r} = [r_x, r_y, r_z]^\top$:

$$\mathbf{R}(\mathbf{r}) = \mathbf{I} + \left(\frac{\sin\theta}{\theta}\right)[\mathbf{r}]_\times + \left(\frac{1-\cos\theta}{\theta^2}\right)[\mathbf{r}]_\times^2, \tag{2.18}$$

where $\theta = \|\mathbf{r}\|$ is the Euclidean norm of $\mathbf{r}$.

To retrieve the axis–angle representation of a rotation matrix $\mathbf{R}$, we first computed the angle of rotation from the trace of the rotation matrix $\mathrm{Tr}(\mathbf{R})$ [67]:

$$\theta = \mathrm{acos}\left(\frac{\mathrm{Tr}(\mathbf{R})-1}{2}\right). \tag{2.19}$$

Then the rotation axis $\mathbf{a}$ can be calculated as follows [67]:

$$\mathbf{a} = \frac{1}{2\sin\theta}\begin{bmatrix} R_{32}-R_{23} \\ R_{13}-R_{31} \\ R_{21}-R_{12} \end{bmatrix}. \tag{2.20}$$

doi:10.6342/NTU201800854

Another way to transform between rotation vectors and rotation matrices is applying the *exponential map* and its inverse, the *logarithm map*. These two techniques are presented in the theory of *Lie groups*, and we refer the interested readers to [68] for more detailed information.

If the rotation angle is very small, i.e., $\theta \approx 0$, then we can use the approximations $\sin\theta \approx \theta$ and $\cos\theta \approx 1 - (\frac{\theta^2}{2})$. Therefore, from (2.18):

$$\mathbf{R}(\mathbf{r}) \approx \mathbf{I} + [\mathbf{r}]_\times + \frac{1}{2}[\mathbf{r}]_\times^2 \approx \mathbf{I} + [\mathbf{r}]_\times = \begin{bmatrix} 1 & -r_z & r_y \\ r_z & 1 & -r_x \\ -r_y & r_x & 1 \end{bmatrix} = \hat{\mathbf{R}}(\mathbf{r}), \quad (2.21)$$

which has the similar form to (2.13).

2.1.3 Quaternions

A quaternion is a 4-tuple (i.e., a vector with four components) consisting of a complex number with three different *imaginary* parts, which gives a simple way to encode the axis–angle representation.

A quaternion $\mathbf{q}$ is generally represented in the form:

$$\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}, \quad (2.22)$$

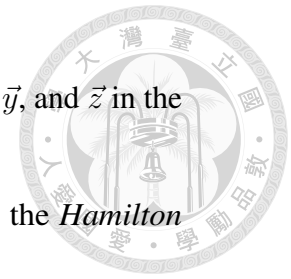
where q_0, q_1, q_2, and q_3 are real numbers, and $\mathbf{i}$, $\mathbf{j}$, and $\mathbf{k}$ are fundamental *quaternion units*. A quaternion unit is a symbol that has no other value than itself. By analogy with complex numbers, the term $q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ is also called the *imaginary part* (or *vector part*) of $\mathbf{q}$, and q_0 is the *real part* (or *scalar part*) of $\mathbf{q}$. The basic rules for multiplication are

$$\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{ijk} = -1, \quad (2.23)$$

which behave similarly to the square of the imaginary unit i of complex numbers. From this follows:

$$\mathbf{ij} = -\mathbf{ji} = \mathbf{k}, \mathbf{jk} = -\mathbf{kj} = \mathbf{i}, \mathbf{ki} = -\mathbf{ik} = \mathbf{j}, \quad (2.24)$$

doi:10.6342/NTU201800854



which behave similarly to pairwise cross products of unit vectors $\vec{x}$, $\vec{y}$, and $\vec{z}$ in the directions of orthogonal coordinate system axes.

For two quaternions $\mathbf{q}$ and $\mathbf{q}'$, their product $\mathbf{q}'' = \mathbf{q}\mathbf{q}'$, called the *Hamilton product*, is determined by the products of the quaternion units:

$$\mathbf{q}'' \equiv \begin{bmatrix} q_0'' \\ q_1'' \\ q_2'' \\ q_3'' \end{bmatrix} = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ q_1 & q_0 & -q_3 & q_2 \\ q_2 & q_3 & q_0 & -q_1 \\ q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \begin{bmatrix} q_0' \\ q_1' \\ q_2' \\ q_3' \end{bmatrix}. \tag{2.25}$$

It should be noted that the multiplication of quaternions is associative and distributes over vector addition, but it is not commutative. We list other quaternion properties as follows:

- Norm: $\|\mathbf{q}\|^2 = q_0^2 + q_1^2 + q_2^2 + q_3^2$,
- Conjugate quaternion: $\bar{\mathbf{q}} = q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k}$,
- Inverse quaternion: $\mathbf{q}^{-1} = \frac{\bar{\mathbf{q}}}{|\mathbf{q}|^2}$,
- Unit quaternion: $\|\mathbf{q}\| = 1$,
- Inverse of unit quaternion: $\mathbf{q}^{-1} = \bar{\mathbf{q}}$.

Rotation through an angle of θ around the unit rotation axis $\mathbf{a} = [a_x, a_y, a_z]^\top$ can be represented by a *unit quaternion* (or *rotation quaternion*):

$$\mathbf{q} = e^{\frac{\theta}{2}(a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k})} = \cos\frac{\theta}{2} + (a_x\mathbf{i} + a_y\mathbf{j} + a_z\mathbf{k})\sin\frac{\theta}{2}, \tag{2.26}$$

and the detailed derivation can be found in [69]. In this case, q_0, q_1, q_2, and q_3 equal to $\cos\frac{\theta}{2}$, $a_x\sin\frac{\theta}{2}$, $a_y\sin\frac{\theta}{2}$, and $a_z\sin\frac{\theta}{2}$, respectively. The desired rotation can be applied to a 3D vector $\mathbf{p}$ by evaluating the *conjugation* of $\mathbf{p}$ by $\mathbf{q}$ using the Hamilton product:

$$\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}, \tag{2.27}$$

doi:10.6342/NTU201800854

where $\mathbf{p}$ and $\mathbf{p}'$ are represented by quaternions with zero real parts:

$$\mathbf{p} = p_x\mathbf{i} + p_y\mathbf{j} + p_z\mathbf{k}, \quad \mathbf{p}' = p'_x\mathbf{i} + p'_y\mathbf{j} + p'_z\mathbf{k}, \tag{2.28}$$

and $\mathbf{p}'$ is the new vector after the rotation. It follows that conjugation by the product of two quaternions is the composition of conjugations by these quaternions. For instance, if $\mathbf{q}_1$ and $\mathbf{q}_2$ are unit quaternions, then the rotation (i.e., conjugation) by $\mathbf{q}_1\mathbf{q}_2$ is:

$$\mathbf{q}_1\mathbf{q}_2\mathbf{p}(\mathbf{q}_1\mathbf{q}_2)^{-1} = \mathbf{q}_1\mathbf{q}_2\mathbf{p}\mathbf{q}_2^{-1}\mathbf{q}_1^{-1} = \mathbf{q}_1(\mathbf{q}_2\mathbf{p}\mathbf{q}_2^{-1})\mathbf{q}_1^{-1}, \tag{2.29}$$

which is the same as rotating (i.e., conjugating) $\mathbf{p}$ by $\mathbf{q}_2$ and then by $\mathbf{q}_1$. The real part of the result is necessarily zero. In this case, the two rotation quaternions can also be first combined into one equivalent quaternion by the relation $\mathbf{q}' = \mathbf{q}_1\mathbf{q}_2$, where $\mathbf{q}'$ corresponds to the rotation $\mathbf{q}_2$ followed by the rotation $\mathbf{q}_1$.

A quaternion rotation $\mathbf{p}' = \mathbf{q}\mathbf{p}\mathbf{q}^{-1}$ can also be algebraically manipulated into a matrix rotation $\mathbf{p}' = \mathbf{R}\mathbf{p}$, in which $\mathbf{R}$ is the rotation matrix:

$$\mathbf{R}(q_0, q_1, q_2, q_3) = \begin{bmatrix} q_0^2 + q_1^2 - q_2^2 - q_3^2 & 2q_1q_2 - 2q_0q_3 & 2q_1q_3 + 2q_0q_2 \\ 2q_1q_2 + 2q_0q_3 & q_0^2 - q_1^2 + q_2^2 - q_3^2 & 2q_2q_3 - 2q_0q_1 \\ 2q_1q_3 - 2q_0q_2 & 2q_2q_3 + 2q_0q_1 & q_0^2 - q_1^2 - q_2^2 + q_3^2 \end{bmatrix}. \tag{2.30}$$

When converting a rotation matrix to a quaternion, several straightforward methods tend to be unstable when the trace of the rotation matrix is very close to zero. We refer the interested readers to [70] for a more stable method of converting a rotation matrix to a quaternion.

The rotation axis $\mathbf{a}$ and angle θ corresponding to a quaternion $\mathbf{q} = q_0 + q_1\mathbf{i} + q_2\mathbf{j} + q_3\mathbf{k}$ can be extracted as follows:

$$(a_x, a_y, a_z) = \frac{(q_1, q_2, q_3)}{\sqrt{1 - q_0^2}}, \quad \theta = 2\,\mathrm{acos}(q_0), \tag{2.31}$$

where $\mathrm{acos}(\cdot)$ is the *arccosine* function. Please note that when the quaternion approaches a real quaternion (i.e., a quaternion with zero imaginary part), the axis is not well-defined due to *degeneracy*.

doi:10.6342/NTU201800854

2.2 Evaluation Metrics

In order to evaluate the performance of between different object pose recovering methods properly, we have to use an appropriate metric first. In fact, there have been many metrics defined for evaluating object pose estimation and tracking methods, which will be introduced next. In the following paragraphs, we use $\mathbf{p} \equiv (\mathbf{R}, \mathbf{t})$ and $\tilde{\mathbf{p}} \equiv (\tilde{\mathbf{R}}, \tilde{\mathbf{t}})$ as the estimated pose and the ground-truth pose, respectively,

2.2.1 Rotation & Translation Errors

The rotation error is defined as the angle of the relative rotation between the estimated and ground-truth rotations. While the rotations are represented as matrices, we can get the estimated rotation error according to (2.19):

$$E_r(\mathbf{R}) = \operatorname{acos}\left(\frac{\operatorname{Tr}(\Delta\mathbf{R}) - 1}{2}\right), \tag{2.32}$$

where $\Delta\mathbf{R} = \mathbf{R}^{-1} \cdot \tilde{\mathbf{R}} = \mathbf{R}^{\top} \cdot \tilde{\mathbf{R}}$ is the relative rotation matrix. In addition, if the rotations are represented as unit quaternions (i.e., $\mathbf{q}$ and $\tilde{\mathbf{q}}$), then we can efficiently compute the rotation error as follows:

$$E_r(\mathbf{q}) = 2 \operatorname{acos}\left(\operatorname{dot}(\mathbf{q}, \tilde{\mathbf{q}})\right), \tag{2.33}$$

where $\operatorname{dot}(\mathbf{q}, \tilde{\mathbf{q}}) = q_0\tilde{q}_0 + q_1\tilde{q}_1 + q_2\tilde{q}_2 + q_3\tilde{q}_3$ is the dot product of $\mathbf{q}$ and $\tilde{\mathbf{q}}$. The reason behind (2.33) is that the relative rotation is represented as follows:

$$\Delta\mathbf{q} = \mathbf{q}^{-1}\tilde{\mathbf{q}} = \bar{\mathbf{q}}\tilde{\mathbf{q}} = (q_0 - q_1\mathbf{i} - q_2\mathbf{j} - q_3\mathbf{k})(\tilde{q}_0 + \tilde{q}_1\mathbf{i} + \tilde{q}_2\mathbf{j} + \tilde{q}_3\mathbf{k}). \tag{2.34}$$

And the rotation angle of $\Delta\mathbf{q}$ is $2\operatorname{acos}(\delta q_0)$ according to (2.31), in which δq_0 is the real part of $\Delta\mathbf{q}$ and is equivalent to $\operatorname{dot}(\mathbf{q}, \tilde{\mathbf{q}})$.

The translation error can be defined as either the *absolute difference* between the estimated and ground-truth translations directly in physical units (e.g., meters):

$$E_t(\mathbf{t}) = \|\mathbf{t} - \tilde{\mathbf{t}}\|, \tag{2.35}$$

doi:10.6342/NTU201800854

or the *relative difference* in percentage terms:

$$E_t(\mathbf{t}) = \frac{\|\mathbf{t} - \tilde{\mathbf{t}}\|}{\|\tilde{\mathbf{t}}\|} \times 100\%. \tag{2.36}$$

Success Rate (SR). We define a pose to be successfully estimated if its rotation and translation errors are both under predefined thresholds. For example, Shotton *et al*. [71] and Tseng *et al*. [3] use the thresholds of 5°&5cm and 20°&10%, respectively. The success rate according to the metric of *rotation & translation errors* is defined as the percentage of the successfully estimated poses. Sometimes we also use measures of the average rotation and translation errors, and they computed only for successfully estimated poses.

2.2.2 3D Distance

This metric is used to compute the averaged 3D distance between points transformed using the estimated pose and the ground-truth pose [5]:

$$E_{3D} = \frac{1}{m} \sum_{\mathbf{x} \in \mathcal{M}} \|\mathbf{R}\mathbf{x} + \mathbf{t} - (\tilde{\mathbf{R}}\mathbf{x} + \tilde{\mathbf{t}})\|, \tag{2.37}$$

where $\mathbf{x}$ is a 3D point of the target object model, m is the number of points on the model, and $\mathcal{M}$ is the set of all 3D points of this model.

Success Rate (SR). For the metric of *3D Distance*, a pose is considered to be successfully estimated if the 3D distance error E_{3D} is less than the product of kd, where d is the diameter (i.e., the longest distance between vertices) of $\mathcal{M}$ and k is a pre-defined threshold. In [5], Hinterstoisser *et al*. set the value of k to be 0.1, which means the computed average distance is within 10% of the model diameter.

2.2.3 2D Projection

Sometimes the previous measures are not very well suited when applying visual effects to 2D images (e.g., in AR applications). For instance, the translation accuracy in z-axis is less critical for the visual impression than the accuracy in

doi:10.6342/NTU201800854

x-axis and y-axis. This metric focuses on the matching of pose estimation on 2D images [16]:

$$E_{2D} = \frac{1}{m} \sum_{\mathbf{x} \in \mathcal{M}} \|\mathbf{K}(\mathbf{R}\mathbf{x} + \mathbf{t}) - \mathbf{K}(\tilde{\mathbf{R}}\mathbf{x} + \tilde{\mathbf{t}})\|, \tag{2.38}$$

where $\mathbf{x}$ is a 3D point of the target object model, m is the number of points on the model, $\mathcal{M}$ is the set of all 3D points of this model, and $\mathbf{K}$ is the intrinsic matrix of the camera as defined in (2.1).

Success Rate (SR). We say that a pose is correctly estimated if the 2D projection error E_{2D} is less than a pre-defined threshold. Brachmann *et al.* [16] use 5px (i.e., 5 pixels) as the threshold for the metric of *2D projection*.

doi:10.6342/NTU201800854

Chapter 3

Related Work

In this chapter, we briefly review methods related to object pose estimation and tracking in the literature. Among these methods, the feature-based approaches are the most common and effective ones if the target objects are full of texture. Feature-based approaches typically run a two-stage pipeline: a) feature detection and matching, b) geometric verification of the matched features using PnP algorithms. Therefore, we first introduce these two techniques in the following sections. We use PnP algorithms to estimate the pose of an object pose from a calibrated camera given a set of n 3D points of the object and their corresponding 2D projections in the image. In contrast, the Kabsch Algorithm is a method for computing the relative rigid transformation between two paired sets of points in the same coordinate space. Recently, image or point cloud alignment methods (e.g., the Lucas-Kanade and Iterative Closest Point methods) based on optimization problems have regained public attention for their accuracy and robustness, and they can be utilized for not only pose tracking but pose refinement. To find a local minimum of an objection function effectively in an optimization problem, the line search strategy is frequently applied next to obtaining a descent direction along which the objective function will be reduced. After introducing these elemental methods, we finally present existing object pose estimation and tracking approaches as well as benchmark datasets used for evaluating pose recovering algorithms.

doi:10.6342/NTU201800854

3.1 Feature Detection and Matching

Establishing feature correspondences across different images typically involves three distinct steps. First, features with rich visual information are detected in both images. The SIFT detector [9] leverages difference of Gaussians (DoG) to accelerate the detection process in different scales, while the SURF [72] detector uses a Haar wavelet approximation of the determinant of the Hessian matrix. In addition, the KAZE detector [73], followed by the Accelerated-KAZE (AKAZE) detector [74], uses non-linear diffusion filtering techniques [75] to build the scale space instead of Gaussian blurring to preserve object boundaries. As these detectors are still computationally expensive, several methods including FAST [76] and AGAST [77] have been developed for improvement of execution speed.

Second, a feature representation based on a local patch centered at a detected feature is constructed. Although the SIFT descriptor [9] and the SURF descriptor [72] have been shown to perform robustly in numerous tasks, the incurred computational cost is high as the feature dimensionality is high. Subsequently, binary descriptors, such as BRIEF [78], BRISK [79], ORB [80], and FREAK [81], are designed for improvement of execution speed.

Third, a feature point is associated with another in the other image. While a method is expected to detect plenty of distinct features accurately in one image and match most of them across different views of the same object, some correspondences are incorrectly determined in practice, and most PnP methods do not handle these outliers well. Outliers are typically rejected at a preliminary stage using projective transformation models or P3P algorithms [82, 83, 84] in combination with RANSAC-based schemes [11, 85, 86].

3.2 PnP Algorithms

Given n 3D reference points in the object-space coordinate system and their corresponding 2D projections, the PnP problem aims to retrieve the rigid transformation

doi:10.6342/NTU201800854

of the target object with respect to the camera. In the past, iterative PnP algorithms, e.g., LM [87] and RPP [88], determine the orientation and position of an object by minimizing an appropriate objective function iteratively. These methods perform well when reliable initial estimates are provided although at the expense of execution time. Recently, several non-iterative methods without requiring good initial estimates have been proposed. The EPnP method [12] uses four virtual control points to represent the 3D reference points and performs at the linear computational complexity. This problem formulation and use of linearization strategies facilitate the PnP methods perform efficiently. Numerous approaches have since been developed to improve the accuracy by replacing the linear formulation with polynomial solvers, e.g., DLS [89], RPnP [90], UPnP [91], OPnP [13], REPPnP [92], and CEPPnP [93]. Among these approaches, the REPPnP method integrates an outlier rejection technique into the pose estimation pipeline, and thus its input correspondences are not all necessary to be inliers.

3.3 Kabsch Algorithm

The Kabsch algorithm is a method for computing the optimal rotation and translation of two paired sets of points in N-dimensional space as to minimize the root mean squared deviation (RMSD) between them [94]. For instance, given two paired sets of 3D points $\mathbf{x}_i \leftrightarrow \mathbf{y}_i$, one can compute the rigid transformation:

$$\mathbf{y}_i = \mathbf{R}\mathbf{x}_i + \mathbf{t}, \quad i = 1, \ldots, n, \tag{3.1}$$

in a closed form solution. The first step is to compute the centroids of these two point sets:

$$\bar{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i, \quad \bar{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{y}_i, \tag{3.2}$$

and subtract the centroids from points:

$$\hat{\mathbf{x}}_i = \mathbf{x}_i - \bar{\mathbf{x}}, \quad \hat{\mathbf{y}}_i = \mathbf{y}_i - \bar{\mathbf{y}}, \quad i = 1, \ldots, n. \tag{3.3}$$

doi:10.6342/NTU201800854

The next step consists of computing a covariance matrix:

$$\mathbf{H} = \sum_{i=1}^{n} \hat{\mathbf{x}}_i \hat{\mathbf{y}}_i^\top. \tag{3.4}$$

Then we calculate the *singular value decomposition* (SVD) of the covariance matrix [95]:

$$\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^\top. \tag{3.5}$$

Finally, we can obtain the optimal rotation and translation as follows:

$$\mathbf{R} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} \mathbf{U}^\top, \quad \mathbf{t} = \bar{\mathbf{y}} - \mathbf{R}\bar{\mathbf{x}}, \tag{3.6}$$

where $d = \det(\mathbf{V}\mathbf{U}^\top)$ is used for correcting the rotation matrix $\mathbf{R}$ to ensure a right-handed coordinate system.

3.4 Lucas-Kanade Method

The Lucas-Kanade (LK) method is a widely used differential method for tackling the image alignment problem [96]. Its goal is to minimize the sum of squared error between two images with respect to the geometric parameters $\mathbf{p} = [p_1, \ldots, p_m]^\top$ (e.g., parameters of affine, projective, or rigid transformation):

$$f(\mathbf{p}) = \sum_{\mathbf{x}} \left(\mathcal{I}_c \left(\mathbf{w}(\mathbf{x}, \mathbf{p}) \right) - \mathcal{I}_t(\mathbf{x}) \right)^2, \tag{3.7}$$

where $\mathcal{I}_c$ is the camera image, $\mathcal{I}_t$ is the target image, $\mathbf{x} = [x, y]^\top$ is the pixel location, and $\mathbf{w}$ is the warp mapping the pixel location $\mathbf{x}$ in $\mathcal{I}_t$ to the sub-pixel location $\mathbf{w}(\mathbf{x}, \mathbf{p}) = [u(\mathbf{x}, \mathbf{p}), v(\mathbf{x}, \mathbf{p})]^\top$ in $\mathcal{I}_c$. Since this optimization problem is non-linear because of the presence of the functions $\mathcal{I}_c(\cdot)$ and $\mathcal{I}_t(\cdot)$, there is no closed form solution for (3.7). Consequently, the LK method assumes that a current estimate of the geometry parameters $\mathbf{p}$ is known and then iteratively solves the optimization problem for increments to the parameters $\Delta\mathbf{p}$:

$$f(\Delta\mathbf{p}) = \sum_{\mathbf{x}} \left(\mathcal{I}_c \left(\mathbf{w}(\mathbf{x}, \mathbf{p}_c + \Delta\mathbf{p}) \right) - \mathcal{I}_t \left(\mathbf{x} \right) \right)^2, \tag{3.8}$$

doi:10.6342/NTU201800854

where $\mathbf{p}_c$ is the current estimate of $\mathbf{p}$. Then the parameters are updated by $\mathbf{p}_c \leftarrow \mathbf{p}_c + \Delta\mathbf{p}$ until a satisfactory estimation result is met.

The LK method (which is a Gauss-Newton algorithm) is derived as follows. The non-linear expression in (3.8) is linearized by performing a first-order Taylor expansion of $\mathcal{I}_c\left(\mathbf{w}(\mathbf{x}, \mathbf{p}_c + \Delta\mathbf{p})\right)$ with respect to the second argument of $\mathbf{w}$ around $\mathbf{p}_c$:

$$f(\Delta\mathbf{p}) \approx \sum_{\mathbf{x}} \left(\mathcal{I}_c\left(\mathbf{w}(\mathbf{x}, \mathbf{p}_c)\right) + \mathbf{J}(\mathbf{x}, \mathbf{p}_c)\Delta\mathbf{p} - \mathcal{I}_t(\mathbf{x})\right)^2. \tag{3.9}$$

In this expression, $\mathbf{J}(\mathbf{x}, \mathbf{p}_c)$ is the $1 \times m$ Jacobian matrix of the camera image $\mathcal{I}_c$ with respect to $\mathbf{p}$ (in numerator-layout notation):

$$\mathbf{J}(\mathbf{x}, \mathbf{p}_c) = \nabla\mathcal{I}_c\left(\mathbf{w}(\mathbf{x}, \mathbf{p}_c)\right) \left.\frac{\partial\mathbf{w}(\mathbf{x}, \mathbf{p})}{\partial\mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c}, \tag{3.10}$$

where $\nabla\mathcal{I}_c\left(\mathbf{w}(\mathbf{x}, \mathbf{p}_c)\right) = \left[\frac{\partial\mathcal{I}_c}{\partial u}, \frac{\partial\mathcal{I}_c}{\partial v}\right]$ is the gradient of $\mathcal{I}_c$ evaluated at $\mathbf{w}(\mathbf{x}, \mathbf{p}_c) = [u(\mathbf{x}, \mathbf{p}_c), v(\mathbf{x}, \mathbf{p}_c)]^\top$. On the right side of (3.10), the second term is the $2 \times m$ Jacobian matrix of the warp $\mathbf{w}$ with respect to the geometry parameters $\mathbf{p}$ around $\mathbf{p}_c$:

$$\left.\frac{\partial\mathbf{w}(\mathbf{x}, \mathbf{p})}{\partial\mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c} = \left.\begin{bmatrix} \frac{\partial u}{\partial p_1} & \frac{\partial u}{\partial p_2} & \cdots & \frac{\partial u}{\partial p_m} \\ \frac{\partial v}{\partial p_1} & \frac{\partial v}{\partial p_2} & \cdots & \frac{\partial v}{\partial p_m} \end{bmatrix}\right|_{\mathbf{p}=\mathbf{p}_c}. \tag{3.11}$$

Since $f(\Delta\mathbf{p})$ in (3.9) is a quadratic form with respect to $\Delta\mathbf{p}$, minimizing $f(\Delta\mathbf{p})$ is a least squares problem and has a closed form solution for $\Delta\mathbf{p}$:

$$\Delta\mathbf{p} = \mathbf{H}\left(\mathbf{p}_c\right)^{-1} \sum_{\mathbf{x}} \mathbf{J}\left(\mathbf{x}, \mathbf{p}_c\right)^\top \left(\mathcal{I}_t(\mathbf{x}) - \mathcal{I}_c\left(\mathbf{w}(\mathbf{x}, \mathbf{p}_c)\right)\right), \tag{3.12}$$

where $\mathbf{H}(\mathbf{p}_c) = \sum_{\mathbf{x}} \mathbf{J}\left(\mathbf{x}, \mathbf{p}_c\right)^\top \mathbf{J}\left(\mathbf{x}, \mathbf{p}_c\right)$ is the $m \times m$ (Gauss-Newton approximation to the) *Hessian* matrix. After computing an increment $\Delta\mathbf{p}$ by (3.12), the current estimate of the parameters is updated by:

$$\mathbf{p}_c \leftarrow \mathbf{p}_c + \Delta\mathbf{p}. \tag{3.13}$$

The steps (3.12) and (3.13) are iterated several times until the estimates of parameters converge. Typically the test for convergence is whether the norm of the increment $\|\Delta\mathbf{p}\|$ is less than a threshold ϵ, i.e., $\|\Delta\mathbf{p}\| < \epsilon$.

doi:10.6342/NTU201800854

The optimization problem can also be solved in an alternative way. First, we turn (3.9) into vectorized form:

$$f(\Delta\mathbf{p}) \approx \|\mathbf{I}_c + \mathbf{J}_c\Delta\mathbf{p} - \mathbf{I}_t\|^2,$$

$$\mathbf{I}_c = \begin{bmatrix} \mathcal{I}_c\left(\mathbf{w}(\mathbf{x}_1, \mathbf{p}_c)\right) \\ \mathcal{I}_c\left(\mathbf{w}(\mathbf{x}_2, \mathbf{p}_c)\right) \\ \vdots \\ \mathcal{I}_c\left(\mathbf{w}(\mathbf{x}_n, \mathbf{p}_c)\right) \end{bmatrix}, \ \mathbf{J}_c = \begin{bmatrix} \mathbf{J}(\mathbf{x}_1, \mathbf{p}_c) \\ \mathbf{J}(\mathbf{x}_2, \mathbf{p}_c) \\ \vdots \\ \mathbf{J}(\mathbf{x}_n, \mathbf{p}_c) \end{bmatrix}, \ \mathbf{I}_t = \begin{bmatrix} \mathcal{I}_t(\mathbf{x}_1) \\ \mathcal{I}_t(\mathbf{x}_2) \\ \vdots \\ \mathcal{I}_t(\mathbf{x}_n) \end{bmatrix}. \tag{3.14}$$

To address this *quadratic minimization* problem, we set the derivative of $f(\Delta\mathbf{p})$ with respect to $\Delta\mathbf{p}$ equal to zero (in numerator-layout notation):

$$2\mathbf{J}_c^\top \left(\mathbf{I}_c + \mathbf{J}_c\Delta\mathbf{p} - \mathbf{I}_t\right)^\top = \mathbf{0} \quad \longrightarrow \quad \mathbf{J}_c^\top\mathbf{J}_c\Delta\mathbf{p} = \mathbf{J}_c^\top(\mathbf{I}_t - \mathbf{I}_c). \tag{3.15}$$

The equation on the right side of (3.15) is called *normal equation* [97]. We can then obtain the following closed-form solution for $\Delta\mathbf{p}$ by solving the system of linear equations in (3.15):

$$\Delta\mathbf{p} = \mathbf{H}^{-1}\mathbf{J}_c^\top(\mathbf{I}_t - \mathbf{I}_c), \quad \mathbf{H} = \mathbf{J}_c^\top\mathbf{J}_c, \tag{3.16}$$

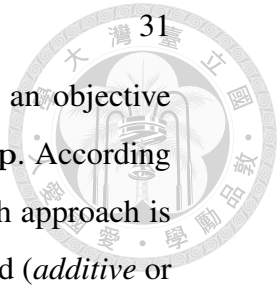
which has the same result as (3.12). We refer the interested readers to [98, 99] for more information on *matrix calculus*. A general form of (3.16) is:

$$\Delta\mathbf{p} = \mathbf{J}_c^\dagger(\mathbf{I}_t - \mathbf{I}_c), \tag{3.17}$$

where $\mathbf{J}_c^\dagger = \left(\mathbf{J}_c^\top\mathbf{J}_c\right)^{-1}\mathbf{J}_c^\top$ is referred to as the *Moore-Penrose pseudoinverse* of $\mathbf{J}_c$. Another way to compute the pseudoinverse is using SVD, which may be more accurate and numerically stable. If $\mathbf{J}_c = \mathbf{U}\Sigma\mathbf{V}^\top$ is the SVD of $\mathbf{J}_c$, then $\mathbf{J}_c^\dagger = \mathbf{V}\Sigma^\dagger\mathbf{U}^\top$. For the rectangular diagonal matrix Σ, its pseudoinverse $\Sigma^\dagger$ can be obtained by taking the reciprocal of each non-zero element on the diagonal, leaving the zeros in place, and then transposing the matrix.

Several optimization methods have been proposed, and a wide variety of extensions have been made to the original formulation of the LK algorithm. These iterative algorithms have the same structure: In each iteration, they first compute

doi:10.6342/NTU201800854



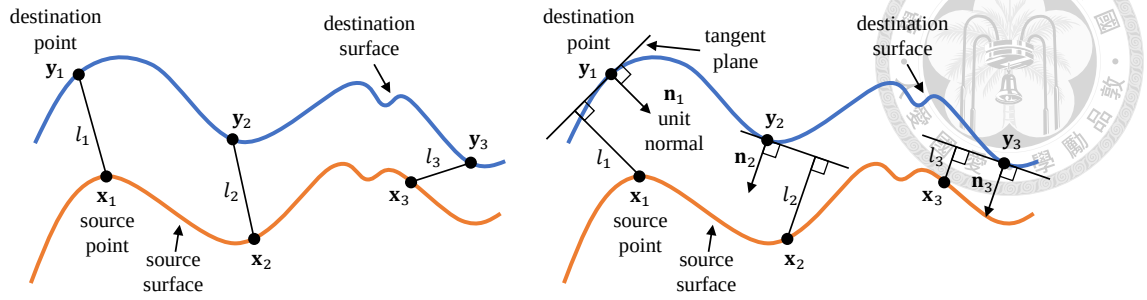
the increment $\Delta\mathbf{p}$ of the current parameters $\mathbf{p}_c$ which minimize an objective function similar to (3.8), and then update the parameters $\mathbf{p}_c$ with $\Delta\mathbf{p}$. According to [100], there are mainly four different kinds of approaches. Each approach is classified depending on: (1) the type of objective function employed (*additive* or *compositional*) and (2) the direction of warping (*forward* or *inverse*). The canonical LK formulation presented in the previous paragraph is regarded as the forwards additive (FA) algorithm. In addition, there are also the forward compositional (FC) algorithm [101], the inverse additive (IA) algorithm [102], and the inverse compositional (IC) algorithm [103]. Different from methods employing first-order approximation mentioned above, Benhimane and Malis [104] propose the efficient second-order minimization (ESM) method which applies a second-order approximation. We refer the interested readers to [100, 105] for more detailed information about these iterative optimization algorithms and their comparisons.

3.5 Iterative Closest Point

The Iterative closest point (ICP) algorithm is a widely used approach in aligning 3D surfaces given an initial estimation of the rigid body transformation [106]. It iteratively refines the transformation from a source point set (i.e., transformed) to a destination point set (i.e., fixed). This approach allows integrating data from different sources into a bigger model. In addition, it can also be used for tracking target object poses in 3D space. The ICP method contrasts with the Kabsch algorithm in that the former treats correspondences as variables to be estimated, whereas the latter requires correspondences between point sets as the input data. Pomerleau *et al.* [107] provide an excellent survey of the different ICP variants during the last twenty years as well as their use cases.

In the original ICP method proposed by Besl and McKay [106], points in one set are paired with their closest points in the other set to form correspondences. Then a *point-to-point* error metric is used, where the sum of the squared distance

doi:10.6342/NTU201800854



(a) Point-to-point error. (b) Point-to-plane error.

Figure 3.1: Two error metrics mostly employed in ICP methods.

between corresponding points is minimized, as illustrated in Figure 3.1(a). The process is iterated until the error (or the difference of errors in between two consecutive iterations) becomes smaller than a predefined threshold. For each iteration, the best rigid transformation according to the point-to-point error metric has a closed-form solution and can be achieved by using the Kabsch algorithm described in Section 3.3. In contrast, Chen and Medioni [108] apply the *point-to-plane* error metric, in which the sum of the squared distance between points and the tangent planes at their corresponding closest points is minimized, as illustrated in Figure 3.1(b). Different from the point-to-point error, the point-to-plane error is usually minimized by using non-linear least squares approaches, such as the Gauss-Newton algorithm and the Levenberg-Marquardt method [109]. Although each iteration of the point-to-plane ICP approach is generally slower than the point-to-point one, the former has been proved to have better convergence rates [110, 111]. Moreover, when the relative rotation between two surfaces is small, the nonlinear least-squares optimization problem can even be approximated with a linear one, so as to speed up the computation. We explain how to achieve the linearization in the next paragraph.

When using the point-to-plane error metric, our goal is to minimize the following objective function with respect to the rotation $\mathbf{R}$ and translation $\mathbf{t}$:

$$f(\mathbf{R},\mathbf{t}) = \sum_{i}^{n} \left(\mathbf{n}_i^{\top}(\mathbf{R}\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i)\right)^2, \tag{3.18}$$

doi:10.6342/NTU201800854

where $\mathbf{x}_i$ is a source point, $\mathbf{y}_i$ is a destination point, and $\mathbf{n}_i$ is the unit normal vector at $\mathbf{y}_i$. To linearize (3.18), we assume the relative rotation between the source and destination surfaces is small and replace the original rotation $\mathbf{R}$ with the small angle approximation $\hat{\mathbf{R}}(\mathbf{r})$ presented in (2.21):

$$\begin{aligned} f\left(\hat{\mathbf{R}}(\mathbf{r}),\mathbf{t}\right) &= \sum_i^n \left(\mathbf{n}_i^\top \left(\hat{\mathbf{R}}(\mathbf{r})\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\right)\right)^2 \\ &= \sum_i^n \left(\mathbf{n}_i^\top \left(\mathbf{x}_i + [\mathbf{r}]_\times \mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\right)\right)^2 \\ &= \sum_i^n \left(\mathbf{n}_i^\top \left(\mathbf{x}_i - [\mathbf{x}_i]_\times \mathbf{r} + \mathbf{t} - \mathbf{y}_i\right)\right)^2 \\ &= \sum_i^n \left(\begin{bmatrix} -\mathbf{n}_i^\top [\mathbf{x}_i]_\times & \mathbf{n}_i^\top \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{t} \end{bmatrix} - \mathbf{n}_i^\top(\mathbf{y}_i - \mathbf{x}_i)\right)^2 \\ &= \sum_i^n \left(\begin{bmatrix} (\mathbf{x}_i \times \mathbf{n}_i)^\top & \mathbf{n}_i^\top \end{bmatrix} \begin{bmatrix} \mathbf{r} \\ \mathbf{t} \end{bmatrix} - \mathbf{n}_i^\top(\mathbf{y}_i - \mathbf{x}_i)\right)^2 . \end{aligned} \tag{3.19}$$

Similar to the vectorization operation from (3.9) to (3.14), we turn (3.19) into vectorized form:

$$f(\mathbf{p}) = \|\mathbf{A}\mathbf{p} - \mathbf{b}\|^2,$$

$$\mathbf{A} = \begin{bmatrix} (\mathbf{x}_1 \times \mathbf{n}_1)^\top & \mathbf{n}_1^\top \\ (\mathbf{x}_2 \times \mathbf{n}_2)^\top & \mathbf{n}_2^\top \\ \vdots & \\ (\mathbf{x}_n \times \mathbf{n}_n)^\top & \mathbf{n}_n^\top \end{bmatrix}, \ \mathbf{p} = \begin{bmatrix} \mathbf{r} \\ \mathbf{t} \end{bmatrix}, \ \mathbf{b} = \begin{bmatrix} \mathbf{n}_1^\top(\mathbf{y}_1 - \mathbf{x}_1) \\ \mathbf{n}_2^\top(\mathbf{y}_2 - \mathbf{x}_2) \\ \vdots \\ \mathbf{n}_n^\top(\mathbf{y}_n - \mathbf{x}_n) \end{bmatrix}. \tag{3.20}$$

A closed-form solution for this quadratic minimization problem can then be achieved (by following the arithmetic operations from (3.14) to (3.17)):

$$\mathbf{p} = \begin{bmatrix} \mathbf{r} \\ \mathbf{t} \end{bmatrix} = \mathbf{A}^\dagger \mathbf{b}, \tag{3.21}$$

where $\mathbf{A}^\dagger$ is the pseudoinverse of $\mathbf{A}$. Note that since $\hat{\mathbf{R}}(\mathbf{r})$ may not be a valid rotation, we should instead use the rotation $\mathbf{R}(\mathbf{r})$ described in (2.18) as the final solution (despite the fact that it is not equal to $\hat{\mathbf{R}}(\mathbf{r})$ applied in (3.19)).

doi:10.6342/NTU201800854

3.6 Line Search & Trust Region

In *mathematical optimization*, there are two primary iterative strategies to find a local minimum point $\mathbf{p}^*$ of an objective function $f : \mathbb{R}^m \to \mathbb{R}$. The first one is the *line search* method. It first chooses a descent direction $\mathbf{n}$ along which f will be reduced and then computes a step size α which determines how far the current point $\mathbf{p}$ should be moved along $\mathbf{n}$:

$$\min_{\alpha>0} f(\mathbf{p} + \alpha\mathbf{n}). \tag{3.22}$$

The descent direction $\mathbf{n}$ can be computed by various approaches, such as the *gradient descent* algorithm (or *steepest descent* algorithm), *Newton's method*, and *Quasi-Newton* method. If the objective function f can be expressed as a sum of squares (e.g., (3.7)), then the *Gauss-Newton* algorithm can also be employed to compute $\mathbf{n}$. To solve (3.22) *exactly*, we would derive the maximum benefit from $\mathbf{n}$ by approaches such as the *conjugate gradient* method. However, in most cases, it is not necessary to find the exact minimum of (3.22) in each iteration. Instead, an *inexact* line search approach, such as the *backtracking line search* algorithm, may be used to find an α that *loosely* approximates the minimum along $\mathbf{n}$.

The second one is the *trust region* method (or *restricted step* method). For each iteration, it first gather the information about the objective function f around the current point $\mathbf{p}$ to construct a *model function* m (often a quadratic function) whose behavior near $\mathbf{p}$ is similar to that of f. Then the candidate step $\Delta\mathbf{p}$ is computed by approximately solving the following problem:

$$\min_{\Delta\mathbf{p}} m(\mathbf{p} + \Delta\mathbf{p}), \qquad \|\Delta\mathbf{p}\| \leq \Delta, \tag{3.23}$$

where $\Delta > 0$ is referred to as the *trust-region radius* (since the trust region is usually defined as a ball). The radius will be enlarged in the next iteration if the model function m approximates the objective function f well within the trust region. Otherwise, the radius will be contracted. The fit is evaluated by comparing the ratio between the expected improvement from the approximation m and the

doi:10.6342/NTU201800854

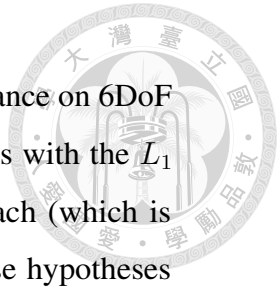
actual improvement observed in f. Simple thresholding of the ratio is used as the criterion for determining whether the trust region should be expanded or contracted. A widely used trust region method is the *Levenberg–Marquardt* algorithm.

The line search and trust region approaches are different in the sense that while the former firstly finds a descent direction and then a step size, the latter chooses a step size (or the size of the trust region) first before computing the descent direction. Both kinds of approaches are advantageous in most cases when solving optimization problems. We refer the interested readers to [109] for more information on mathematical optimization.

3.7 Object Pose Estimation Approaches

Although direct methods [112, 5, 3] have been shown to achieve promising results on texture-less objects, the success is limited to objects that are not occluded. On the other hand, even though feature-based methods [7, 8] can estimate and track pose under partial occlusion, these approaches do not perform well on textureless objects. For the past few years, learning-based approaches have gained increasing attention as they perform well in various conditions. During the first period, most learning-based methods are developed based on decision forests [14, 15, 113, 114, 16, 115], where a set of local patches are sampled from training images. Instead of determining object pose directly, Brachmann *et al.* [15] train a decision forest that stores a distribution over a set of intermediates, i.e., object coordinates, at each leaf node. Given a test image, patches are first densely sampled and evaluated by the decision forest to obtain the estimated object coordinates, and then rigid transformation hypotheses between 3D-to-3D correspondences can be computed by the Kabsch algorithm [94] presented in Section 3.3. The final pose is determined by the minimal cost of an objective function with a RANSAC-based scheme. This method [15] is further improved in various ways, such as: (1) replacing the cost function with a learned alternative using a convolutional

doi:10.6342/NTU201800854



neural network (CNN) for better measurement of the geodesic distance on 6DoF pose manifolds [113]; (2) refining the regressed object coordinates with the L_1 regularized loss function [16]; (3) using a *policy gradient* approach (which is called *PoseAgent*, a *reinforcement learning* agent) to improve pose hypotheses directly [116]; and (4) employing a *global* geometry check strategy to generate fewer but better pose hypotheses [117].

Recently, research in most pose estimation tasks has been dominated by deep neural networks. At first, Wohlhart *et al*. [118] apply CNNs for direct object pose recovering (rotation only) from holistic template images. Nonetheless, these input images should have been cropped around target objects from original camera frames by employing some detection techniques beforehand. In the last few years, there have been many object detectors which can find the bounding boxes around objects effectively and efficiently, such as Faster R-CNN [119, 120, 121, 122], YOLO [123, 124], and SSD [125]. Kehl *et al*. [126] accordingly detect the target objects and estimate their 6DoF poses simultaneously through extending the SSD approach to cover the full 6D pose space. Different from these two methods [118, 126] which cast pose estimation into classification tasks, the PoseCNN paradigm [127] estimate the object pose by regressing convolutional features extracted inside the bounding box of the object to parameters of rotation and translation. In addition, there are also algorithms which work by firstly predicting 2D projections of the 3D points, which are either corners of the object's bounding box [128, 129, 130] or semantic object keypoints [131], and then the 6DoF pose can be computed from the 2D-to-3D correspondences with a PnP algorithm. Rad *et al*. [130] further apply the *transfer learning* (or *domain transfer*) technique to learn a mapping from the exemplary representations of real images to the exemplary representations of synthetic images. Consequently, they can just exploit only *synthetic* images when training a deep network to estimate the 6DoF object pose from a *real* image. By contrast with these *holistic* methods, Kehl *et al*. [17] propose a voting-based approach using auto-encoder descriptors of *local patches* for 6DoF pose hypotheses

doi:10.6342/NTU201800854

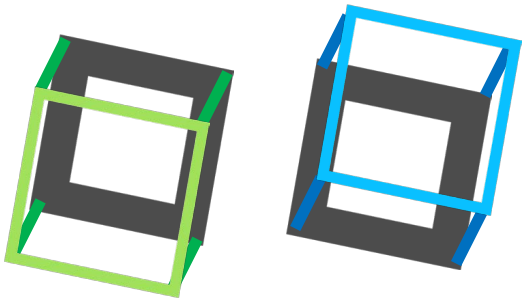


Figure 3.2: The pose ambiguity can be regarded as a geometric illusion. There appears to be more than one 3D geometrical explanation based on the same perspective-projected marker in the camera image.

generation.

Although object poses can be estimated in frames by these pose estimation approaches, they may not suit AR applications because they are usually not accurate enough to generate stable pose sequences. Nonetheless, one can still use these methods to compute a rough initial pose and use this pose as the input data for some pose tracking or refining algorithms.

3.7.1 Pose Disambiguation for Planar Objects

If the target object is a plane, then it may cause the *pose ambiguity* problem, as illustrated in Figure 3.2. Pose ambiguity is related to situations where the error function has several local minima for a given configuration, which is the leading cause of flipping estimated poses in an image sequence [132, 133], as presented in Figure 3.3. This problem occurs not only under orthographic projection but also for perspective transformation, especially when the planar target object is significantly tilted with respect to camera views. A typical approach for pose disambiguation is first to find all possible poses which are stationary points with local minima of a designed objective function, and then the one with smallest objective values is considered as the estimated pose. Empirically, the number of

doi:10.6342/NTU201800854

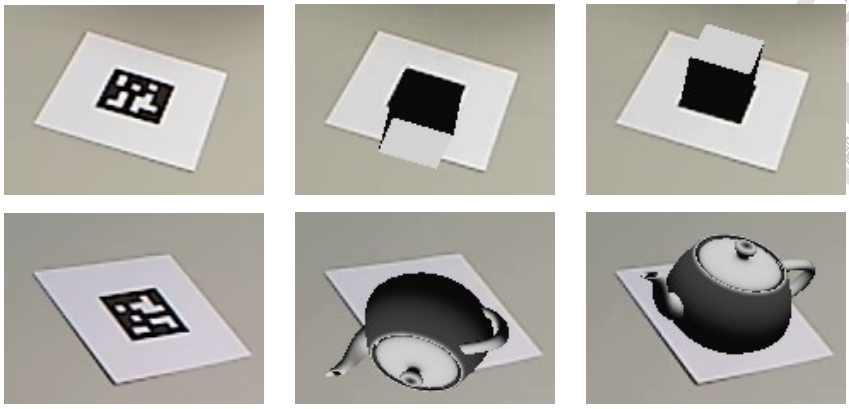


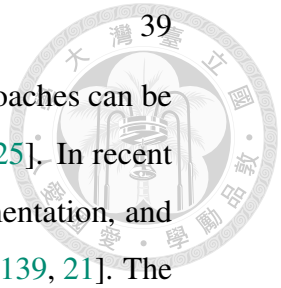
Figure 3.3: Pose ambiguity in real cases. The images in the first column are the original images. Images with a synthetic model rendered according to each ambiguous pose are shown in the last two columns.

ambiguous poses is two in general. Schweighofer and Pinz [88] observe that two local minima exist for cases with images of a planar target object viewed by a perspective camera, and a method is developed to determine a unique solution based on an iterative pose estimation method [87]. The PnP problem can be posed as a minimization problem [13], and all the stationary points can be determined by using the Gröbner basis method [134]. In addition, given a pose solution, the other ambiguous pose can also be generated by reflecting the first pose with respect to a plane whose normal vector is the line-of-sight from the camera image center to the planar target center [135].

3.8 Object Pose Tracking Approaches

Object pose tracking can be regarded as an energy minimization problem from an initial estimate of pose parameters. To obtain an accurate pose, an energy function should reflect the geodesic distance on the 6DoF manifold that can be computed efficiently. An early yet comprehensive review of model-based tracking is conducted by Lepetit and Fua [136]. One of the classical methods for object pose tracking is the feature-based paradigm, which computes the object pose based

doi:10.6342/NTU201800854

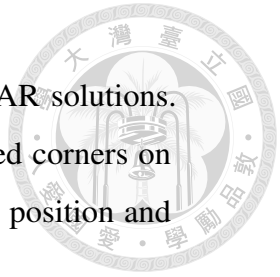


on tracked natural features from frame to frame [137]. These approaches can be further improved by adopting the particle filtering technique [24, 25]. In recent years, pose tracking approaches combining region-based 2D segmentation, and 2D-to-3D pose estimation become more and more popular [138, 19, 139, 21]. The pose parameters are computed based on iteratively minimizing the distance of point correspondences between the segmented and projected contour of the target object. Prisacariu and Reid [19] present the PWP3D method which solves the pose recovering problem with a pixel-wise minimization scheme. Other approaches based on the PWP3D method mainly focus on improving the segmentation results and gradient descent search strategies [139, 21]. Since the release of the low-cost depth device Kinect, many of the pose tracking approaches have applied the ICP algorithm to align 3D point clouds between a CAD model and real scenes [18, 140]. Kehl *et al*. [141] further propose a combined tracking approach which leverages both the region-based method and the ICP algorithm. In addition, there are also learning-based methods of object pose tracking. Tan *et al*. use random forests to learn more robust features that better handle occlusion [142], and they further combine this method with an optimization approach, which minimizes an energy function to find the best transformation between the source and the target, to improve the tracking accuracy and reduce jitter [143]. Furthermore, Garon and Lalonde [144] leverage a deep neural network to automatically track the 6DoF pose of an object robustly even under clutter and occlusion; Li *et al*. [145] propose a deep iterative matching network for 6DoF object pose refinement, which can also be utilized in pose tracking applications.

3.8.1 Binary Square Fiducial Marker Tracking Solutions

The 2D binary square fiducial marker is an easiest-to-construct, and maybe the most famous 6DoF tracking solution. It has been used extensively for both recognition and tracking. Libraries for efficient identification and localization of binary square markers, such as ARToolKit [146], ARToolKitPlus [147], ARTag [148, 149],

doi:10.6342/NTU201800854



and ArUco [150, 151], have become a building block for many AR solutions. The typical output of such a library is a sparse set of corresponded corners on the recognized marker, which can then be used to solve for 6DoF position and orientation by PnP algorithms presented in Section 3.2.

3.8.2 Pen Tracking Paradigms

5DoF and 6DoF tracking of pens have been an active area of research in the computer vision and human-computer interaction communities. The IrCube [152] and IrPen [153] trackers rely on setting up a source localization problem involving a cluster of directed LEDs, achieving an accuracy of 10 mm in a 20×20 cm^2 area. The Lumitrack approach [154] uses laser projections of coded patterns and a linear optical sensor to track at 800 Hz with an accuracy of 5 mm. The Light chisel system [155] consists of two LEDs inside a diffuse cylinder fiducial tracked by stereo cameras at an accuracy of 2 mm over a $56\times31\times33$ cm^3 volume. A pen can also be tracked from a light-field camera [156] through a lenslet array with an accuracy of 3 mm.

3.8.3 Commercial Tracking Systems

Consumer solutions for 6DoF tracking typically combine micro-electromechanical systems (MEMS) inertial measurement with laser positioning (e.g., HTC Vive [157]), optical tracking (e.g., Oculus Touch [158] and PS Move [159]), or magnetic tracking (e.g., Razer Hydra [160]).

Motion capture is another widely used method for high-fidelity 6DoF tracking. In a mocap system, such as OptiTrack [161], Vicon [162], and Qualisys [163], typically a large array of strobing cameras observes a set of passive retroreflective fiducials. Triangulation and tracking are used to obtain the absolute position and orientation of the tracked object at better than millimeter accuracy.

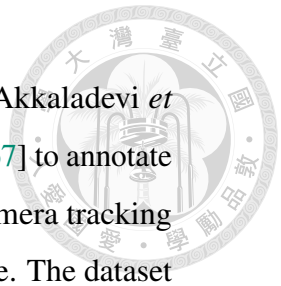
doi:10.6342/NTU201800854

3.9 Benchmark Datasets

Numerous datasets have been developed to evaluate algorithms in areas related to 3D pose estimation and tracking. The dataset presented by Lieberknecht *et al.* [164] contains 40 sequences of eight different textured 2D objects and five unconstrained motions (e.g., zoom-in and translation). A dataset with 96 videos from six textured planar targets and varying geometric distortions as well as lighting conditions is constructed by Gugglitz *et al.* [4]. The homography transformation parameters are provided in this dataset. Since a rolling-shutter camera is used, it may be difficult to obtain the exact 6DoF pose from the homography parameters when the relative motion is significant.

Hinterstoisser *et al.* [5] construct a dataset of 18,000 images with 15 textureless 3D objects, which is further extended for multi-instance 3D object detection and pose estimation [14]. Similarly, a dataset with 20 textured and textureless objects is proposed [15] where each one is recorded under three different lighting conditions. In addition, Hodan *et al.* [165] propose the T-LESS dataset that features thirty commodity electrical parts which have no significant texture. There is also a dataset [144] consisting of sequences in which the occlusion is varied from low to high levels. For the datasets mentioned above, both color and depth images are recorded using handheld Kinect V1 or Kinect V2 cameras. The target objects are attached to a planar board surrounded with fiducial markers, which provide the corresponding poses. Since markers cannot be accurately localized in a blurry image, the recorded targets need to be static in front of the camera, and thus these datasets do not contain distortions that are crucial for performance evaluation of pose tracking in real-world scenarios. The real pose is also arduous to compute because of the rolling-shutter effect which will change the appearance of markers whenever there exists some camera movement. Different from using fiducial markers, the ground-truth object poses in the datasets [26, 25] are manually labeled and less accurate. Even the poses estimated by Krull *et al.* [25] and Xiang *et al.* [127] are further refined by the ICP method, the estimates are not

doi:10.6342/NTU201800854



accurate due to noisy measurements of depth values. In contrast, Akkaladevi *et al*. [166] utilize a camera tracking solution called ReconstructMe [167] to annotate poses. However, the ground-truth pose accuracy depends on the camera tracking results performed by ReconstructMe, which may not be very reliable. The dataset proposed by Choi and Christensen [24] consists of four synthetically generated sequences of four models. The main drawback of this synthetic dataset is the lack of distortions in both RGB-D images and motion blurs. We summarize the characteristics of existing benchmark datasets for pose estimation and tracking in Table 3.1.

doi:10.6342/NTU201800854

Table 3.1: Benchmark datasets for object pose estimation. Using a programmable robotic arm, we can record images under different motion patterns and different speed. The recorded sequences hence contain different distortions that are crucial for performance evaluation of pose tracking algorithms for real-world scenarios.

Benhmark	Device	Mechanism	Pose Establishment	Video Clips	# 2D Targets	# 3D Targets	# Motion Patterns	# Frames
Lieberknecht [164]	Marlin F-080C	Handheld	Marker-based	Yes	8	-	5	48,000
Gauglitz [4]	Fire-i	Manually Operated Contraption	Direct Alignment	Yes	6	-	16	6,889
Hinterstoisser [5]	Kinect V1	Handheld	Marker-based	No	-	15	-	18,000
Tejani [14]	Kinect V1	Handheld	Marker-based	No	-	3	-	5,229
Brachmann [15]	Kinect V1	Handheld	Marker-based	No	-	20	3	10,000
Rennie [26]	Kinect V1	Robotic Arm	Manual	No	-	24	-	10,368
Krull [25]	Kinect V1	Handheld	ICP	Yes	-	3	-	1,100
Choi [24]	Synthetic	-	Synthetic	Yes	-	4	-	4,000
Akkaladevi [166]	Primesense Carmine 1.09	Handheld	ReconstructMe	Yes	-	4	-	4,000
Hodan [165]	Kinect V2	Handheld	Marker-based	No	-	30	-	49,000
Garon [144]	Kinect V2	Handheld	Marker-based	Yes	-	4	9	7,500
Xiang [127]	Asus Xtion Pro Live	Handheld	ICP	Yes	-	21	3	133,827
Proposed	Kinect V2	Programmable Robotic Arm	Checkerboard-based	Yes	6	6	23	100,956

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

Chapter 4

OPT: A Benchmark Dataset for 6DoF Object Pose Tracking

In this work, we propose a large-scale benchmark dataset of RGB-D video sequences for both 2D and 3D objects with ground-truth information, as shown in Figure 1.1. The proposed benchmark dataset contains 690 color and depth videos of varying degrees of textured and geometric objects with over 100,000 frames. These videos are annotated with different imaging conditions (i.e., *Translation*, *Zoom*, *In-plane Rotation*, *Out-of-plane Rotation*, *Flashing Light*, *Moving Light*, and *Free Motion*) and speed recorded with a Kinect V2 sensor mounted on a programmable robotic arm. A 3D printer renders the objects in the benchmark dataset with distinct textures. The ground-truth poses are computed using a designed checkerboard and checkerbox for 2D and 3D objects. Due to the global-shutter infrared camera with fast shutter speed from the Kinect V2 sensor, we can annotate the ground-truth poses by leveraging the clear infrared images under fast motions.

The contributions of this work are summarized below:

Benchmark dataset. We design a benchmark dataset for 6DoF object pose tracking. It consists of 690 videos under seven varying conditions with five speeds. It is a large dataset where images are acquired from a moving camera for performance

doi:10.6342/NTU201800854

evaluation of both 2D and 3D object pose tracking algorithms. Furthermore, the proposed dataset can also be used in other computer vision tasks such as 3D feature tracking and matching.

Performance evaluation. Each pose tracking method is extensively evaluated and analyzed using more than 100,000 frames including both 2D and 3D objects. Since the state-of-the-art simultaneous localization and mapping (SLAM) methods [2, 60] are able to track and relocalize camera pose in real time, we also evaluate these approaches by adapting them to object pose tracking scenarios. We present the extensive performance evaluation of the state-of-the-art methods using the proposed benchmark dataset. Finally, we discuss the potential research directions in this field. The proposed benchmark dataset is available online at media.ee.ntu.edu.tw/research/OPT. Below we describe how we collect data and compute the 6DoF pose of the target object with addressing the rolling-shutter issues in detail.

4.1 Acquiring Images

The color, depth, and infrared images of each sequence are obtained from a Kinect V2 sensor mounted on a programmable KUKA KR 16-2 CR robot arm, as illustrated in Figure 4.1. The robotic arm, which has six axes and repeatability of 0.05 mm, can be programmed to move in complex trajectories precisely. Each 2D object shown in Figure 4.2 is a printed pattern with size 133.6×133.6 mm^2 surrounded by a checkerboard glued to an acrylic plate. Each 3D object shown in Figure 4.3 is generated by a 3D printer with resolution 300×450 dpi and 0.1 mm layer thickness. The length, width, and height of 3D objects illustrated in Figure 1.1 are in the ranges of (57.0, 103.6), (57.0, 103.6), and (23.6, 109.5), respectively in mm. We describe how the ground-truth 6DoF pose of a target object is obtained based on the 2D checkerboard or 3D checkerbox in Section 4.2.

The object motions in the proposed benchmark dataset are (regarded as moving

doi:10.6342/NTU201800854

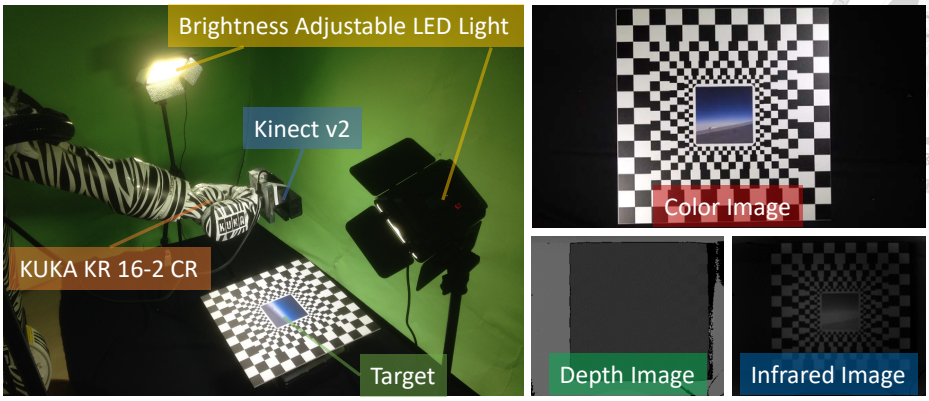
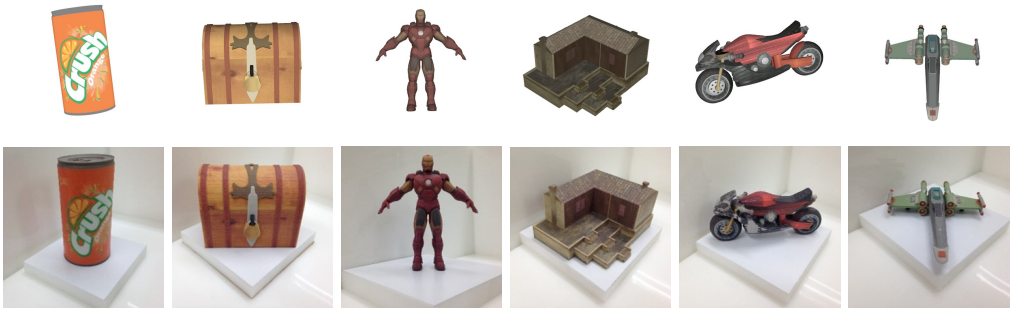


Figure 4.1: Sequences in the proposed dataset are recorded with a Kinect V2 sensor mounted on a programmable robotic arm. Note that we normalize the intensity of the depth image in this figure for clarity.

Low Normal Rich

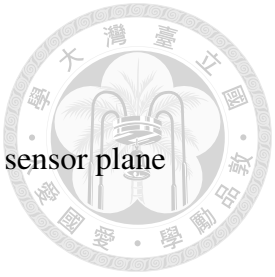
Figure 4.2: 2D objects with low (*Wing*, *Duck*), normal (*City*, *Beach*), and rich (*Firework*, *Maple*) texture.



Simple Normal Complex

Figure 4.3: 3D objects with simple (*Soda*, *Chest*), normal (*Ironman*, *House*), and complex (*Bike*, *Jet*) geometry.

doi:10.6342/NTU201800854



object rather than the camera):

Translation. An object moves along a circle parallel to the camera sensor plane with motion blur in all directions.

Zoom. An object moves forward first and then backward.

In-plane Rotation. An object rotates along an axis perpendicular to the camera sensor plane.

Out-of-plane Rotation. An object rotates along an axis parallel to the camera sensor plane.

Flashing Light. The light source is turned on and off repeatedly, and the object moves slightly.

Moving Light. The light source moves and results in illumination variations while the object moves slightly.

Free Motion. An object moves in arbitrary directions.

The objects move at five speeds in *Translation*, *Zoom*, *In-plane Rotation*, and *Out-of-plane Rotation* such that the videos are close to real-world scenarios with different image distortions (e.g., motion blurs). For each 3D object, videos from four camera perspectives are recorded. A square region on the bottom plane is hollowed out to fit the base of a 3D object as shown in Figure 4.4. Table 4.1 lists the properties of all motion patterns in the proposed dataset.

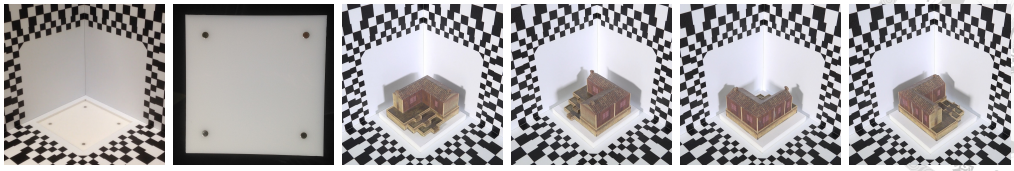
Since the Kinect V2 sensor drops frames occasionally, it may affect some tracking approach which exploits a motion model to perform trajectory prediction and obtain a better initial pose in the next frame. To address this issue, we develop a method to record the RGB-D video sequences instead of using the original recording software by Microsoft. The GUI of our recording program is shown in Figure 4.5. This program will automatically detect if some frames are dropped during recording and display an error message when the unfortunate happens. We record the sequences repeatedly until all of them are complete (i.e., exact 30 fps). We also ensure that the sequences only contain the desired motions by removing the start and end of the sequences where the object is stationary.

doi:10.6342/NTU201800854

Table 4.1: Evaluated motion patterns. For each speed level, there are six sequences with 2D models and 24 sequences with 3D models (6 models × 4 sides).

Motion Pattern	Model	Level	# Frames	Avg. Rot. Vel. [deg/s]	Avg. Trans. Vel. [mm/s]
Translation	2D	1	241	0.69428	37.916
		2	83	0.79381	110.3443
		3	52	0.88252	175.9481
		4	36	1.1011	252.0285
		5	30	1.2767	297.7782
	3D	1	174	0.69185	32.8919
		2	61	0.85972	93.2347
		3	38	1.0905	147.2741
		4	27	1.5142	199.8588
		5	23	1.9685	222.9882
Zoom	2D	1	295	0.66289	28.3413
		2	104	0.7268	80.7141
		3	65	0.73599	129.6973
		4	47	0.78002	180.9885
		5	38	0.83148	224.9915
	3D	1	341	0.87975	31.6376
		2	119	1.1265	90.9394
		3	76	1.1053	143.1166
		4	55	1.1661	199.1822
		5	44	1.2035	250.0844
In-plane Rotation	2D	1	209	6.5114	4.3954
		2	75	18.2884	8.7469
		3	46	30.0623	14.2698
		4	35	39.8567	19.0389
		5	29	48.5156	23.6787
	3D	1	370	3.3929	5.2248
		2	127	9.9255	13.1589
		3	78	16.2043	20.9461
		4	53	23.9756	30.522
		5	41	31.1681	39.1322
Out-of-plane Rotation	2D	1	555	3.8118	5.3987
		2	189	11.2437	14.0884
		3	116	18.3775	22.7303
		4	81	26.4373	32.4912
		5	63	34.1305	41.9536
	3D	1	600	3.7498	9.2109
		2	202	11.0287	24.0732
		3	123	18.1562	39.1878
		4	85	26.3867	56.7088
		5	64	35.1654	74.6798
Flashing Light	2D	-	161	0.61427	4.8458
	3D	-	154	0.63751	5.03
Moving Light	2D	-	164	0.60217	4.7679
	3D	-	154	0.64017	5.0404
Free Motion	2D	-	784	24.9678	127.2679
	3D	-	323	173.992	5.800

doi:10.6342/NTU201800854



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

Figure 4.4: The checkerbox is designed so that the 3D object can be changed with four different sides. (a) The hollowed part on the bottom plane of the checkerbox. (b) The bottom view of the base of the 3D object. This base can adhere to the checkbox with four magnet pairs. (c) The front view, (d) left view, (e) back view, and (f) right view of a target.

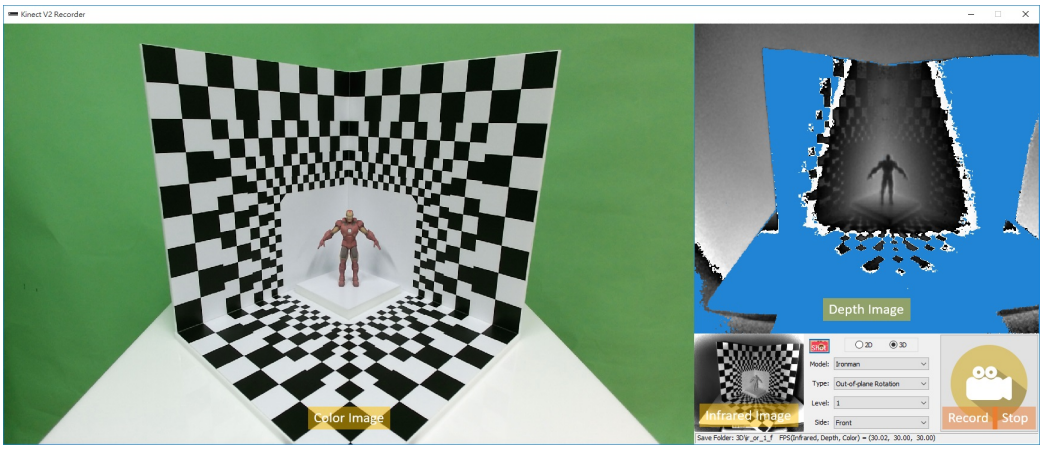


Figure 4.5: GUI for recording RGB-D sequences captured by Kinect V2.

4.2 Obtaining Ground-truth Object Pose

We estimate the intrinsic and extrinsic camera parameters using the calibration toolbox [168]. It is worth noting that depth and infrared images, as shown in Figure 4.1, are obtained from the same sensor (i.e., depth camera). Therefore, we calibrate depth camera using infrared images. The estimated intrinsic parameters are shown in Table 4.2. Next, we conduct an extrinsic calibration [168] of the two

doi:10.6342/NTU201800854

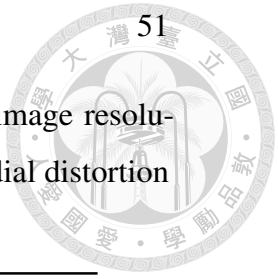


Table 4.2: Intrinsic parameters of the used Kinect V2. (w, h): image resolution; (f_x, f_y): focal length; (c_x, c_y): principal point; (r_1, r_2, r_3): radial distortion coefficients; and (t_1, t_2): tangential distortion coefficients.

Camera Type	Color Camera	Depth Camera
Shutter Type	Rolling Shutter	Global Shutter
FPS (Hz)	30	30
(w, h)	(1920,1080)	(512,424)
(f_x, f_y)	(1060.197,1060.273)	(366.736,366.458)
(c_x, c_y)	(965.809,561.9526)	(254.026,207.470)
(r_1, r_2, r_3)	(0.0435,-0.0183,-0.031)	(0.107,-0.297,0.114)
(t_1, t_2)	(0.000905,0.000709)	(0.0013,-0.00026)

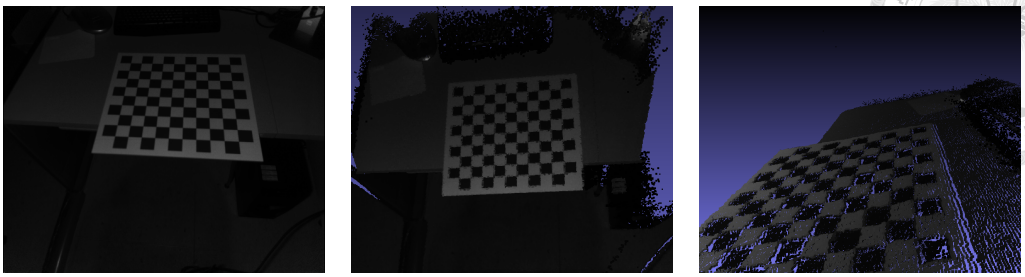
cameras resulting in the transformation matrix:

$$\mathbf{T}_{d2c} = \begin{bmatrix} \mathbf{R}_{d2c} & \mathbf{t}_{d2c} \\ \mathbf{0} & \mathbf{1} \end{bmatrix} = \begin{bmatrix} 1.0000 & -0.0053 & 0.0038 & -52.51 \\ 0.0053 & 1.0000 & -0.0041 & 0.602 \\ -0.0038 & 0.0041 & 1.0000 & -0.326 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.1}$$

from the depth camera coordinate system to color camera coordinate system.

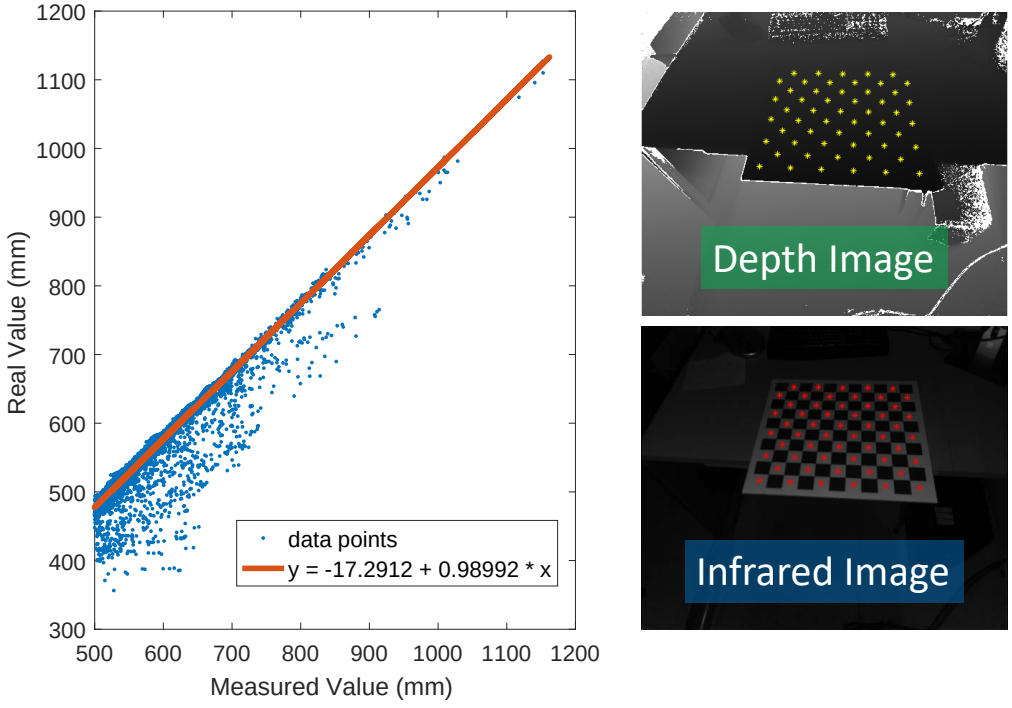
We find that the measured distance within dark regions in the depth maps are less accurate, as illustrated in Figure 4.6. This is also discussed in [1], as the Kinect V2 sensor uses an indirect time-of-flight system based on light modulation to estimate depth. Since some measured points may be affected by points within dark regions during calibration, we marginalize the unreliable measurements by applying the robust regression with Bisquare weighting, as shown in Figure 4.7. We also rectify the depth images according to the regression results between these two types of images. In addition, since the transformation matrix $\mathbf{T}_{d2c}$ between the depth and camera coordinate is estimated, the depth image in the color camera coordinate system (and vice versa) can be obtained according to $\mathbf{T}_{d2c}$. We illustrate

doi:10.6342/NTU201800854



(a) (b) (c)

Figure 4.6: (a) Infrared image. (b) Point cloud corresponding to (a) from one viewpoint. (c) Point cloud from one another viewpoint. The measured distance within a dark region is larger than real cases.



(a) (b)

Figure 4.7: (a) Deviations between real and measured depth values. We perform the robust regression with Bisquare weighting. (b) Points are sampled in the center of white blocks since the measured values on a dark surface are less reliable [1].

doi:10.6342/NTU201800854

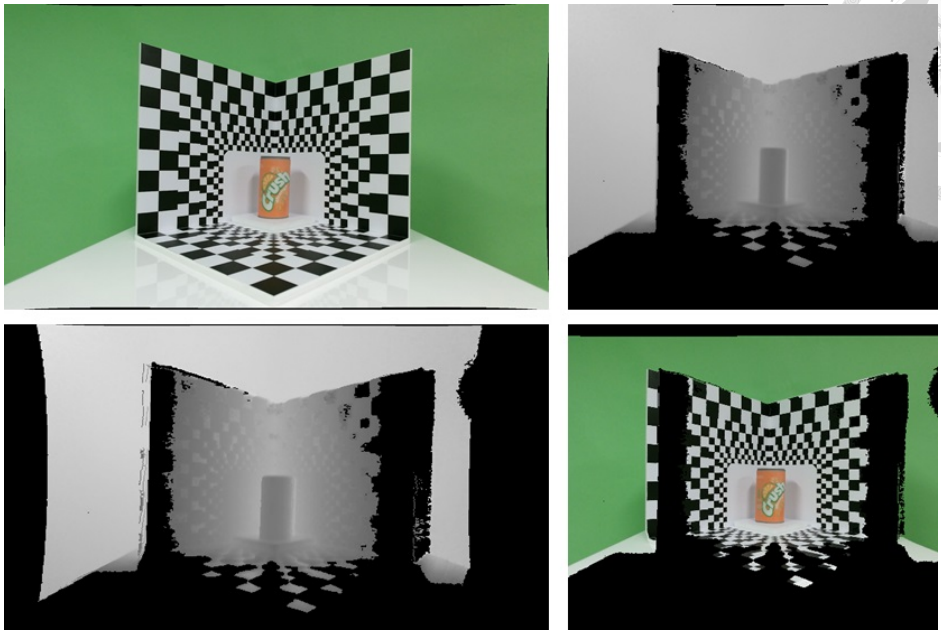


Figure 4.8: Original images (top row) and new mapped images (bottom row) in the other coordinate system.

the mapping results between two coordinate systems in Figure 4.8.

After rectifying the images, we obtain the ground-truth pose using the camera parameters and the checkerboard (or checkerbox) around an object as follows. The positions of a few crossed points are initialized with known 2D-to-3D correspondences in the first frame of each sequence and updated by the nearest corners using [169]. Other crossed points can then be obtained with an initial pose $\mathbf{p}_0$ estimated according to the correspondences with the OPnP method [13]. The location of each point is refined with a sub-pixel corner detection method [168]. A point may be discarded if it is close to another crossed point for robust pose estimation. We compute the object pose $\mathbf{p}$ according to the refined points with the OPnP method [13] again and refine $\mathbf{p}$ with the Newton method. Figure 4.9 shows an example of object pose estimation in the first frame of each sequence. We determine the corresponding points in the current and following frames with the KLT tracker [96], and estimate poses according to these points with the scheme

doi:10.6342/NTU201800854

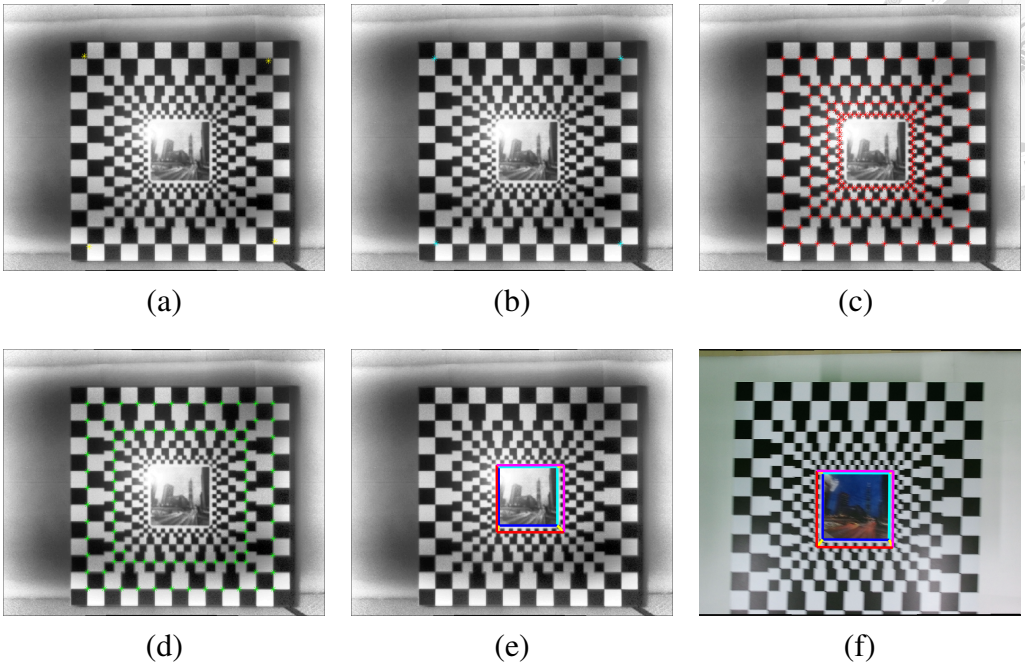


Figure 4.9: Ground-truth object pose annotation. (a) We first initialize a few points with known 2D-to-3D correspondences. (b) The nearest corner points of the initialized points are detected. (c) The other corner points are computed with an initial pose $\mathbf{p}_0$ according to the initial correspondences. (d) We later refine these points and discard non-robust ones. (e) The final pose $\mathbf{p}$ is estimated according to the remaining points. (f) The object pose in the related color image is computed according to the estimated transformation matrix.

mentioned above. As such, the object pose in each frame can be obtained sequentially. The annotation process is performed with a handcrafted program, as shown in Figure 4.10. The checkerboard (or checkerbox) is designed with increasing block size from center to border. This pattern facilitates detecting a sufficient number of corner points when the target object is either near or far from the sensor as illustrated in Figure 1.1. Because of the symmetric form, the crossed corner points can also be better localized than the corners of fiducial markers used in existing datasets [164, 5, 14, 15]. The exact 3D target position related to the base is calibrated manually.

doi:10.6342/NTU201800854

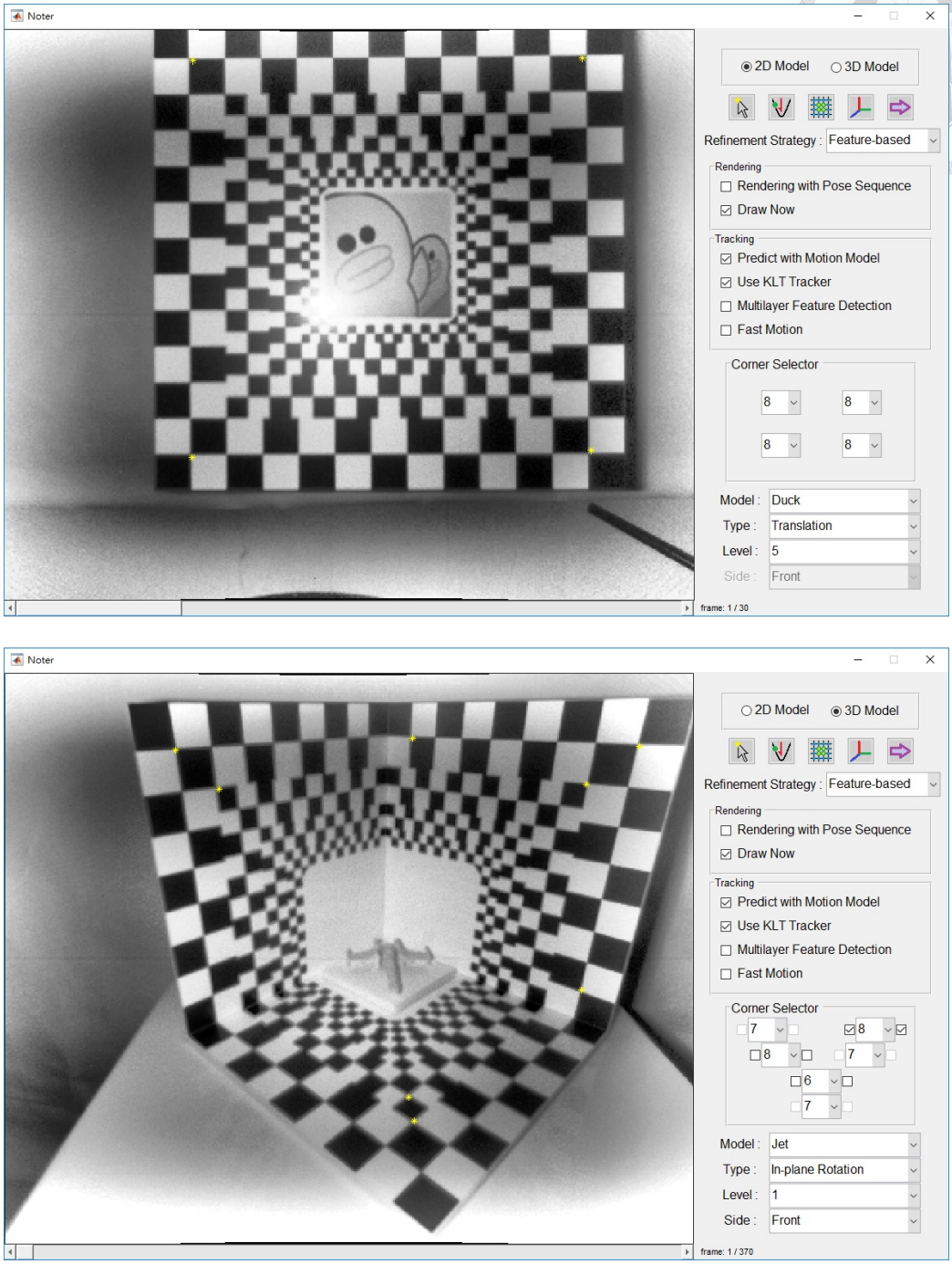
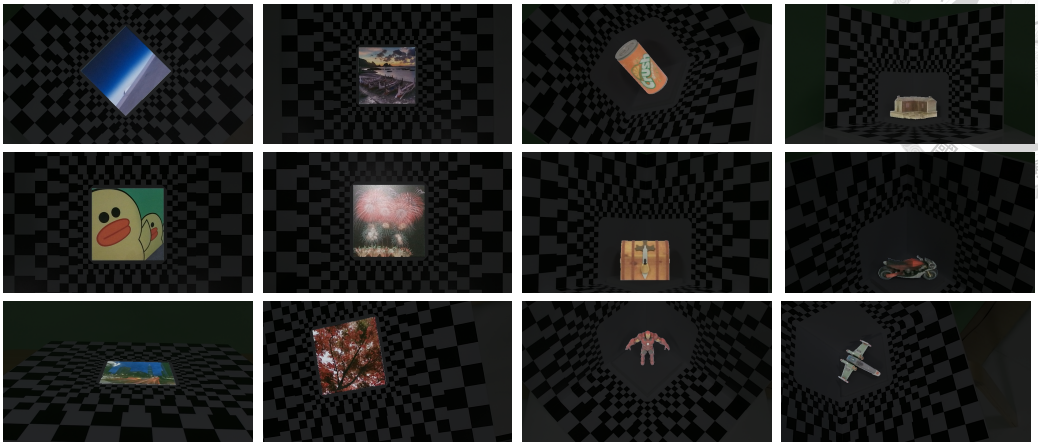


Figure 4.10: The GUI of our handcrafted program for annotating the ground-truth poses of 2D target objects (top) and 3D target objects (bottom). To establish the 2D-to-3D correspondences before applying a PnP algorithm, we first specify some 3D points by the *corner selector* panel and then mark the corresponding 2D points on the image.

doi:10.6342/NTU201800854

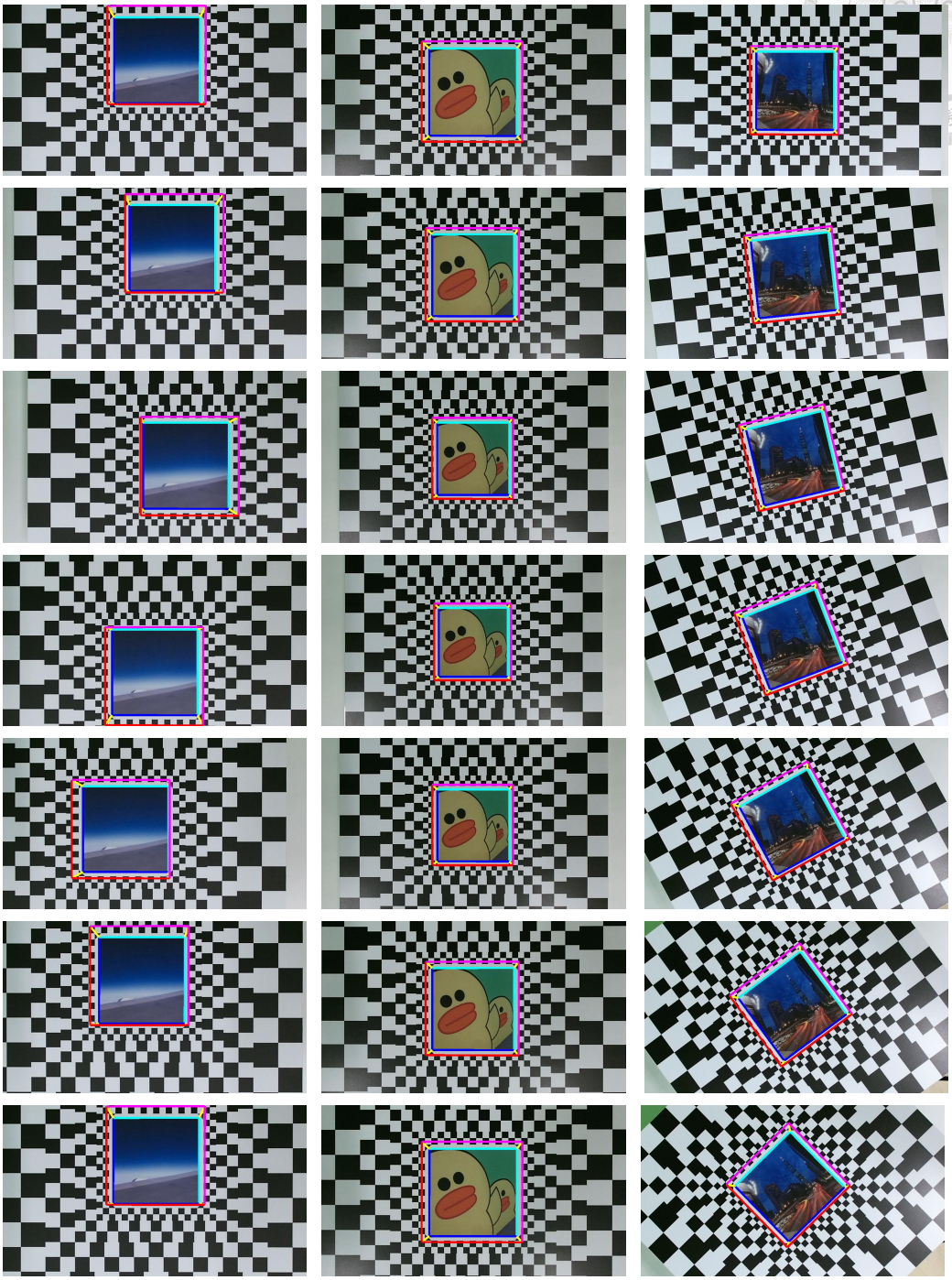


(a) 2D Models (b) 3D Models

Figure 4.11: Camera frames for (a) 2D and (b) 3D objects blended with masks. The mask is generated using the corresponding pose and the 3D models.

We use the infrared images to obtain the ground-truth object pose $\mathbf{p}$ instead of using color images which can be distorted due to the rolling-shutter effect, as the skewed image illustrated in Figure 4.9(f). Furthermore, the exposure time of the infrared camera is much shorter such that infrared images contain less motion blur. The object pose in the color images of the first and following frames are obtained by transforming $\mathbf{p}$ according to the transformation matrix $\mathbf{T}_{d2c}$. As the intensity contrast of the original infrared image is relatively low (as shown in Figure 4.1), the images shown in Figure 4.9(a)–(e) are processed with a tone mapping algorithm for presentation purpose. In addition, we generate the mask image related to each frame according to the ground-truth pose, as illustrated in Figure 4.11. These mask images are used for cropping target templates for the training purpose. Video sequences with annotated poses of different motion patterns are shown in Figure 4.12 to Figure 4.19. For sequences with 3D objects, we present the annotated ground-truth pose by rendering monochrome semi-transparent wire-frame models onto camera frames. Regarding the *Flashing Light* motion pattern, the upper and lower regions of some frames have different intensity values due to the rolling-shutter effect.

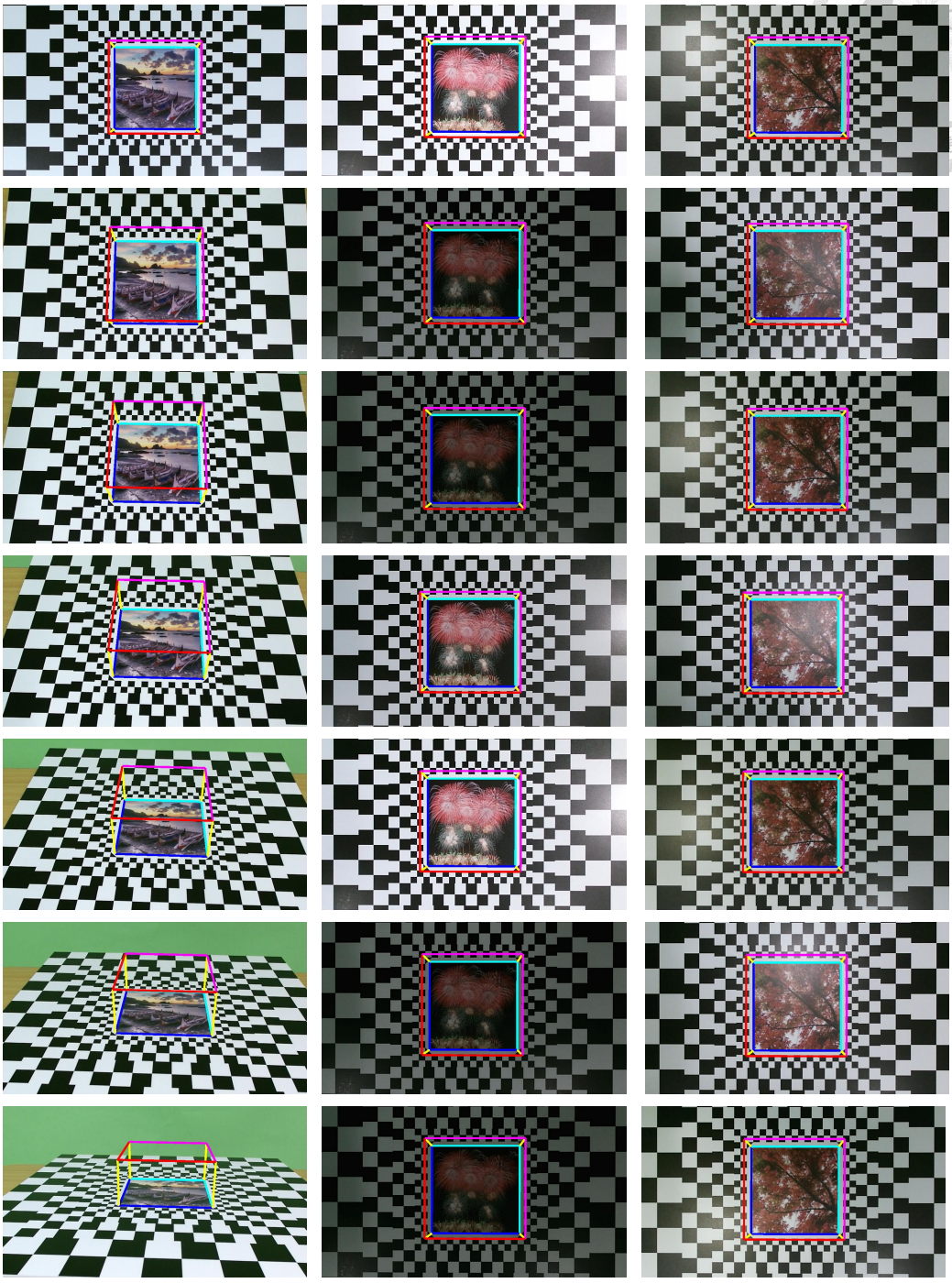
doi:10.6342/NTU201800854



(a) Translation (b) Zoom (c) In-plane Rotation

Figure 4.12: Image sequences of different motion patterns with annotated poses. (a) Translation (*Wing*). (b) Zoom (*Duck*). (c) In-plane Rotation (*City*).

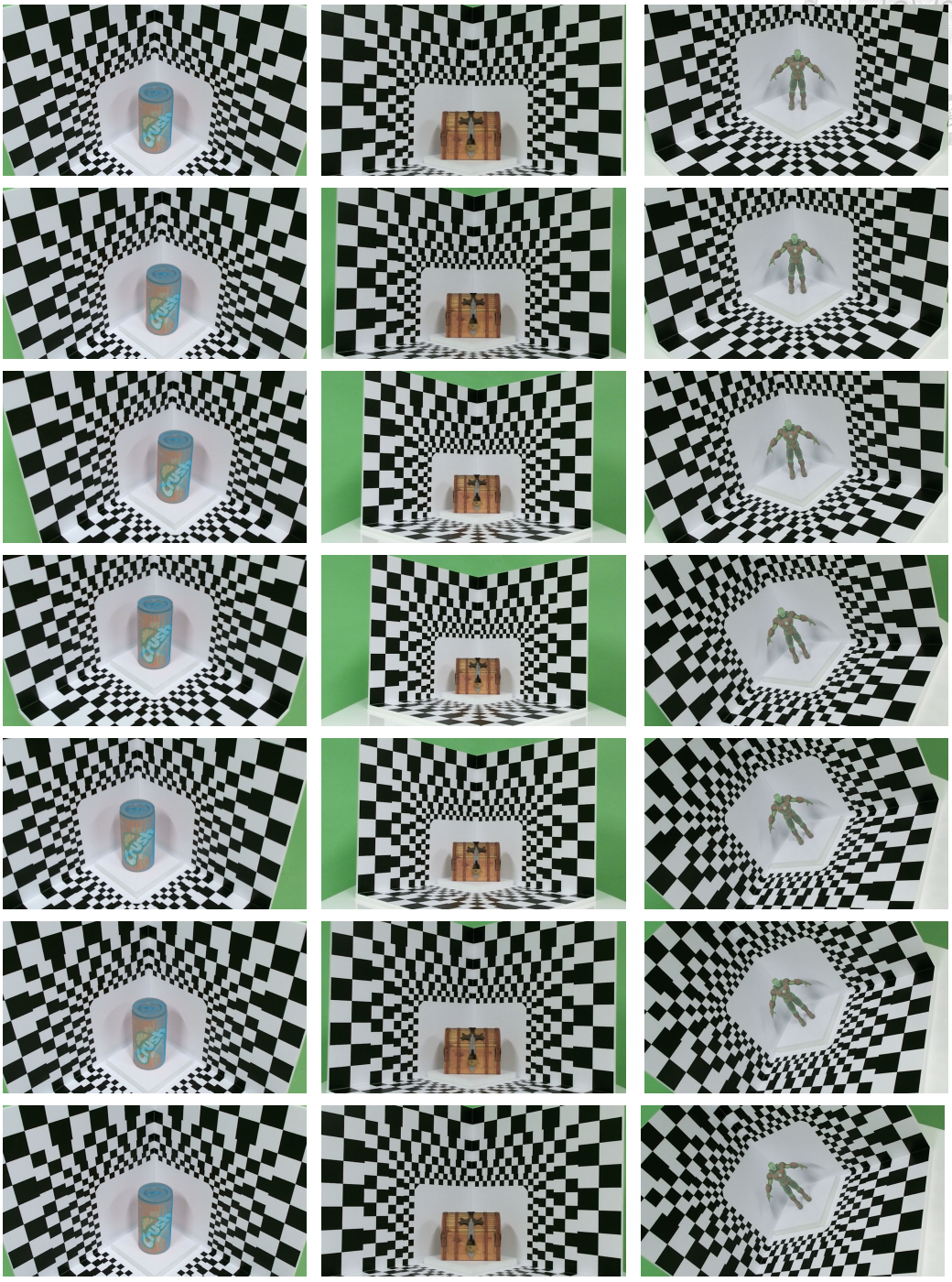
doi:10.6342/NTU201800854



(a) Out-of-plane Rotation (b) Flashing Light (c) Moving Light

Figure 4.13: Image sequences of different motion patterns with annotated poses. (a) Out-of-plane Rotation (*Beach*). (b) Flashing Light (*Firework*). (c) Moving Light (*Maple*).

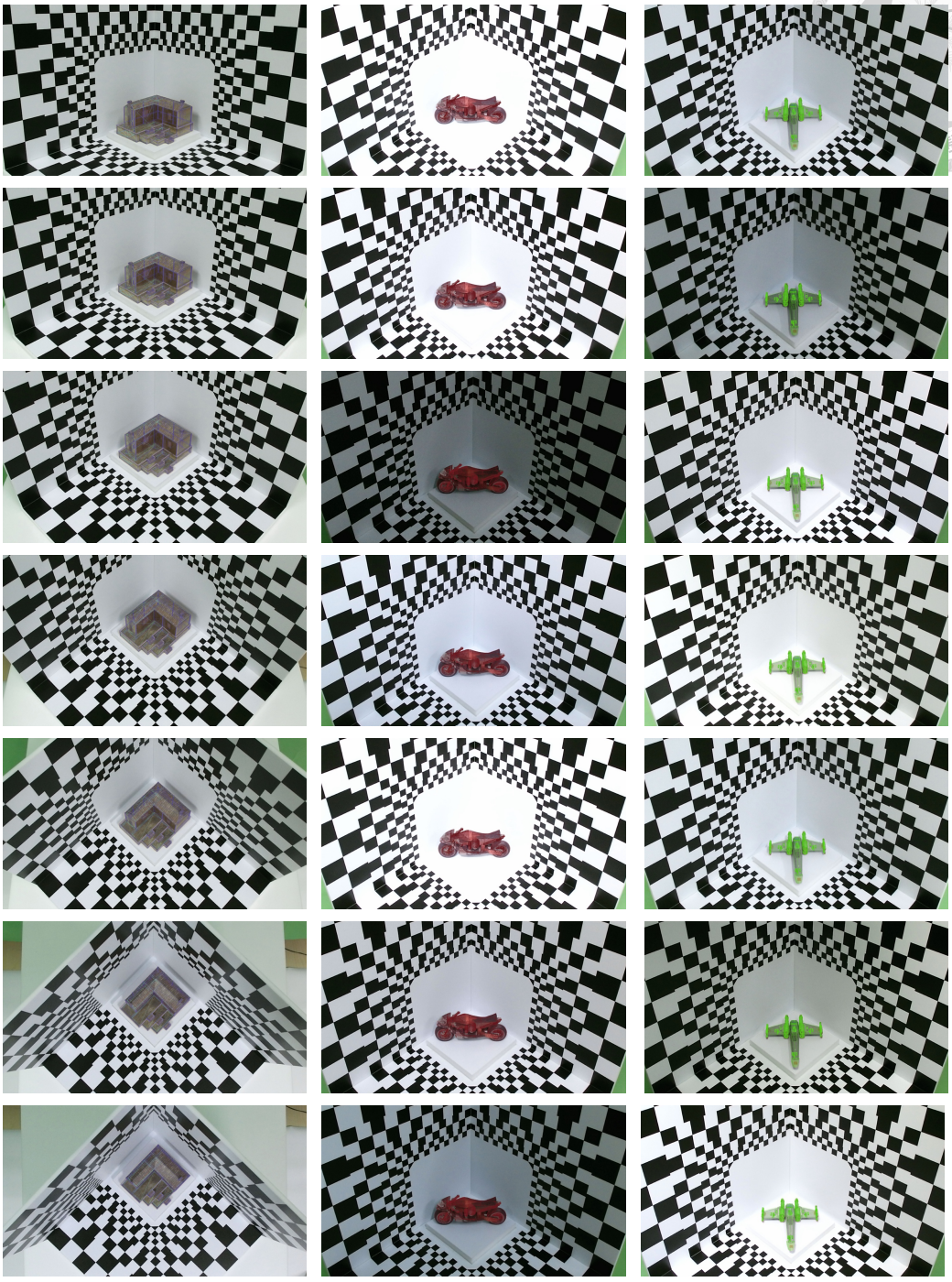
doi:10.6342/NTU201800854



(a) Translation (b) Zoom (c) In-plane Rotation

Figure 4.14: Images with wire-frame models rendered according to annotated poses. (a) Translation (*Soda*). (b) Zoom (*Chest*). (c) In-plane Rotation (*Ironman*).

doi:10.6342/NTU201800854



(a) Out-of-plane Rotation (b) Flashing Light (c) Moving Light

Figure 4.15: Images with wire-frame models rendered according to annotated poses. (a) Out-of-plane Rotation (*House*). (b) Flashing Light (*Bike*). (c) Moving Light (*Jet*).

doi:10.6342/NTU201800854

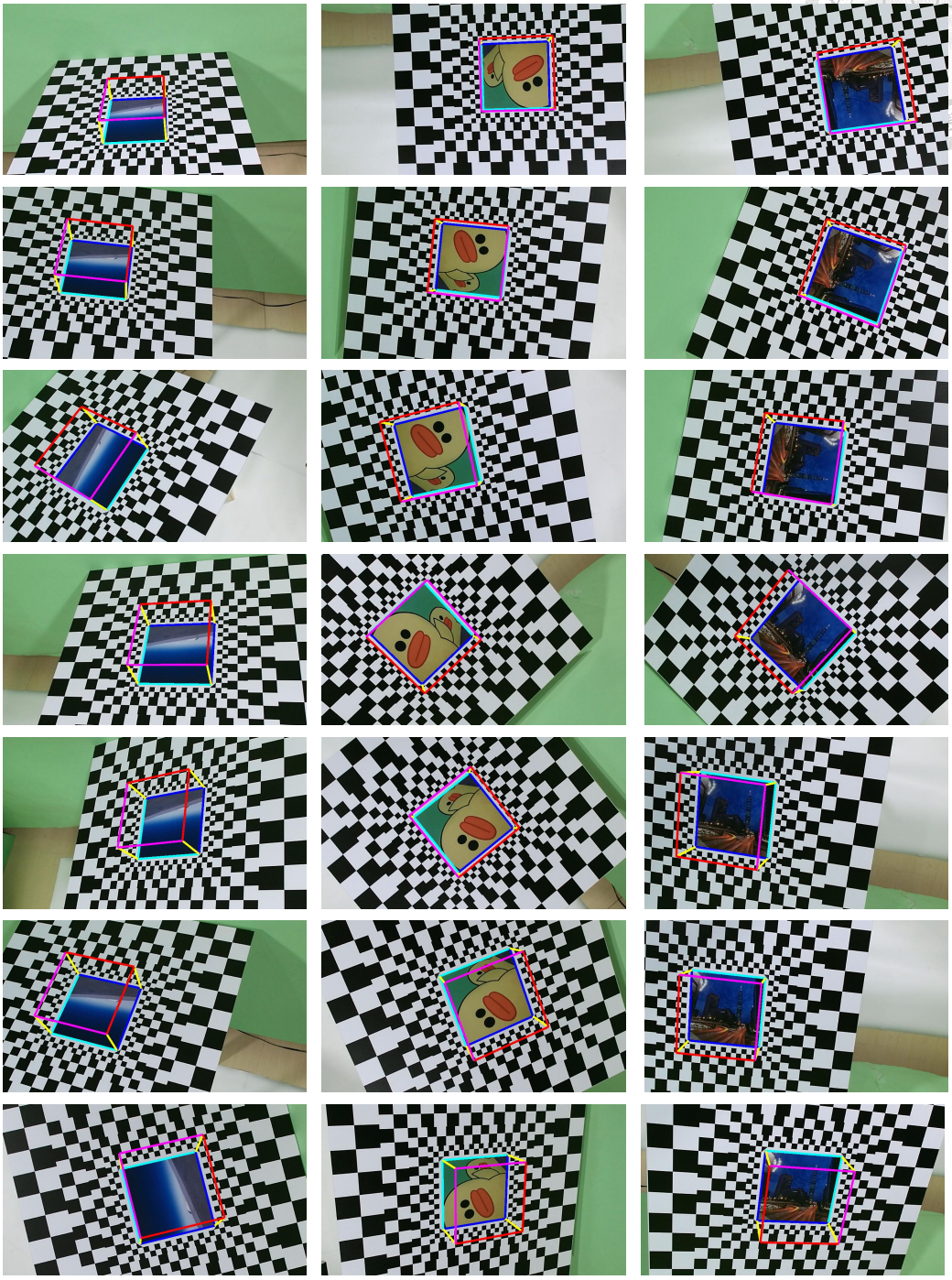


Figure 4.16: Images of motion pattern *free motion* with 2D targets. In this case, we hold the Kinect V2 device manually. These sequences are recorded with combining different motion patterns and speed levels.

doi:10.6342/NTU201800854

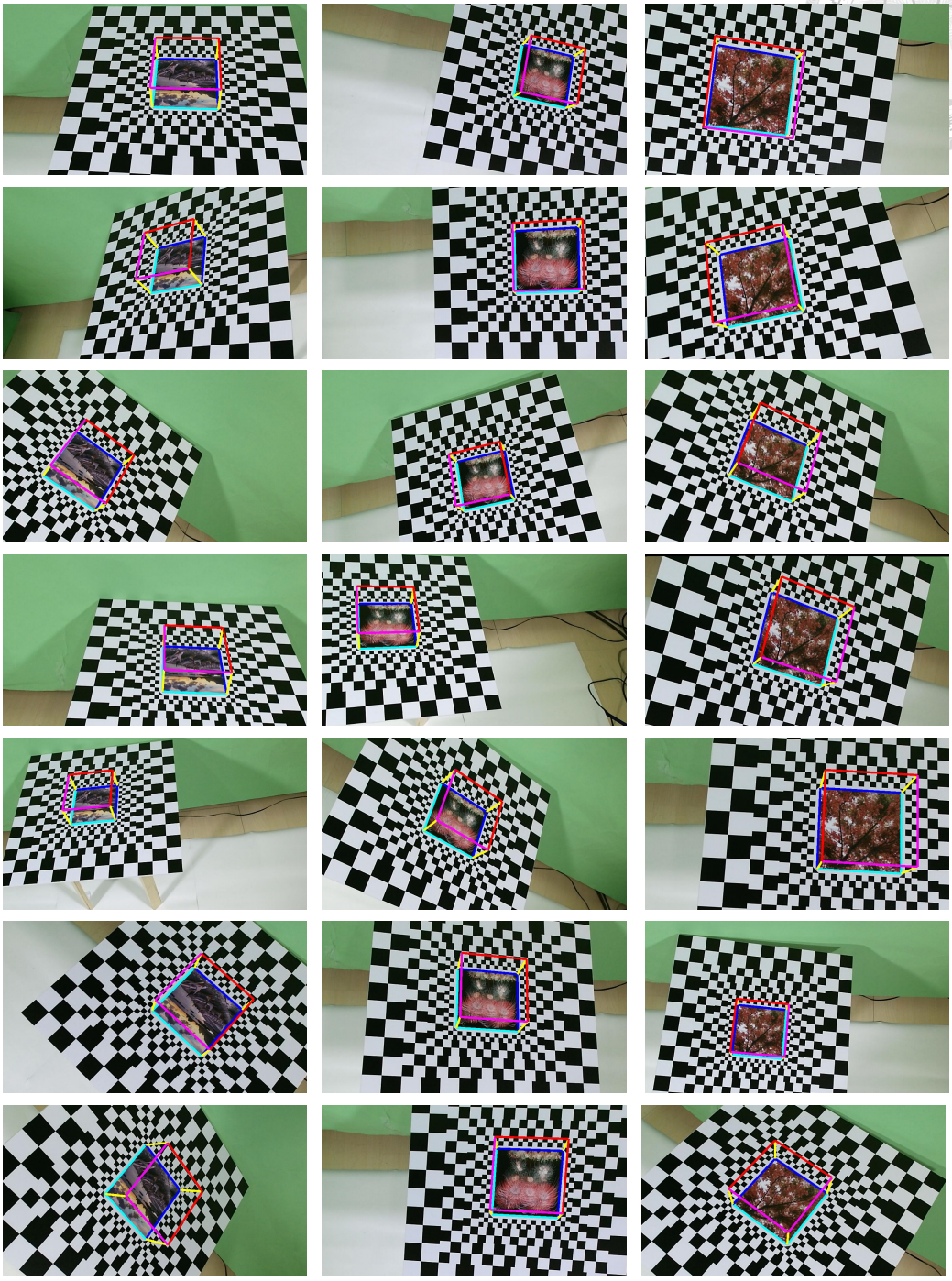


Figure 4.17: Images of motion pattern *free motion* with 2D targets. In this case, we hold the Kinect V2 device manually. These sequences are recorded with combining different motion patterns and speed levels.

doi:10.6342/NTU201800854

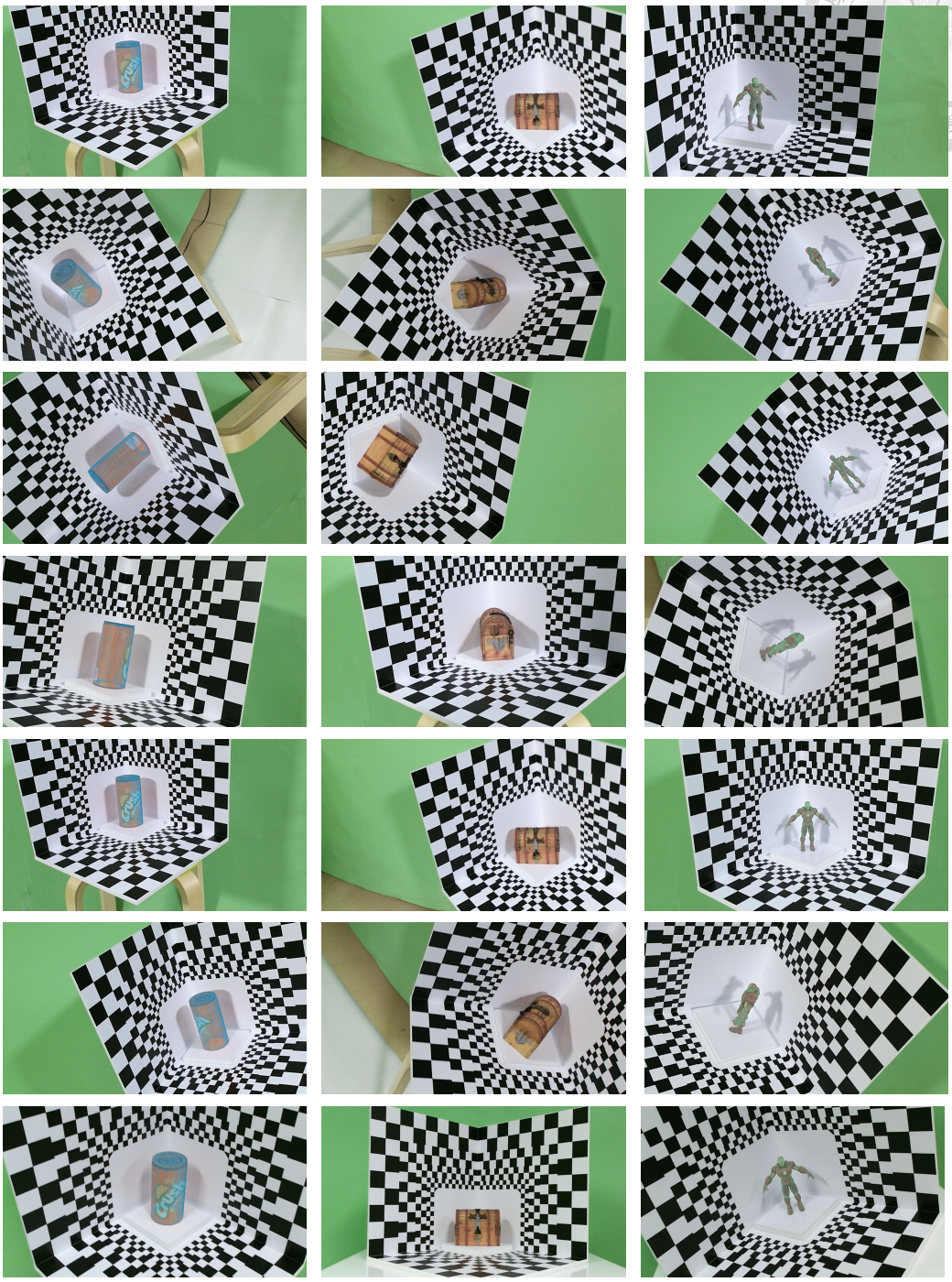


Figure 4.18: Images of motion pattern *free motion* with 3D targets. In this case, we hold the Kinect V2 device manually. These sequences are recorded with combining different motion patterns and speed levels.

doi:10.6342/NTU201800854

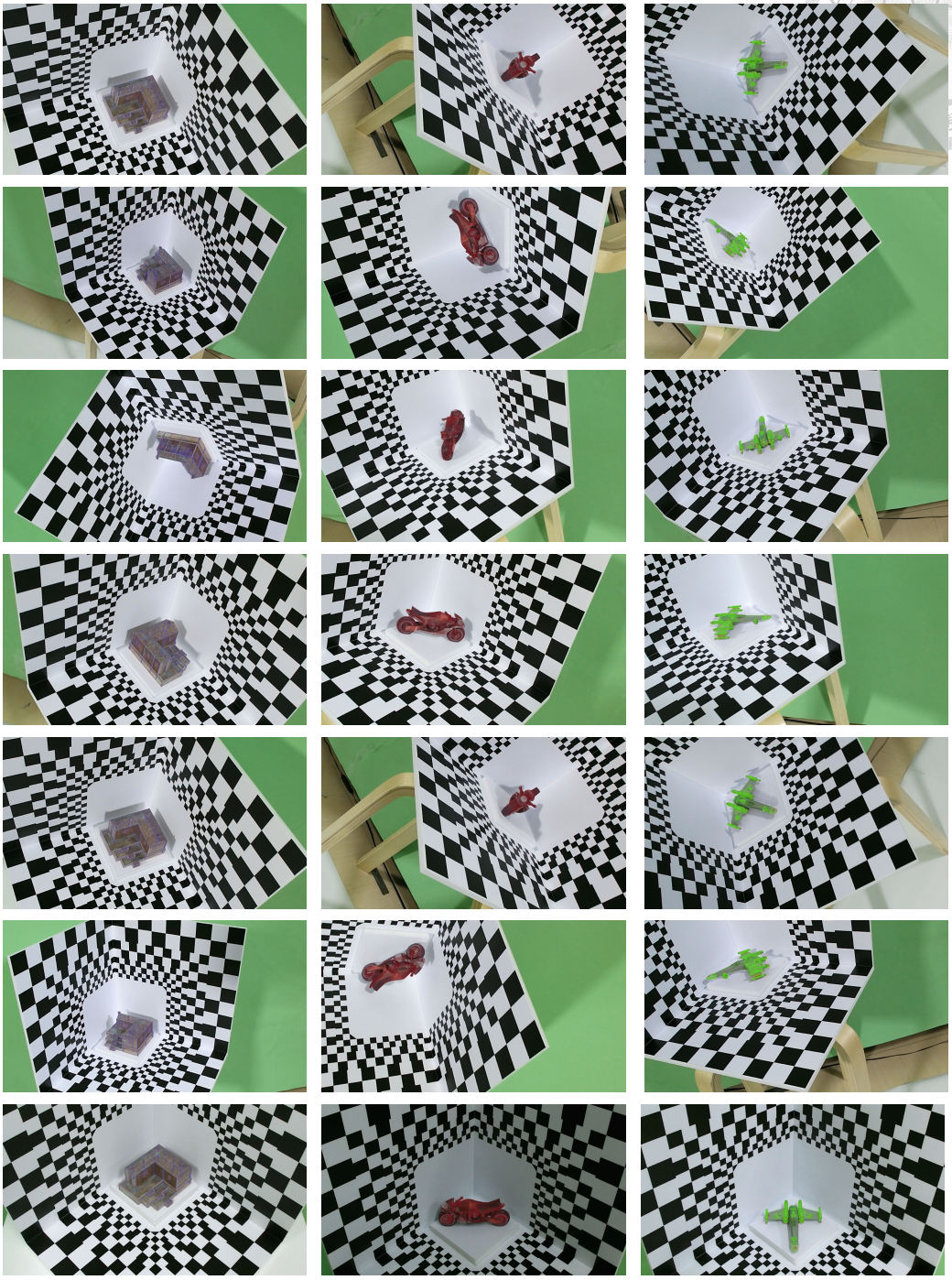


Figure 4.19: Images of motion pattern *free motion* with 3D targets. In this case, we hold the Kinect V2 device manually. These sequences are recorded with combining different motion patterns and speed levels.

doi:10.6342/NTU201800854

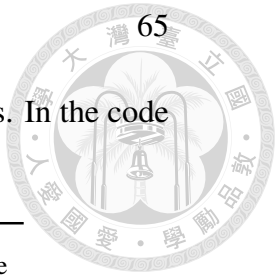


Table 4.3: Evaluated algorithms. Run time is measured in seconds. In the code column, C: C/C++, M: Matlab, CU: CUDA.

Algorithm	Description	Code	Run Time
SIFT [9]	Feature detector	C, M	5.287 s
ASIFT [10]	Feature detector	C, M	50.995 s
OPnP [13]	PnP algorithm	M	0.156 s
IPPE [135]	PnP algorithm	M	0.044 s
DPE16 [3]	Pose estimator (2D)	C, M, CU	4.811 s
UDP [16]	Pose estimator (3D)	C	9.262 s
PWP3D [19]	Pose tracker (3D)	C, CU	0.066 s
OS2 [2]	SLAM approach (sparse)	C	0.067 s
EF [60]	SLAM approach (dense)	C, CU	0.078 s

4.3 Evaluation Methodology

In this work, we evaluate pose tracking algorithms for both 2D and 3D target objects. Table 4.3 lists the main characteristics of the algorithms being assessed. We further explain our evaluation metrics in Section 4.3.2.

4.3.1 Evaluation Algorithms

To estimate the pose of a planar target, we look into feature-based approaches, and evaluate algorithms with a combination of two feature detectors (i.e., SIFT [9] and ASIFT [10]) and two PnP algorithms (i.e., OPnP [13] and IPPE [135]) for pose estimation. Note that the IPPE approach is actually a homography decomposition method as it takes the coefficients of a homography as inputs, whereas the homography can be obtained from the 2D-to-3D correspondences by using the Direct Linear Transform (DLT) [170] algorithm. Therefore, we still regard it as a PnP approach used for planar objects. The RANSAC-based schemes [11] are then applied to

doi:10.6342/NTU201800854

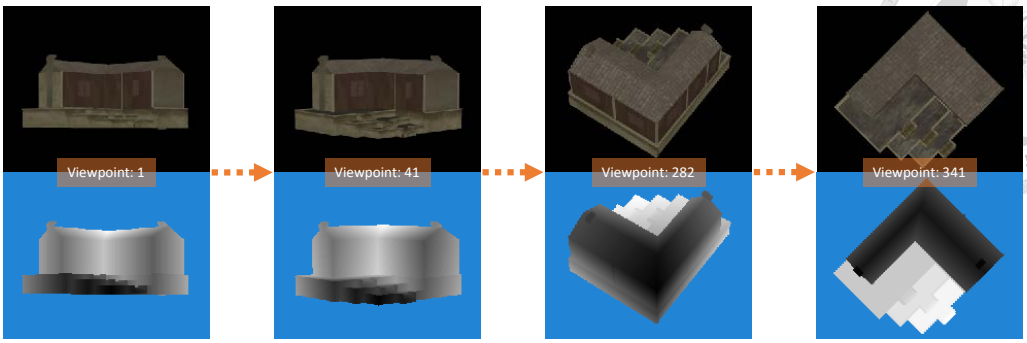


Figure 4.20: Synthetic frames rendered from 341 viewpoints on half of the recursively divided icosahedron

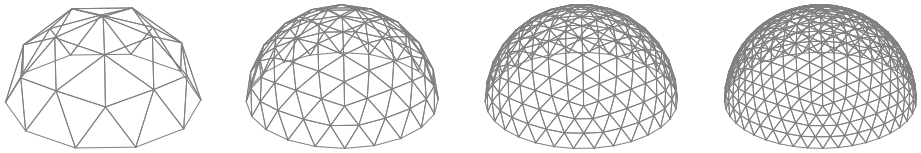
these feature-based methods to remove incorrect feature correspondences. We implement a CUDA-based direct pose estimator, which is ten times faster than the original DPE16 approach [3] with equivalent accuracy.

To recover 3D object poses, we evaluate two state-of-the-art approaches (i.e., UDP [16] and PWP3D [19]) for pose estimation and pose tracking. We note numerous camera pose trackers have been released recently that achieve real-time performance by leveraging the reconstructed environment maps. Two state-of-the-art approaches in this field (i.e., ORB-SLAM2 [2] and ElasticFusion [60]) are used for evaluation. The ORB-SLAM2 method tracks camera poses based on sparse features, and the ElasticFusion scheme solves a minimization problem based on intensity and depth values. These camera pose trackers are evaluated by deactivating them within background regions of a video sequence. Foreground and background regions are separated according to the geometric model and related ground-truth pose, as illustrated in Figure 4.11.

For each feature-based approach (including the ORB-SLAM2 method), the average number of detected features in a camera frame is set to be around 3,000.

For SLAM-based approaches, we construct a 3D map of the target object for evaluation. Each map is constructed with synthetic frames created by rendering a mesh from 341 viewpoints on one half of a recursively divided icosahedron, as shown in Figure 4.20. The half of recursively divided icosahedron is illustrated

doi:10.6342/NTU201800854



(a) 26 vertices (b) 91 vertices (c) 196 vertices (d) 341 vertices

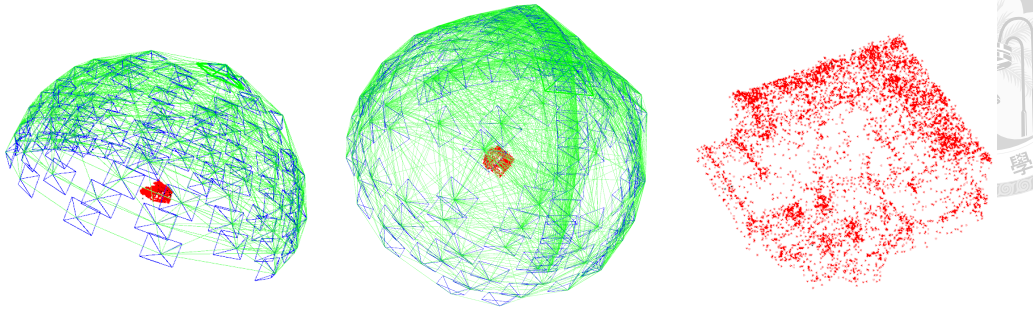
Figure 4.21: Half of the recursively divided (from left to right) icosahedron.

in Figure 4.21. ORB-SLAM2 uses a set of feature points to represent its environment map and performs relocalization by feature matching from between this map and the input camera frame. We find the relocalization process in ORB-SLAM2 system can be further accelerated by leveraging real camera frames, and therefore we add frames of motion pattern *Out-of-plane Rotation* (whose camera trajectory is like a cross) into the mapping process, as illustrated in Figure 4.22. Consequently, it bootstraps (i.e., estimate the object pose) in the beginning of each sequence. On the contrary, the ElasticFusion approach builds its map with dense surfels, as shown in Figure 4.23. It instead uses randomized ferns to perform relocalization [171]. However, it does not work well in our experiments. Therefore it needs a guided pose in the first frame. For both SLAM approaches, all maps keep unchanged (i.e., no points for surfels will be further added) during the tracking process. We experimentally set the ICP weight of the joint cost function to be 0.5 for better tracking results in the ElasticFusion approach. As the SLAM approaches are able to deal with 2D cases, we also evaluate these methods with 2D objects.

Since the UDP method does not perform well if it is trained on synthetic images (as discussed in [16] and confirmed in our comparative study), we select about 10% of the images from the proposed dataset as the training data for UDP.

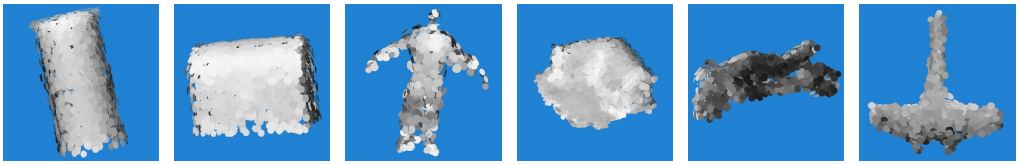
In the PWP3D method, we set foreground and background distributions to be unchanged (i.e., use the distributions from the first frame for all frames in the sequence) as it can achieve better tracking results. The mask image of the first camera frame is also used for the PWP3D scheme to set the color distribution of

doi:10.6342/NTU201800854



(a) (b) (c)

Figure 4.22: Model maps generated by the ORB-SLAM2 method. (a) This map is built with synthetic frames created by rendering mesh from 341 viewpoints on half of the recursively divided icosahedron. The green wire-frame model, blue line, and red point stand for cameras, correlations between cameras, and detected feature point, respectively. We refer the reader to [2] for more details. (b) To accelerate the relocalization process, we further exploit the real captured frames to build the model map. (c) The feature-based model map produced by the ORB-SLAM2 method.



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

Figure 4.23: The surfel-based models: (a) *Soda*, (b) *Chest*, (c) *Ironman*, (d) *House*, (e) *Bike*, and (f) *Jet* generated by the mapping process of the ElasticFusion method.

foreground and background regions.

For the iterative energy minimization schemes (i.e., PWP3D and ElasticFusion), the ground-truth pose in the first frame is provided and object pose tracking is performed subsequently. To fairly compare different approaches, the result of the first frame in each video sequence is not considered.

doi:10.6342/NTU201800854

4.3.2 Evaluation Metrics

Given the ground-truth rotation matrix $\tilde{\mathbf{R}}$ and translation vector $\tilde{\mathbf{t}}$, we compute the error of the estimated pose ($\mathbf{R}$, $\mathbf{t}$) by using the *3D Distance* metric (2.37) presented in Section 2.2.2. For a 2D object, we define the model points as vertices of a bounding box, whose height is half of its side length, as illustrated in Figure 1.1. The pose is considered to be successfully estimated if E_{3D} is less than kd where d is the diameter (i.e., the longest distance between vertices) of the target object and k is a pre-defined threshold. As the precision plot has been commonly adopted to measure the overall tracking performance recently [172], we evaluate a method by the percentage of frames with correct estimations under different values of k in a precision plot. A method with a higher area under curve (AUC) scores achieves better pose estimation results.

4.4 Evaluation Results

All the experiments are carried out on a machine with an Intel Core i7-6700K processor, 32 GB RAM, and an NVIDIA GTX 960 GPU. The RGB-D video frame size is 1920×1080. Each approach for 2D and 3D target objects is evaluated on 20,988 images and 79,968 images, respectively. The iterative energy minimization approaches (i.e., ElasticFusion and PWP3D) tend to lose track of all frames once the matching baseline is too wide. We thus evaluate the ElasticFusion+ and PWP3D+ methods (variants of ElasticFusion and PWP3D) by feeding the ground-truth pose in the previous frame for re-initialization when a failure occurs which is determined by visual inspection.

4.4.1 Overall Performance

The overall experimental results are shown in Figure 4.24 and Figure 4.25. The maximum coefficient k defined in (2.37) is set to 0.2 in the plots, with AUC scores ranging from 0 to 20.

doi:10.6342/NTU201800854

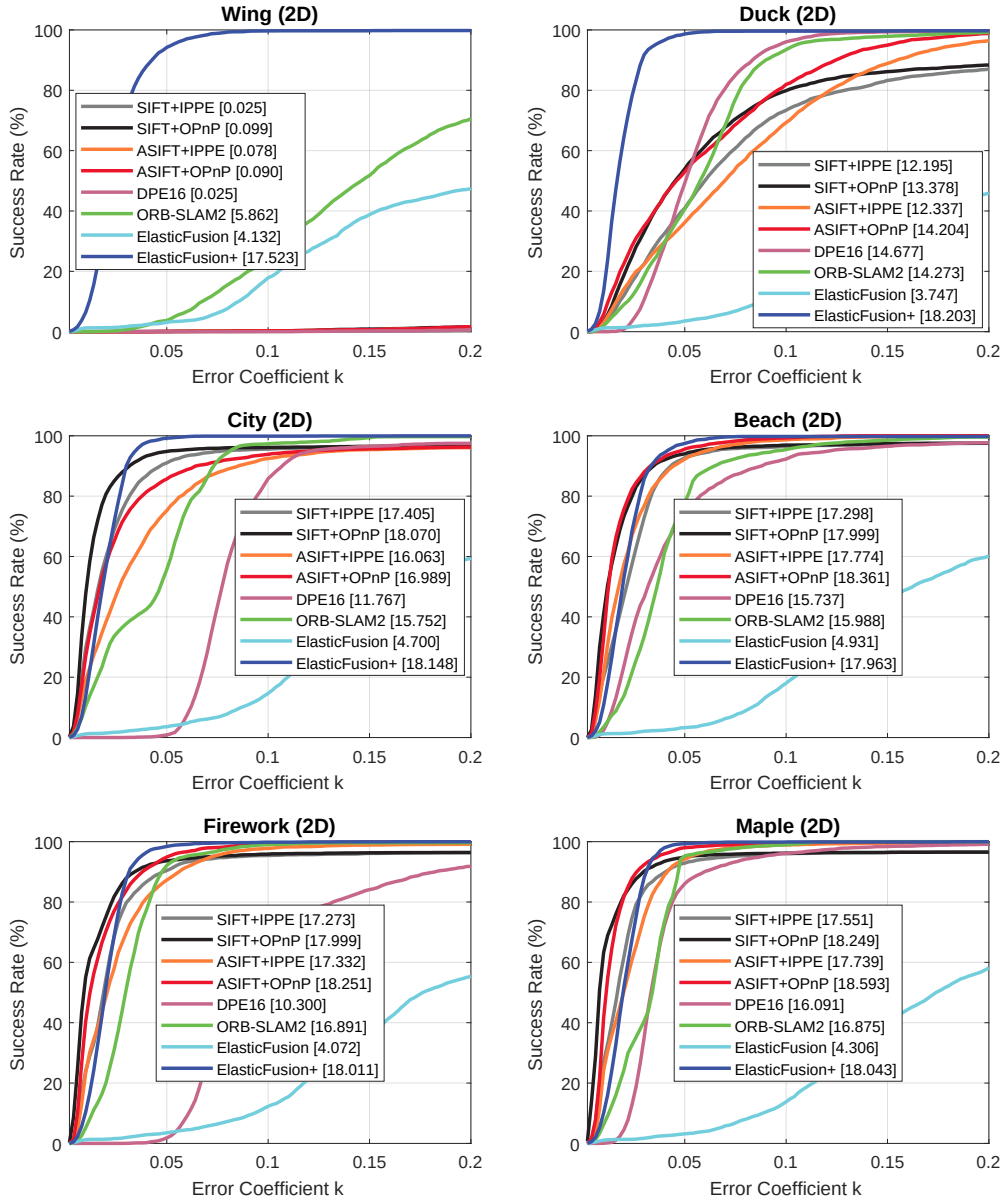


Figure 4.24: Overall performance for 2D objects on the proposed benchmark dataset. The AUC score for each approach is shown in the legend.

2D objects. The average score of tracking the *Wing* sequence is lower than the others since the target object contains less texture or structure. There exist many ambiguous pose candidates that cannot be distinguished by all evaluated approaches as the corresponding cost values are similar. In contrast, although the object in the *Duck* sequence does not contain much texture, the DPE16 method is able to

doi:10.6342/NTU201800854

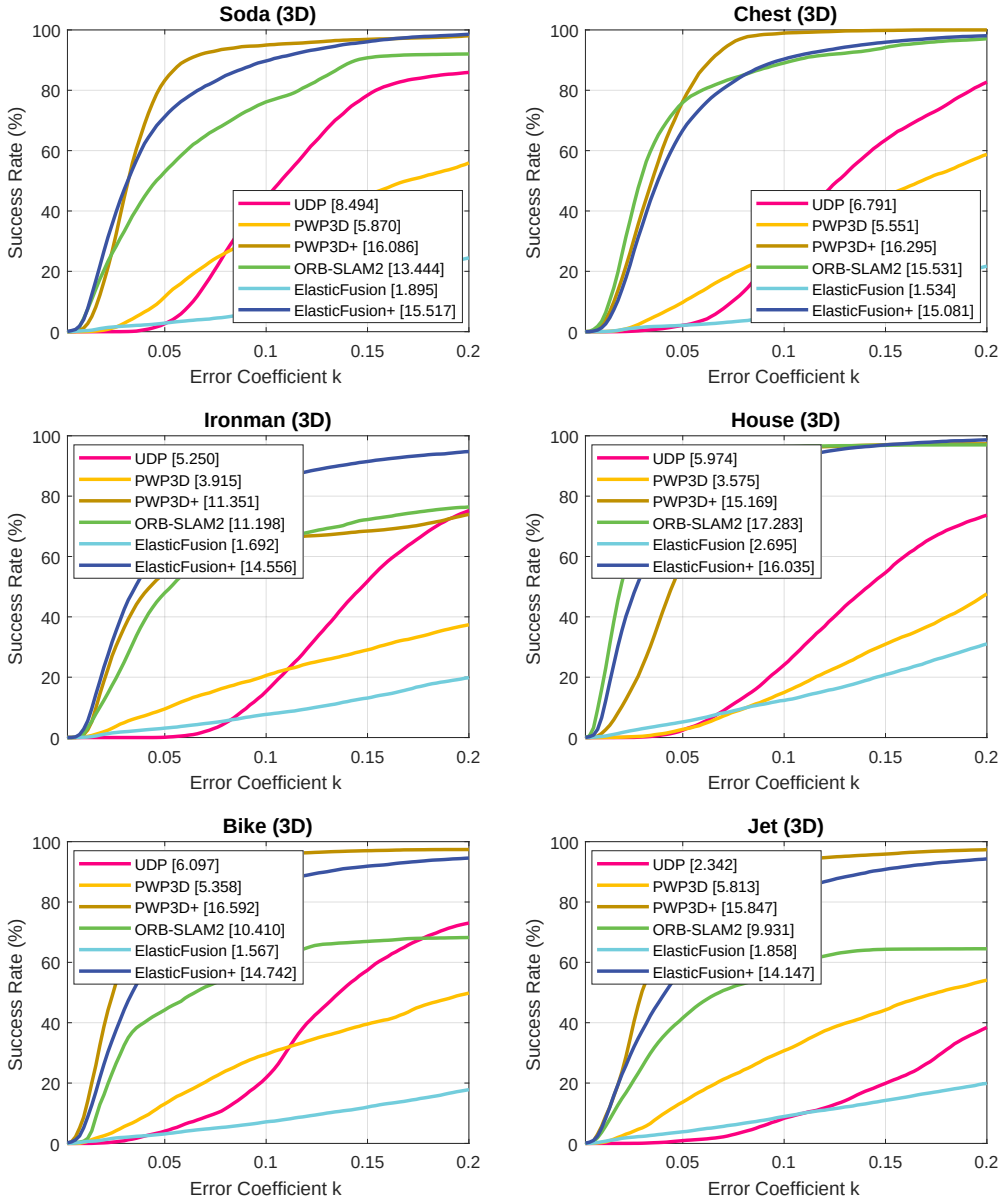
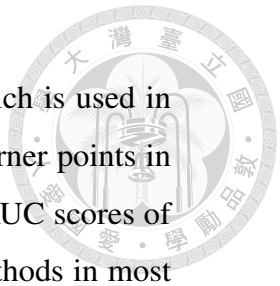


Figure 4.25: Overall performance for 3D objects on the proposed benchmark dataset. The AUC score for each approach is shown in the legend.

estimate poses well based on the distinct contour. The feature-based schemes outperform direct methods when a sufficient number of features can be extracted from a target object, as shown in the other four cases.

Despite the IPPE algorithm is designed for pose estimation of planar objects, it does not perform as well as the OPnP algorithm that is able to estimate pose

doi:10.6342/NTU201800854



in more general scenarios. As the FAST-based detector [76], which is used in the ORB-SLAM2 method, is designed for efficiently detecting corner points in an image, it does not localize features accurately. Therefore the AUC scores of the ORB-SLAM2 method are lower than those of SIFT-based methods in most cases. It is worth noticing that the ORB-SLAM2 method performs well based on the feature-based scheme as it achieves wide baseline matching, which prevents the tracker from getting stuck in local minimum. In contrast, the ElasticFusion method tends to lose track of the target object when the initial pose is not accurate since the energy minimization scheme is sensitive to perturbation caused by the introduced distortion in this work.

3D objects. Since the tracking accuracy and area of an object within one frame are in positive correlation, most approaches achieve better performance on tracking the *Soda*, *Chest*, and *House* sequences. Similar to tracking 2D objects, methods with energy minimization scheme do not perform well on the 3D dataset. However, they also show the ability to refine poses under the short-baseline conditions. We note that although the AUC scores of the ElasticFusion+ and PWP3D+ methods seem to be higher than the other approaches, it does not mean that they outperform others because their tasks are significantly simplified as the ground truth of the previous pose is given when a failure occurs. As the UDP algorithm does not have any further pose refinement scheme, the estimated pose accuracy is not as high as the other approaches. Both PWP3D and ElasticFusion methods are prone to losing track of the target when its appearance changes drastically.

4.4.2 Performance Analysis by Attributes

In this section, we show experimental results for each method with respect to different lighting and movement conditions.

2D objects. We present the pose tracking results under two different lighting conditions and freestyle condition movements in Table 4.4. As both ORB [80]

doi:10.6342/NTU201800854

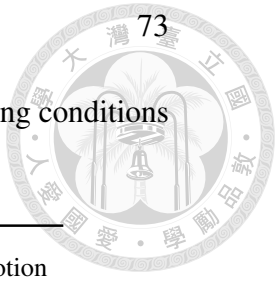


Table 4.4: AUC scores of evaluated approaches in the dynamic lighting conditions and the freestyle motion conditions.

Model	Approach	Flashing Light	Moving Light	Free Motion
2D	SIFT+IPPE	14.194	13.902	13.904
	SIFT+OPnP	15.380	15.183	14.408
	ASIFT+IPPE	13.996	13.584	12.808
	ASIFT+OPnP	15.312	14.902	13.461
	DPE16	12.996	7.516	9.793
	ORB-SLAM2	14.879	14.128	14.986
	ElasticFusion	1.974	7.479	2.948
	ElasticFusion+	16.981	18.173	18.107
3D	UDP	5.170	7.245	3.857
	PWP3D	5.084	4.907	2.890
	PWP3D+	13.071	14.434	16.041
	ORB-SLAM2	15.906	15.987	9.104
	ElasticFusion	1.444	2.005	0.278
	ElasticFusion+	14.598	12.299	10.871

and SIFT [9] are less sensitive to illumination change, the feature-based methods perform well in sequences under lighting variations. In contrast, the DPE16 algorithm does not track object poses well under different lighting conditions as the direct methods operate on the pixel values without extracting features that are designed to handle illumination changes.

The pose tracking results of 2D objects in different motion patterns and speeds are presented in Figure 4.26. Due to fast camera speeds, the recorded images in the translation case contain significant motion blur. As the feature-based approaches are not able to determine useful correspondences in blurry images, these methods do not track poses well. On the other hand, the DPE algorithm performs well with

doi:10.6342/NTU201800854

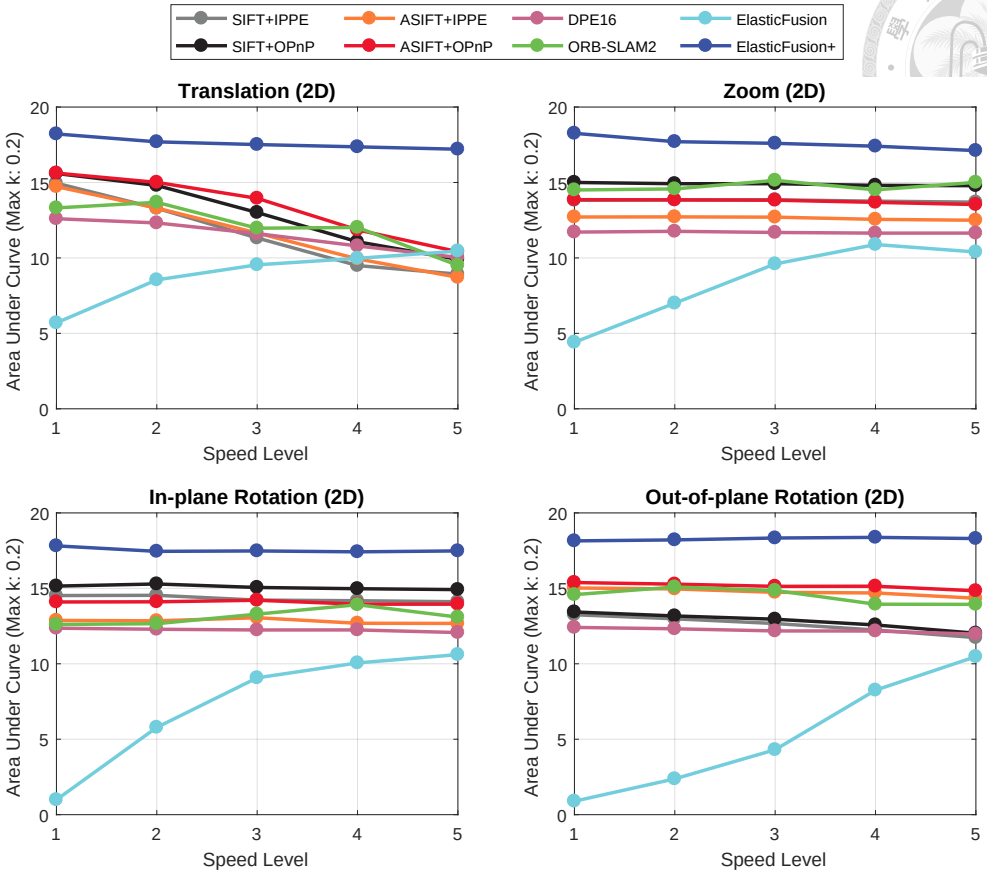


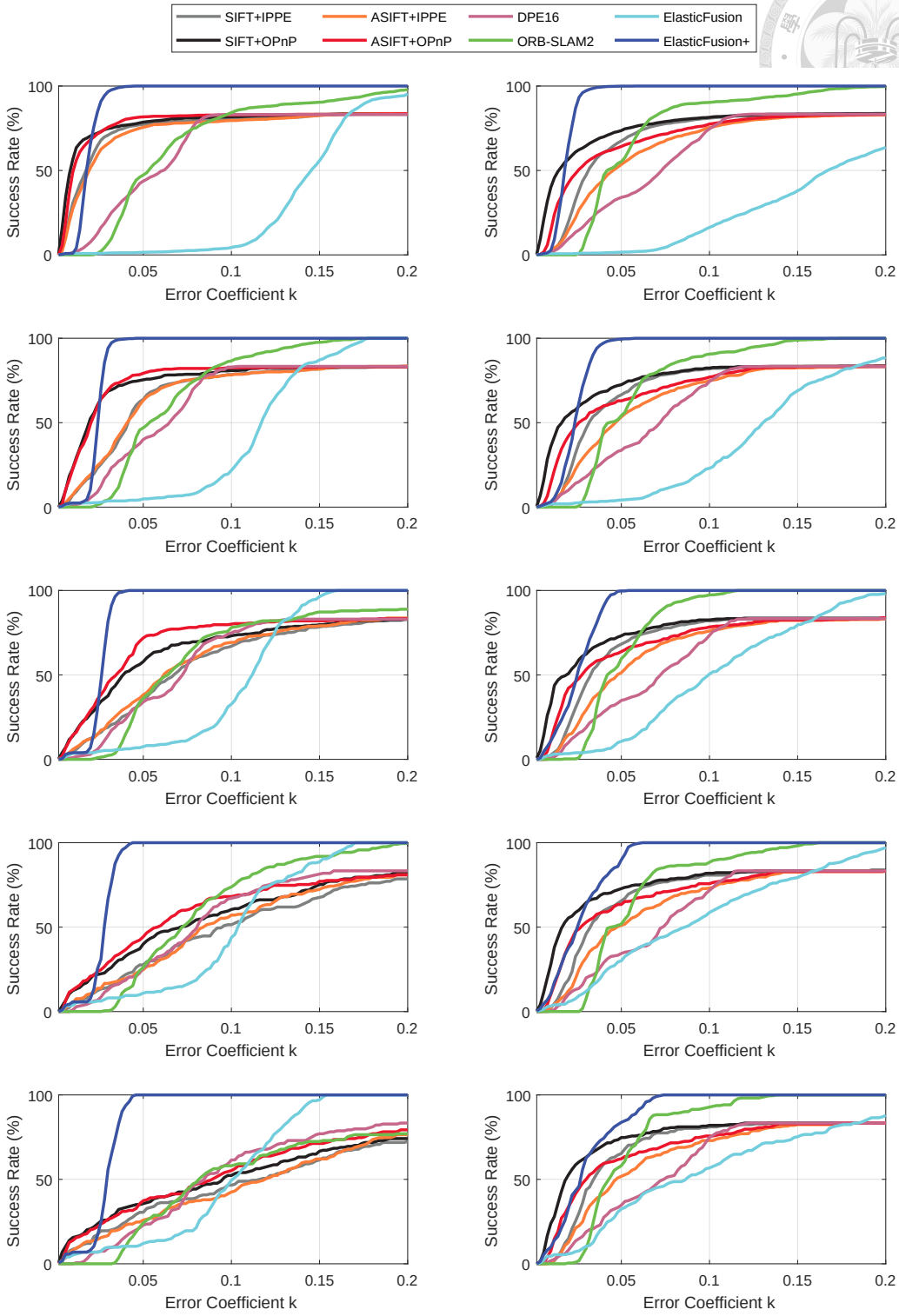
Figure 4.26: Performance by attributes with different speeds for 2D objects on the proposed benchmark dataset. Level 5 stands for the highest speed.

different camera speeds as it can handle objects with less texture.

The ASIFT algorithm outperforms other feature-based approaches in the sequences with *Out-of-plane Rotation* since it is designed to account for the affine transformation. We note the ElasticFusion method performs better at higher camera speed. This may be attributed to the fact that the decreased frame number of high-speed sequences also reduces the changes that iterative minimization approaches lose track. As in-depth analysis of this issue requires different experimental setups which are beyond the scope of this work, we will address it in future work.

The precision plots for the performance of the evaluated methods on all the attribute subsets are shown from Figure 4.27 to Figure 4.29.

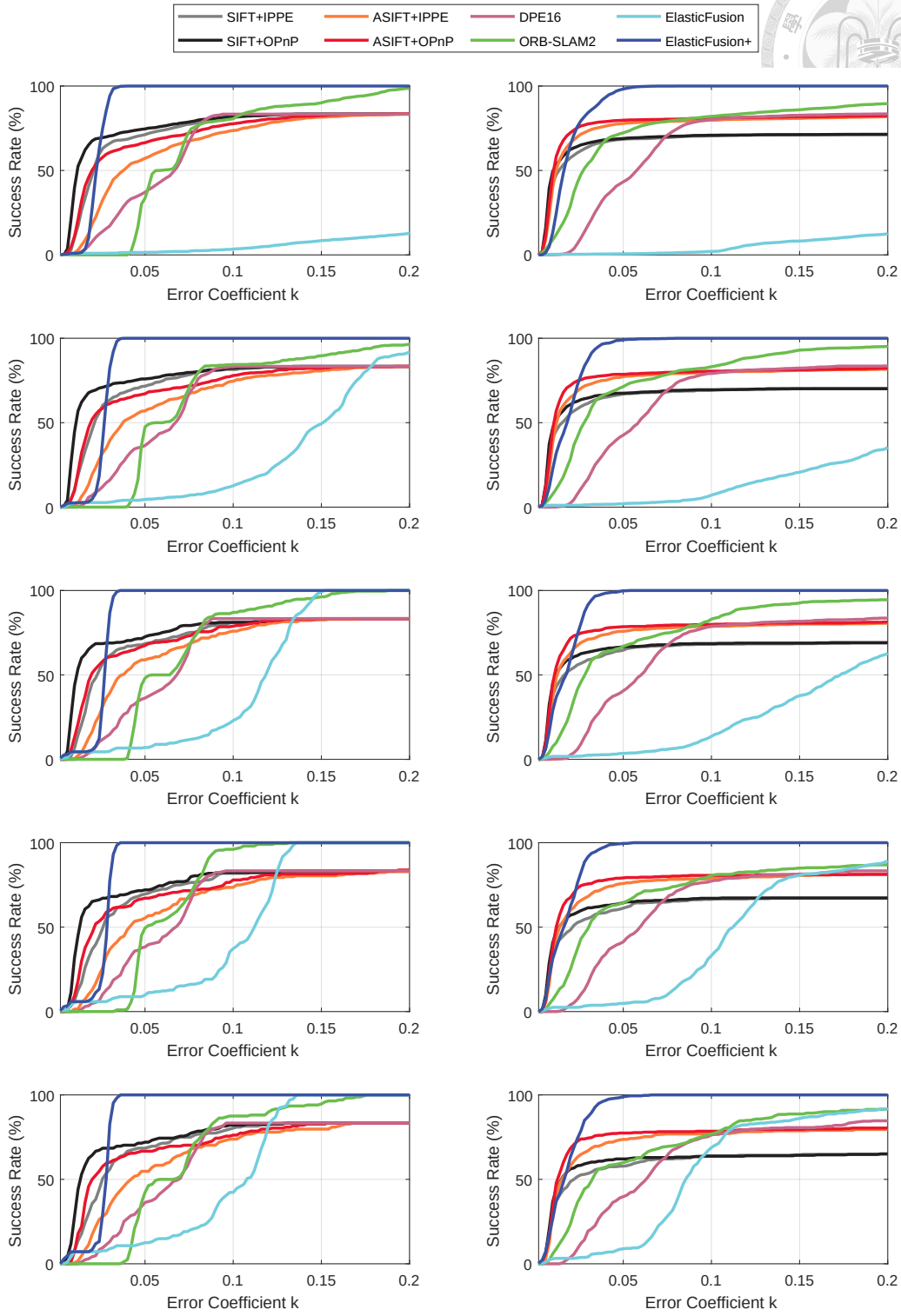
doi:10.6342/NTU201800854



(a) Translation (b) Zoom

Figure 4.27: Precision plots for 2D object (a) *Translation* and (b) *Zoom* sub-datasets. From top to bottom: lowest speed (i.e., level 1) to highest speed (i.e., level 5).

doi:10.6342/NTU201800854



(a) In-plane Rotation (b) Out-of-plane Rotation

Figure 4.28: Precision plots for 2D object (a) *In-plane Rotation* and (b) *Out-of-plane Rotation* sub-datasets. From top to bottom: lowest speed (i.e., level 1) to highest speed (i.e., level 5).

doi:10.6342/NTU201800854

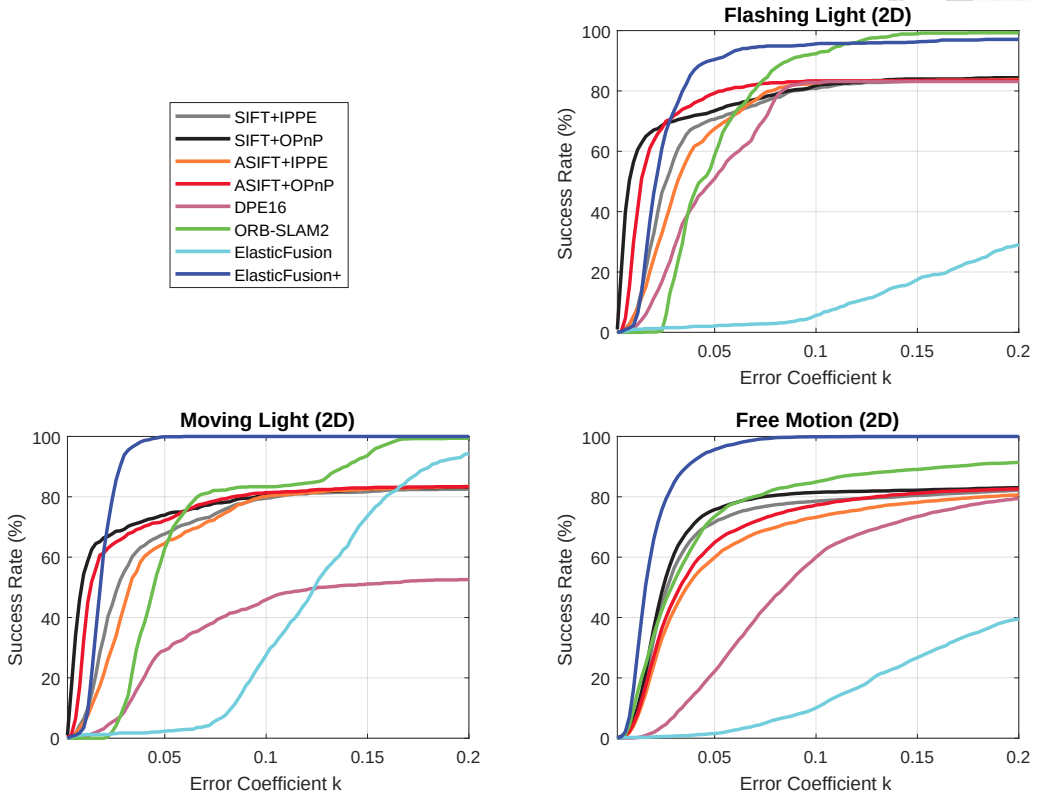


Figure 4.29: Precision plots for 2D object *Flashing Light*, *Moving Light*, and *Free Motion* sub-datasets.

3D objects. Since we only change the visible light in the experiments mentioned above with illumination variations, the depth images are not significantly affected. Compared to the pose tracking results of most approaches under standard ambient lighting, the performance difference on 3D objects is not significant. In contrast, as the PWP3D method recovers the object pose using color frames only, the pose tracking results are worse than those under normal light.

The pose tracking results of 3D targets in different motion patterns and speeds are shown in Figure 4.30. We note all approaches perform worse when the target object moves zoom in front of the camera. One reason is the size change of a target object in two consecutive frames. For the ICP-based approaches, e.g., ElasticFusion, it is difficult to align two point sets of different sizes. For the segmentation-based approaches, e.g., PWP3D, it is crucial to set a gradient step in

doi:10.6342/NTU201800854

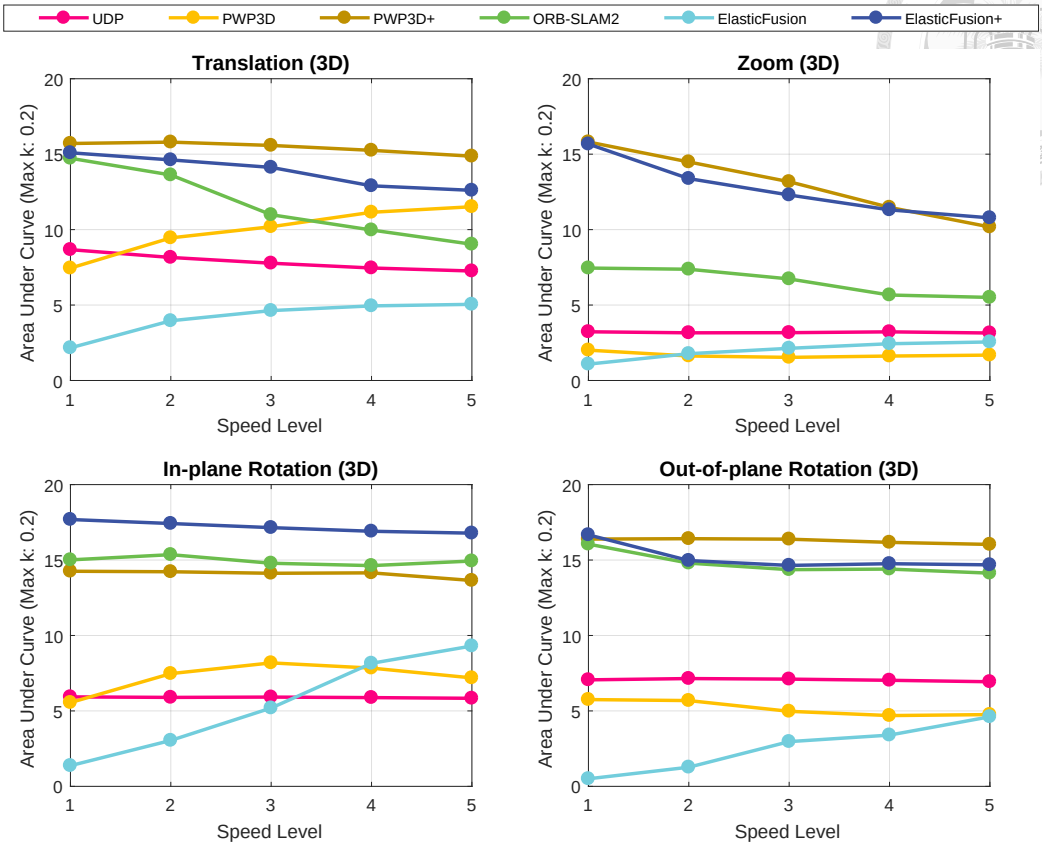


Figure 4.30: Performance by attributes with different speeds for 3D objects on the proposed benchmark dataset. Level 5 stands for the highest speed.

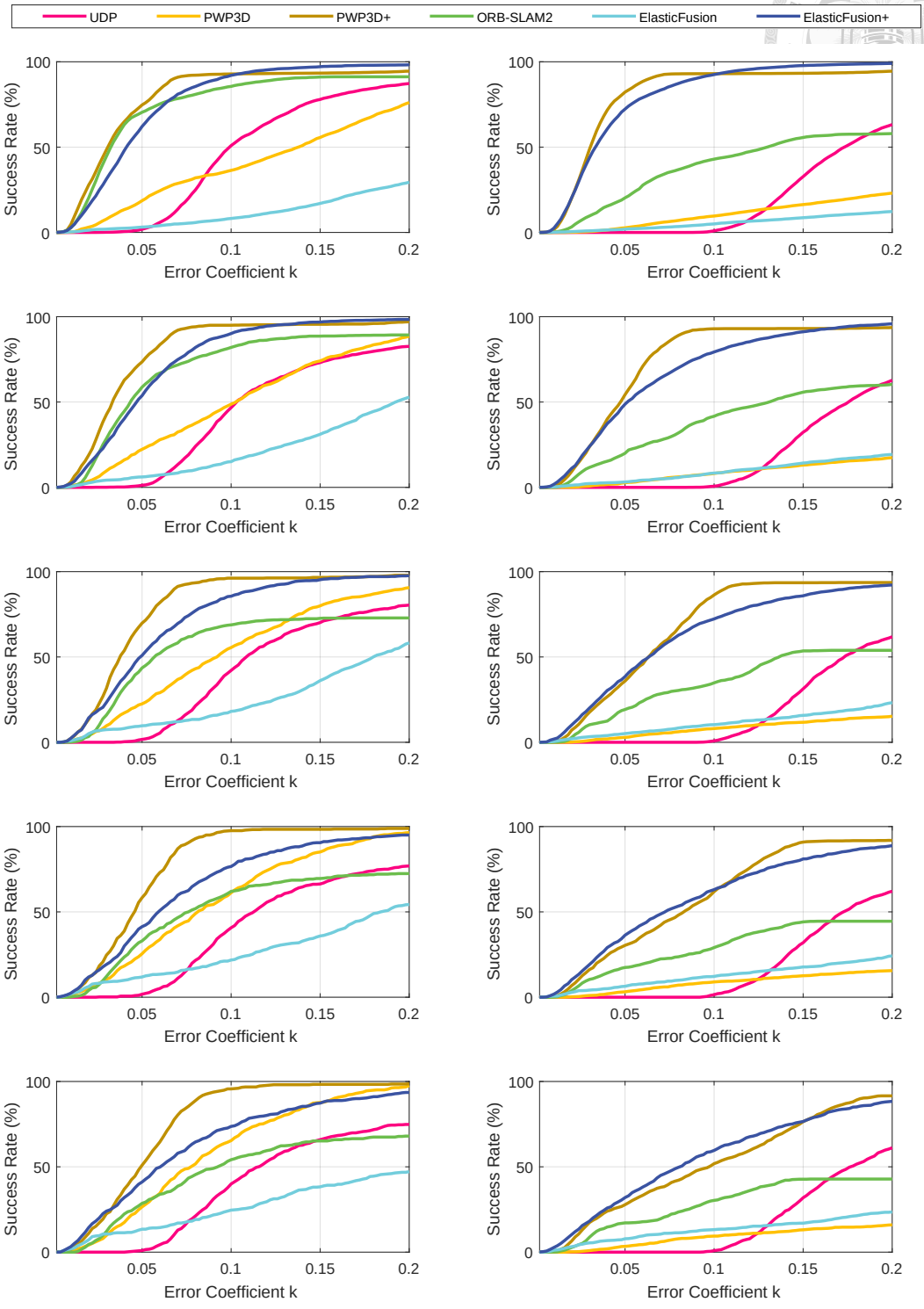
the z-direction, as discussed in [21]. We also notice that the depth values captured by Kinect V2 occasionally change significantly even under the static conditions, as illustrated in Figure 4.7. As such, the evaluated approaches may occasionally lose track of objects when the camera is not moving.

The precision plots for the performance of the evaluated methods on all the attribute subsets are shown from Figure 4.31 to Figure 4.33.

4.4.3 Discussion

Based on the comparative study on the benchmark datasets, we highlight some components that are essential for advancing the field of pose tracking on 2D and 3D

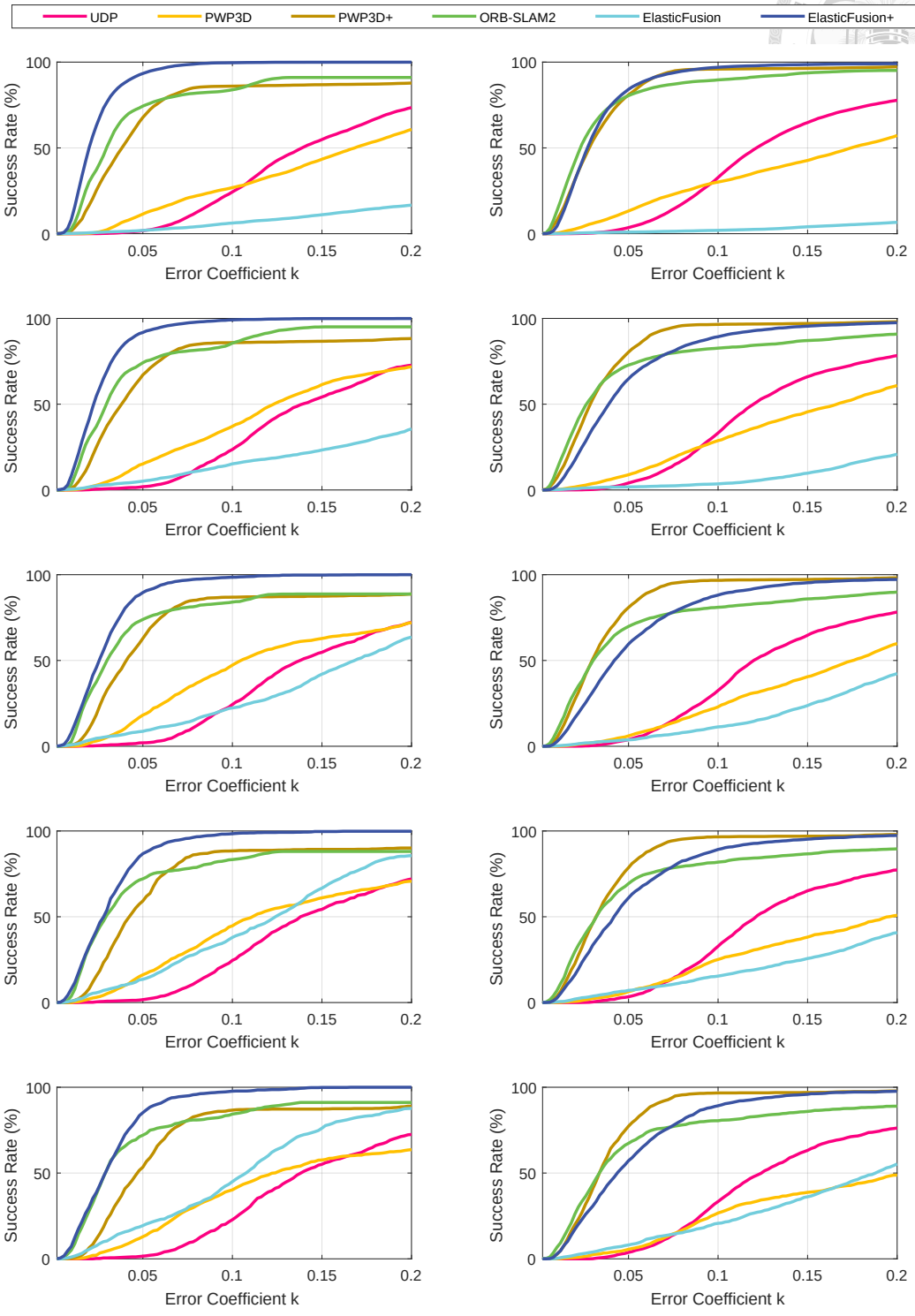
doi:10.6342/NTU201800854



(a) Translation (b) Zoom

Figure 4.31: Precision plots for 3D object (a) *Translation* and (b) *Zoom* sub-datasets. From top to bottom: lowest speed (i.e., level 1) to highest speed (i.e., level 5).

doi:10.6342/NTU201800854



(a) In-plane Rotation (b) Out-of-plane Rotation

Figure 4.32: Precision plots for 3D object (a) *In-plane Rotation* and (b) *Out-of-plane Rotation* sub-datasets. From top to bottom: lowest speed (i.e., level 1) to highest speed (i.e., level 5).

doi:10.6342/NTU201800854

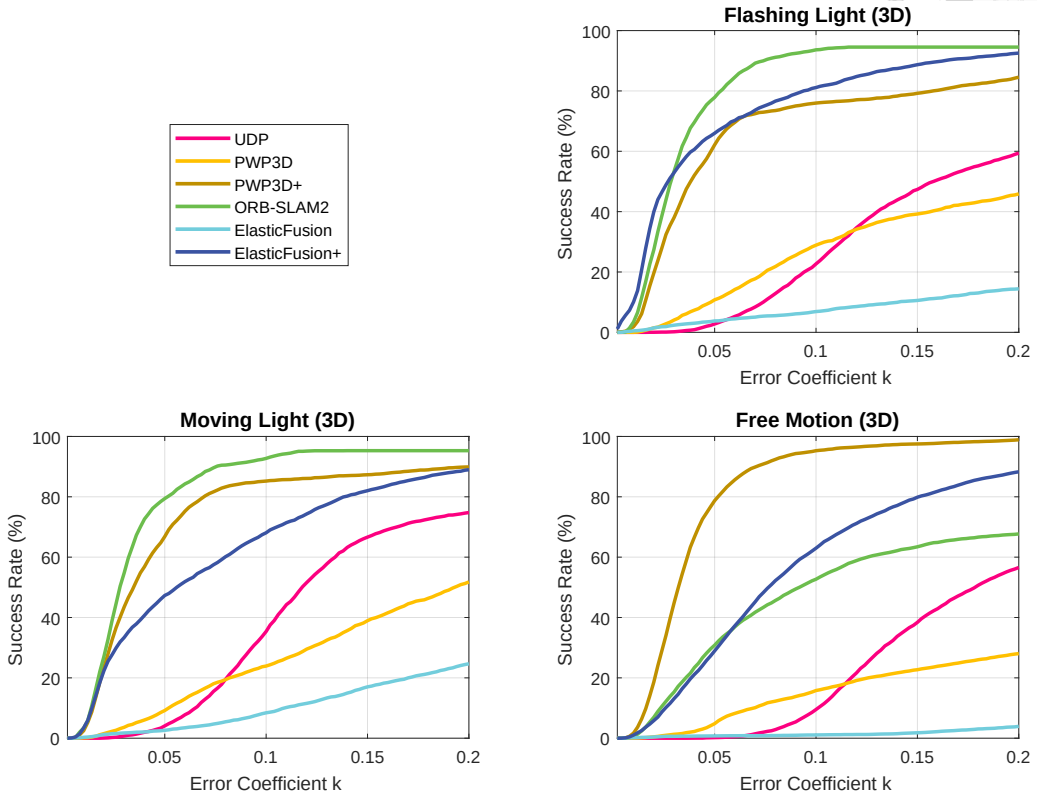
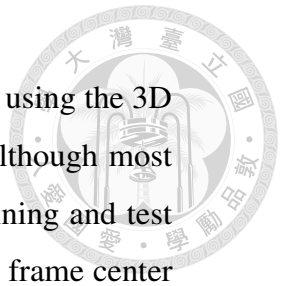


Figure 4.33: Precision plots for 3D object *Flashing Light*, *Moving Light*, and *Free Motion* sub-datasets.

objects. First, approaches with wider baseline matching strategies can alleviate the issue with losing track of objects more effectively than the gradient-descent-based energy minimization methods. It is crucial to develop more effective schemes to reduce the risk of getting stuck in a local minimum for methods based on energy minimization. It will be of great interest to use multiple solutions, hierarchical optimization, and particle filters to alleviate these issues.

Second, it is essential to equip the segmentation-based approach with a robust foreground and background classifier for effective pose tracking. Although the measured depth values are noisy (as shown in Figure 4.6 and 4.7), it would still be more accessible to segment the foreground object from the background scene even when in cluttered environment. In more challenging scenes, CNN-based segmentation approaches [173, 174] can also be used.

doi:10.6342/NTU201800854



Third, more accurate pose tracking results may be achieved by using the 3D coordinates (x, y, z) as training data rather than depth values. Although most learning-based approaches use the raw RGB-D images as the training and test data [118, 16], the training data is usually generated at the camera frame center with the fixed translation vector, but with different rotation matrices. However, the object appearances vary at different positions in a camera frame even with the same rotation, as illustrated in Figure 4.34. It is also possible that local regions at two different poses may be very similar in appearance. That is, the RGB-D appearance and the pose are not in one-to-one correspondence. Thus, incorrect results may be obtained if the object poses are estimated based on the raw RGB-D values. The situation can be even worse if we change the camera model to another one with different intrinsic parameters. The results from this comparative study show that better results can be achieved by using RGB-XYZ values [2, 60] as these values and the model pose are bijective. Furthermore, we can still use the same trained model with a different camera for recovering the object pose.

4.5 Summary

In this work, we propose a large-scale benchmark dataset and perform thorough performance evaluation under various conditions close to real-world scenarios. The proposed benchmark dataset contains 690 color and depth videos with over 100,000 frames. These videos are recorded under seven different movement and lighting conditions with five speeds. We select six 2D target planes with three different texture levels, and six 3D target objects with three different geometric level. The ground-truth poses are annotated by leveraging the clear infrared images recorded by the global-shutter infrared camera with fast shutter speed from the Kinect V2 sensor, which enables us to record sequence even under quick motions.

Based on the benchmark experiments, we discuss some tracking components that are essential for improving the tracking performance. This large-scale perfor-

doi:10.6342/NTU201800854

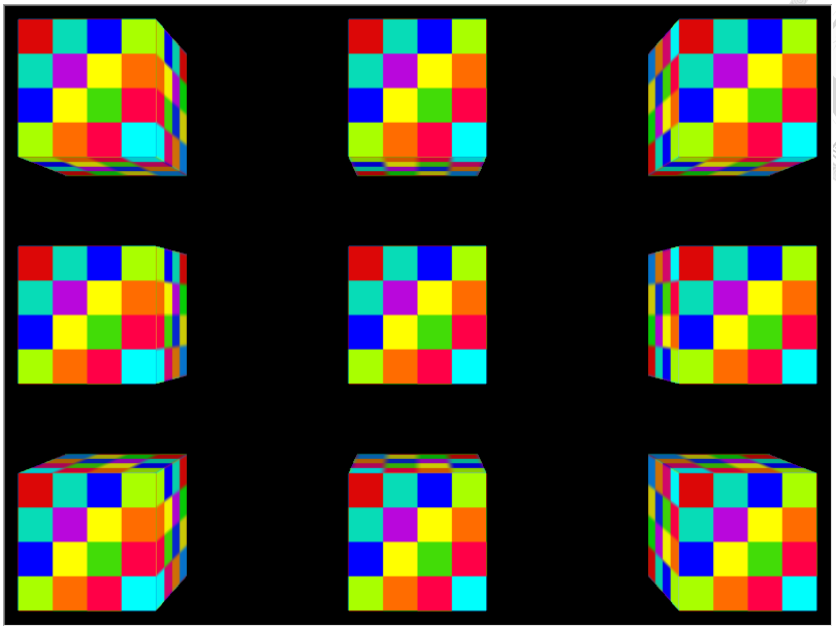


Figure 4.34: The appearances of cubes are different with the same rotation (which is an identity matrix in this image) at different positions. It is challenging to effectively recover the accurate object pose based on the raw RGB-D values if the training data is only generated at the camera frame center with different rotation matrices. Ambiguous results may be obtained with different rotation in this condition. For example, we may get a pose result with inaccurate rotation for the up-right cube in this image since there exists another candidate which has a more similar RGB-D appearance with different rotation at the camera frame center.

mance evaluation facilitates a better understanding of the state-of-the-art object pose tracking approaches, and provide a platform for gauging new algorithms. We note that considerable progress has recently been made to improve the state-of-the-art methods for pose tracking [114, 115, 139, 21]. Our future work will focus on extending the datasets (e.g., change the background to cluttered one and adding partial occlusion) and evaluate more methods once they are made publicly available.

doi:10.6342/NTU201800854

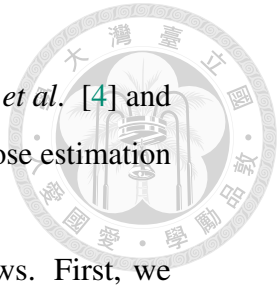
doi:10.6342/NTU201800854

Chapter 5

DPE: Direct Pose Estimation for Planar Objects

In this work, we propose a direct method to estimate the 6DoF poses of a planar target from a calibrated camera by measuring the similarity between the projected planar target object $\mathcal{O}_t$ and observed camera image $\mathcal{I}_c$ based on appearance. As the proposed method is based on a planar object rather than a 3D model, the pose ambiguity problem as discussed in prior arts is inevitably bound to occur [175, 88, 176, 133]. Pose ambiguity is related to situations where the error function has several local minima for a given configuration, which is the main cause of flipping estimated poses in an image sequence. Based on image observations, one of the ambiguous poses with local minima, according to an error function, is the correct pose. Therefore, after obtaining an initial rough pose using an approximate pose estimation scheme, we determine all ambiguous poses and refine the estimates until they converge to local minima. The final pose is chosen as the one with the lowest error among these refined ambiguous poses. We show some pose estimation results by the proposed method in Figure 1.2. Extensive experiments are conducted to validate the proposed algorithm in this work. In particular, we evaluate the proposed algorithm on different types of templates with varying levels of degraded images caused by blur, intensity, tilt angle, and compression noise. Furthermore, we

doi:10.6342/NTU201800854



evaluate the proposed algorithm on the dataset proposed by Gauglitz *et al*. [4] and our OPT dataset presented in Chapter 4 against the state-of-the-art pose estimation methods.

The main contributions of this work are summarized as follows. First, we propose an efficient direct pose estimation algorithm for planar targets undergoing arbitrary 3D perspective transformations. Second, we show the proposed pose estimation algorithm performs favorably against the state-of-the-art feature-based approaches in terms of robustness and accuracy. Third, we demonstrate the proposed pose refinement method not only improves the accuracy of estimated results but also alleviates the pose ambiguity problem effectively. The source code and datasets are available on our project website at media.ee.ntu.edu.tw/research/DPE.

Based on our prior study in [3], in this work, we extend and construct an image pyramid for the APE method as described in Section 5.1, and we apply a new PR approach based on the LK algorithm as described in Section 5.2. We show experimental results with significant improvements regarding accuracy and efficiency compared to the previous work in Section 5.3.

5.1 Approximate Pose Estimation

We first normalize the target image $\mathcal{O}_t$ and the camera image $\mathcal{I}_c$ with pixel values in the range $[0, 1]$. Let $T_{\mathbf{p}}$ be the transformation at pose $\mathbf{p}$ in (2.1). Assume a reference point $\mathbf{x}_i = [x_i, y_i, 0]^\top$ in the target image is transformed separately to $\mathbf{u}_{i1}$ and $\mathbf{u}_{i2}$ in the camera image with two different poses $\mathbf{p}_1$ and $\mathbf{p}_2$. It has been shown by Korman *et al*. [177] that if any distance between $\mathbf{u}_{i1}$ and $\mathbf{u}_{i2}$ is smaller than a positive value ε, with upper bound in the Big-O notation [178]:

$$\forall \mathbf{x}_i \in \mathcal{O}_t : d(T_{\mathbf{p}_1}(\mathbf{x}_i), T_{\mathbf{p}_2}(\mathbf{x}_i)) = O(\varepsilon), \tag{5.1}$$

then the following equation holds:

$$|E_{a_1}(\mathbf{p}_1) - E_{a_1}(\mathbf{p}_2)| = O(\varepsilon \bar{\mathcal{V}}), \tag{5.2}$$

doi:10.6342/NTU201800854

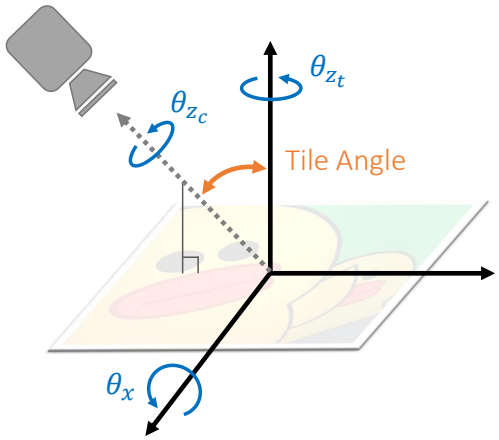


Figure 5.1: Illustration of rotation angle: θ_x indicates the tilt angle between the camera and the target image when the rotation is factored as $\mathbf{R} = \mathbf{R}_z(\theta_{z_c})\mathbf{R}_x(\theta_x)\mathbf{R}_z(\theta_{z_t})$.

where $\bar{\mathcal{V}}$ denotes the mean variation of $\mathcal{O}_t$, which represents the mean value over the entire target image of the maximal difference between each pixel and any of its neighbors. The mean variation $\bar{\mathcal{V}}$ can be constrained by filtering $\mathcal{O}_t$. In addition, the error function $E_{a_1}(\cdot)$ is defined in (2.4). The main result is that the difference between $E_{a_1}(\mathbf{p}_1)$ and $E_{a_1}(\mathbf{p}_2)$ is bounded in terms of ε. In the proposed direct method, we only need to consider a limited number of poses by constructing an ε-covering pose set $\mathcal{S}$ [179] based on (5.1) and (5.2).

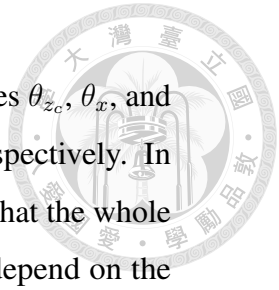
5.1.1 Constructing the ε-covering Set

As illustrated in Figure 5.1, in this stage, we factor the rotation as follows [65]:

$$\begin{aligned}\mathbf{R} &= \mathbf{R}_z(\theta_{z_c})\mathbf{R}_x(\theta_x)\mathbf{R}_z(\theta_{z_t}) \\ &= \begin{bmatrix} c_{z_c}c_{z_t} - c_x s_{z_c} s_{z_t} & -c_x c_{z_t} s_{z_c} - c_{z_c} s_{z_t} & s_x s_{z_c} \\ c_{z_t} s_{z_c} + c_x c_{z_c} s_{z_t} & c_x c_{z_c} c_{z_t} - s_{z_c} s_{z_t} & -s_x c_{z_c} \\ s_x s_{z_t} & s_x c_{z_t} & c_x \end{bmatrix},\end{aligned} \tag{5.3}$$

where we use the notation $c_a = \cos(\theta_a)$ and $s_a = \sin(\theta_a)$ for $a = z_c, x, z_t$. Moreover, $\mathbf{R}_x(\cdot)$ and $\mathbf{R}_z(\cdot)$ are defined in (2.7). Therefore, the object pose is

doi:10.6342/NTU201800854



now parameterized as $\mathbf{p} = [\theta_{z_c}, \theta_x, \theta_{z_t}, t_x, t_y, t_z]^\top$. These Euler angles θ_{z_c}, θ_x, and θ_{z_t} are in the range $[-180°, 180°]$, $[0°, 90°]$, and $[-180°, 180°]$, respectively. In addition, the translation parameters t_x, t_y, and t_z are bounded such that the whole target image would be within the camera image, and the bounds depend on the camera intrinsic parameters. Furthermore, we set an upper bound for t_z since it is not practical to detect an extreme tiny target image in the camera image.

A pose set $\mathcal{S}$ is constructed such that any two consecutive poses, $\mathbf{p}_k$ and $\mathbf{p}_k + \Delta\mathbf{p}_k$ on each dimension satisfy (5.1) in $\mathcal{S}$. To construct the set, the coordinates of $\mathbf{x}_i \in \mathcal{I}_t$ are normalized to the range $[-1, 1]$. Starting with t_z, we derive the following equation by using (2.1) for each $\mathbf{x}_i$:

$$\begin{aligned} & d(T_{\mathbf{p}_{t_z}}(\mathbf{x}_i), T_{\mathbf{p}_{t_z+\Delta t_z}}(\mathbf{x}_i)) \\ &= \sqrt{\left[\left(\frac{f_x x_i}{t_z}\right) - \left(\frac{f_x x_i}{t_z + \Delta t_z}\right)\right]^2 + \left[\left(\frac{f_y y_i}{t_z}\right) - \left(\frac{f_y y_i}{t_z + \Delta t_z}\right)\right]^2} \\ &= O\left(\frac{1}{t_z} - \frac{1}{t_z + \Delta t_z}\right). \end{aligned} \tag{5.4}$$

To satisfy the constraint in (5.1), we use the step size with tight bound in the Big-Theta notation [178]:

$$\Delta t_z = \Theta\left(\frac{\varepsilon t_z^2}{1 - \varepsilon t_z}\right), \tag{5.5}$$

which represents that (5.4) can be bounded if we construct $\mathcal{S}$ using (5.5) on dimension t_z.

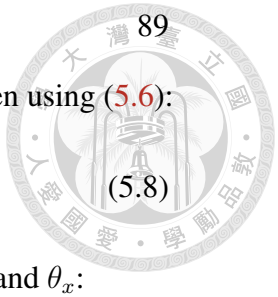
Since θ_x describes the tilt angle between the camera and target image as shown in Figure 5.1, we obtain the following equation based on t_z:

$$\begin{aligned} d(T_{\mathbf{p}_{\theta_x}}(\mathbf{x}_i), T_{\mathbf{p}_{\theta_x+\Delta\theta_x}}(\mathbf{x}_i)) &= \sqrt{\alpha_{\theta_x}^2 + \beta_{\theta_x}^2} \\ &= O\left(\frac{1}{t_z - \sin(\theta_x + \Delta\theta_x)} - \frac{1}{t_z - \sin(\theta_x)}\right), \end{aligned} \tag{5.6}$$

where:

$$\begin{aligned} \alpha_{\theta_x} &= \left(\frac{f_x x_i}{y_i \sin\theta_x + t_z}\right) - \left(\frac{f_x x_i}{y_i \sin(\theta_x + \Delta\theta_x) + t_z}\right), \\ \beta_{\theta_x} &= \left(\frac{f_y y_i \cos\theta_x}{y_i \sin\theta_x + t_z}\right) - \left(\frac{f_y y_i \cos(\theta_x + \Delta\theta_x)}{y_i \sin(\theta_x + \Delta\theta_x) + t_z}\right). \end{aligned} \tag{5.7}$$

doi:10.6342/NTU201800854



In addition, to satisfy the constraint in (5.1), we set the step size when using (5.6):

$$\Delta\theta_x = \Theta\left(\sin^{-1}\left(t_z - \frac{1}{\varepsilon + \frac{1}{t_z - \sin(\theta_x)}}\right) - \theta_x\right). \tag{5.8}$$

Let $\theta_{z_t'} = \theta_{z_t} + \Delta\theta_{z_t}$, we obtain the following equation based on t_z and θ_x:

$$\begin{aligned} d(T_{\mathbf{p}_{\theta_{z_t}}}(\mathbf{x}_i), T_{\mathbf{p}_{\theta_{z_t}+\Delta\theta_{z_t}}}(\mathbf{x}_i)) &= \sqrt{f_x^2\alpha_{\theta_{z_t}}^2 + f_y^2 c_x^2 \beta_{\theta_{z_t}}^2} \\ &\leq \sqrt{f_x^2\alpha_{\theta_{z_t}}^2 + f_y^2 \beta_{\theta_{z_t}^2}} \\ &= O\left(\frac{\Delta\theta_{z_t}}{t_z + k\sin(\theta_x)}\right), \end{aligned} \tag{5.9}$$

where k denotes any constant in the range of $\left[-\sqrt{2}, \sqrt{2}\right]$, and:

$$\begin{aligned} \alpha_{\theta_{z_t}} &= \frac{c_{z_t}x - s_{z_t}y}{s_x(s_{z_t}x + c_{z_t}y) + t_z} - \frac{c_{z_t'}x - s_{z_t'}y}{s_x(s_{z_t'}x + c_{z_t'}y) + t_z}, \\ \beta_{\theta_{z_t}} &= \frac{s_{z_t}x + c_{z_t}y}{s_x(s_{z_t}x + c_{z_t}y) + t_z} - \frac{s_{z_t'}x + c_{z_t'}y}{s_x(s_{z_t'}x + c_{z_t'}y) + t_z}. \end{aligned} \tag{5.10}$$

An illustrative example of (5.10) is shown in Figure 5.2. To make (5.10) satisfy the constraint in (5.1), we set the step size:

$$\Delta\theta_{z_t} = \Theta\left(\varepsilon\left(t_z + k\sin(\theta_x)\right)\right), \tag{5.11}$$

where larger k means larger bounded steps for constructing $\mathcal{S}$. We set k to be 0 for $\Delta\theta_{z_t}$ in the proposed method.

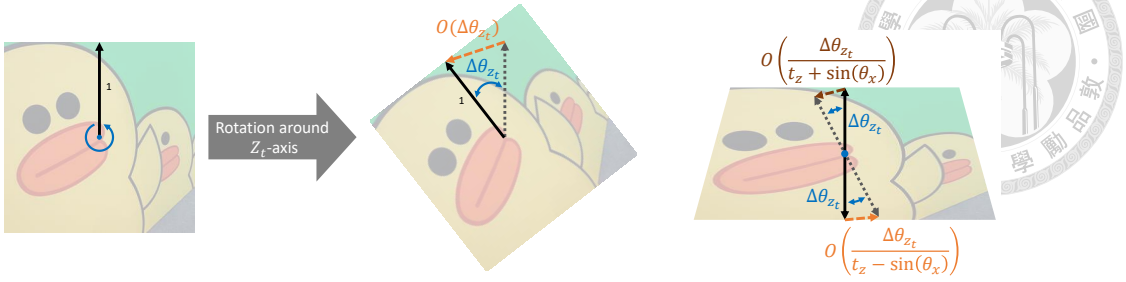
As θ_{z_t} denotes 2D image rotation of the planar target, it does not influence the bounded steps for θ_{z_c}. Let $\theta_{z_c'} = \theta_{z_c} + \Delta\theta_{z_c}$, we obtain the following equation depending on the current t_z and θ_x:

$$\begin{aligned} d(T_{\mathbf{p}_{\theta_{z_c}}}(\mathbf{x}_i), T_{\mathbf{p}_{\theta_{z_c}+\Delta\theta_{z_c}}}(\mathbf{x}_i)) &= \sqrt{f_x^2\alpha_{\theta_{z_c}}^2 + f_y^2 \beta_{\theta_{z_c}}^2} \\ &= O\left(\frac{\Delta\theta_{z_c}}{t_z + k\sin(\theta_x)}\right), \end{aligned} \tag{5.12}$$

where:

$$\begin{aligned} \alpha_{\theta_{z_c}} &= \frac{c_{z_c}x - c_x s_{z_c}y}{s_x y + t_z} - \frac{c_{z_c'}x - c_x s_{z_c'}y}{s_x y + t_z}, \\ \beta_{\theta_{z_c}} &= \frac{s_{z_c}x + c_x c_{z_c}y}{s_x y + t_z} - \frac{s_{z_c'}x + c_x c_{z_c'}y}{s_x y + t_z}. \end{aligned} \tag{5.13}$$

doi:10.6342/NTU201800854



(a) (b)

Figure 5.2: (a) 2D illustration of rotation around Z_t-axis. The linear distance (orange solid line) between points before and after applying rotation is bounded by the arc length (brown dotted line). (b) 3D illustration of rotation around Z_t-axis. The linear distance between points is a function of tilt angle θ_x.

We can realize (5.13) in a similar way to (5.10). To make (5.13) satisfy the constraint in (5.1), we set the step size:

$$\Delta\theta_{z_c} = \Theta(\varepsilon(t_z + k\sin(\theta_x))) = \Theta(\varepsilon(t_z)), \tag{5.14}$$

which k is set to 0.

As the bounded steps for t_x and t_y are also influenced by horizontal distance t_z and tilt angle θ_x only, we have:

$$\begin{aligned} d(T_{\mathbf{p}_{t_x}(\mathbf{x}_i)}, T_{\mathbf{p}_{t_x+\Delta t_x}}(\mathbf{x}_i)) &= \sqrt{f_x^2\alpha_{t_x}^2 + f_y^2\beta_{t_x}^2} \\ &= O\left(\frac{\Delta t_x}{t_z + k\sin(\theta_x)}\right), \end{aligned} \tag{5.15}$$

where:

$$\begin{aligned} \alpha_{t_x} &= \frac{x + t_x}{s_x y + t_z} - \frac{x + t_x + \Delta t_x}{s_x y + t_z}, \\ \beta_{t_x} &= \frac{y}{s_x y + t_z} - \frac{y}{s_x y + t_z}. \end{aligned} \tag{5.16}$$

And:

$$\begin{aligned} d(T_{\mathbf{p}_{t_y}(\mathbf{x}_i)}, T_{\mathbf{p}_{t_y+\Delta t_y}}(\mathbf{x}_i)) &= \sqrt{f_x^2\alpha_{t_y}^2 + f_y^2\beta_{t_y}^2} \\ &= O\left(\frac{\Delta t_y}{t_z + k\sin(\theta_x)}\right), \end{aligned} \tag{5.17}$$

doi:10.6342/NTU201800854

Table 5.1: Bounded step size on each dimension in the pose domain for constructing the ε-covering pose set.

Dimension	Step Size
θ_{z_c}	$\Theta(\varepsilon t_z)$
θ_x	$\Theta\left(\sin^{-1}\left(t_z - \frac{1}{\varepsilon + \frac{1}{t_z - \sin(\theta_x)}}\right) - \theta_x\right)$
θ_{z_t}	$\Theta(\varepsilon t_z)$
t_x	$\Theta\left(\varepsilon\left(t_z - \sqrt{2}\sin(\theta_x)\right)\right)$
t_y	$\Theta\left(\varepsilon\left(t_z - \sqrt{2}\sin(\theta_x)\right)\right)$
t_z	$\Theta\left(\frac{\varepsilon t_z^2}{1-\varepsilon t_z}\right)$

where:

$$\begin{aligned} \alpha_{t_y} &= \frac{x}{s_x y + t_z} - \frac{x}{s_x y + t_z}, \\ \beta_{t_y} &= \frac{y + t_y}{s_x y + t_z} - \frac{y + t_y + \Delta t_y}{s_x y + t_z}. \end{aligned} \tag{5.18}$$

To make (5.16) and (5.18) satisfy the constraint in (5.1), we set these step sizes:

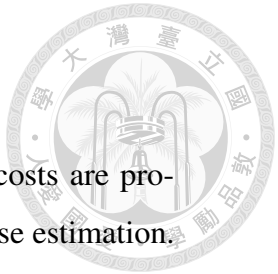
$$\Delta t_x = \Theta\left(\varepsilon\left(t_z + k\sin(\theta_x)\right)\right) = \Theta\left(\varepsilon\left(t_z - \sqrt{2}\sin(\theta_x)\right)\right), \tag{5.19}$$

$$\Delta t_y = \Theta\left(\varepsilon\left(t_z + k\sin(\theta_x)\right)\right) = \Theta\left(\varepsilon\left(t_z - \sqrt{2}sin(\theta_x)\right)\right). \tag{5.20}$$

as k is set to $-\sqrt{2}$ for practical consideration. Table 5.1 summarizes the bounded step size on each dimension for the ε-covering pose set.

Finally, the pose set is constructed recursively starting from t_z based on the bounded step shown in Table 5.1. We then determine values of θ_x based on its bounded step which is influenced by t_z. The remaining pose parameters θ_{z_c}, θ_{z_t}, t_x, and t_y are determined based on each of their bounded steps, which are affected only by t_z and θ_x and independent of each other.

doi:10.6342/NTU201800854



5.1.2 Coarse-to-Fine Estimation

As the parameter space is large, the computational and memory costs are prohibitively high if the ε-covering set is used straightforwardly for pose estimation. In this work, we develop a coarse-to-fine approach for fast and accurate pose estimation. The pose set $\mathcal{S}$ is first constructed with a coarse ε. After obtaining the best pose $\mathbf{p}_b$ and the associated error measure $E_{a_1}(\mathbf{p}_b)$, we select the poses within a threshold:

$$\mathcal{S}_L = \{\mathbf{p}_L \mid E_{a_1}(\mathbf{p}_L) < E_{a_1}(\mathbf{p}_b) + L\}, \tag{5.21}$$

to be considered in the next step. Here the constant L is a threshold empirically determined. Based on $\mathcal{S}_L$, we create sets with finer ε':

$$\mathcal{S}' = \{\mathbf{p}' \mid \exists \mathbf{p}_L \in \mathcal{S}_L : (5.1) \text{ holds for } \mathbf{p}', \mathbf{p}_L \text{ and } \varepsilon'\}, \tag{5.22}$$

and repeat this process until we obtain the desired precision parameter ε^*. In our implementation, the initial ε is set to be 0.25 and is diminished by multiplying a scale factor of 0.662 in each iteration. The precision parameter ε^* is set to meet the condition that for each point in the target image, the maximum distance between neighboring points in the camera image transformed by poses in the ε-covering pose set is less than 1 pixel. Empirically, ε^* would be around 0.01. The best pose in the last set is considered as the approximate estimate.

5.1.3 Approximate Error Measure

If we approximate the error measure E'_{a_1} with random sampling only a portion of pixels instead of computing E_{a_1} with sampling all pixels in $\mathcal{O}_t$, according to Hoeffding's inequality [180], E'_{a_1} is close to E_{a_1} within a precision parameter δ if the number of sampling pixels m is sufficiently large:

$$P(|E'_{a_1} - E_{a_1}| > \delta) \leq 2e^{-2\delta^2 m}, \tag{5.23}$$

where $P(\cdot)$ represents the probability measure. This inequality suggests that if m is properly selected, the approximation error between E'_{a_1} and E_{a_1} can be

doi:10.6342/NTU201800854

bounded with high probability. In other words, E'_{a_1} is a close approximation of E_{a_1} within the probably approximately correct (PAC) framework [181]. With this approximation, the runtime of estimating the error measure can be significantly reduced by inspecting only a small fraction of pixels in a target image. We normalize the intensity term and add the chroma components to the appearance distance measure to account for lighting variation.

5.1.4 Pyramidal Implementation

To constrain the mean variation $\bar{\mathcal{V}}$ in (5.2), it is common to blur $\mathcal{O}_t$ (and $\mathcal{I}_c$) before carrying out the proposed approximate pose estimation method. Since a blurry image has a texture similar to that of a lower resolution image, we construct an image pyramid instead of directly blurring images. It is worth using a lower resolution image for pose estimation from some perspectives. First, when we sample pixels on a smaller image, the cache miss rate will be lower and thus reduce memory traffic. Second, we can also sample a smaller amount of pixels in (5.23) when using low-resolution images. Starting from the lowest resolution image, we proceed to the next level (i.e., higher resolution image) when the distance in (5.1) is smaller than one pixel for all transformations. Empirically, the pyramid implementation can increase the runtime performance significantly while achieving similar or even higher accuracy and robustness for pose estimation.

5.2 Pose Refinement

We obtain a coarse pose $\mathbf{p}' \equiv (\mathbf{R}', \mathbf{t}')$ using the proposed approximate pose estimation scheme. However, this estimate is bounded based on the distance in the appearance space rather than the pose space. Thus the estimated and actual poses may be significantly different even when the appearance distance is small, mainly when the tilt angle of a target image is large. In the meanwhile, the pose ambiguity problem is likely to occur as illustrated in Figure 1.2. As such, we propose a pose

doi:10.6342/NTU201800854

refinement method to improve accuracy and address the ambiguity problem of estimates.

5.2.1 Determining Candidate Poses

In order to address the pose ambiguity problem, we first transform four corner points $\mathbf{x}_{c1}$, $\mathbf{x}_{c2}$, $\mathbf{x}_{c3}$, and $\mathbf{x}_{c4}$ in the target image $\mathcal{O}_t$ to $\mathbf{u}_{c1}$, $\mathbf{u}_{c2}$, $\mathbf{u}_{c3}$, and $\mathbf{u}_{c4}$ in the observed camera image $\mathcal{I}_c$ with $\mathbf{p}'$, respectively. We then compute all stationary points of the error function (2.3) based on the Gröbner basis method [134]. Only the stationary points with the two smallest objective values in (2.3) are plausible poses, and these two ambiguous poses $\mathbf{p}'_1$ and $\mathbf{p}'_2$ are both chosen as the candidate poses.

5.2.2 Refining Candidate Poses

After obtaining the two candidate poses, we further refine the estimates using a dense image alignment method which minimizes the SSD error E_{a_2} in (2.5) (instead of the SAD error E_{a_1} in (2.4) as it is not continuously differentiable) by the LK-based approach. For each candidate pose $\mathbf{p}_c$, we solve the nonlinear least squares problem using the Gauss-Newton method. To approximate how the image changes with respect to pose, we use the first-order Taylor series as follows:

$$\begin{aligned}\Delta\mathbf{p}^* &= \underset{\Delta\mathbf{p}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{I}_c\left(\mathbf{u}_i\left(\mathbf{p}_c+\Delta\mathbf{p}\right)\right)-\mathcal{O}_t\left(\mathbf{x}_i\right)\right)^2 \\ &\approx \underset{\Delta\mathbf{p}}{\operatorname{argmin}} \sum_{i=1}^{n}\left(\mathcal{I}_c\left(\mathbf{u}_i\left(\mathbf{p}_c\right)\right)+\left.\frac{\partial\mathcal{I}_c}{\partial\mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c}\Delta\mathbf{p}-\mathcal{O}_t\left(\mathbf{x}_i\right)\right)^2.\end{aligned} \tag{5.24}$$

Different from the method described in Section 5.1, here the pose $\mathbf{p}$ is parameterized as a 6D vector consisting of the 3D rotation vector (which is presented in Section 2.1.2) and the 3D translation vector:

$$\mathbf{p}=\begin{bmatrix}\mathbf{r}\\ \mathbf{t}\end{bmatrix},\ \mathbf{r}=\begin{bmatrix}r_x\\ r_y\\ r_z\end{bmatrix}\in\mathbb{R}^3,\ \mathbf{t}=\begin{bmatrix}t_x\\ t_y\\ t_z\end{bmatrix}\in\mathbb{R}^3. \tag{5.25}$$

doi:10.6342/NTU201800854

To compute $\Delta\mathbf{p}$ in each iteration, we set the first derivative of (6.3) to zero and solve the resulting system of linear equations:

$$\mathbf{J}_c\Delta\mathbf{p} = \mathbf{O}_t - \mathbf{I}_c, \tag{5.26}$$

where $\mathbf{O}_t$ and $\mathbf{I}_c$ are vector forms of $\mathcal{O}_t(\mathbf{x}_i)$ and $\mathcal{I}_c(\mathbf{u}_i)$, respectively. In (5.26), $\mathbf{J}_c$ is the Jacobian matrix of $\mathbf{I}_c$ with respect to $\mathbf{p}$ at the pose $\mathbf{p} = \mathbf{p}_c$ and computed by the chain rule (in the numerator-layout notation):

$$\mathbf{J}_c = \left.\frac{\partial \mathbf{I}_c}{\partial \mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c} = \begin{bmatrix} \frac{\partial \mathcal{I}_c(\mathbf{u}_1)}{\partial \mathbf{p}} \\ \frac{\partial \mathcal{I}_c(\mathbf{u}_2)}{\partial \mathbf{p}} \\ \vdots \\ \frac{\partial \mathcal{I}_c(\mathbf{u}_n)}{\partial \mathbf{p}} \end{bmatrix}, \tag{5.27}$$

$$\frac{\partial \mathcal{I}_c}{\partial \mathbf{p}} = \frac{\partial \mathcal{I}_c}{\partial \mathbf{u}} \left[\frac{\partial \mathbf{u}}{\partial \mathbf{r}}, \frac{\partial \mathbf{u}}{\partial \mathbf{t}}\right] = \left[\frac{\partial \mathcal{I}_c}{\partial u}, \frac{\partial \mathcal{I}_c}{\partial v}\right] \left[\frac{\partial \mathbf{u}}{\partial \hat{\mathbf{x}}} \frac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{R}}} \frac{\partial \hat{\mathbf{R}}}{\partial \mathbf{r}}, \frac{\partial \mathbf{u}}{\partial \hat{\mathbf{x}}}\right], \tag{5.28}$$

$$\frac{\partial \mathbf{u}}{\partial \hat{\mathbf{x}}} = \begin{bmatrix} \frac{f_x}{\hat{z}} & 0 & -\frac{f_x \hat{x}}{\hat{z}^2} \\ 0 & \frac{f_y}{\hat{z}} & -\frac{f_y \hat{y}}{\hat{z}^2} \end{bmatrix}, \frac{\partial \hat{\mathbf{x}}}{\partial \hat{\mathbf{R}}} = \begin{bmatrix} x & y & 0 & 0 & 0 & 0 \\ 0 & 0 & x & y & 0 & 0 \\ 0 & 0 & 0 & 0 & x & y \end{bmatrix}, \tag{5.29}$$

where $\hat{\mathbf{R}} = [R_{11}, R_{12}, R_{21}, R_{22}, R_{31}, R_{32}]^\top$ denotes the vector with elements in the left two columns of the rotation matrix $\mathbf{R}$, and

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & t_x \\ R_{21} & R_{22} & t_y \\ R_{31} & R_{32} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{5.30}$$

is the camera-space coordinate transformed from the object-space coordinate $\mathbf{x} = [x, y, 0]^\top$.

In addition, the derivative of $\hat{\mathbf{R}}$ with respect to $\mathbf{r}$ can be obtained using the following formula [182]:

$$\frac{\partial \mathbf{R}}{\partial r_a} = \frac{r_a [\mathbf{r}]_\times + [\mathbf{r} \times (\mathbf{I} - \mathbf{R})\mathbf{e}_i]_\times}{\|\mathbf{r}\|^2}\mathbf{R}, \quad a = x, y, z, \tag{5.31}$$

doi:10.6342/NTU201800854

where $\mathbf{I}$ and $\mathbf{e}_i$ are the identity matrix and the i-th vector of the standard basis in $\mathbb{R}^3$, respectively. In addition, $[\mathbf{r}]_\times$ represents the cross-product matrix for the vector $\mathbf{r}$ which is as defined in (2.17).

A closed form solution of (5.26) is:

$$\Delta\mathbf{p} = \left(\mathbf{J}_c^\top \mathbf{J}_c\right)^{-1} \mathbf{J}_c^\top \left(\mathbf{I}_t - \mathbf{I}_c\right). \tag{5.32}$$

As the least squares problem is nonlinear, the Gauss-Newton iteration method does not always converge with fixed step size. We thus perform a backtracking line search to scale the step size after each iteration of computing (5.32). We shrink $\Delta\mathbf{p}$ by $\Delta\mathbf{p} \leftarrow \alpha\Delta\mathbf{p}$ until it meets the Armijo-Goldstein condition:

$$E_{a_2}(\mathbf{p}_c + \Delta\mathbf{p}) \leq E_{a_2}(\mathbf{p}_c) + c\nabla E_{a_2}(\mathbf{p}_c)^\top \Delta\mathbf{p}, \tag{5.33}$$

where $\nabla E_{a_2}(\mathbf{p}_c)$ is the local function gradient. We set $\alpha = 0.5$ and $c = 10^{-4}$ empirically in this work. The candidate pose $\mathbf{p}_c$ is refined by $\mathbf{p}_c \leftarrow \mathbf{p}_c + \Delta\mathbf{p}$ until the vector norm $\|\Delta\mathbf{p}\|$ is less than a predefined threshold $\varepsilon_{\Delta\mathbf{p}}$.

Finally, the pose corresponding to the smaller E_{a_2} is selected from the two refined candidate poses. The main steps of the proposed pose estimation method are summarized in Algorithm 1. It should be noted that we also perform the pyramid implementation for the refinement process to increase both the accuracy and efficiency.

5.3 Experimental Results

We evaluate the proposed algorithm for the 6DoF pose estimation problem using a synthetic image dataset that we develop and two real image benchmark datasets. As the color of each template in the real image benchmark datasets is slightly changed after being generated by a printer and then viewed by a camera, we calibrate each template in the two real image benchmark datasets before carrying out performance evaluation.

doi:10.6342/NTU201800854

Algorithm 1: Direct 6DoF Pose Estimation.

Input: Target image $\mathcal{O}_t$, camera image $\mathcal{I}_c$, intrinsic parameters, and parameters ε^*, $\varepsilon_{\Delta\mathbf{p}}$;

Output: Estimated pose result $\mathbf{p}^*$;

Build image pyramids for $\mathcal{O}_t$ and $\mathcal{I}_c$;
Start from images with lowest resolution;
Create an ε-covering pose set $\mathcal{S}$;
Find $\mathbf{p}_b$ from $\mathcal{S}$ with E'_{a_1} according to (5.23);
while $\varepsilon > \varepsilon^*$ **do**
    Obtain the set $\mathcal{S}_L$ according to (5.21);
    Diminish ε;
    if $d < 1$ according to (5.1) **then**
        Change to the next image resolution;
    end if
    Replace $\mathcal{S}$ according to (5.22);
    Find $\mathbf{p}_b$ from $\mathcal{S}$ with E'_{a_1} according to (5.23);
end while
Determine the candidate poses $\mathbf{p}_1$ and $\mathbf{p}_2$ with $\mathbf{p}_b$;
for $i = 1 \rightarrow 2$ **do**
    Let $\mathbf{p}_c = \mathbf{p}_i$;
    repeat
        Compute $\mathbf{J}_c$ according to (5.27);
        Compute $\Delta\mathbf{p}$ according to (5.32);
        while Condition according to (5.33) is not met **do**
            $\Delta\mathbf{p} \leftarrow \alpha\Delta\mathbf{p}$
        end while
        $\mathbf{p}_c \leftarrow \mathbf{p}_c + \Delta\mathbf{p}$
    until $\|\Delta\mathbf{p}\| < \varepsilon_{\Delta\mathbf{p}}$
    Let $\mathbf{p}_i = \mathbf{p}_c$;
end for
Return the pose $\mathbf{p}^*$ with smaller E_{a_2} from $\mathbf{p}_1$ and $\mathbf{p}_2$;

doi:10.6342/NTU201800854

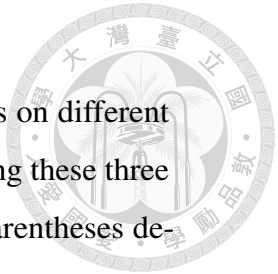


Table 5.2: Average runtime (measured in seconds) for approaches on different datasets. Although SIFT-based approach is the fastest method among these three different schemes, its performance is quite limited. Numbers in parentheses denote the average runtime of the CUDA implementation of the proposed method, which can be executed more efficiently on a GPGPU platform as it can be easily parallelized.

Method		Dataset		
		Synthetic	VT	OPT
SIFT-based Approach	SIFT	7.431	3.608	11.261
	RANSAC	0.010	0.005	0.098
	IPPE/OPnP	0.001/0.009	0.001/0.008	0.001/0.008
	Total	*7.446*	*3.618*	*11.364*
ASIFT-based Approach	ASIFT	10.903	15.806	38.884
	RANSAC	0.004	0.003	0.055
	IPPE/OPnP	0.001/0.009	0.001/0.008	0.001/0.008
	Total	*10.912*	*15.814*	*38.944*
DPE	APE	10.549 (**1.505**)	17.920 (**1.217**)	18.545 (**0.994**)
	PR	0.571 (**0.117**)	0.694 (**0.180**)	0.214 (**0.088**)
	Total	*11.120* (***1.622***)	*18.615* (***1.397***)	*18.759* (***1.082***)

All the experiments are completed using MATLAB on a machine with an Intel Core i7-6700K 4.0 GHz processor and 32 GB RAM. In addition, we implement the proposed direct method on an NVIDIA GTX 970 GPU using CUDA based on [183]. Table 5.2 shows average runtime for different algorithms.

We compare the proposed algorithm with feature-based pose estimation methods. The proposed direct pose estimation (DPE) algorithm is constructed with the approximate pose estimation (APE) and pose refinement (PR) approaches. Based on preliminary experiments, we determine the SIFT [9] representation performs better than other alternative features in terms of repeatability and accuracy. Similar

doi:10.6342/NTU201800854

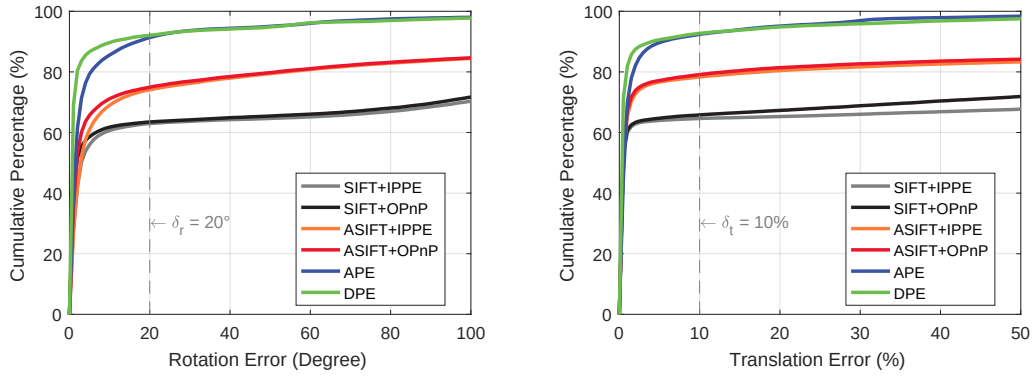


Figure 5.3: Cumulative percentage of poses whose rotation or translation errors are under values specified in the x-axis over experiments. The vertical dashed lines correspond to the thresholds used to detect unsuccessfully estimated poses. There is a total of 36,277 poses estimated by each pose estimation approach.

observations have also be reported in the literature [4]. As the ASIFT [10] method is considered the state-of-the-art affine-invariant method to determine correspondences under large view changes, we use both the SIFT and ASIFT representations in the evaluation against feature-based schemes. The RANSAC-based method [11] is then used to eliminate outliers before an object pose is estimated by a PnP algorithm. It has been shown that, among the PnP algorithms [88, 12, 13, 91, 135], the OPnP [13] and IPPE [135] algorithms achieve the state-of-the-art results in terms of efficiency and precision for planar targets. Thus, we use these two algorithms as the pose estimator in the feature-based methods.

Given the ground-truth rotation matrix $\tilde{\mathbf{R}}$ and translation vector $\tilde{\mathbf{t}}$, we compute the rotation error $E_r(^\circ)$ of the estimated rotation matrix $\mathbf{R}$ in degrees according to (2.32). The translation error $E_t(\%)$ of the estimated translation vector $\mathbf{t}$ is measured by the relative difference between $\tilde{\mathbf{t}}$ and $\mathbf{t}$ according to (2.36). We define a pose to be successfully estimated if its both errors are under predefined thresholds. We use $\delta_r = 20^\circ$ and $\delta_t = 10\%$ as the thresholds on rotation error and translation error empirically, as shown in Figure 5.3. The success rate (SR) is defined as the percentage of the successfully estimated poses within each test condition. In the

doi:10.6342/NTU201800854

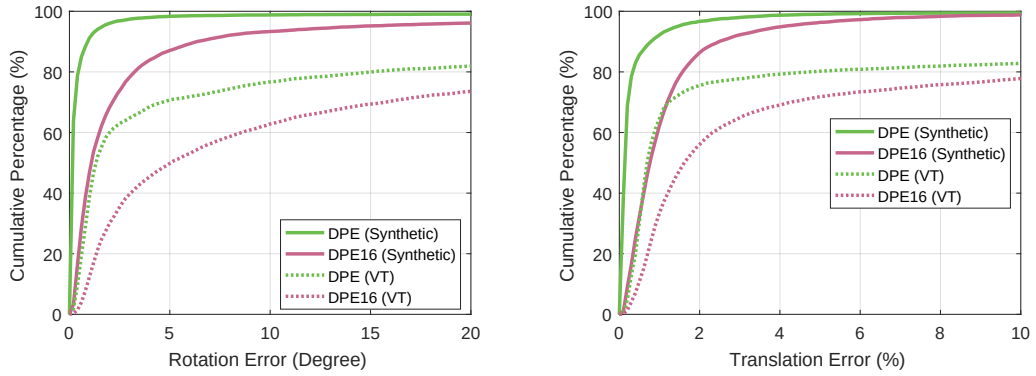


Figure 5.4: Cumulative percentage of poses whose rotation or translation errors are under thresholds specified in the x-axis over experiments on the same datasets used by [3], i.e., the proposed synthetic dataset and the visual tracking dataset built by Gauglitz *et al.* [4].

following sections, the average rotation and translation errors are computed only for successfully estimated poses.

We compare the DPE algorithm proposed in this work with the algorithm proposed in the previous work (i.e., DPE16) [3] on the same datasets [3]. Figure 5.4 shows that the proposed DPE algorithm performs accurately and robustly against the DPE16 method. For presentation clarify, we do not show the evaluation results of the DPE16 method in the following sections.

5.3.1 Synthetic Image Dataset

For our experiments, we use a set of synthetic images consisting of 8400 test images covering 21 different test conditions. Each test image is generated from warping a template image according to the randomly generated pose with the tilt angle in the range [0°, 75°] with a randomly chosen background image as illustrated in Figure 5.5. The template image size is 640×480 pixels. These templates are classified into four different classes, namely *low texture*, *repetitive texture*, *normal texture*, and *high texture* [164] as shown from top to bottom in Figure 5.5.

doi:10.6342/NTU201800854

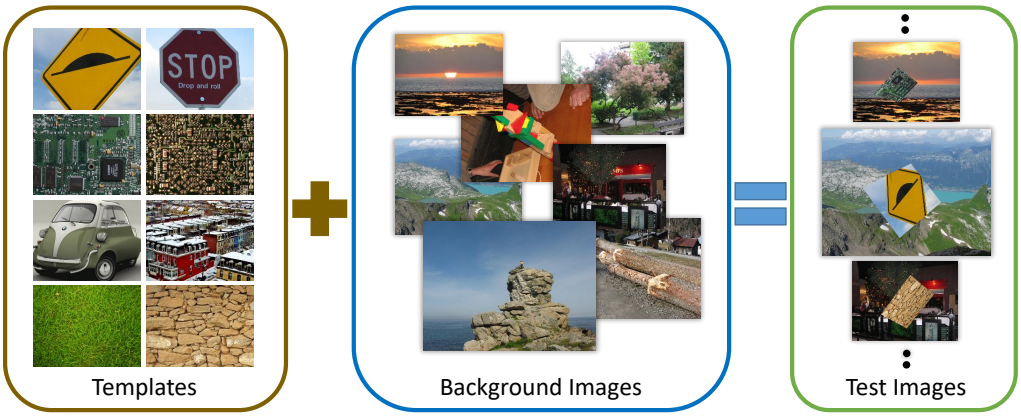


Figure 5.5: A synthetic test image was generated from a warping template image according to a randomly generated pose on a randomly chosen background image.

Each class is represented by two targets. The background images are from the database [184] and resized to 800×600 pixels.

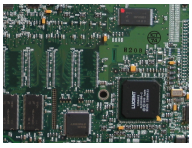
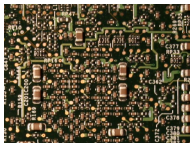
Undistorted Images. The pose estimation results of the SIFT-based, ASIFT-based, and proposed direct methods on the undistorted test images are shown in Table 5.3. For each image, the average rotation error E_r, translation error E_t, and success rate are presented. The evaluation results show that the proposed DPE method performs accurately and robustly against feature-based approaches on various template images. In addition, the proposed refinement approach can effectively improve accuracy that is first estimated by the APE method.

In most cases, the feature-based approaches do not estimate pose accurately on textureless template images or template images with feature points that are similar to each other. Although the IPPE algorithm is designed for pose estimation of planar objects, it does not perform as well as the OPnP algorithm that is able to estimate pose more accurately in general scenarios.

Degraded Images. We evaluate these approaches using all templates with different types of image degradation: (a) *Gaussian Blur* with kernel width of {1, 2, 3, 4, 5} pixels, (b) *JPEG Compression* with the quality parameter set to {90, 80, 70, 60, 50},

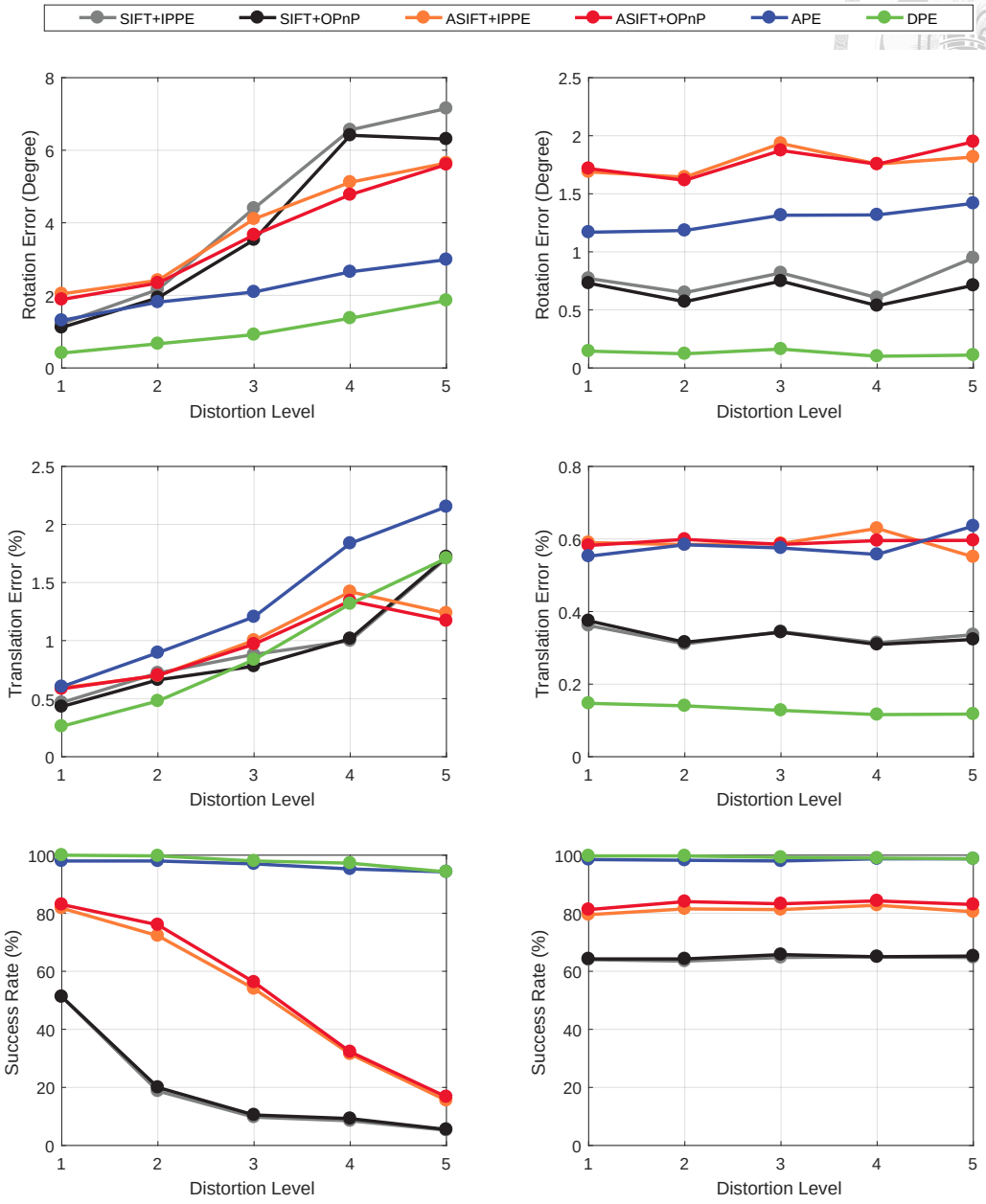
doi:10.6342/NTU201800854

Table 5.3: Evaluation results for feature-based approaches and the proposed direct methods with undistorted test images in terms of average numbers of rotation error E_r, translation error E_t, and success rate in each test condition. The best values are highlighted in bold.

	Bump Sign			Stop Sign			Lucent			MacMini Board		
												
Method	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)
SIFT+IPPE	0.85	0.34	*40.0*	1.90	0.54	*86.0*	0.23	0.25	*28.0*	0.32	0.24	*86.0*
SIFT+OPnP	0.76	0.40	*40.0*	1.18	0.46	*86.0*	0.20	0.24	*28.0*	0.25	0.24	*86.0*
ASIFT+IPPE	9.70	2.92	*20.0*	2.96	0.81	*94.0*	1.48	0.43	***100***	1.65	0.51	*94.0*
ASIFT+OPnP	8.20	2.22	*22.0*	2.72	0.74	***100***	1.38	0.41	***100***	1.53	0.45	*96.0*
APE	1.10	0.33	***100***	1.44	0.42	***100***	0.90	0.47	*98.0*	2.56	1.23	*94.0*
DPE	**0.39**	**0.17**	***100***	**0.42**	**0.24**	***100***	**0.16**	**0.14**	***100***	**0.16**	**0.12**	***98.0***
	Isetta			Philadelphia			Grass			Wall		
												
Method	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)
SIFT+IPPE	0.74	0.35	*92.0*	0.56	0.40	*98.0*	1.15	0.50	*30.0*	0.28	0.37	*96.0*
SIFT+OPnP	0.56	0.32	*92.0*	0.55	0.43	*98.0*	1.48	0.47	*30.0*	0.25	0.36	*96.0*
ASIFT+IPPE	1.59	0.57	***100***	1.29	0.34	*98.0*	2.17	0.52	*52.0*	1.96	0.36	*90.0*
ASIFT+OPnP	1.40	0.50	*98.0*	1.26	0.35	***100***	1.33	0.37	*52.0*	1.80	0.36	*94.0*
APE	1.03	0.35	***100***	1.63	0.49	***100***	1.96	0.91	***100***	1.57	0.68	*98.0*
DPE	**0.21**	**0.16**	***100***	**0.21**	**0.11**	***100***	**0.15**	**0.14**	***100***	**0.17**	**0.13**	***100***

(c) *Intensity Change* with pixel intensity scale factor set to {0.9,0.8,0.7,0.6,0.5}, and (d) *Tilt Angle* in the range of {[0°, 15°), [15°, 30°), [30°, 45°), [45°, 60°), and [60°, 75°)}. Figure 5.6 and Figure 5.7 shows the evaluation results. The proposed DPE algorithm performs favorably against the other feature-based methods on blurry images. Although the translation errors of the proposed method appear to be larger than those of feature-based methods, these errors are computed only on successfully estimated poses. As the proposed method can estimate template

doi:10.6342/NTU201800854



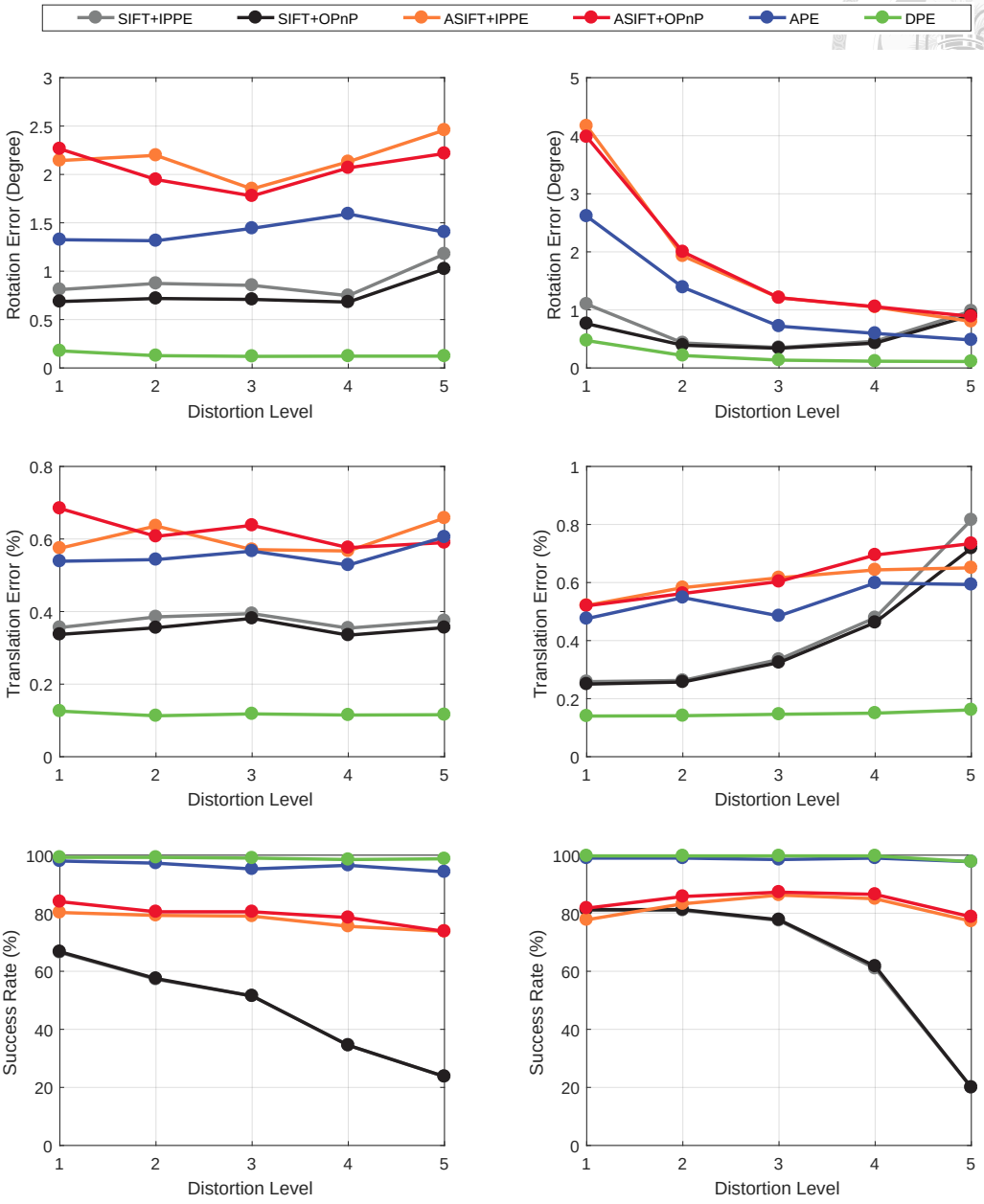
(a) Gaussian Blur (b) JPEG Compression

Figure 5.6: Experimental results on synthetic data under (a) *Gaussian Blur* and (b) *JPEG Compression* conditions.

poses successfully even under blur conditions, the errors are larger due to slightly inaccurate pose estimates in blurry images.

All approaches are able to deal with certain levels of distortion with JPEG

doi:10.6342/NTU201800854



(a) Intensity Change (b) Tilt Angle

Figure 5.7: Experimental results on synthetic data under (a) *Intensity Change* and (b) *Tilt Angle* conditions.

compression noise.

For images with intensity changes, the SIFT-based methods perform worse than other approaches as fewer features are detected in low contrast images by the

doi:10.6342/NTU201800854

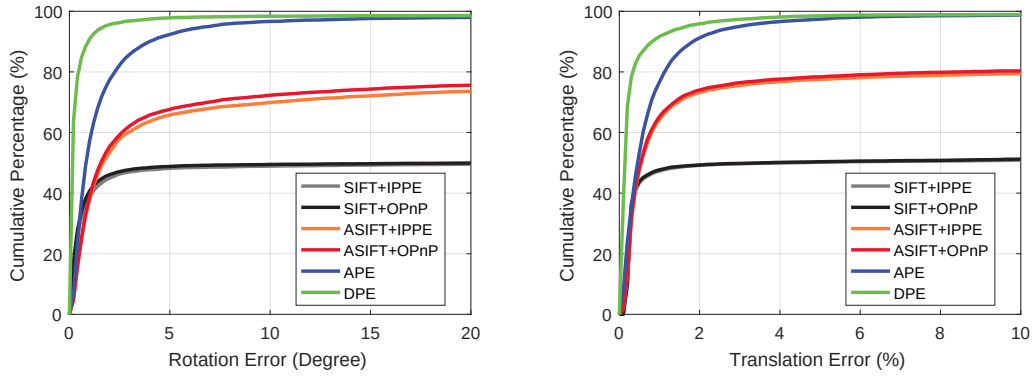


Figure 5.8: Cumulative percentage of poses whose rotation or translation errors are under thresholds specified in the x-axis over experiments on the proposed synthetic image dataset. There is a total of 8400 poses estimated by each pose estimation approach.

SIFT detector. We note that the SIFT-based methods can still perform well under low-intensity conditions when we adjust the feature detection threshold to extract more features.

Although the SIFT-based approaches can detect and match features accurately under small tilt angles, these methods frequently fail when the tilt angles are larger. In contrast, the proposed algorithm and the ASIFT-based methods are able to estimate 6DoF poses relatively well even the template images are perspectively distorted in the camera images.

We show the overall evaluation results on the proposed synthetic image dataset in Figure 5.8. Overall, the proposed direct method performs favorably against the feature-based approaches with the success rate of 98.90%. The success rate of the SIFT-based and ASIFT-based approaches are 49.65% and 74.26%, respectively.

Refinement Analysis. To improve pose estimation accuracy, we propose a refinement method that minimizes the appearance distance between the template and camera images using an LK-based scheme as described in Section 5.2. Figure 5.9 shows pose estimation results with and without the refinement approach on the

doi:10.6342/NTU201800854

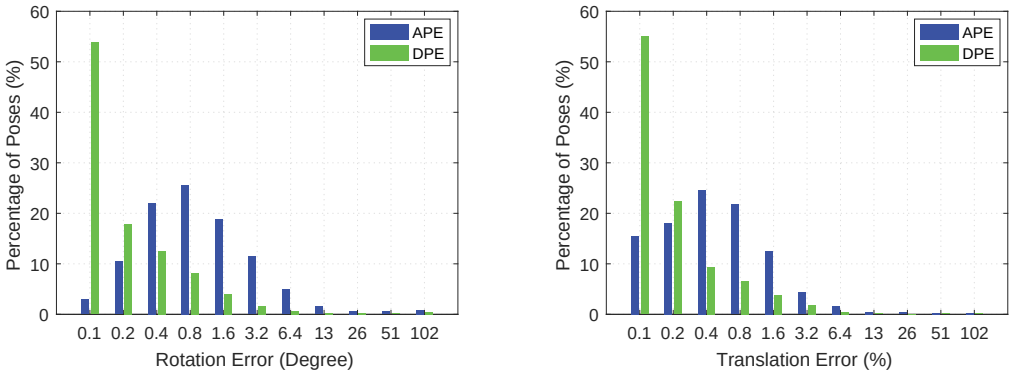


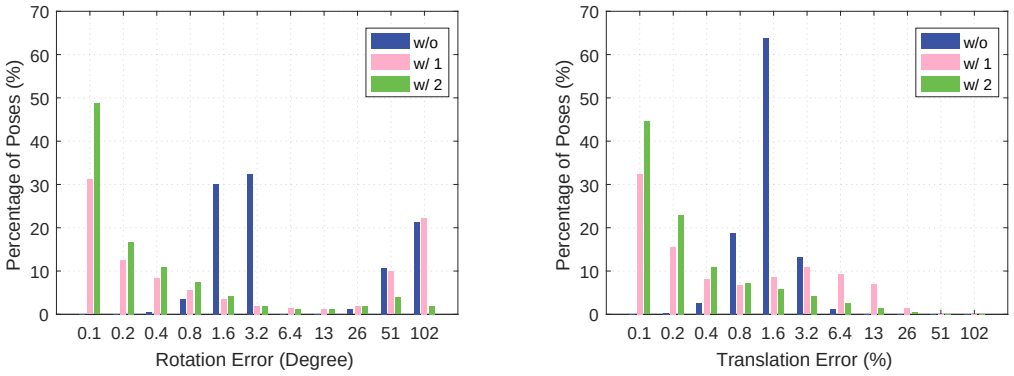
Figure 5.9: Pose estimation results with refinement approach (DPE) and without refinement approach (APE). The average value of rotation and translation errors are both reduced by the proposed refinement approach.

synthetic dataset. The rotation and translation errors can be reduced by 1.951° and 0.670% respectively with proposed refinement scheme. Sample images rendered with poses estimated by the proposed algorithm with and without the refinement scheme on the synthetic image dataset are shown in Figure 1.2.

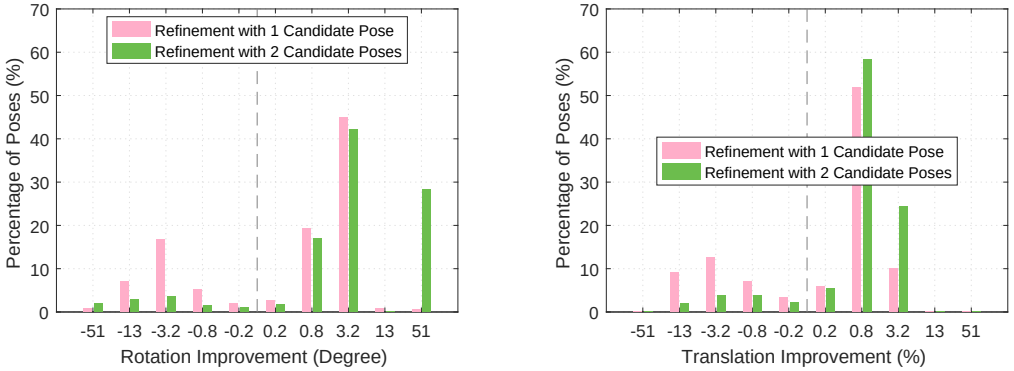
We design another experiment to demonstrate the proposed algorithm is able to disambiguate plausible poses. A template image from the synthetic dataset is warped according to pose $\mathbf{p}_t$. Two ambiguous pose, $\mathbf{p}_{a_1}$ and $\mathbf{p}_{a_2}$, can be obtained from $\mathbf{p}_t$ using the functional minimization method [13]. One of the two plausible poses $\mathbf{p}'_a$ is randomly chosen and added with some Gaussian noise. The refinement approach is then applied to $\mathbf{p}'_a$ for estimating the pose of the warped template image. Finally, we compute E_r and E_t of both the initial noisy pose $\mathbf{p}'_a$ and the refined pose $\mathbf{p}_r$ according to $\mathbf{p}_t$. Thus, if the proposed refinement approach can disambiguate the plausible pose $\mathbf{p}'_a$, the rotation error can be reduced significantly. All images in the synthetic dataset are used for the experiment.

We compare the proposed refinement method with the refinement approach with only one candidate pose in Algorithm 1, and present the results in Figure 5.10. While the rotation errors of ambiguous poses are usually large (which causes the pose flipping), the proposed refinement approach can disambiguate the object pose

doi:10.6342/NTU201800854



(a) Distribution of errors



(b) Improvements

Figure 5.10: Results of the proposed method without refinement (w/o), refinement with one candidate (w/ 1), and refinement with two candidates (w/ 2). (a) The rotation errors are reduced significantly in the ambiguous cases, but the translation errors are relatively not because the translation terms of ambiguous poses are quite similar in most cases. (b) The difference of pose errors before and after applying two kinds of refinement approaches. While the proposed refinement approach can disambiguate the object pose effectively, approach with only one candidate pose suffers from the risk of getting trapped into a local minimum.

effectively and reduce the rotation errors significantly (which result in smoother pose estimations throughout an image sequence). Table 5.4 shows that the proposed refinement method can help improve estimation accuracy in terms of rotation and translation and address the pose ambiguity problem effectively.

doi:10.6342/NTU201800854

Table 5.4: Evaluation results for different pose refinement approaches on the synthetic image dataset in the refinement analysis experiment.

Approach	$E_r(^\circ)$	$E_t(\%)$	SR(%)
Without refinement	2.235	1.369	*66.82*
Refinement with 1 candidate pose	0.734	0.461	*65.49*
Refinement with 2 candidate poses	**0.558**	**0.416**	***92.05***

5.3.2 Visual Tracking Dataset

We analyze the performance of the proposed algorithm and state-of-the-art methods on the visual tracking (VT) dataset [4] which contains 96 videos and 6889 frames with 6 templates. These videos are recorded under different moving and lighting conditions with motion-blurs. The camera image size in this dataset is 640×480 pixels. And since the templates have different primary resolutions, we resize each template to 570×420 pixels uniformly. It is a challenging database for pose estimation due to significant viewpoint changes, drastic illumination differences, and noisy camera images.

The evaluation results of the proposed and feature-based methods on six templates under different conditions are shown in Table 5.5 and Table 5.6. Different from synthetic images, the color appearance of a template image may change significantly within a video sequence in this real image dataset. The DPE algorithm performs favorably against the feature-based methods under most conditions, especially when distinct features cannot be found on a template image.

While PnP algorithms perform well in pose estimation, the success hinges on whether the feature can be well matched. As shown in Figure 5.11, feature-based approaches do not perform well when motion blurs occur. Similarly, feature-based methods do not estimate pose well on videos listed in Table 5.5 and Table 5.6 due to motion blurs. On the other hand, the proposed algorithm can estimate poses well under blur conditions. As motion blurs are likely to occur in AR applications,

doi:10.6342/NTU201800854

Table 5.5: Experimental results on the visual tracking dataset [4] under *Unconstrained*, *Panning*, and *Rotation* conditions. The best results (excluding the proposed direct pose tracking method) for each condition are highlighted in bold.

		Bricks			Building			Mission			Paris			Sunset			Wood		
																			
Condition	Method	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)
Unconstrained	SIFT+IPPE	2.98	1.07	*0.40*	2.60	0.89	*6.60*	1.64	0.72	*60.6*	1.61	0.66	*44.0*	3.22	**1.32**	*26.6*	2.04	0.53	*5.00*
	SIFT+OPnP	2.37	0.98	*0.40*	2.60	0.88	*6.80*	1.48	0.73	*61.6*	1.44	0.65	*43.8*	**2.81**	1.43	*28.4*	1.43	**0.41**	*5.00*
	ASIFT+IPPE	2.67	1.05	*37.0*	2.80	0.92	*31.4*	3.03	1.11	*58.0*	1.57	0.72	*91.0*	3.40	2.10	*44.4*	2.54	1.03	*33.0*
	ASIFT+OPnP	1.92	0.93	*37.6*	2.48	0.88	*31.4*	2.35	0.91	*57.4*	1.31	0.85	*91.0*	3.03	2.11	*46.0*	2.20	1.01	*34.8*
	APE	2.12	1.45	***52.0***	1.11	0.92	*88.0*	1.57	1.14	*98.0*	0.75	0.76	*99.0*	4.24	3.89	*43.4*	1.41	2.24	*53.0*
	DPE	**1.00**	**0.72**	***52.0***	**0.93**	**0.66**	***88.4***	**1.17**	**0.70**	***98.2***	**0.47**	**0.54**	***99.8***	3.49	4.12	***46.8***	**0.67**	1.53	***56.0***
	DPT	1.11	0.94	*90.2*	1.23	0.81	*92.2*	1.17	0.70	*98.4*	0.81	0.65	*96.0*	3.69	4.46	*49.9*	0.88	1.43	*91.2*
Panning	SIFT+IPPE	–	–	*0.00*	1.29	0.55	*10.0*	2.18	0.65	*96.0*	3.34	1.04	*40.0*	5.49	0.75	*20.0*	–	–	*0.00*
	SIFT+OPnP	–	–	*0.00*	0.81	**0.54**	*10.0*	2.39	**0.61**	***100***	4.45	1.12	*50.0*	6.79	1.27	*24.0*	–	–	*0.00*
	ASIFT+IPPE	5.91	1.52	*80.0*	7.89	0.98	*2.00*	4.95	1.15	*82.0*	5.85	**0.91**	*44.0*	13.8	2.56	*10.0*	9.94	1.42	*4.00*
	ASIFT+OPnP	5.80	1.40	*80.0*	19.2	5.59	*2.00*	4.95	0.98	*88.0*	5.93	1.16	*62.0*	15.5	3.34	*10.0*	16.0	2.02	*4.00*
	APE	4.27	0.50	***96.0***	1.56	1.08	***100***	1.79	0.96	***100***	3.94	1.05	*74.0*	4.53	**0.56**	***100***	6.03	1.24	*56.0*
	DPE	**1.04**	**0.29**	***96.0***	**0.38**	0.63	***100***	**0.90**	0.89	***100***	**1.52**	0.97	***86.0***	**2.75**	0.68	***100***	**1.05**	**0.93**	***60.0***
	DPT	1.64	0.36	*95.9*	0.38	0.64	*100*	0.95	0.89	*100*	1.51	0.98	*100*	2.68	0.71	*100*	1.33	0.92	*79.6*
Rotation	SIFT+IPPE	1.65	0.34	*44.0*	2.79	**0.50**	*56.0*	1.17	0.42	***100***	1.71	0.37	*98.0*	5.97	0.57	*74.0*	3.76	0.40	*62.0*
	SIFT+OPnP	1.74	0.37	*46.0*	2.69	0.52	*56.0*	1.05	0.41	***100***	1.61	0.33	***100***	5.61	0.66	*84.0*	2.51	**0.39**	*70.0*
	ASIFT+IPPE	2.83	0.39	***100***	6.15	1.24	*76.0*	2.35	**0.36**	***100***	1.35	0.33	***100***	6.68	0.91	*72.0*	3.39	0.47	*94.0*
	ASIFT+OPnP	1.78	0.39	***100***	5.09	1.11	*74.0*	1.66	0.37	***100***	1.23	0.36	***100***	5.69	0.90	*78.0*	2.88	0.49	*98.0*
	APE	1.20	0.25	***100***	2.00	0.66	***100***	1.11	0.42	***100***	0.71	**0.25**	***100***	2.18	0.66	***100***	1.64	0.66	***100***
	DPE	**0.84**	**0.24**	***100***	**1.50**	0.59	***100***	**0.31**	0.46	***100***	**0.56**	0.29	***100***	**0.90**	**0.52**	***100***	**0.98**	0.55	***100***
	DPT	0.84	0.24	*100*	1.50	0.59	*100*	0.32	0.46	*100*	0.55	0.29	*100*	0.88	0.51	*100*	0.99	0.54	*100*

doi:10.6342/NTU201800854

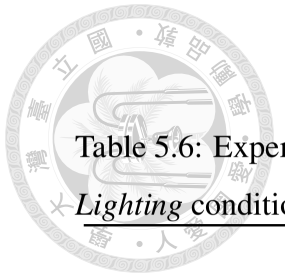


Table 5.6: Experimental results on the visual tracking dataset [4] under *Perspective Distortion*, *Zoom*, *Static Lighting*, and *Dynamic Lighting* conditions. The best results (excluding the proposed direct pose tracking method) for each condition are highlighted in bold.

		Bricks			Building			Mission			Paris			Sunset			Wood		
Condition	Method	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)
Perspective Distortion	SIFT+IPPE	2.99	0.46	*58.0*	4.38	**0.40**	*34.0*	2.77	0.43	*76.0*	3.98	0.40	*76.0*	6.56	0.87	*58.0*	4.70	0.59	*20.0*
	SIFT+OPnP	1.45	0.30	*58.0*	2.62	0.45	*34.0*	**0.68**	0.53	*76.0*	1.53	0.45	*76.0*	4.79	0.74	*62.0*	6.23	**0.43**	*24.0*
	ASIFT+IPPE	3.01	**0.25**	***72.0***	4.99	0.43	*68.0*	3.74	**0.35**	*80.0*	3.07	**0.34**	*84.0*	4.96	**0.64**	*58.0*	3.69	0.75	*66.0*
	ASIFT+OPnP	1.55	0.29	***72.0***	3.51	0.54	*68.0*	1.95	0.39	*80.0*	1.78	0.51	*84.0*	3.73	0.87	*62.0*	2.07	0.82	*66.0*
	APE	1.81	0.94	*56.0*	0.97	0.77	***92.0***	1.35	0.56	***86.0***	0.69	0.42	***90.0***	2.44	2.32	*68.0*	1.74	1.34	***68.0***
	DPE	**0.89**	0.29	*56.0*	**0.74**	0.51	***92.0***	0.81	0.52	***86.0***	**0.43**	0.46	***90.0***	**1.47**	1.96	***78.0***	**0.56**	0.86	***68.0***
	DPT	0.72	0.34	*93.9*	0.71	0.51	*100*	0.84	0.61	*95.9*	0.57	0.68	*98.0*	1.61	1.63	*75.5*	0.62	1.13	*87.8*
Zoom	SIFT+IPPE	2.51	0.53	*6.00*	3.28	0.34	*26.0*	4.01	0.42	***100***	3.09	0.40	***100***	9.75	0.94	*60.0*	4.23	0.45	*40.0*
	SIFT+OPnP	1.15	0.38	*6.00*	3.14	0.30	*28.0*	2.30	**0.40**	*98.0*	2.73	0.43	***100***	7.42	0.91	*60.0*	2.83	0.46	*42.0*
	ASIFT+IPPE	4.91	0.76	*64.0*	4.60	0.56	*58.0*	5.24	0.67	*76.0*	2.54	**0.20**	*74.0*	10.5	1.05	*50.0*	4.10	**0.43**	*48.0*
	ASIFT+OPnP	3.32	0.65	*64.0*	3.95	0.52	*58.0*	3.36	0.48	*80.0*	1.67	0.36	*76.0*	6.47	1.18	*56.0*	4.33	0.50	*54.0*
	APE	3.37	0.77	***94.0***	1.73	0.33	***100***	3.13	0.63	***100***	1.22	0.55	***100***	5.58	**0.74**	***100***	3.79	1.06	***100***
	DPE	**1.14**	**0.33**	***94.0***	**0.86**	**0.27**	***100***	**1.94**	0.51	***100***	**0.50**	0.45	***100***	**2.50**	0.80	***100***	**0.87**	0.61	***100***
	DPT	1.16	0.33	*100*	0.87	0.27	*100*	1.98	0.51	*100*	0.52	0.45	*100*	2.43	0.80	*100*	0.93	0.58	*100*
Static Lighting	SIFT+IPPE	1.51	0.83	*27.5*	2.75	0.98	*20.0*	1.09	0.48	*81.3*	1.56	0.79	*72.5*	2.28	**0.87**	*57.5*	1.01	**0.50**	*21.3*
	SIFT+OPnP	1.49	0.91	*28.7*	2.42	1.18	*20.0*	0.77	**0.43**	*81.3*	1.58	0.86	*72.5*	**1.94**	0.91	*60.0*	1.00	0.52	*21.3*
	ASIFT+IPPE	1.20	**0.81**	***75.0***	2.77	0.88	*42.5*	1.43	0.48	***100***	1.28	**0.65**	***100***	2.66	1.73	*47.5*	1.80	0.58	*52.5*
	ASIFT+OPnP	**1.09**	0.82	***75.0***	2.41	0.82	*42.5*	1.27	0.45	***100***	1.23	0.76	***100***	2.45	1.59	*62.5*	1.46	0.58	*52.5*
	APE	1.75	1.44	*71.3*	0.90	0.50	***100***	0.95	0.60	***100***	1.24	0.72	***100***	2.97	3.59	*81.3*	1.61	1.85	***85.0***
	DPE	1.20	1.06	*71.3*	**0.85**	**0.40**	***100***	**0.61**	0.51	***100***	**1.03**	0.68	***100***	2.24	2.44	***82.5***	**0.94**	0.78	***85.0***
	DPT	1.20	1.05	*100*	0.85	0.39	*100*	0.61	0.51	*100*	1.02	0.68	*100*	2.85	3.13	*100*	0.91	0.72	*100*
Dynamic Lighting	SIFT+IPPE	1.38	0.41	*13.0*	1.81	0.89	*17.0*	1.16	0.55	*78.0*	1.12	0.47	*38.0*	1.45	**0.67**	*44.0*	1.08	**0.42**	*28.0*
	SIFT+OPnP	1.37	0.43	*13.0*	1.59	0.90	*17.0*	0.98	0.58	*77.0*	1.13	0.52	*38.0*	**1.29**	0.70	*48.0*	1.01	0.43	*28.0*
	ASIFT+IPPE	1.22	**0.36**	*62.0*	2.81	1.10	*38.0*	1.53	0.54	***100***	0.95	0.48	***100***	3.31	1.33	*47.0*	1.79	0.56	*48.0*
	ASIFT+OPnP	1.14	0.38	***63.0***	3.01	1.15	*37.0*	1.42	0.55	***100***	0.92	0.53	***100***	2.60	1.33	*48.0*	1.47	0.59	*51.0*
	APE	1.25	0.71	*40.0*	**1.06**	0.68	***98.0***	0.99	0.70	***100***	0.65	**0.33**	*84.0*	3.26	3.10	*72.0*	1.26	1.31	***52.0***
	DPE	**1.00**	0.47	*40.0*	1.20	**0.65**	***98.0***	**0.47**	**0.52**	***100***	**0.63**	0.41	*84.0*	2.75	3.19	***77.0***	**0.82**	0.72	***52.0***
	DPT	1.00	0.45	*100*	1.20	0.66	*100*	0.46	0.52	*100*	0.63	0.42	*100*	3.29	3.67	*100*	0.81	0.63	*100*

doi:10.6342/NTU201800854

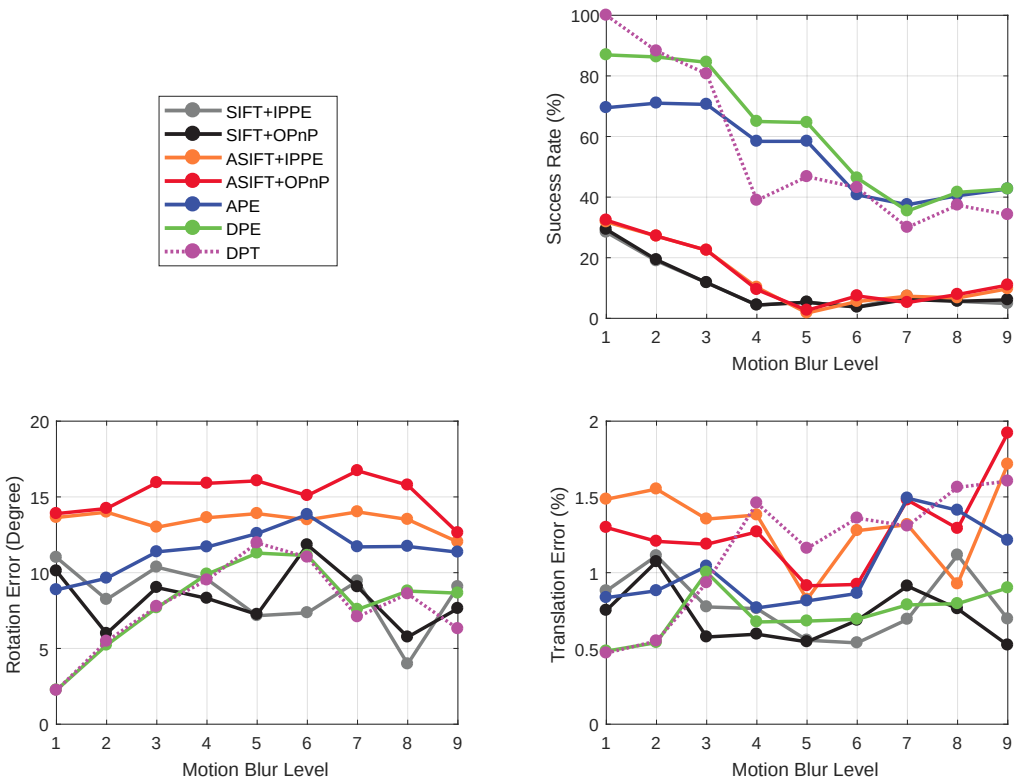
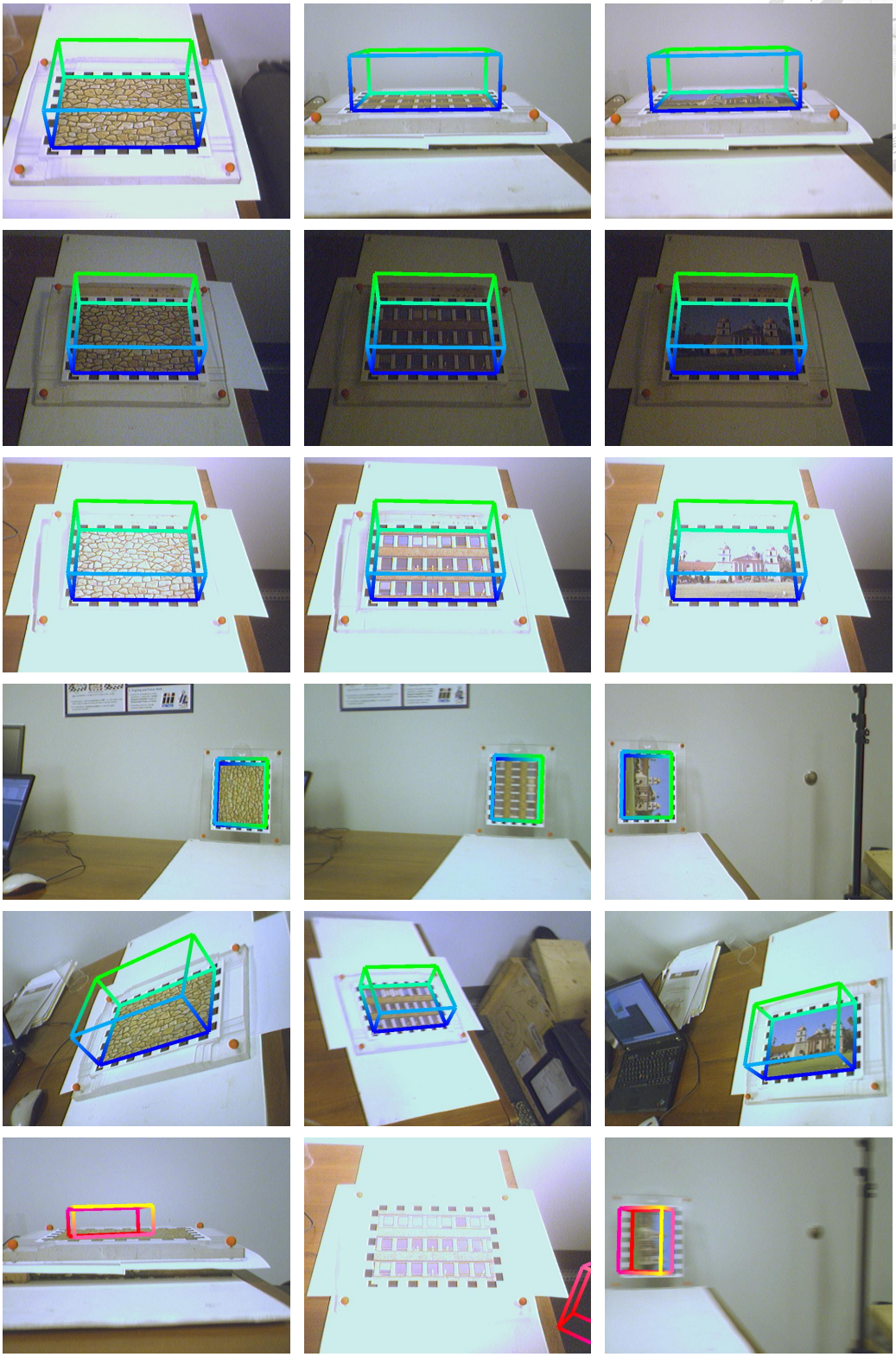


Figure 5.11: Experimental results on the visual tracking dataset [4] under varying motion blur levels, where level 9 stands for the strongest motion blur.

the proposed algorithm can be better applied to estimate 6DoF pose than feature-based approaches. However, if the target object appears an extremely flat color in a camera image, the proposed method is likely to fail because the appearance between the template and its local patches are almost indistinguishable.

Sample pose estimation results from the proposed DPE method are shown in Figure 5.12 and Figure 5.13, in which the success cases are represented with rendered cyan boxes, and the failure cases are represented with rendered magenta boxes. The cumulative percentage of estimated poses according to different translation and rotation errors are shown in Figure 5.14. Overall, the proposed direct method performs favorably against the feature-based approaches within the success rate of 77.76%. The success rate of the SIFT-based and ASIFT-based approaches are 29.98% and 48.52% respectively.

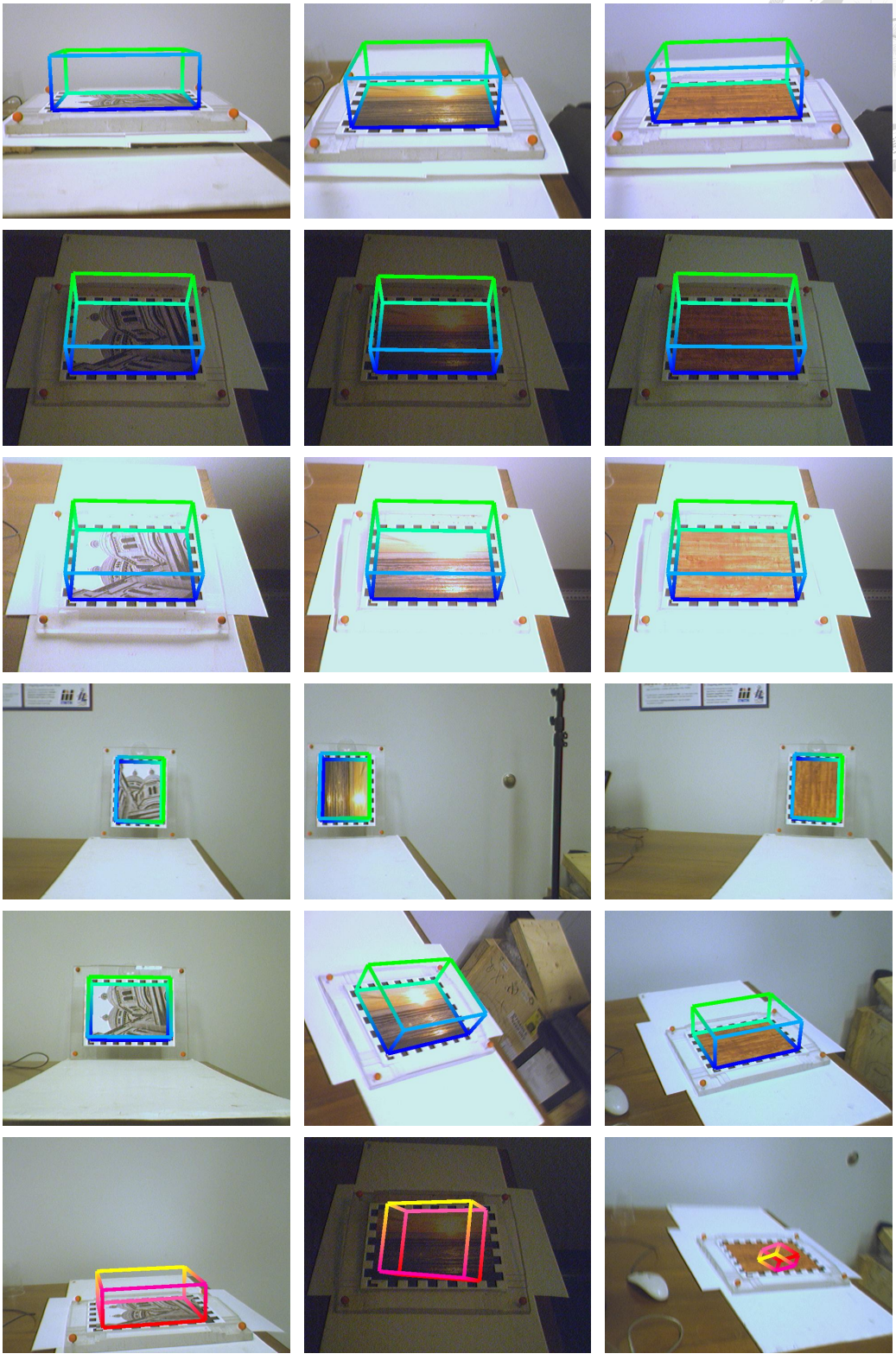
doi:10.6342/NTU201800854



(a) Bricks (b) Building (c) Mission

Figure 5.12: Estimation results by the proposed DPE method on the visual tracking dataset [4] under different conditions.

doi:10.6342/NTU201800854



(a) Paris (b) Sunset (c) Wood

Figure 5.13: Estimation results by the proposed DPE method on the visual tracking dataset [4] under different conditions.

doi:10.6342/NTU201800854

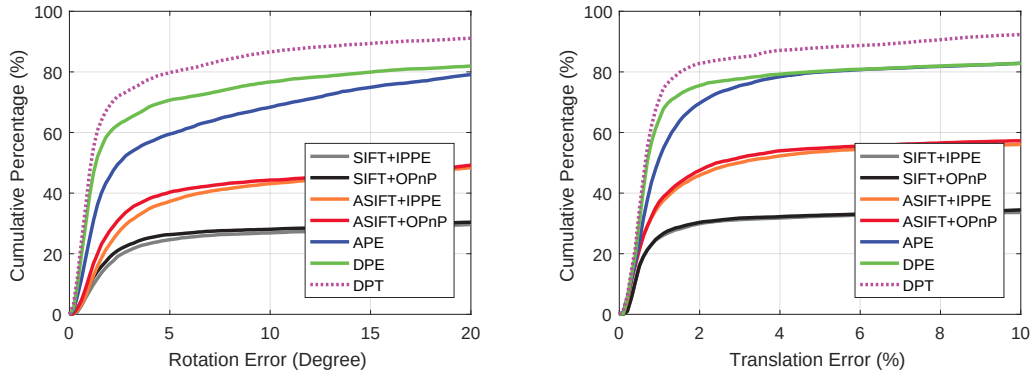


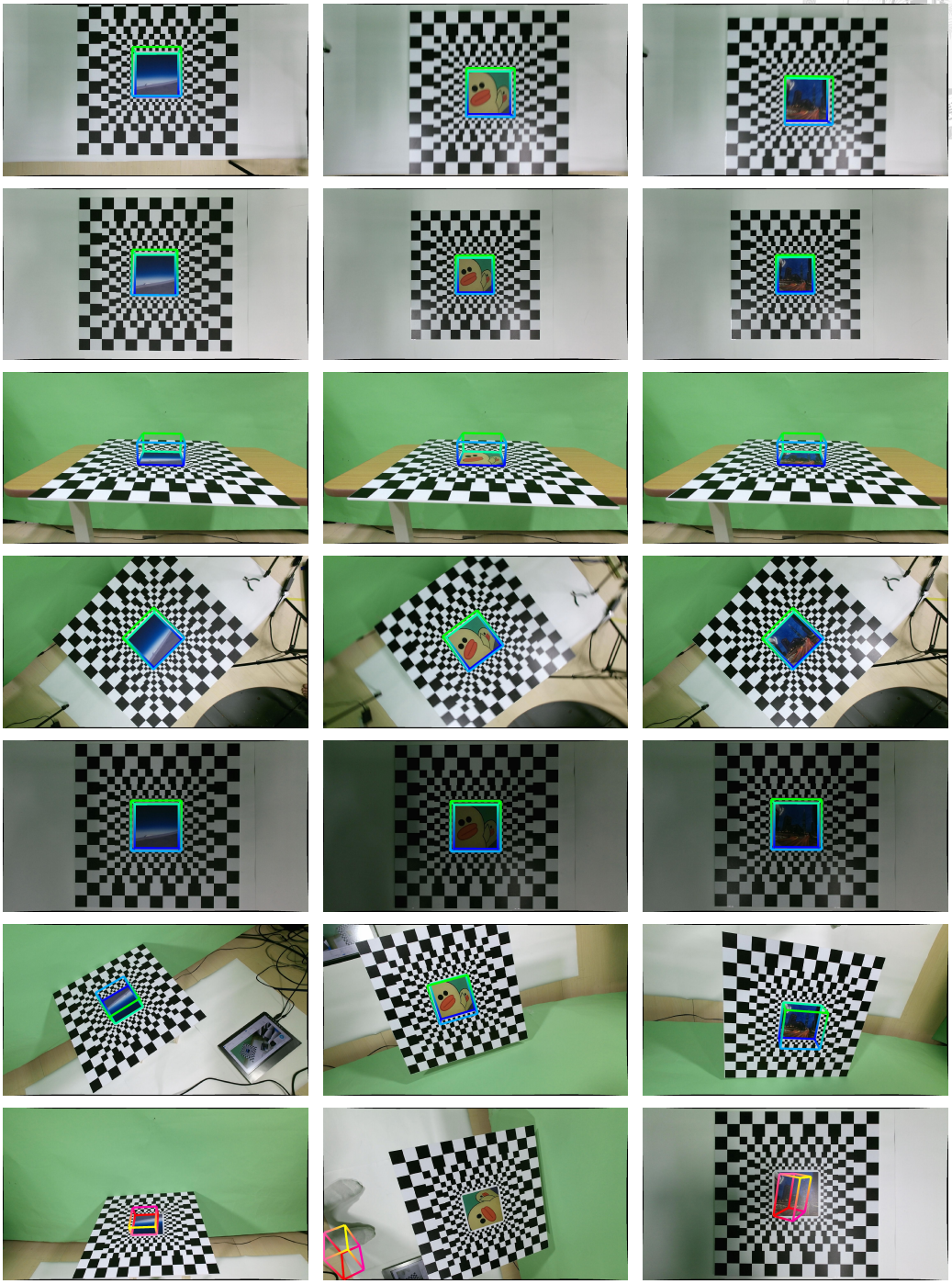
Figure 5.14: Cumulative percentage of poses whose rotation or translation errors are under thresholds specified in the x-axis over experiments on the visual tracking dataset [4]. There is a total of 6889 poses estimated by each pose estimation approach.

Note that the proposed pose refinement approach can also be regarded as a direct pose tracking (DPT) algorithm. The evaluation results of the DPT method on the VT dataset are shown in Table 5.5, Table 5.6, Figure 5.11, and Figure 5.14. If the DPT method loses track of the object pose (namely the rotation or translation error is larger than the pre-defined threshold, i.e., δ_r and δ_t), we reset the initial object pose in the current frame as the object pose in the previous frame. Overall, the proposed DPT method can track object poses well. The DPT algorithm can be integrated with the DPE method for more robust performance with specific re-initialization schemes (e.g., periodic restarts).

5.3.3 Object Pose Tracking Dataset

We evaluate the proposed algorithm and feature-based methods on the object pose tracking (OPT) benchmark dataset presented in Chapter 4. For 2D objects, it contains 138 videos with 20,988 frames. Sample images rendered according to the pose estimated by the proposed DPE method on this OPT dataset are shown in Figure 5.15 and Figure 5.16, where the success cases are represented with

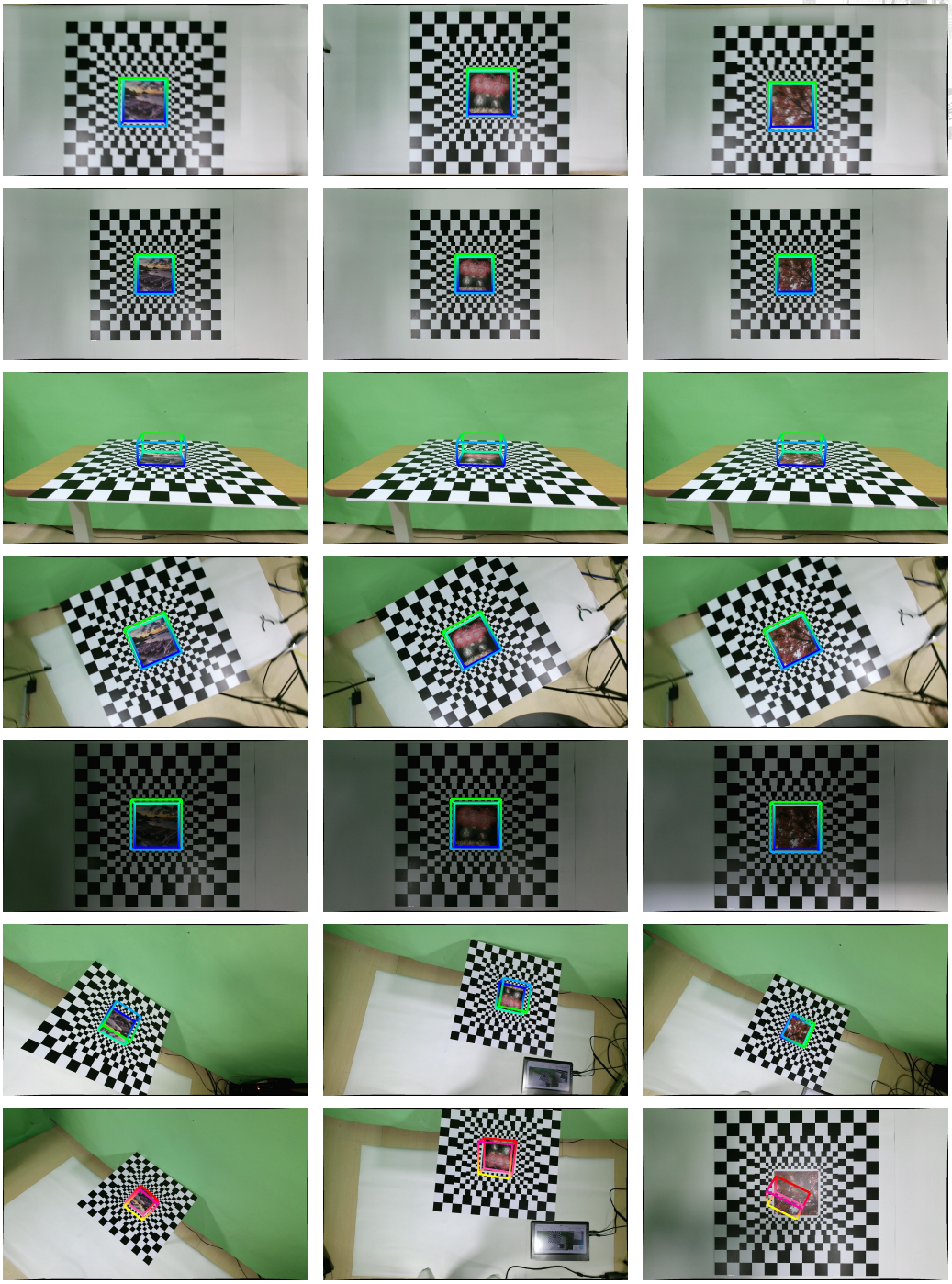
doi:10.6342/NTU201800854



(a) Wing (b) Duck (c) City

Figure 5.15: Estimation results by the proposed DPE method on the OPT dataset presented in Chapter 4 under different conditions.

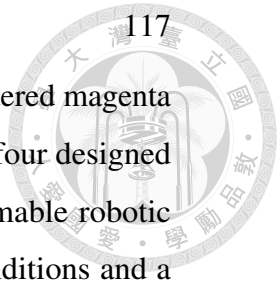
doi:10.6342/NTU201800854



(a) Beach (b) Firework (c) Maple

Figure 5.16: Estimation results by the proposed DPE method on the OPT dataset presented in Chapter 4 under different conditions.

doi:10.6342/NTU201800854



rendered cyan boxes, and the failure cases are represented with rendered magenta boxes. We note that videos in the OPT dataset are recorded under four designed motion patterns and five camera speeds controlled by a programmable robotic arm. Furthermore, these videos contain two different lighting conditions and a free-motion case. The frame size in this dataset is 1920×1080 pixels, and we resize each template to 300×300 pixels.

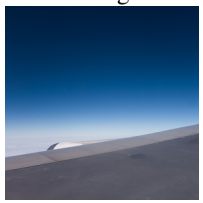
The pose tracking results of all evaluated algorithms under *Flashing Light*, *Moving Light*, and *Free Motion* conditions with six templates and different texture levels are shown in Table 5.7. Similar to the results in Section 5.3.1 and Section 5.3.2, feature-based methods do not perform well on the template images with less texture or structure. In contrast, the proposed DPE method is able to track object poses well except the *Wing* image. When a template image does not contain sufficient structural information, the proposed direct method may estimate erroneous poses which cover only parts of the template image, as shown in the failure cases in Figure 5.15 and Figure 5.16. The proposed method does not perform well on images when drastic color distortion occurs, e.g., under *Moving Light* condition, as the appearance distance metric is less effective in such scenarios.

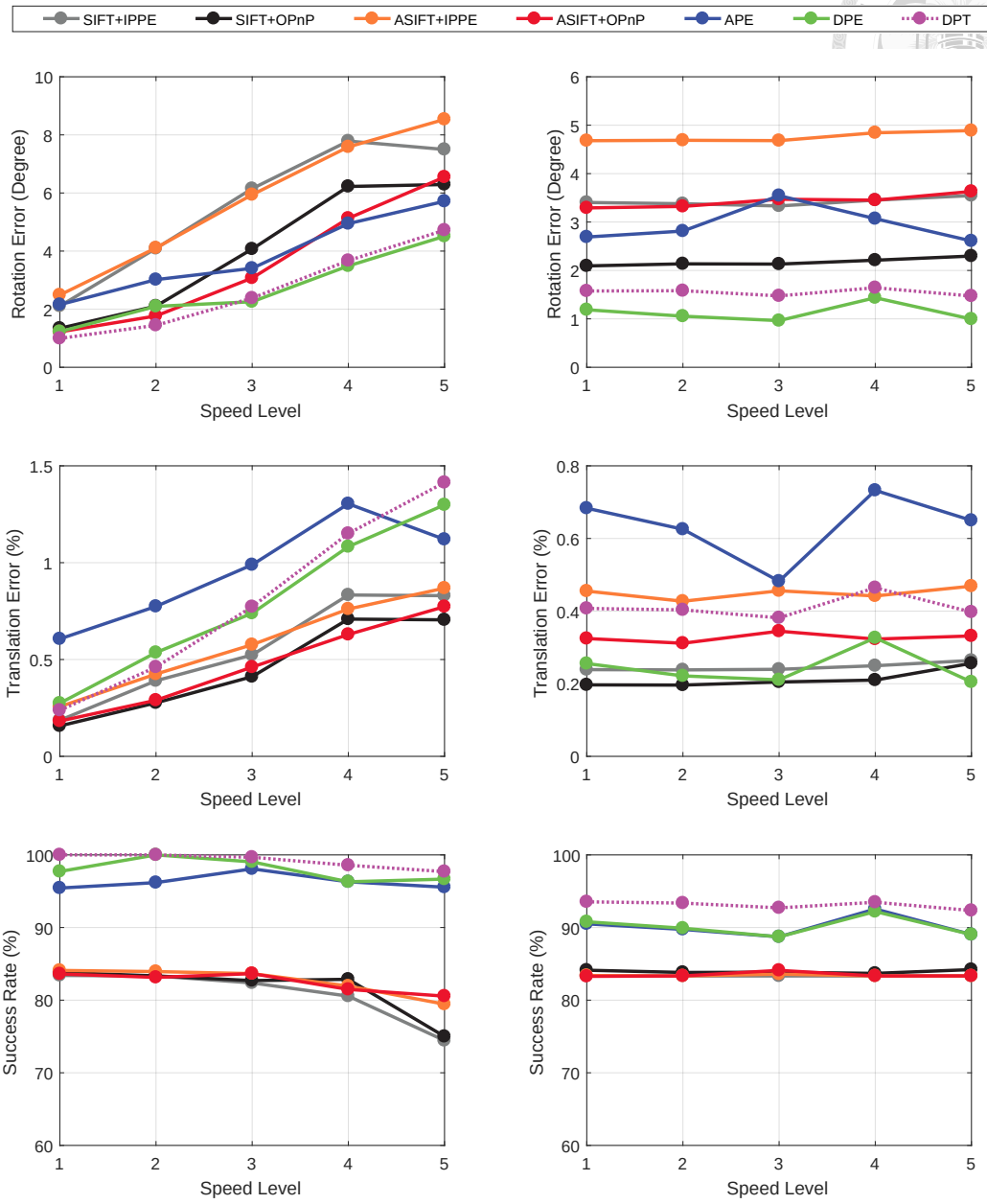
The pose tracking results of the template images in different motion patterns and speed are shown in Figure 5.17 and Figure 5.18. Since the images in the *Translation* condition are more blurry than those in other motion patterns at higher speed, the plot trends of the evaluation results under this condition are similar to those under the *Gaussian Blur* conditions in Figure 5.6. In contrast, the other three motion patterns do not result in blurry images at the highest speed, the performance of all approaches under conditions at different speeds are similar. As all the evaluated approaches are scale and rotation invariant, they all perform favorably on template images with the *Zoom* and *In-plane Rotation* patterns. However, the success rates of SIFT-based methods are lower in the *Out-of-plane Rotation* motion pattern as they are not invariant under perspective distortion.

We evaluate the proposed DPT algorithm on the OPT dataset to analyze the

doi:10.6342/NTU201800854

Table 5.7: Experimental results on the OPT dataset under different conditions. The best results (excluding the proposed direct pose tracking method) for each condition are highlighted in bold.

		Wing			Duck			City			Beach			Firework			Maple		
																			
Condition	Method	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)	$E_r(°)$	$E_t(\%)$	SR(%)
Flashing Light	SIFT+IPPE	10.6	2.96	*1.24*	6.99	0.43	***100***	2.11	0.20	***100***	2.80	0.16	***100***	1.80	0.15	***100***	1.63	0.12	***100***
	SIFT+OPnP	14.2	2.33	*9.32*	5.94	0.33	***100***	**0.86**	**0.10**	*99.4*	0.83	0.09	***100***	**0.23**	0.07	***100***	**0.35**	0.08	***100***
	ASIFT+IPPE	14.6	3.27	*4.35*	6.36	0.50	***100***	3.01	0.28	***100***	1.79	0.19	***100***	2.73	0.25	***100***	2.43	0.22	***100***
	ASIFT+OPnP	17.5	2.84	*3.11*	3.58	0.34	***100***	1.47	0.16	***100***	0.88	0.18	***100***	0.91	0.16	***100***	1.35	0.12	***100***
	APE	10.4	**1.51**	*36.0*	2.12	0.22	***100***	1.95	0.56	***100***	1.28	0.28	***100***	2.00	0.34	***100***	1.98	0.41	***100***
	DPE	**8.32**	1.52	***42.2***	**0.72**	**0.05**	***100***	1.08	0.19	***100***	**0.50**	**0.09**	*99.4*	0.38	**0.04**	***100***	0.50	**0.05**	*98.1*
	DPT	6.46	1.72	*86.3*	0.76	0.05	*99.4*	1.16	0.17	*100*	0.47	0.09	*100*	0.43	0.05	*100*	0.56	0.05	*97.5*
Moving Light	SIFT+IPPE	17.8	0.69	*0.61*	7.54	0.63	*94.5*	2.52	0.22	***100***	2.60	0.15	***100***	1.64	0.13	***100***	1.87	0.15	***100***
	SIFT+OPnP	15.2	2.75	*8.54*	5.95	0.50	*94.5*	**1.02**	**0.11**	*100*	**0.69**	**0.09**	*100*	**0.22**	0.07	***100***	**0.55**	**0.09**	***100***
	ASIFT+IPPE	15.6	2.97	*1.83*	7.13	0.61	***100***	4.71	0.41	*99.4*	1.74	0.20	***100***	2.68	0.27	***100***	2.42	0.20	***100***
	ASIFT+OPnP	19.4	**0.38**	*0.61*	5.10	0.46	***100***	2.73	0.29	*99.4*	0.84	0.15	***100***	0.80	0.18	***100***	0.98	0.12	***100***
	APE	11.3	4.98	*27.4*	4.24	0.37	*99.4*	5.43	0.71	*55.5*	3.64	0.35	*75.0*	3.26	0.54	*95.1*	6.09	1.03	*62.2*
	DPE	**8.41**	4.38	***45.1***	**2.14**	**0.12**	***100***	2.34	0.18	*56.7*	1.51	0.09	*77.4*	0.71	**0.04**	*94.5*	3.20	0.32	*59.8*
	DPT	9.22	1.92	*64.4*	1.96	0.11	*100*	2.59	0.19	*98.8*	1.42	0.10	*99.4*	0.76	0.05	*100*	4.08	0.37	*82.2*
Free Motion	SIFT+IPPE	7.55	3.95	*1.15*	5.80	0.59	*93.2*	1.00	0.28	***100***	0.61	0.42	*99.9*	1.38	0.39	***100***	0.73	0.39	***100***
	SIFT+OPnP	9.81	2.87	*2.04*	3.68	0.57	*96.8*	0.77	0.27	***100***	0.61	0.41	***100***	1.09	0.38	***100***	0.72	0.38	***100***
	ASIFT+IPPE	11.6	**2.54**	*0.38*	7.89	1.18	*90.6*	2.43	0.39	*99.4*	0.95	0.53	*99.9*	1.78	0.39	*98.7*	1.45	0.49	*96.4*
	ASIFT+OPnP	11.4	5.38	*1.15*	6.53	0.90	*96.7*	2.03	0.36	*99.7*	0.91	0.52	***100***	1.55	0.36	*99.7*	1.39	0.49	*99.9*
	APE	6.14	5.16	*56.1*	2.73	0.31	***98.7***	1.35	0.66	***100***	1.53	0.86	*83.7*	1.79	0.55	***100***	3.18	1.98	*98.3*
	DPE	**4.84**	4.41	***59.7***	**1.16**	**0.23**	***98.7***	**0.60**	**0.18**	***100***	**0.54**	**0.27**	*91.1*	**1.05**	**0.30**	***100***	**0.65**	**0.34**	*99.1*
	DPT	4.52	3.14	*69.5*	0.88	0.18	*100*	0.55	0.22	*100*	0.49	0.26	*99.6*	1.02	0.30	*100*	0.58	0.26	*100*

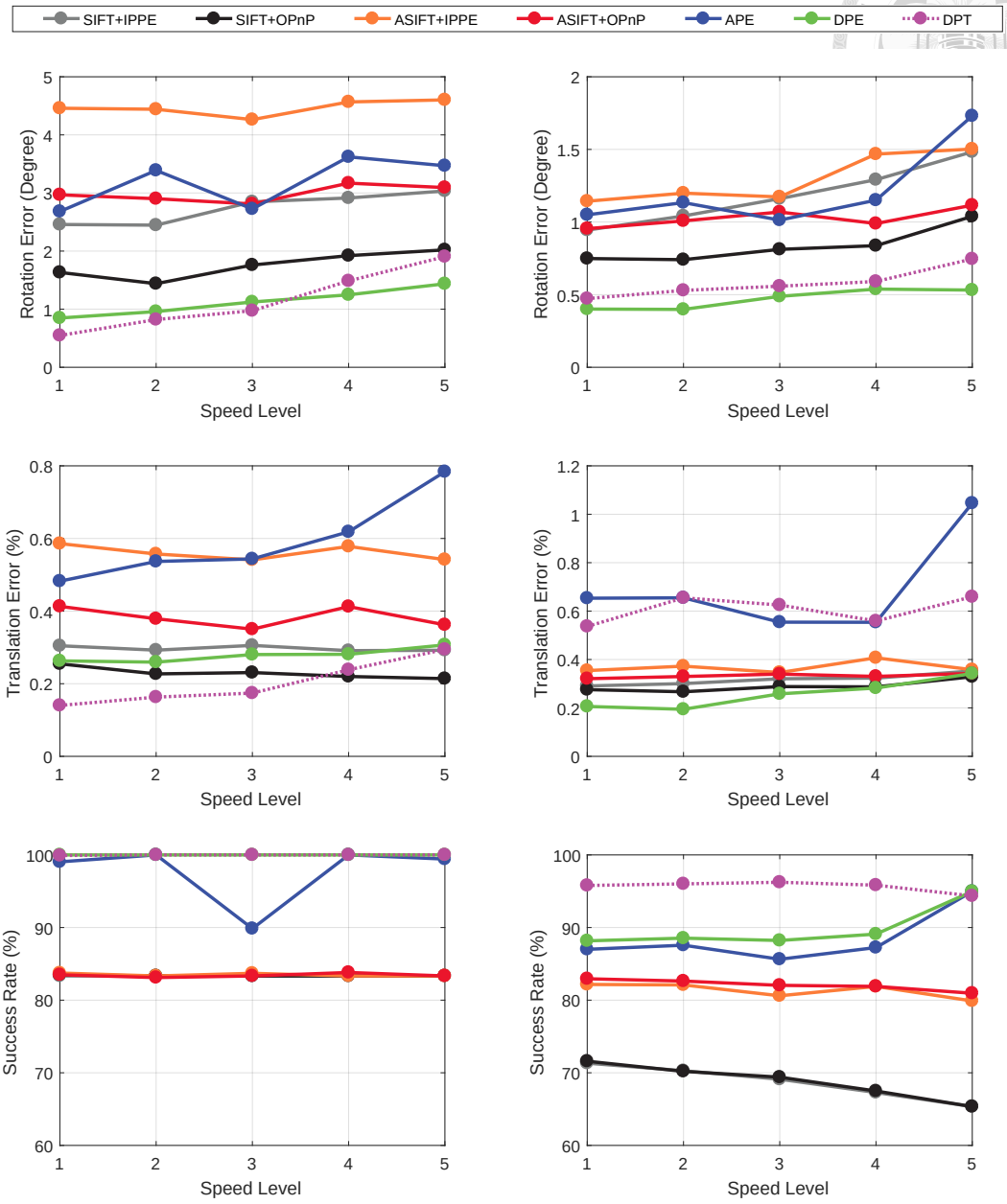


(a) Translation (b) Zoom

Figure 5.17: Experimental results on the OPT dataset in motion patterns (a) *Translation* and (b) *Zoom* with different speeds.

tracking performance using the same experimental setting as that described in Section 5.3.2, Figure 5.17, Figure 5.18, and Table 5.7 show that the DPT algorithm can track object poses well on most template images except one. As discussed

doi:10.6342/NTU201800854



(a) In-plane Rotation (b) Out-of-plane Rotation

Figure 5.18: Experimental results on the OPT dataset in motion patterns (a) *In-plane Rotation* and (b) *Out-of-plane Rotation* with different speeds.

above, the proposed DPT method does not work well on images, e.g., *Wing*, without sufficient structure for pose estimation based on appearance. The curves of cumulative percentages of poses estimated by the evaluated algorithms on the OPT

doi:10.6342/NTU201800854

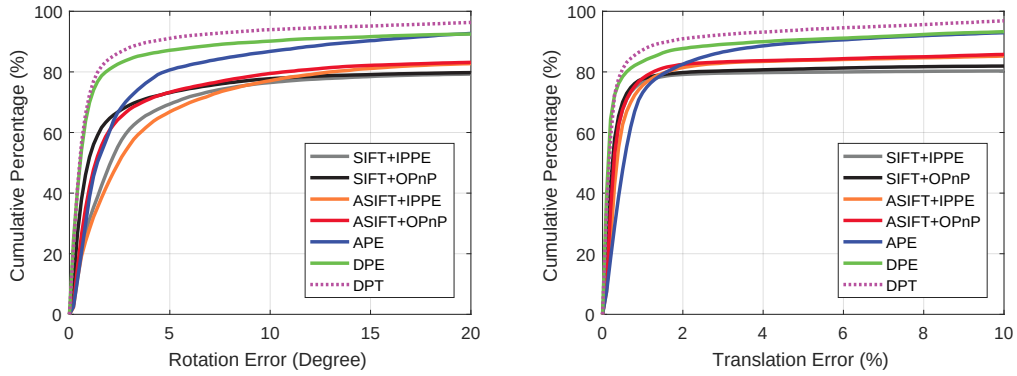


Figure 5.19: Cumulative percentage of poses whose rotation or translation errors are under thresholds specified in the x-axis over experiments on the OPT dataset. There is a total of 20,988 poses estimated by each pose estimation approach.

dataset are shown in Figure 5.19. Overall, the proposed direct method performs favorably against feature-based approaches with a success rate of 91.27%. The success rates of the SIFT-based and ASIFT-based approaches are 79.46% and 82.74%, respectively.

5.4 Summary

In this work, we propose a robust direct method for 6DoF pose estimation based on two main steps. First, the pose of a planar target with respect to a calibrated camera is approximately estimated using an efficient coarse-to-fine scheme. Next, we use the LK-based method to further refine and disambiguate the object pose. Extensive experimental evaluations on both synthetic image and real image datasets demonstrate the proposed algorithm performs favorably against two state-of-the-art feature-based pose estimation approaches in terms of robustness and accuracy under several varying conditions. We have also implemented the proposed algorithm on a GPGPU platform as the algorithm can be easily parallelized.

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

Chapter 6

DodecaPen: Accurate 6DoF Tracking of a Passive Stylus

In this work, we explore a simpler hardware setup that uses a minimal amount of electronics to achieve high accuracy tracking. We propose a system that requires only a single off-the-shelf camera and a passive 3D-printed fiducial with several hand-glued binary square markers printed from a laser printer, as shown in Figure 1.3. Our proposed system has the distinct advantage of ease-of-construction and setup over electronically instrumented solutions. Because there are no electronics (including LEDs) on the stylus, threading wires or charging batteries are not a concern. Neither lasers nor active illumination is required. The only requirements are the use of a 2D office printer, a 3D printer, some glue, and a global shutter camera. Because we need only a single camera, it can be mounted casually on a tripod placed on the user's desk, without concern for re-calibration of multiple cameras. Despite these constraints, we achieve an accuracy of 0.4 mm at 60Hz over a 30×40 cm^2 working area, which is comparable to state-of-the-art professional motion capture (mocap) systems.

The overview of the proposed system is illustrated in Figure 6.1. Given a target object $\mathcal{O}_t$ (the DodecaPen in this work) represented by a dense surface model (triangle mesh) and a camera image $\mathcal{I}_c$, the task is to determine the 6DoF object

doi:10.6342/NTU201800854

doi:10.6342/NTU201800854

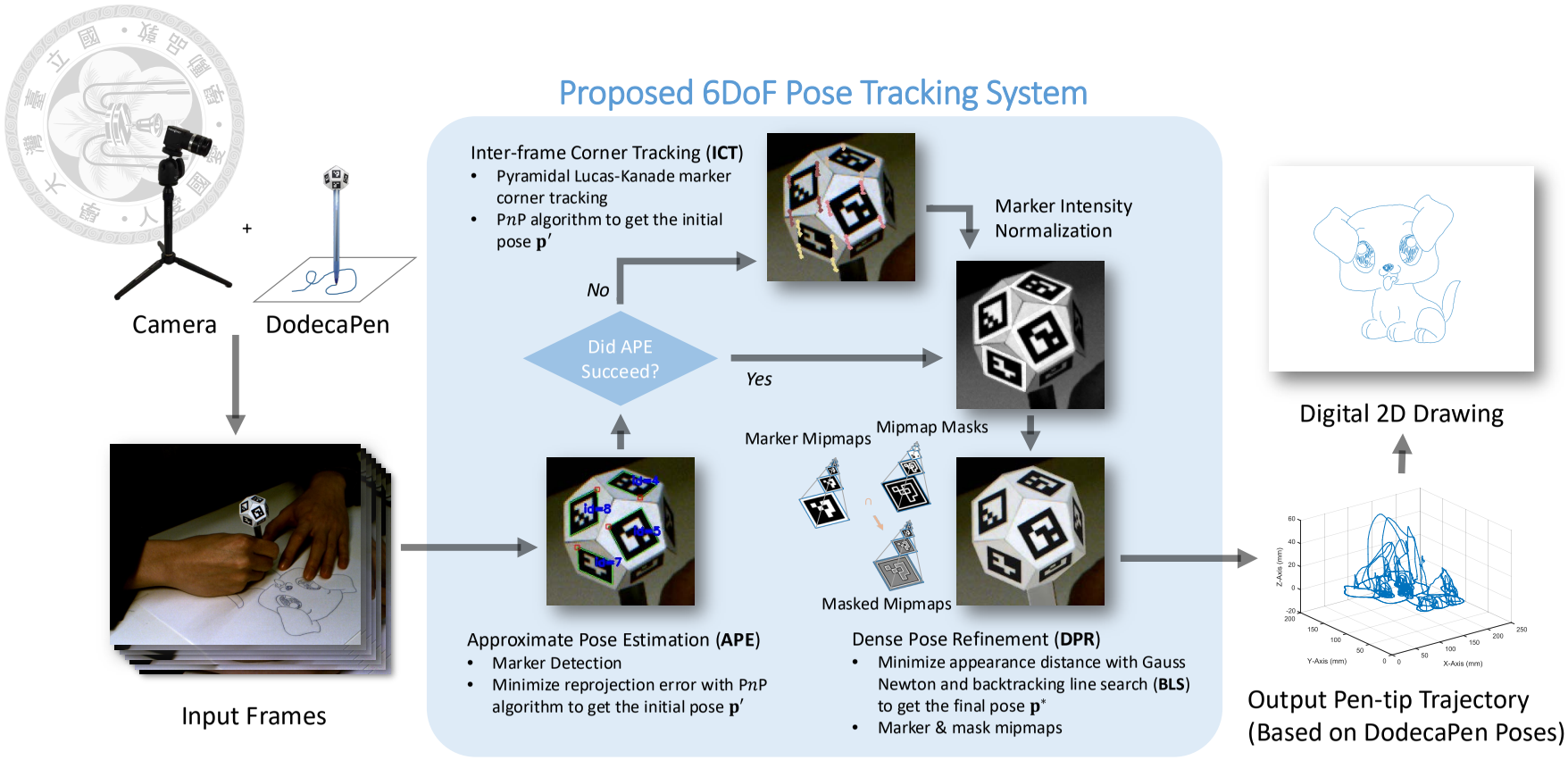


Figure 6.1: System overview. In the *approximate pose estimation* step, we detect the binary square fiducial markers in the input images and estimate the 6DoF pose of the DodecaPen using the PnP algorithm. If fewer than two markers are detected, we use the LK method to track marker corners between frames. In the *dense pose refinement* step, the pose $\mathbf{p}'$ is refined by minimizing the appearance distance between the 3D model of the DodecaPen and image pixels to get the final pose $\mathbf{p}^*$. We generate the pen-tip trajectory in the 3D view from the computed 6DoF pose sequence and visualize the 2D drawing by removing points where the pen tip is lifted off the page.

pose $\mathbf{p}$ of $\mathcal{O}_t$ relative to the camera. Let $\mathbf{x}_i = [x_i, y_i, z_i]^\top, i = 1, \ldots, n, n \geq 3$ be a set of reference points in the local object-space of $\mathcal{O}_t$, and let $\mathbf{u}_i = [u_i, v_i]^\top$ be the corresponding 2D image-space coordinates of $\mathcal{I}_c$. The relationship between them can be obtained using camera projection formulated in (2.2). In this study, the pose $\mathbf{p}$ is formulated as a 6D vector consisting of the 3D rotation vector and the 3D translation vector (which is presented in Section 2.1.2):

$$\mathbf{p} = \begin{bmatrix} \mathbf{r} \\ \mathbf{t} \end{bmatrix}, \ \mathbf{r} = \begin{bmatrix} r_x \\ r_y \\ r_z \end{bmatrix} \in \mathbb{R}^3, \ \mathbf{t} = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \in \mathbb{R}^3. \tag{6.1}$$

The proposed 6DoF pose tracking system primarily comprises two phases: approximate pose estimation (APE) and dense pose refinement (DPR), Once we have computed the 6DoF pose of the dodecahedron, we can recover the pen-tip trajectory and use it to reconstruct the drawing. We rigorously evaluate the performance of the proposed system when we degrade the camera (with shot noise, spatial blur, and reduced spatial resolution). We conclude with demonstrations of this accurate and easy-to-setup 6DoF tracking system for the application of drawing in 2D and 3D as well as object manipulation in a VR environment.

6.1 Dodecahedron Design

Although binary square fiducial markers are commonly attached to cubes [149, 185], pose recovery can fail when only a single marker is visible due to an ambiguity in the PnP problem [3]. By substituting a dodecahedron as the tracked object, we ensure that at least two planes are visible in most cases, eliminating the ambiguity. Despite the fact that there are still other *regular solids* (or *Platonic solids*), as shown in Figure 6.2, the area ratio of a square marker to a triangle face for the other three (i.e., tetrahedron, octahedron, and icosahedron) would be too small for the proposed system to track decently, as illustrated in Figure 6.3. Moreover, for a tetrahedron, its faces captured by a tracking system may be too few as well.

doi:10.6342/NTU201800854

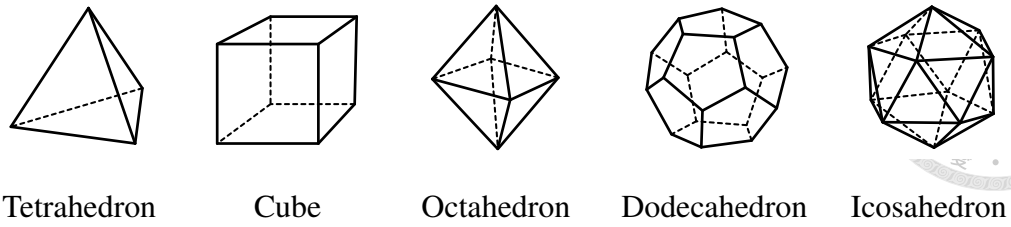


Figure 6.2: Five regular solids in 3D space.

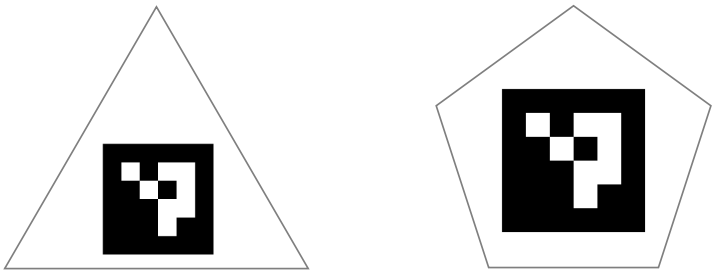


Figure 6.3: The area ratio of a square marker to a triangle face is much smaller than that to a pentagon face.

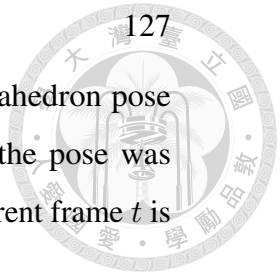
Consequently, the dodecahedron seems to be the best fit for our system.

We use an off-the-shelf 3D printer to create our trackable dodecahedron. Each edge of the resulting dodecahedron is 12.9 mm in length, while the markers glued on its surface have edges of length 10.8 mm and are printed with a laser printer. Each marker is generated with the ArUco library [151] and is encoded as a 6×6 grid where the external cells are set as black.

6.2 Approximate Pose Estimation

We first use the binary square fiducial marker detector provided in the ArUco library [150] to detect markers in input images. This gives us an image-space position and orientation of each marker on the dodecahedron. We use these to recover the 6DoF dodecahedron pose $\mathbf{p}$ by minimizing the *reprojection error* formulated in (2.3). This is a standard PnP problem, which we minimize using the Levenberg-Marquardt method [186]. To accelerate the marker detection process,

doi:10.6342/NTU201800854



we use a constant acceleration motion model to predict the dodecahedron pose and constrain ArUco's search region for the fiducial markers if the pose was successfully recovered in the frame. The predicted pose $\hat{\mathbf{p}}_t$ in the current frame t is computed with the information from the last frame $t-1$:

$$\hat{\mathbf{p}}_t = \mathbf{p}_{t-1} + \dot{\mathbf{p}}_{t-1} + \frac{1}{2}\ddot{\mathbf{p}}_{t-1}, \tag{6.2}$$

where $\dot{\mathbf{p}}$ and $\ddot{\mathbf{p}}$ are the pose velocity and acceleration between frames, respectively. The search region for the current frame is set to be four times the area of the dodecahedron in the last frame to account for fast motion.

6.3 Inter-frame Corner Tracking

We occasionally find that the APE method fails due to motion blur or because most of the markers are strongly tilted relative to the camera. Because PnP cannot work reliably in the case where we detect fewer than two markers, we apply the inter-frame corner tracking (ICT) scheme to generate more constraints for PnP. We use the pyramidal LK optical flow tracker [187] to track the corners of the markers from frame to frame.

Square markers can be challenging for optical flow algorithms because different corners have a very similar appearance, and thus the pyramidal LK implementation frequently finds incorrect correspondences. Therefore, we perform the tracking in two rounds. In the first round, we track each visible marker separately in the camera frame and compute the velocity vectors of each marker by differencing with the previous frame. We reject markers whose velocity is further than three standard deviations from the mean. We then initialize the marker corner tracker using the trusted predictions from the first round and run the tracking for the four corners of each remaining marker a second time with similar outlier removal strategy. The resulting motion tracks are much more reliable.

doi:10.6342/NTU201800854

6.4 Dense Pose Refinement

Unfortunately, the initial pose $\mathbf{p}'$ computed using PnP is too jittery to use in tracking the pen tip. We can substantially improve the pose accuracy using a *dense alignment*, which minimizes the appearance distance E_{a_2} between the image $\mathcal{I}_c$ and the object $\mathcal{O}_t$ pixels across all of the visible marker points, as formulated in (2.5). We solve this nonlinear least squares problem using Gauss-Newton iteration; to approximate how the image changes with respect to pose, we approximate it using a first-order Taylor series as follows:

$$\begin{aligned}\Delta\mathbf{p}^* &= \underset{\Delta\mathbf{p}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{I}_c\left(\mathbf{u}_i\left(\mathbf{p}'+\Delta\mathbf{p}\right)\right)-\mathcal{O}_t\left(\mathbf{x}_i\right)\right)^2 \\ &\approx \underset{\Delta\mathbf{p}}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n}\left(\mathcal{I}_c\left(\mathbf{u}_i\left(\mathbf{p}'\right)\right)+\left.\frac{\partial\mathcal{I}_c}{\partial\mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}'}\Delta\mathbf{p}-\mathcal{O}_t\left(\mathbf{x}_i\right)\right)^2.\end{aligned} \tag{6.3}$$

To solve for $\Delta\mathbf{p}$ in each iteration, we set the first derivative of (6.3) equal to zero, and solve the resulting system of linear equations:

$$\mathbf{J}_c\Delta\mathbf{p} = \mathbf{O}_t - \mathbf{I}_c, \tag{6.4}$$

where $\mathbf{O}_t$ and $\mathbf{I}_c$ are vector forms of $\mathcal{O}_t\left(\mathbf{x}_i\right)$ and $\mathcal{I}_c\left(\mathbf{u}_i\right)$, respectively, and $\mathbf{J}_c$ is the Jacobian matrix of $\mathbf{I}_c$ with respect to $\mathbf{p}$ and is computed by the chain rule (in the numerator-layout notation):

$$\mathbf{J}_c = \left.\frac{\partial\mathbf{I}_c}{\partial\mathbf{p}}\right|_{\mathbf{p}=\mathbf{p}_c} = \begin{bmatrix}\frac{\partial\mathcal{I}_c(\mathbf{u}_1)}{\partial\mathbf{p}}\\ \frac{\partial\mathcal{I}_c(\mathbf{u}_2)}{\partial\mathbf{p}}\\ \vdots\\ \frac{\partial\mathcal{I}_c(\mathbf{u}_n)}{\partial\mathbf{p}}\end{bmatrix}, \tag{6.5}$$

$$\frac{\partial\mathcal{I}_c}{\partial\mathbf{p}} = \frac{\partial\mathcal{I}_c}{\partial\mathbf{u}}\left[\frac{\partial\mathbf{u}}{\partial\mathbf{r}},\frac{\partial\mathbf{u}}{\partial\mathbf{t}}\right] = \left[\frac{\partial\mathcal{I}_c}{\partial u},\frac{\partial\mathcal{I}_c}{\partial v}\right]\left[\frac{\partial\mathbf{u}}{\partial\hat{\mathbf{x}}}\frac{\partial\hat{\mathbf{x}}}{\partial\hat{\mathbf{R}}}\frac{\partial\hat{\mathbf{R}}}{\partial\mathbf{r}},\frac{\partial\mathbf{u}}{\partial\hat{\mathbf{x}}}\right], \tag{6.6}$$

$$\frac{\partial\mathbf{u}}{\partial\hat{\mathbf{x}}} = \begin{bmatrix}\frac{f_x}{\hat{z}} & 0 & -\frac{f_x\hat{x}}{\hat{z}^2}\\ 0 & \frac{f_y}{\hat{z}} & -\frac{f_y\hat{y}}{\hat{z}^2}\end{bmatrix}, \frac{\partial\hat{\mathbf{x}}}{\partial\hat{\mathbf{R}}} = \begin{bmatrix}x & y & z & 0 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & x & y & z & 0 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 0 & x & y & z\end{bmatrix}, \tag{6.7}$$

doi:10.6342/NTU201800854

where $\hat{\mathbf{R}} = [R_{11}, R_{12}, R_{13}, R_{21}, R_{22}, R_{23}, R_{31}, R_{32}, R_{33}]^\top$ denotes the vector with elements of the rotation matrix $\mathbf{R}$, and:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{z} \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & t_x \\ R_{21} & R_{22} & R_{23} & t_y \\ R_{31} & R_{32} & R_{33} & t_z \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{6.8}$$

is the camera-space coordinate transformed from the object-space coordinate $\mathbf{x} = [x, y, z]^\top$.

In addition, the derivative of $\hat{\mathbf{R}}$ with respect to $\mathbf{r}$ can be obtained using the following formula [182]:

$$\frac{\partial \mathbf{R}}{\partial r_a} = \frac{r_a [\mathbf{r}]_\times + [\mathbf{r} \times (\mathbf{I} - \mathbf{R}) \mathbf{e}_i]_\times}{\|\mathbf{r}\|^2} \mathbf{R}, \quad a = x, y, z, \tag{6.9}$$

where $\mathbf{I}$ and $\mathbf{e}_i$ are the identity matrix and the i-th vector of the standard basis in $\mathbb{R}^3$, respectively. In addition, $[\mathbf{r}]_\times$ represents the cross-product matrix for the vector $\mathbf{r}$ which is as defined in (2.17).

A closed form solution of (6.4) is:

$$\Delta \mathbf{p} = \left(\mathbf{J}_c^\top \mathbf{J}_c \right)^{-1} \mathbf{J}_c^\top \left(\mathbf{I}_t - \mathbf{I}_c \right). \tag{6.10}$$

Another way for getting $\Delta \mathbf{p}$ is to use the QR decomposition to solve (6.4), which would take more time to compute but are more numerically stable as well.

Because our least squares problem is nonlinear, Gauss-Newton iteration does not always converge with fixed step size. We thus perform a backtracking line search to scale the step size after each iteration of solving (6.4). We shrink $\Delta \mathbf{p}$ by $\Delta \mathbf{p} \leftarrow \alpha \Delta \mathbf{p}$ until it meets the Armijo-Goldstein condition below:

$$E_{a_2}(\mathbf{p} + \Delta \mathbf{p}) \leq E_{a_2}(\mathbf{p}) + c \nabla E_{a_2}(\mathbf{p})^\top \Delta \mathbf{p}, \tag{6.11}$$

where $\nabla E_{a_2}(\mathbf{p})$ is the local function gradient. We set $\alpha = 0.5$ and $c = 10^{-4}$ empirically.

To ensure intensity invariance and to minimize the residual between the model and image, we normalize the intensity first before solving the dense alignment

doi:10.6342/NTU201800854

problem above. We observe that the primary variation in intensity is due to the normal direction of each plane (and marker) as shown in Figure 6.1. Therefore we normalize the intensity per local marker.

To avoid aliasing effects, we also need to ensure that the model fiducial markers are resampled to be the same size they appear in the image. We generate a mipmap of the binary square fiducial markers ahead of time to enable efficient sampling of the model points at approximately the same scale as the image.

There are large portions of the square marker that do not significantly contribute to the error term, notably in regions of uniform intensity where $\nabla \mathcal{I}_c(\mathbf{u}_i) = 0$ and thus, $\frac{\partial \mathcal{I}_c}{\partial \mathbf{p}} = 0$. We take advantage of this by selectively masking out flat regions ahead of time on our marker as shown in Figure 6.1, dropping regions where $\nabla \mathcal{O}_t(\mathbf{x}_i) = 0$ and hence $\nabla \mathcal{I}_c(\mathbf{u}_i)$ is likely to be zero as well. The white and black colors of the masks in Figure 6.1 represent the active and non-active regions, respectively. The gray color of the final masked markers represent the non-active regions. We show that we can significantly accelerate the algorithm without compromising tracking quality using this masking technique.

6.5 Dodecahedron Calibration

While square markers are easy to print and glue on to the dodecahedron, the manual nature of this process necessarily results in the model error, leading to inaccurate pose tracking results, as we show in Section 6.7. We perform dodecahedron calibration (DC) to determine the precise pose of each marker with respect to the dodecahedron $\mathbf{p}_j$. We first take several dodecahedron photos (24 in this work, as shown in Figure 6.4) and apply a one-time offline bundle adjustment, by minimizing the following cost function:

$$E_a(\{\mathbf{p}_j, \mathbf{p}_k\}) = \sum_i \sum_j \sum_k \left(\mathcal{I}_c\left(\mathbf{u}_i\left(\mathbf{p}_j; \mathbf{p}_k\right)\right) - \mathcal{O}_t\left(\mathbf{x}_i\right)\right)^2, \tag{6.12}$$

with respect to both marker poses $\mathbf{p}_j$ and dodecahedron poses with respect to the camera $\mathbf{p}_k$. Because the problem is ill-posed, we fix one of the marker poses and

doi:10.6342/NTU201800854

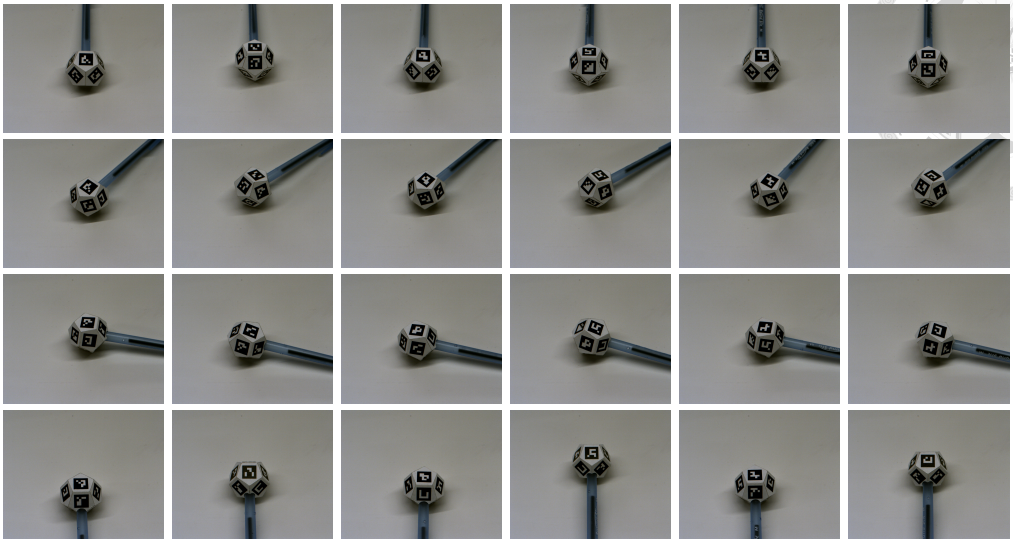


Figure 6.4: Photos used for dodecahedron calibration.

adjust other marker and dodecahedron poses simultaneously using Gauss-Newton iteration, similarly to how we solved (6.4) in Section 6.4. We initialize the marker poses $\mathbf{p}_j$ to their ideal positions on the dodecahedron, and we initialize the camera poses $\mathbf{p}_k$ with the APE approach.

6.6 Pen-tip Calibration

To recover a drawing, we need to know the position of the pen tip. Since the pen tip is a ball, we calibrate the position of the sphere center $\mathbf{c} = [x_c, y_c, z_c]^\top$ with respect to the coordinate frame of the dodecahedron. Given the 6DoF pose of the dodecahedron, we can get the world position of the pen tip (i.e., the ball center) $\mathbf{c}' = [x'_c, y'_c, z'_c]^\top = \mathbf{R}\mathbf{c} + \mathbf{t}$, where $(\mathbf{R}, \mathbf{t})$ is the pose of the dodecahedron. Finally, we can check if the distance between the pen-tip sphere center and the paper surface is less than the radius of the pen ball at runtime to determine if the pen is drawing.

To calibrate the position of the pen tip $\mathbf{c}$, we press the pen tip against a surface to keep it fixed, while moving the dodecahedron, as depicted in Figure 6.5. We track

doi:10.6342/NTU201800854

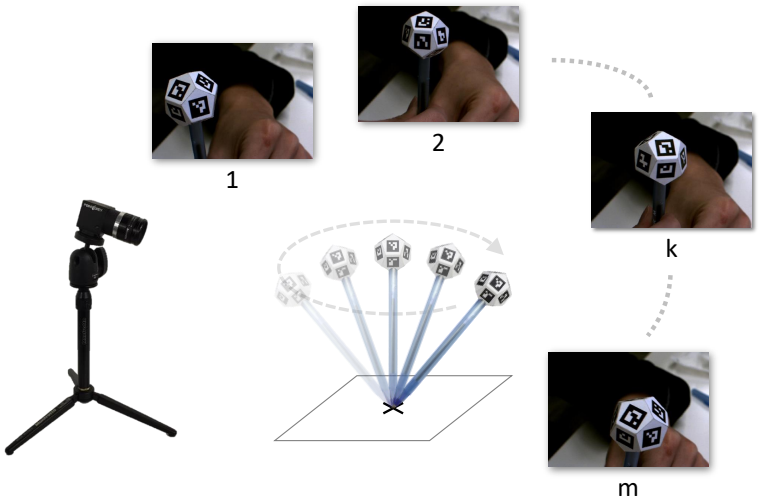


Figure 6.5: Procedures for pen-tip calibration. We press the pen tip against a surface to keep it fixed while moving and rotating the dodecahedron. In the same time, we take the pictures which are used for pen-tip calibration.

the dodecahedron to obtain a number of its poses $(\mathbf{R}_k, \mathbf{t}_k)$, where $k \in [1, m]$. Since the pen-tip center is fixed in world space, we can write the equation $\mathbf{R}_{k_1}\mathbf{c} + \mathbf{t}_{k_1} = \mathbf{R}_{k_2}\mathbf{c} + \mathbf{t}_{k_2}$ for all k_1 and k_2. From m poses, we can obtain $\frac{m(m-1)}{2}$ linear equations, which can be solved to obtain the least squares estimate of the pen-tip position $\mathbf{c}$.

6.7 Experimental Results

We evaluate the proposed method for the 6DoF DodecaPen pose tracking using both synthetic and real datasets, and compare it with an OptiTrack [161] motion caption system. Our system is run on a desktop computer with a 3.6 GHz CPU and 32 GB RAM. We use a Point Grey Flea3 1.3 MP color camera (60 Hz, 1280× 1024) with a Fujinon 12.5 mm f/1.4 lens for an effective horizontal field of view 60 degrees.

Given the ground-truth rotation matrix $\tilde{\mathbf{R}}$ and translation vector $\tilde{\mathbf{t}}$, we compute the rotational error of the estimated rotation matrix $\mathbf{R}$ in degrees according to (2.32). The translation error of the estimated translation vector $\mathbf{t}$ is measured by the

doi:10.6342/NTU201800854

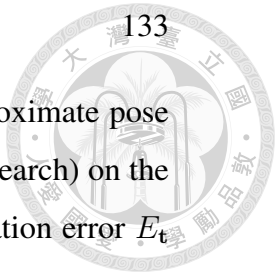


Table 6.1: Evaluation results for different approaches (APE: approximate pose estimation; DPR: dense pose refinement; BLS: backtracking line search) on the synthetic dataset in terms of average rotation error $E_{\mathbf{R}}$ (°), translation error $E_{\mathbf{t}}$ (mm), pen-tip error E_{pen} (mm), and runtimes per frame (ms). The last column shows the average number of iterations for the DPR approach.

Approach	$E_{\mathbf{R}}$	$E_{\mathbf{t}}$	E_{pen}	Time	#Iter.
APE	0.447	5.835	5.854	1.100	–
APE+DPR	0.053	0.356	0.401	6.213	6.178
APE+DPR+BLS	0.053	0.336	0.386	6.140	3.834

ℓ^2 difference between $\tilde{\mathbf{t}}$ and $\mathbf{t}$ defined as (2.35). The pen-tip error E_{pen} is the ℓ^2 difference between two pen-tip positions transformed with either $(\mathbf{R}, \mathbf{t})$ or $(\tilde{\mathbf{R}}, \tilde{\mathbf{t}})$ in the camera coordinate system. The distance between the pen tip and the dodecahedron center is 143 mm. The success rate (SR) is defined as the percentage of the successfully estimated poses within each sequence.

6.7.1 Synthetic Data

We construct a synthetic dataset by generating 24 image sequences with different motion patterns of the virtual DodecaPen, as shown in Figure 6.6. The 6DoF DodecaPen pose sequence in each image sequence is obtained by recording poses of a rigid body with the OptiTrack motion capture system. Each sequence consists of 301 frames with the same resolution and intrinsics as our real camera. We initialize each tracking algorithm with the ground-truth pose for the first frame. Table 6.1 shows that the dense pose refinement (DPR) approach can achieve significantly better accuracy than the approximate pose estimation (APE) approach from sparse constraints alone. With a backtracking linear search (BLS) scheme, we can take fewer Gauss-Newton iterations during the optimization process and also achieve more accurate results compared to not using a line search. It is also

doi:10.6342/NTU201800854

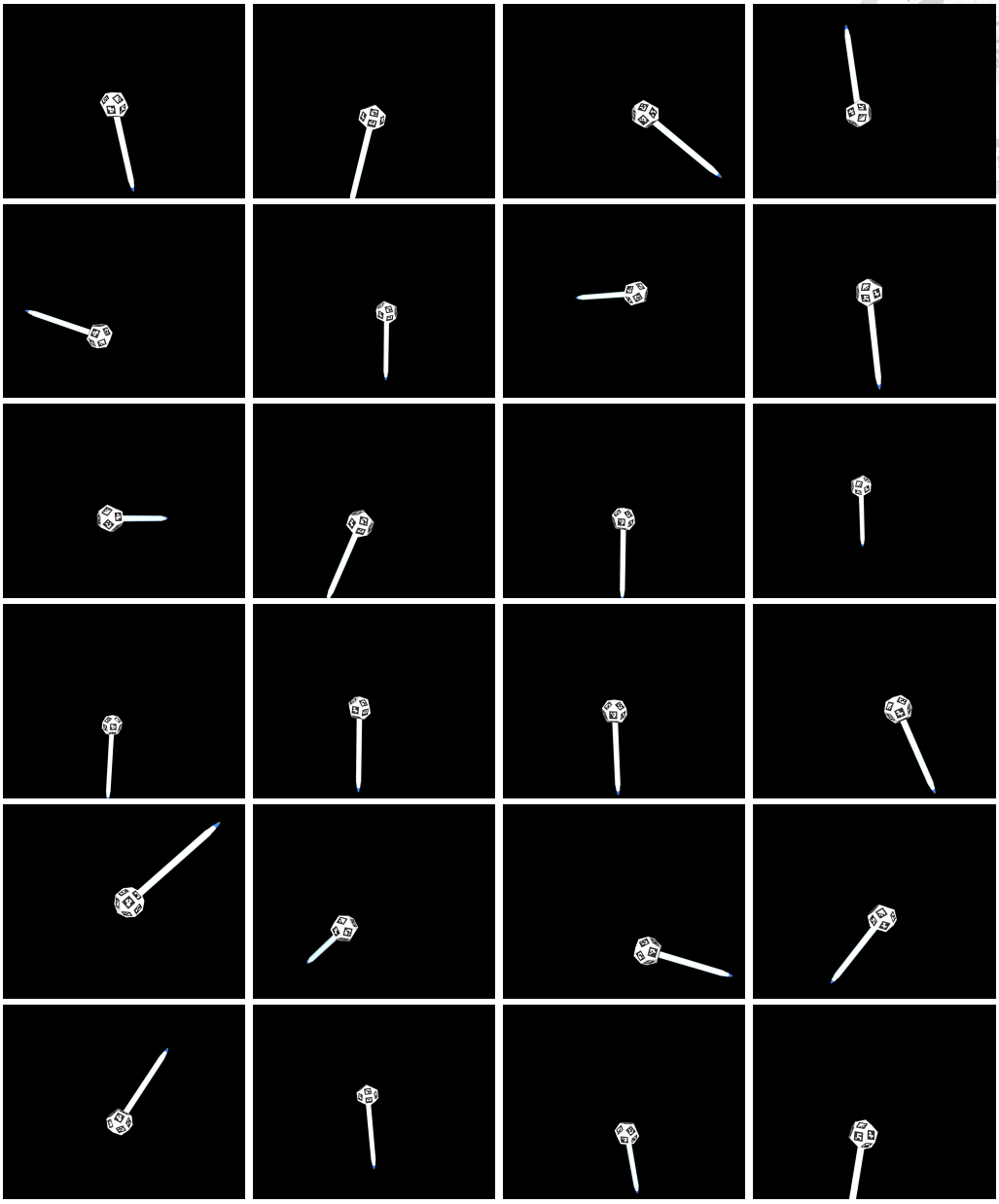


Figure 6.6: We generate synthetic image sequences with 24 motion patterns of the virtual DodecaPen for evaluation.

notable that even though the pen tip is far from the dodecahedron center, the pen-tip error is dominated by the translation error. We show all the pen-tip trajectories generated by approaches in Figure 6.7 to Figure 6.9, and compare them with ground truth. The trajectories generated from the APE approach alone is visibly jittery,

doi:10.6342/NTU201800854

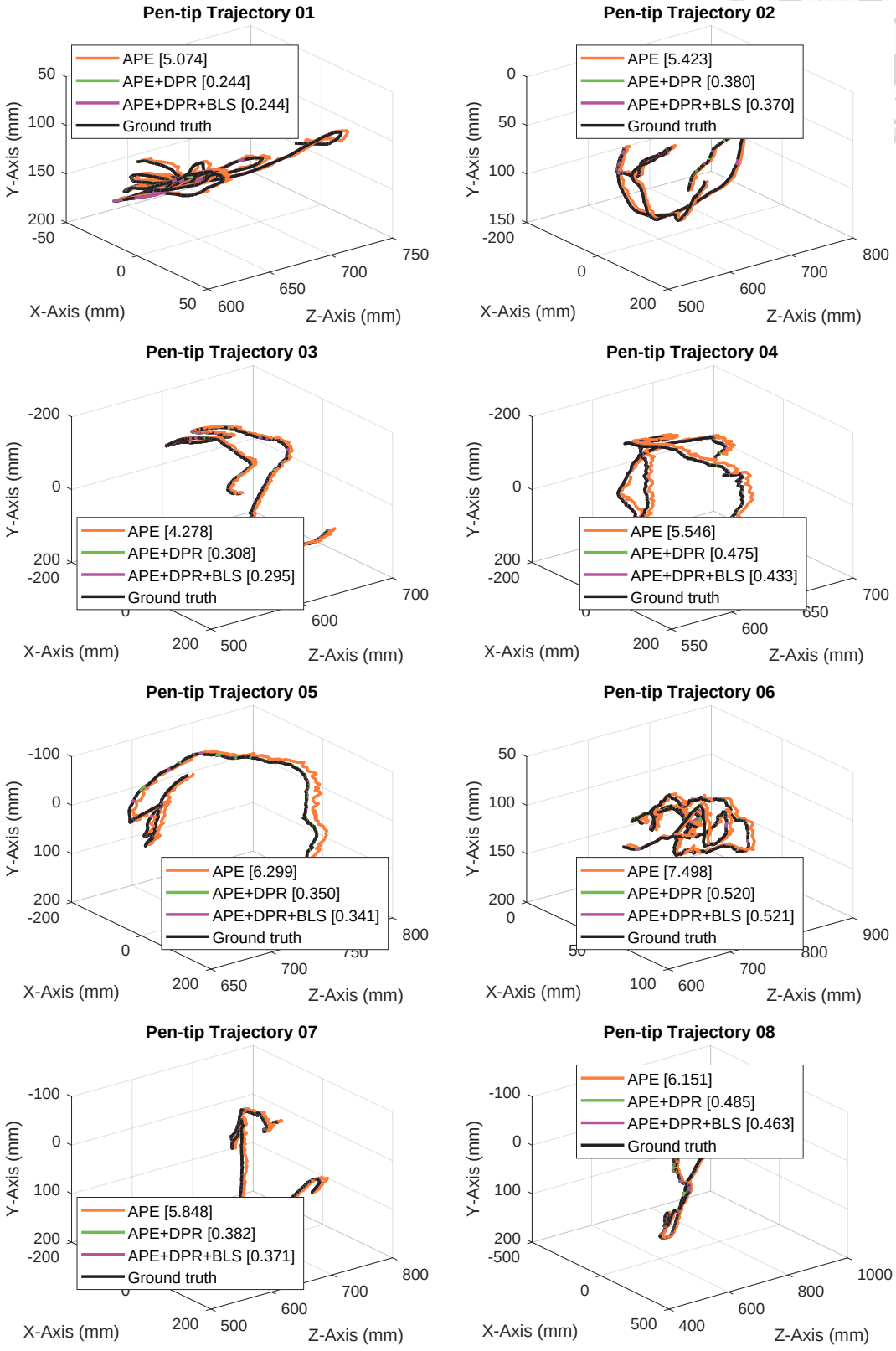


Figure 6.7: Pen-tip trajectories (01–08) generated by different approaches. Average pen-tip errors (mm) are shown in legends.

doi:10.6342/NTU201800854

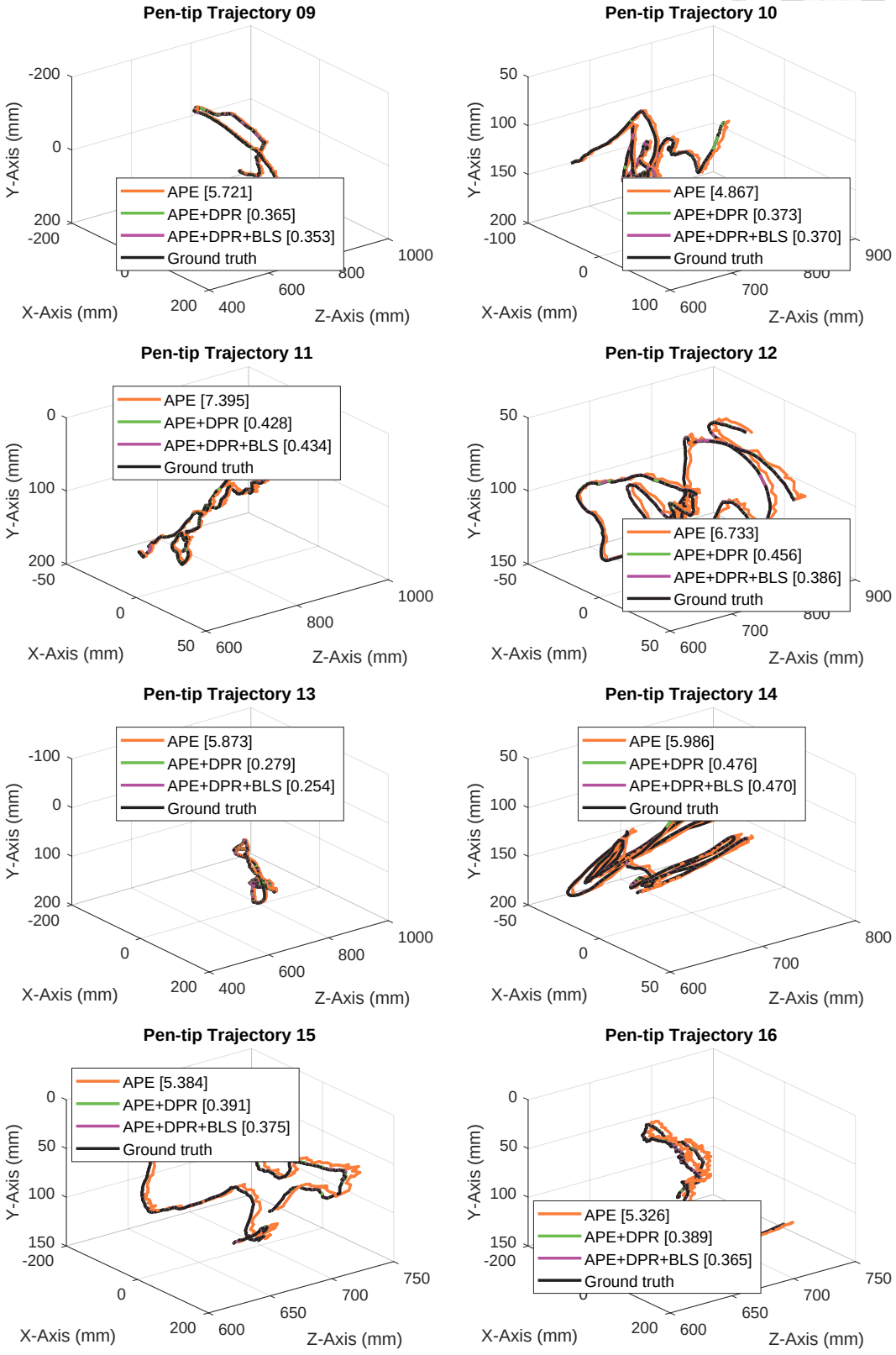


Figure 6.8: Pen-tip trajectories (09–16) generated by different approaches. Average pen-tip errors (mm) are shown in legends.

doi:10.6342/NTU201800854

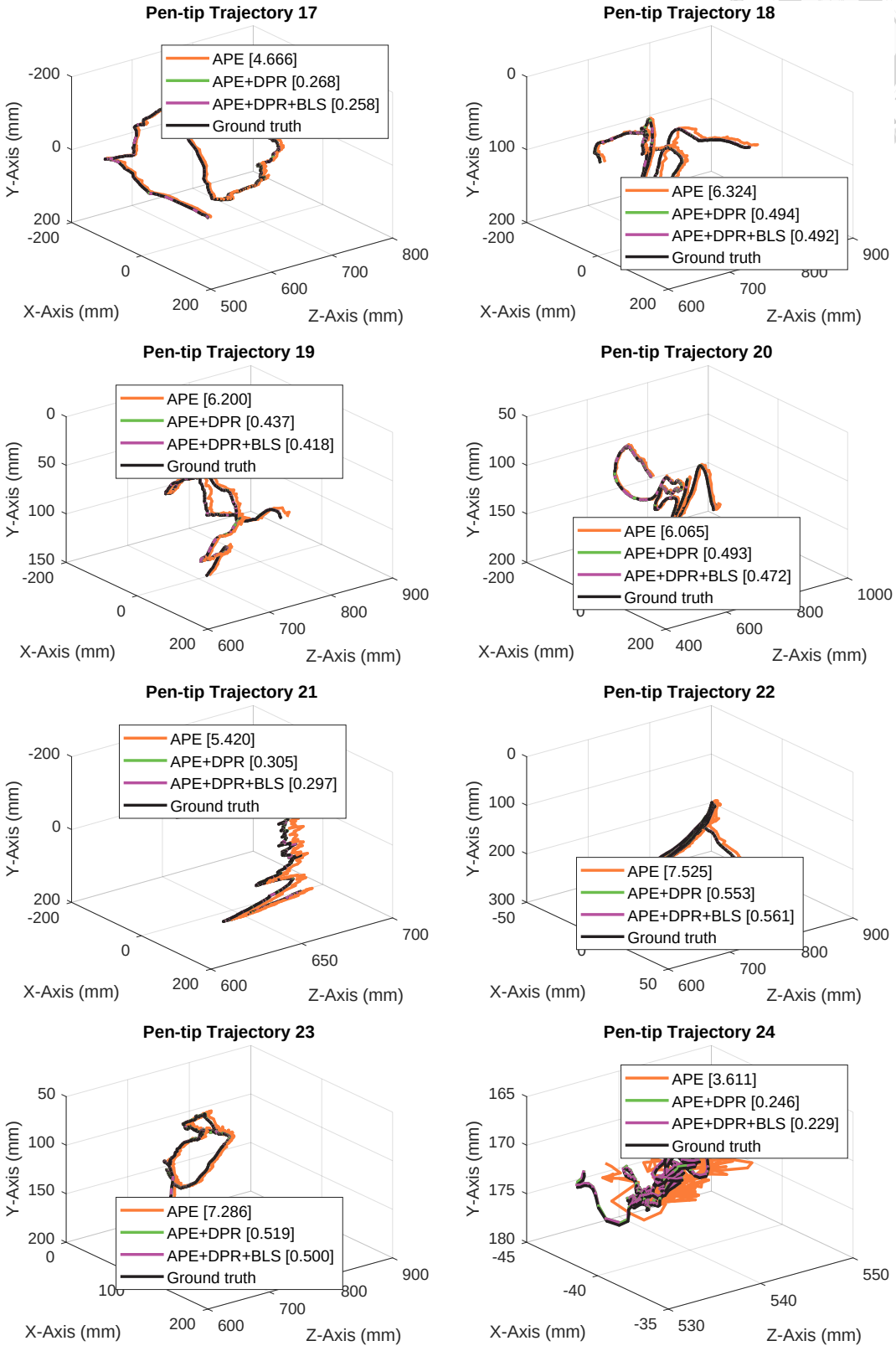
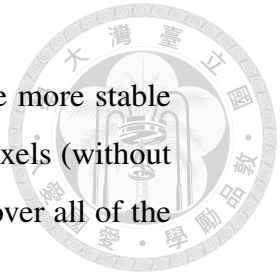


Figure 6.9: Pen-tip trajectories (17–24) generated by different approaches. Average pen-tip errors (mm) are shown in legends.

doi:10.6342/NTU201800854



while those generated by approaches using the DPR approach are more stable and numerically closer to ground truth. The average number of pixels (without considering masking) for the markers on the DodecaPen is 6136 over all of the sequences in the synthetic dataset.

We further evaluate the proposed approaches under varying shot noise, spatial blur, camera resolutions, and mask kernel widths to evaluate the sensitivity of the system to the most common types of degradation to allow practitioners to evaluate the feasibility of this system. There are several observations to note in the results. First, when the input frames are degraded with shot noise, the tracking results without the BLS scheme degrade more rapidly than those with it, as demonstrated in Figure 6.10. We also find that sufficient shot noise can prevent direct alignment from converging without the line search. The BLS scheme is particularly effective when the small residual approximation of Gauss-Newton breaks down with noise.

Second, although the ArUco marker detector can detect markers well for images corrupted with high shot noise, it quickly fails for spatially blurry images, as explained in Figure 6.11. Hence, tracking success rate drops dramatically with spatial blur. In contrast, by adding the inter-frame corner tracking (ICT) scheme, our pose estimation can be quite robust (in terms of tracking success rate) to spatial blur, although accuracy suffers.

Third, the proposed approach still performs favorably even with VGA resolution sensors (i.e., 0.3 megapixels) while the execution time is reduced to 3.0 ms, as shown in Figure 6.12.

Finally, the accuracy seems to be empirically unaffected by different sizes of the mask kernel even when the number of valid pixels used in dense alignment drops from 6136 to 3941, as presented in Figure 6.13.

6.7.2 Real Data

Because the DodecaPen is an actual ball-point pen, we can evaluate the accuracy of our approach by comparing the resulting hand-drawn image and the digital 2D

doi:10.6342/NTU201800854

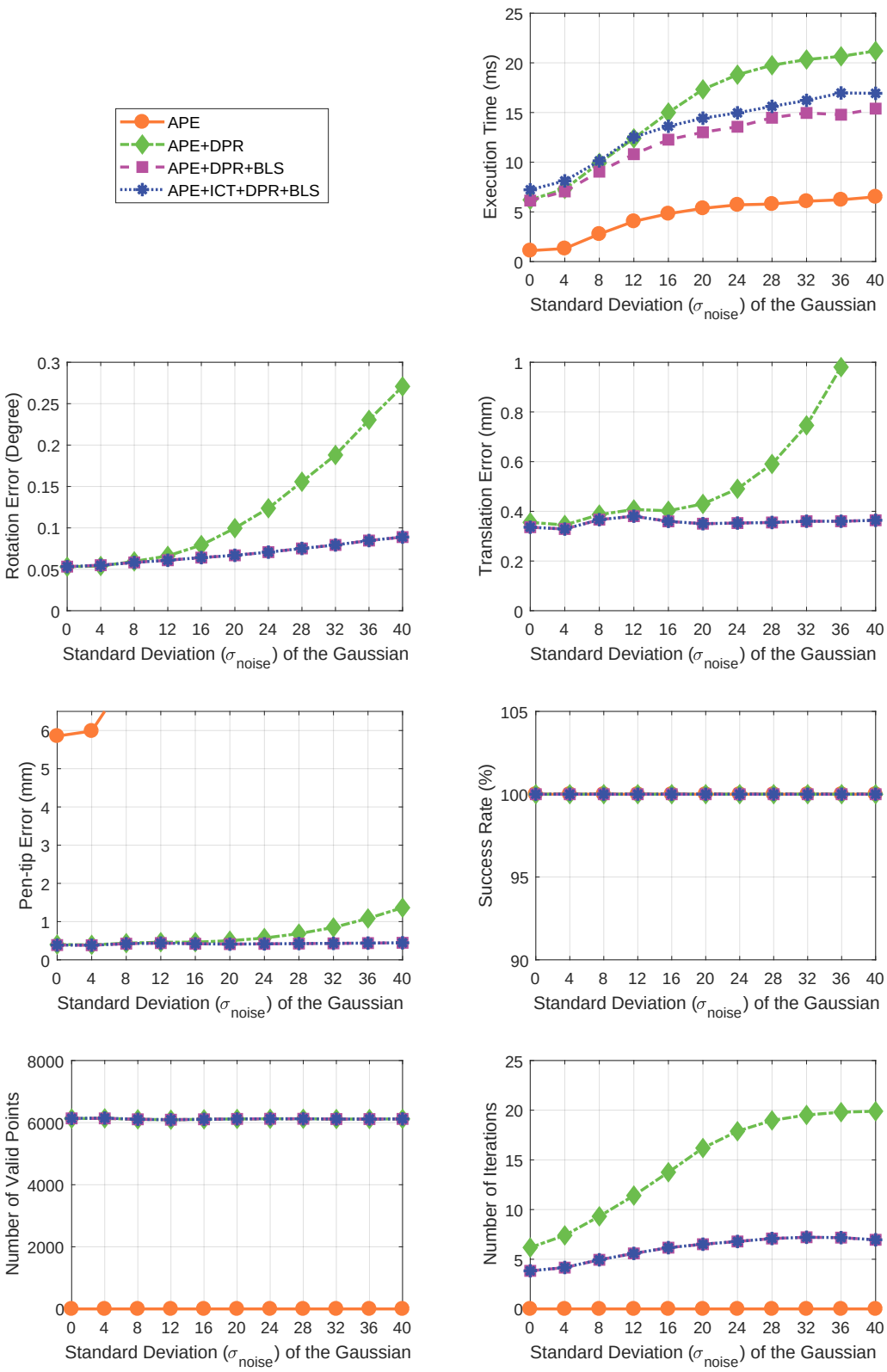


Figure 6.10: Experimental results on synthetic dataset under *Shot Noise* condition with different degradation levels. The standard deviation of the Gaussian noise is set for an intensity range of 0 to 255.

doi:10.6342/NTU201800854

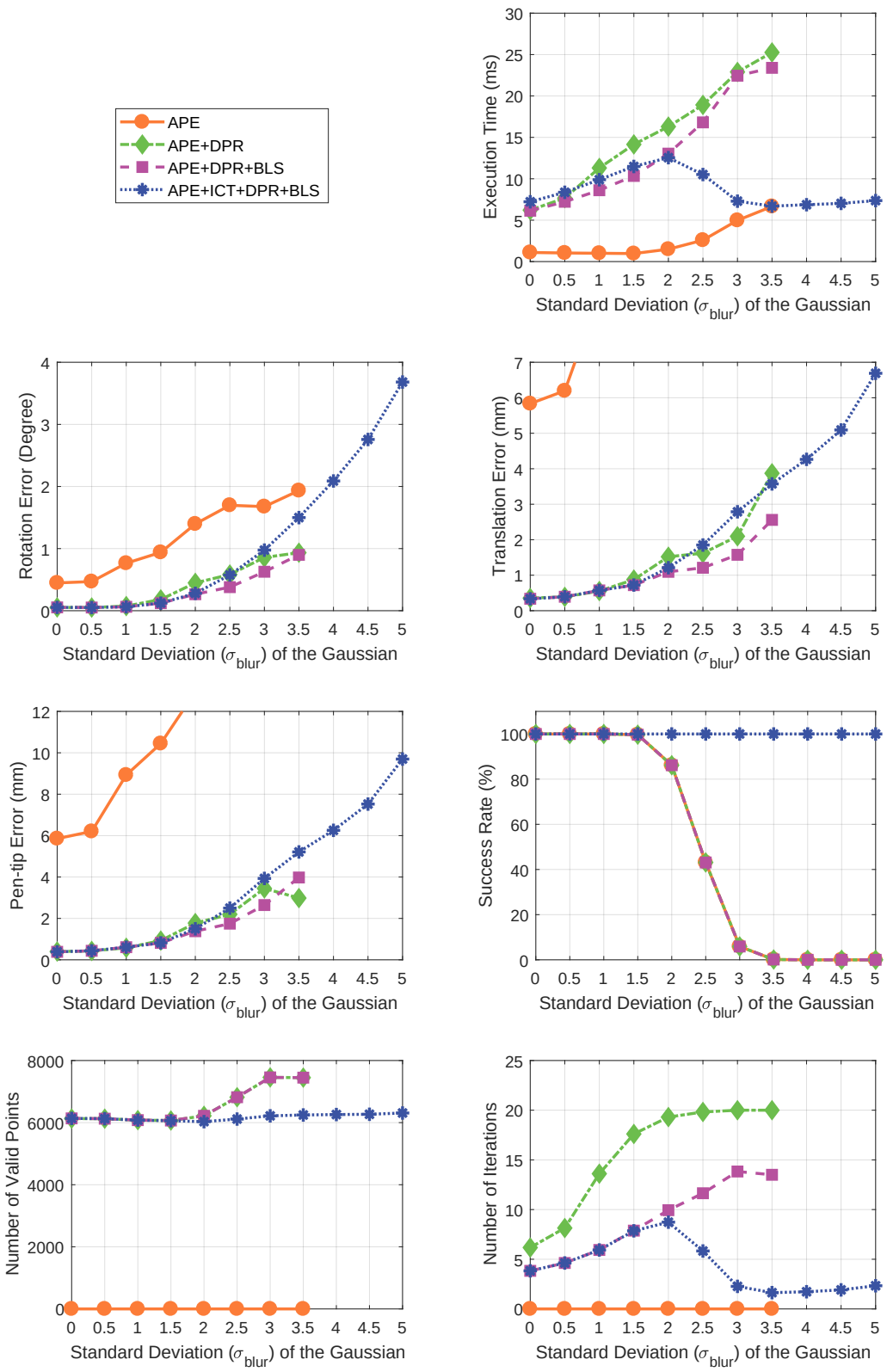


Figure 6.11: Experimental results on synthetic dataset under *Spatial Blur* condition with different degradation levels. The spatial Gaussain blur sigma is in pixels for a 1280×1024 image.

doi:10.6342/NTU201800854

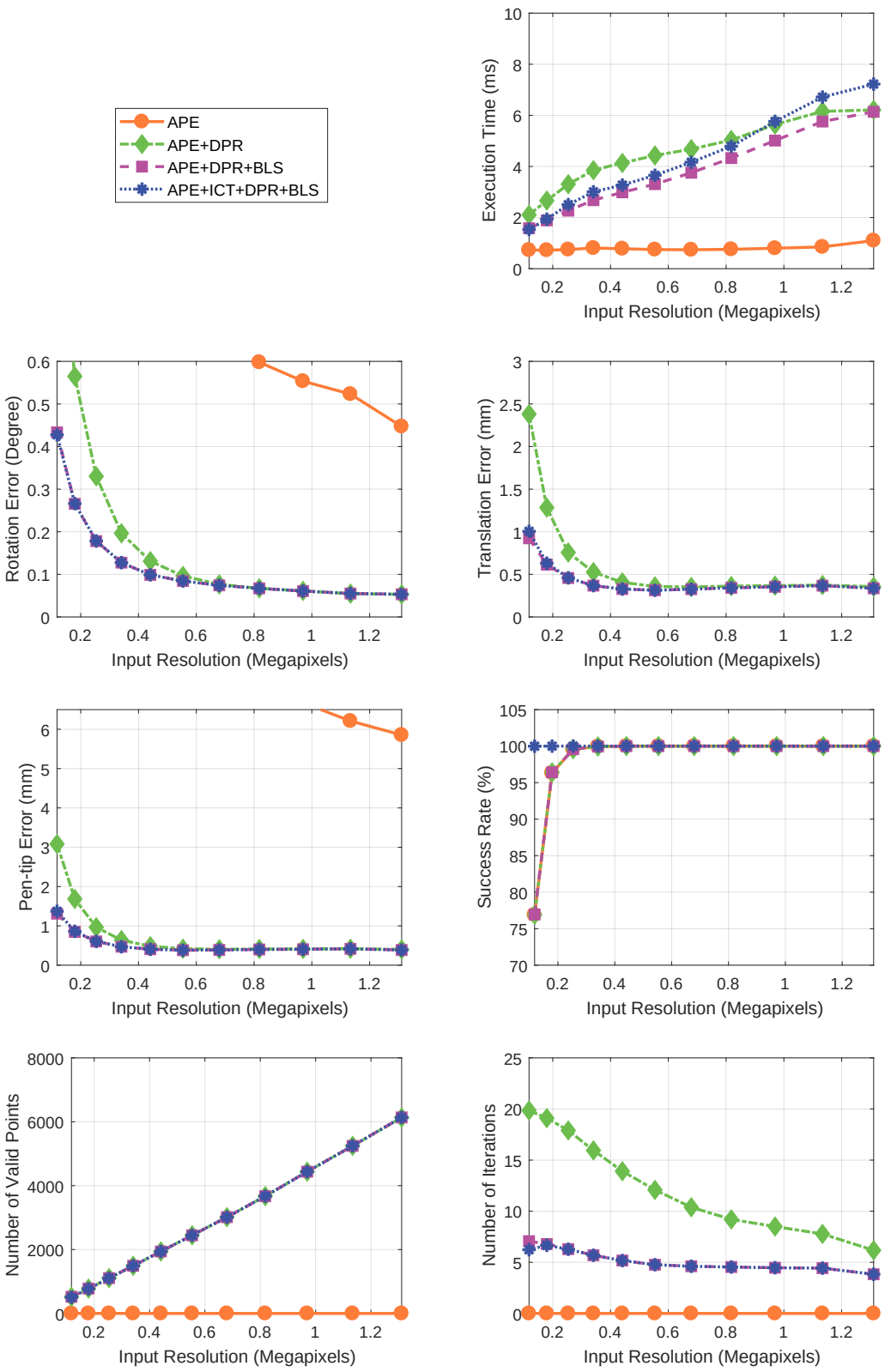


Figure 6.12: Experimental results on synthetic dataset under *Camera Resolution* condition with different degradation levels.

doi:10.6342/NTU201800854

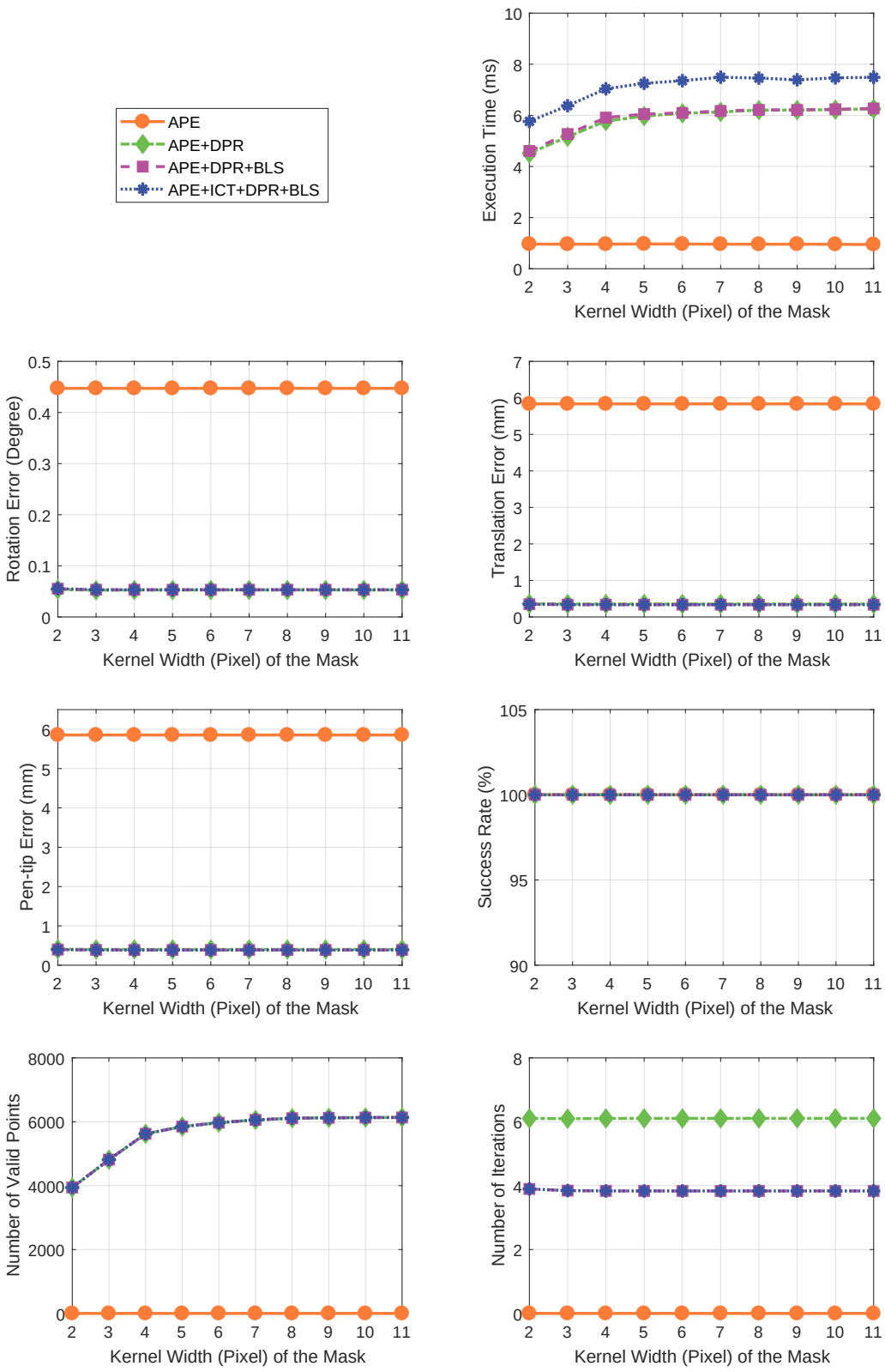
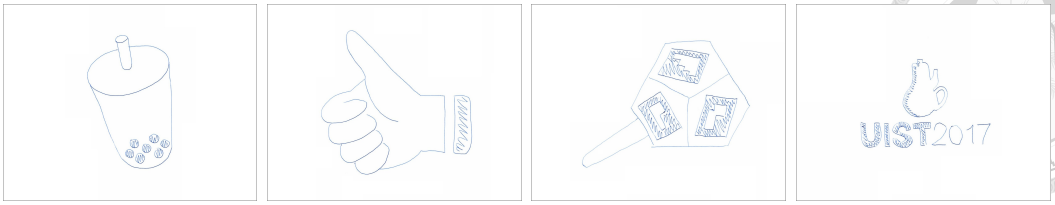


Figure 6.13: Experimental results on synthetic dataset under *Mask Kernel Width* condition with different degradation levels.

doi:10.6342/NTU201800854



(a) Boba (b) Thumb (c) DodecaPen (d) UIST2017

Figure 6.14: The four ground-truth drawings used for real data evaluation. These patterns are drawn on a letter size paper (220×280 mm^2).

drawing produced by our technique. The ground-truth image (on a letter size paper) is obtained from a scanner, as shown in Figure 6.14, while the digital 2D drawing is generated by the built-in plot function in MATLAB. Both images are scaled to a resolution of 1650×1275. The maximum rotation and translation speeds of the dodecahedron in the real dataset are around 80 degree/s and 200 mm/s, respectively. General drawing and writing are covered within these speeds. The relative rigid transformation between the camera and the drawing paper is resolved through calibration.

To compare a drawing generated by the proposed system to a ground-truth drawing, we first binarize both drawings by Otsu's method [188] and obtain a 2D set of drawn points from each image. Next, we overlay these two binary images and find the nearest point in the other image for each point in both point sets according to their coordinates. The mean distances between each point and its nearest neighbor are regarded as the similarity metric, which is applied for our real data evaluation.

We collect four real drawings with different shapes, and the tracking results of our system compared to ground-truth (i.e., scanned) patterns are shown in Figure 6.15 to Figure 6.18. The proposed method can generate drawings virtually identical to ground truth, while results from applying the APE approach alone are visually messy. Furthermore, without dodecahedron calibration, distortions due to model error are clearly visible in the alignment with the ground truth. The accuracy

doi:10.6342/NTU201800854

Figure 6.15: Hand-drawing results of *Boba* generated by different approaches. Each image is blended with the ground-truth drawing and augmented with a text box showing the mean shortest distance (in millimeters) between the generated and ground-truth drawing.

doi:10.6342/NTU201800854

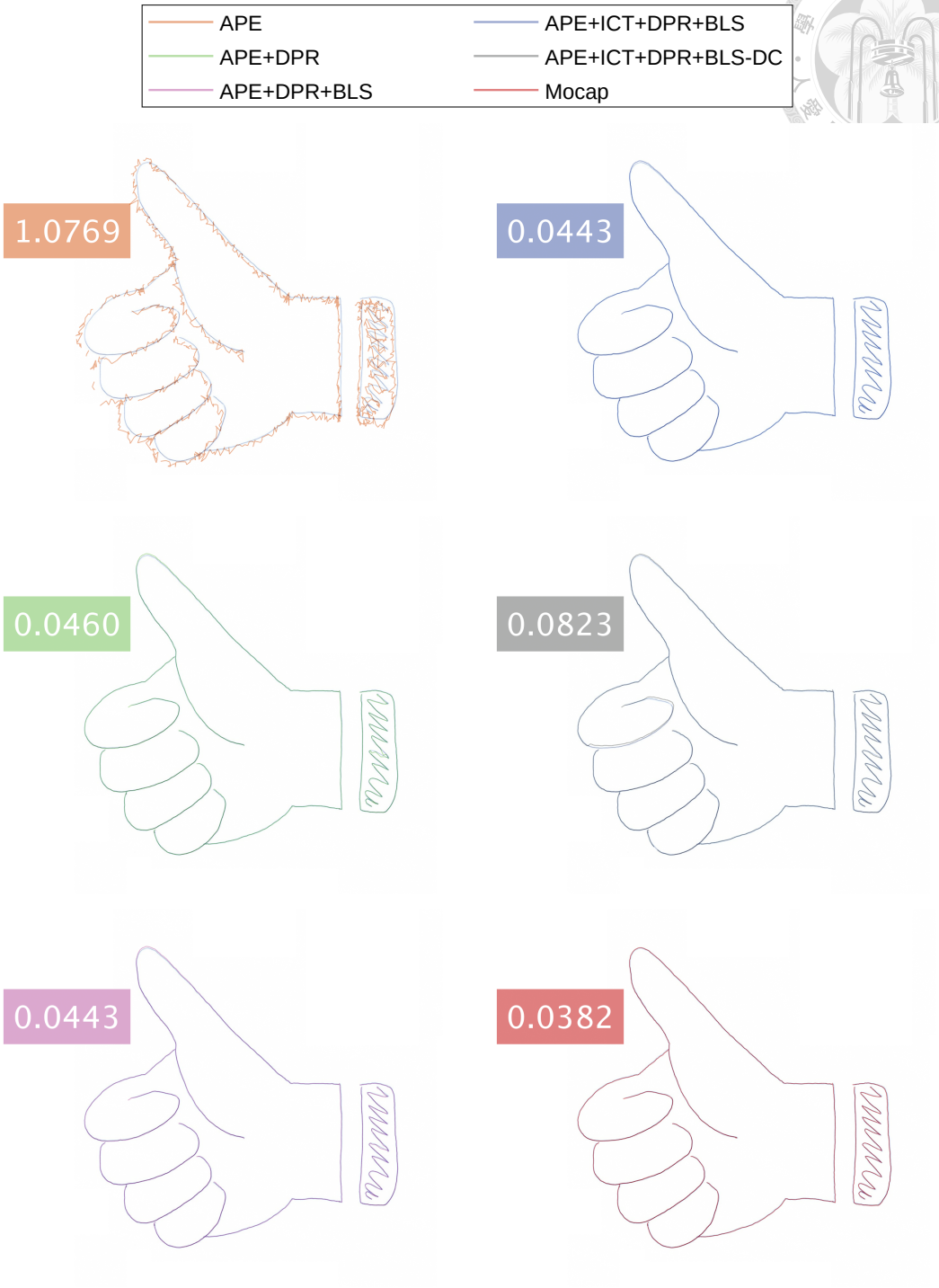


Figure 6.16: Hand-drawing results of *Thumb* generated by different approaches. Each image is blended with the ground-truth drawing and augmented with a text box showing the mean shortest distance (in millimeters) between the generated and ground-truth drawing.

doi:10.6342/NTU201800854

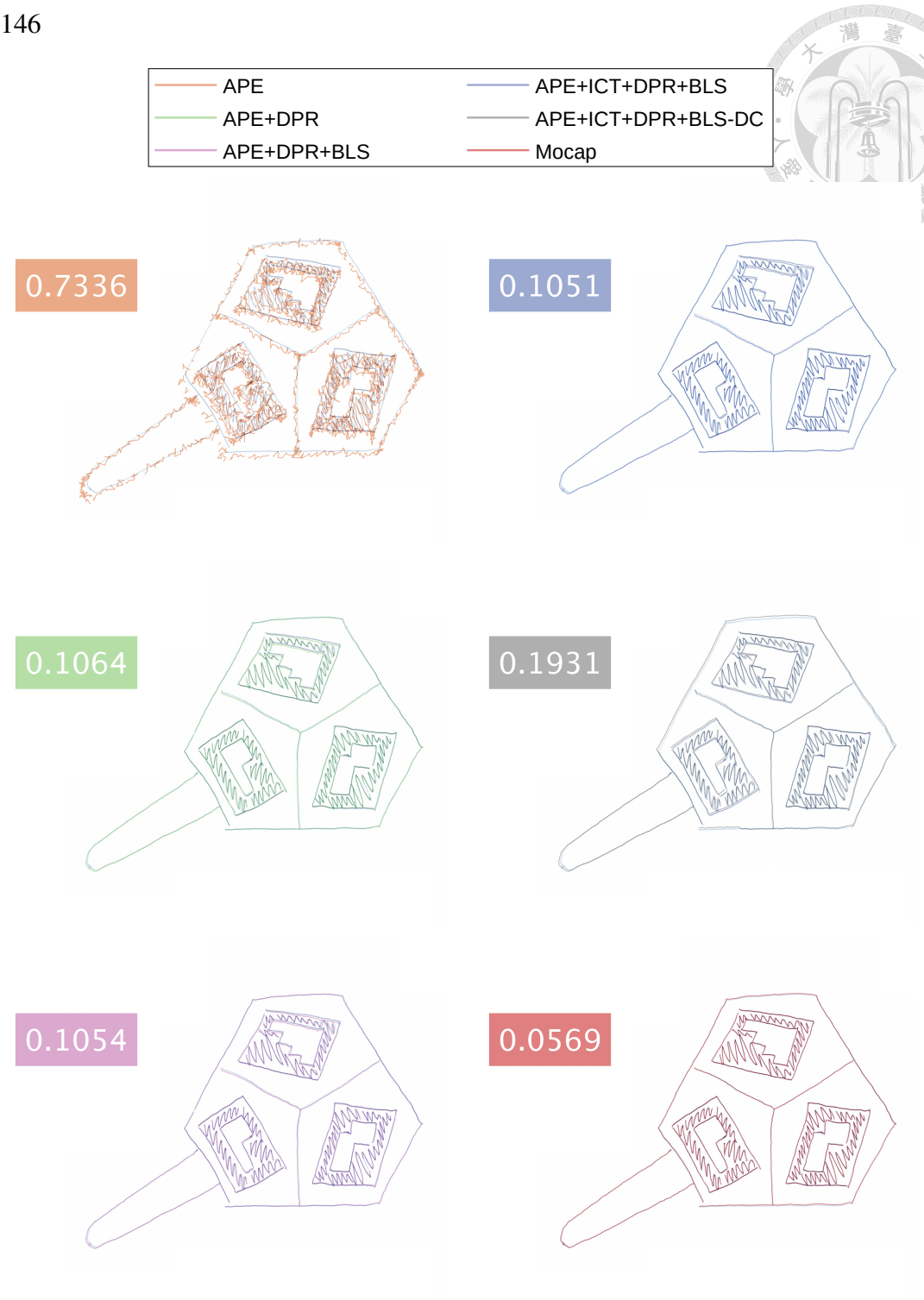


Figure 6.17: Hand-drawing results of *DodecaPen* generated by different approaches. Each image is blended with the ground-truth drawing and augmented with a text box showing the mean shortest distance (in millimeters) between the generated and ground-truth drawing.

doi:10.6342/NTU201800854

Figure 6.18: Hand-drawing results of *UIST2017* generated by different approaches. Each image is blended with the ground-truth drawing and augmented with a text box showing the mean shortest distance (in millimeters) between the generated and ground-truth drawing.

doi:10.6342/NTU201800854

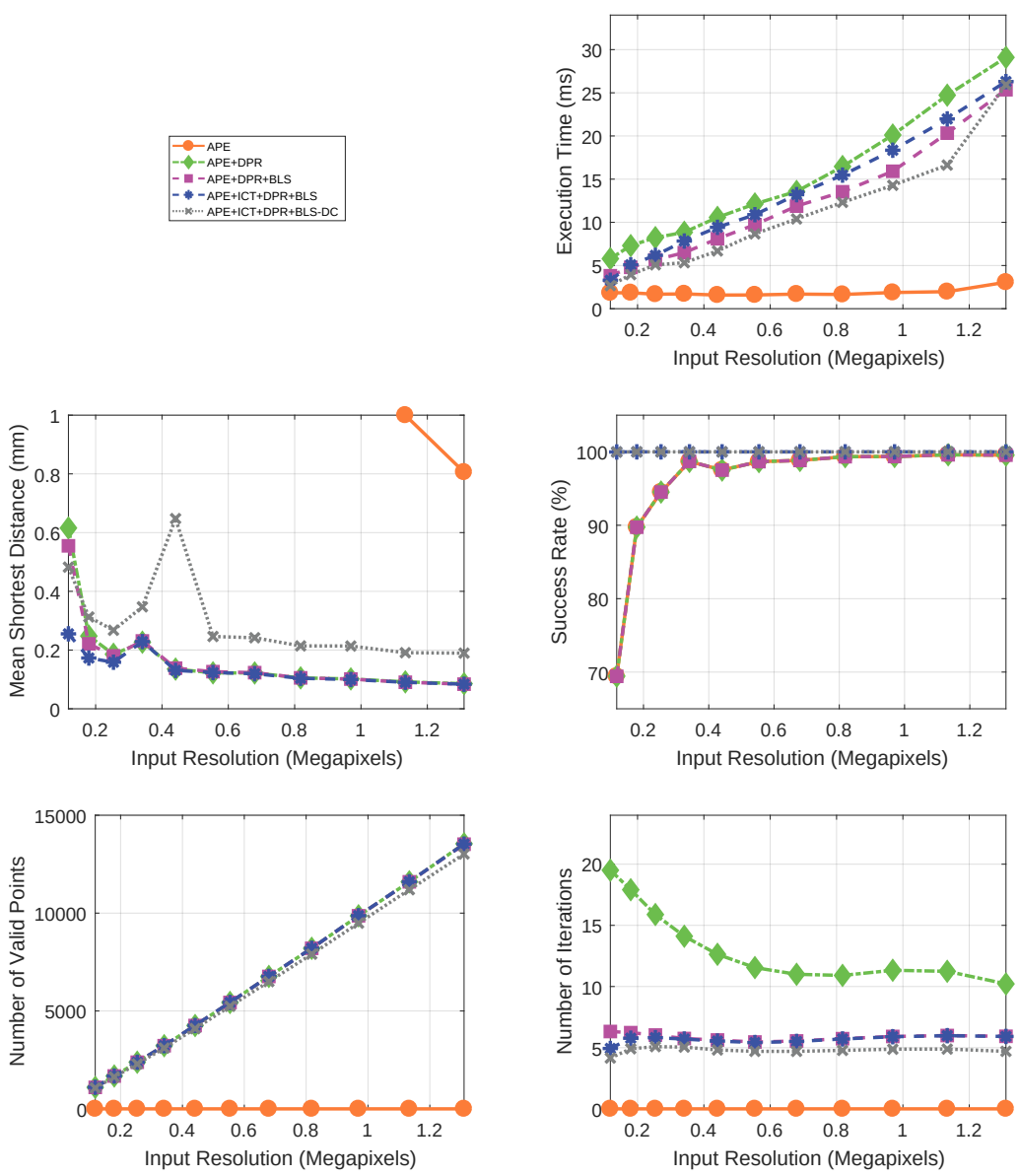


Figure 6.19: Experimental results on real dataset under various camera resolution conditions.

and performance for various camera resolutions and mask kernel conditions are shown in Figure 6.19 and Figure 6.20, respectively. As we have already seen in Section 6.7.1, the proposed method can still perform well (0.5 mm accuracy) even at VGA resolution. And masking does not seem to affect the tracking results, which makes it possible to run the proposed system at 60Hz by choosing the

doi:10.6342/NTU201800854

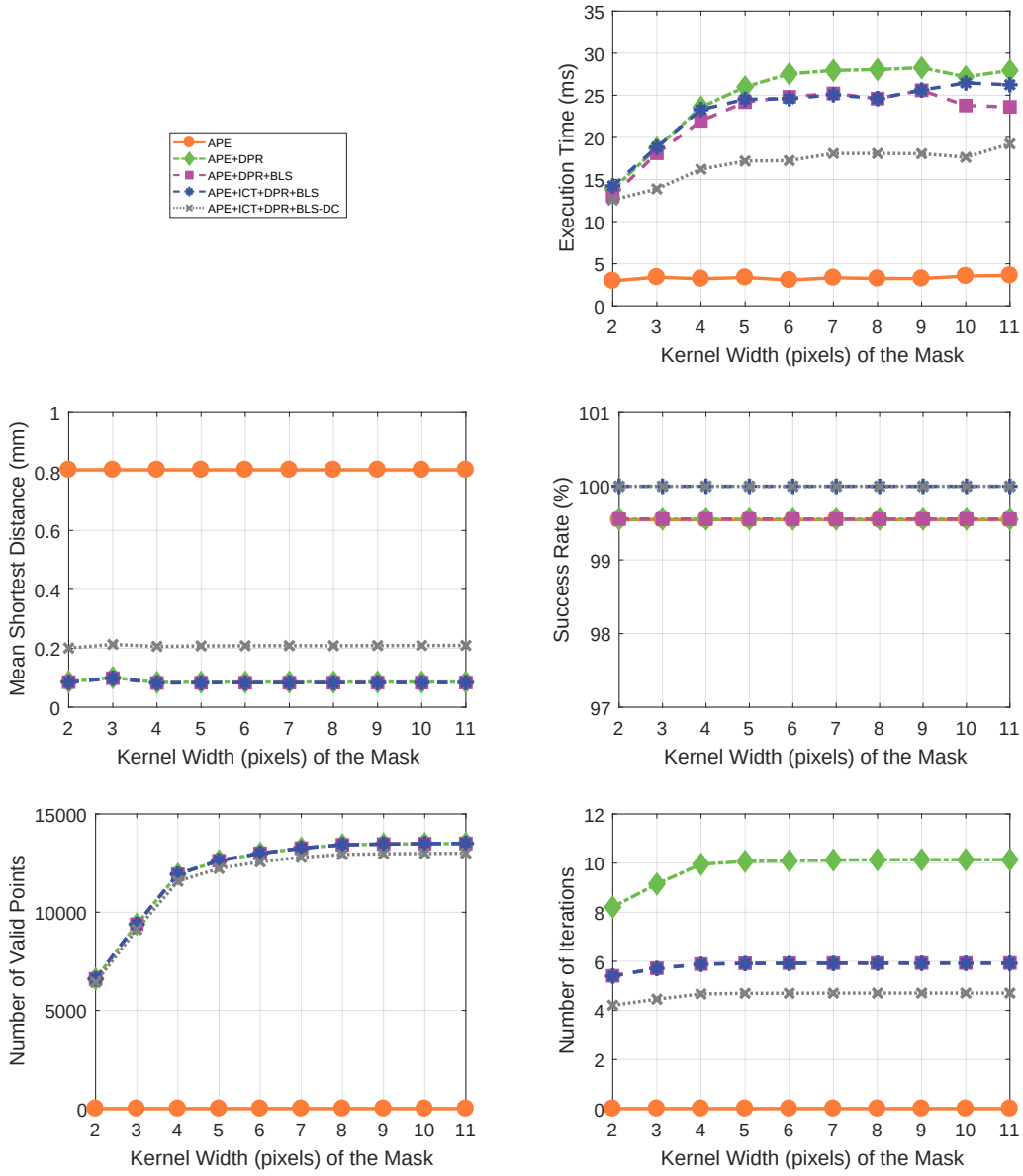


Figure 6.20: Experimental results on real dataset under various camera resolution conditions.

smallest mask kernel size.

In our final comparison, we compare the drawing results generated by the proposed DodecaPen system with those generated by a state-of-the-art motion capture system. The motion capture system is constructed with 16 OptiTrack Prime 17 W (1.7 megapixels, 70 degrees field-of-view) cameras, as shown in Figure 6.21.

doi:10.6342/NTU201800854

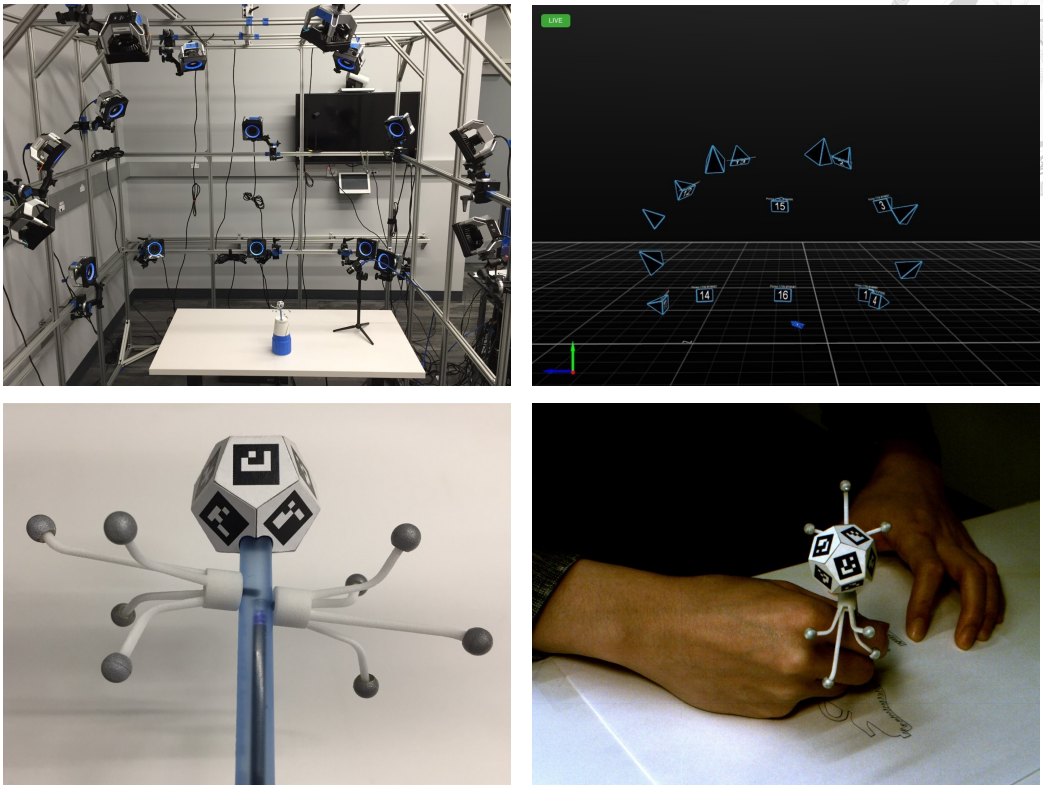


Figure 6.21: Experiments with OptiTrack motion capture system. Top tow: We use 16 OptiTrack cameras. Bottom row: We add eight retroreflective markers to the DodecaPen and shown a sample frame from the DodecaPen tracking camera.

The pen is augmented with eight more retroreflective balls as markers for the mocap system. After calibrating the mocap system, we record image sequences from all 16 motion capture cameras (with a combined 27MP of resolution) as well as the DodecaPen tracking camera (1.3MP) simultaneously. Because motion capture obtains the 3D position from triangulation from multiple cameras, it is interesting to see how accuracy degrades with fewer cameras. Since not every camera contributes to the pose computation on the same level, we make a best effort of selectively reducing the number of cameras in lowest priority order based on the distance to the pen as well as the percentage of the time the markers are blocked from that camera view. The results shown in Figure 6.22 reveal that the proposed method is comparable to a motion capture system with 10 active cameras

doi:10.6342/NTU201800854

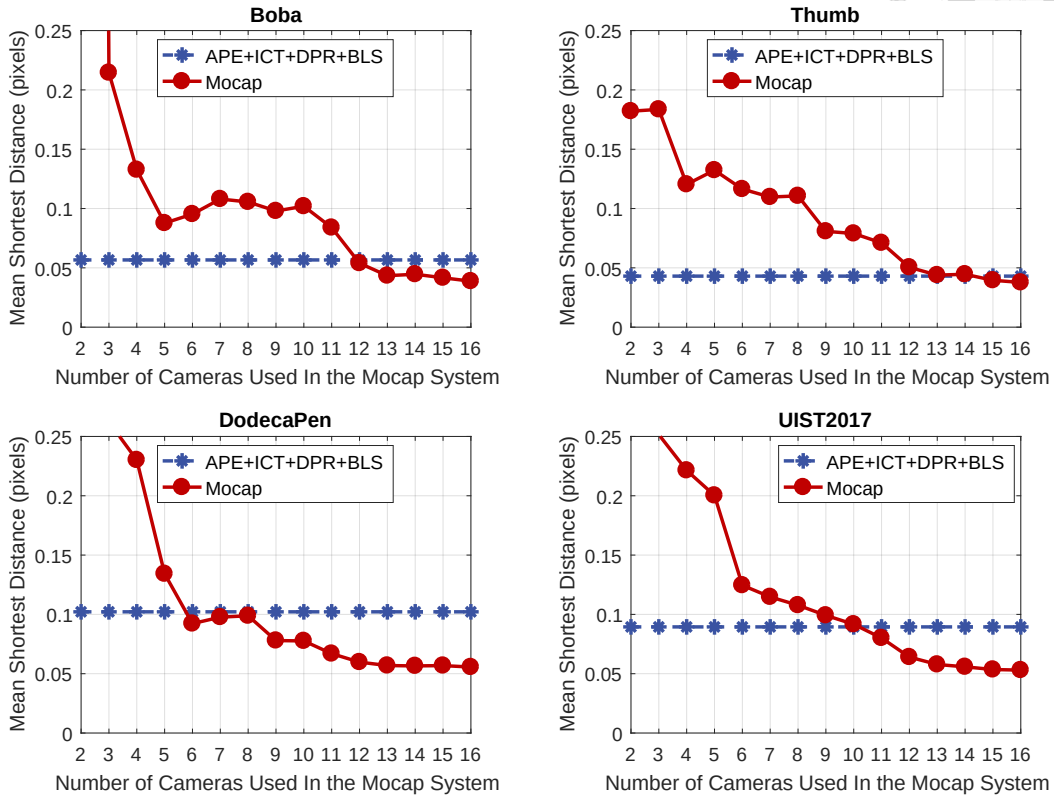


Figure 6.22: Experimental results of the motion capture system with different numbers of active cameras. The accuracy of the proposed method is comparable to a motion capture system with 10 active cameras.

(17 MP). The drawing results generated by the mocap system with 16 cameras are also shown in Figure 6.15 to Figure 6.18 and are virtually indistinguishable from the ground truth.

6.8 Applications

The DodecaPen can provide low-cost writing and drawing capabilities to both 2D and 3D (e.g., VR) applications. We demonstrate both 2D and 3D drawing. Although a pen is typically used for writing and drawing, the pen (via the dodecahedron) can also serve as a handheld proxy for 3D objects.

doi:10.6342/NTU201800854

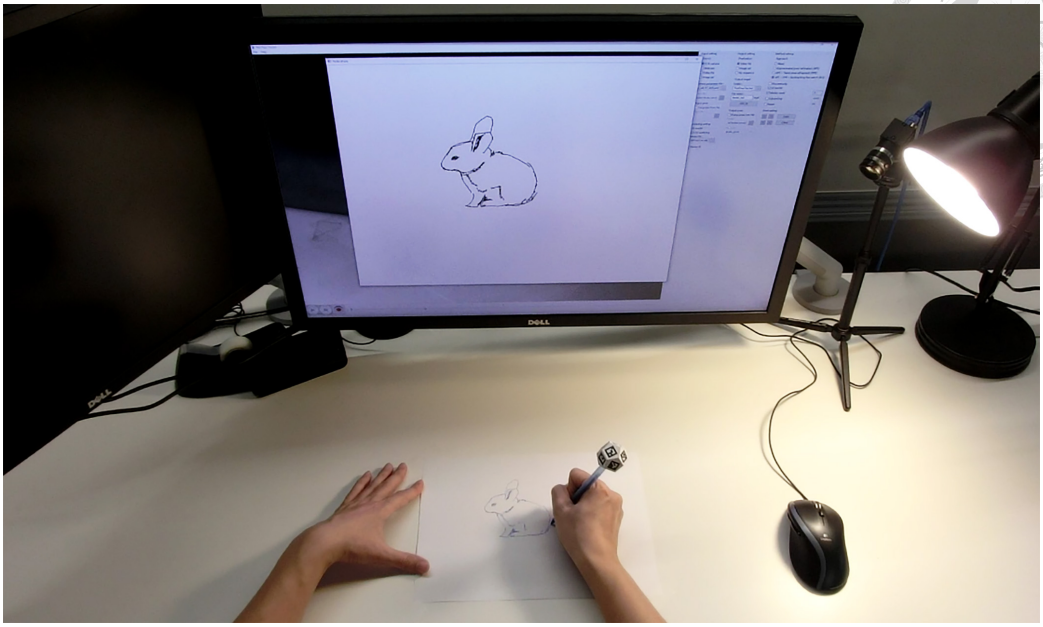


Figure 6.23: The DodecaPen can turn a flat surface into a digital drawing surface.

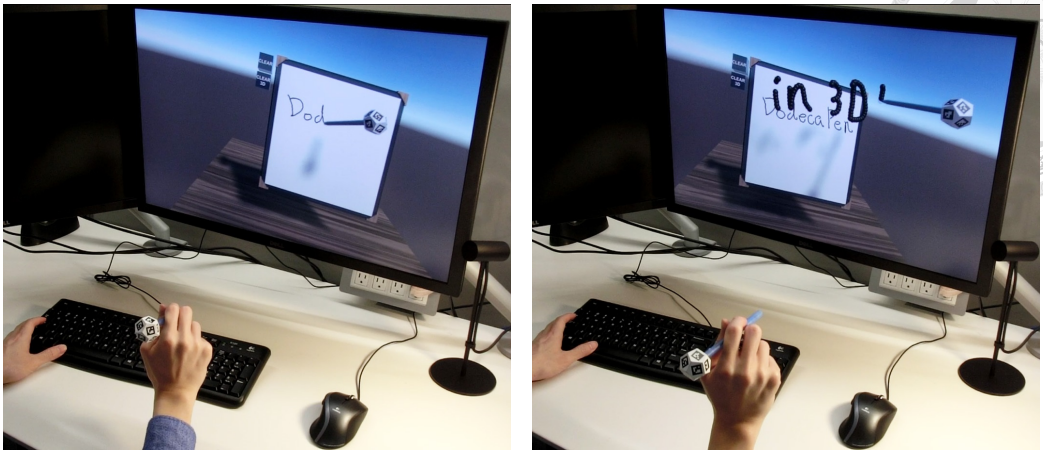
6.8.1 2D Drawing

Our system can turn any flat surface into a digital writing and drawing surface, such as on a desk or whiteboard, as shown in Figure 6.23. Although the DodecaPen requires an external camera, the pen and surface do not require any electronics found in professional graphics tablets [189] and can digitize real graphite or ink without a textured pattern [190]. With 3D tracking, we can utilize the space above the writing surface and enable hover-based interactions [191] as well as multi-layer interactions [192]. Instead of using an external camera, we could embed a camera with a global shutter to our existing devices (e.g., monitors, laptops, mobile devices) and create writable surfaces on the fly.

6.8.2 3D Drawing

In addition to drawing on a 2D surface in a 3D VR environment, we can use the DodecaPen to draw 3D curves, as shown in Figure 6.24. The pen can emit 3D ink for 3D annotation or be used as an instrument for content creation, such as a virtual

doi:10.6342/NTU201800854



(a) Drawing on a 2D surface (b) Drawing in 3D space

Figure 6.24: In a VR environment, the DodecaPen can (a) draw on a midair 2D surface or (b) emit 3D ink when the spacebar is pressed.

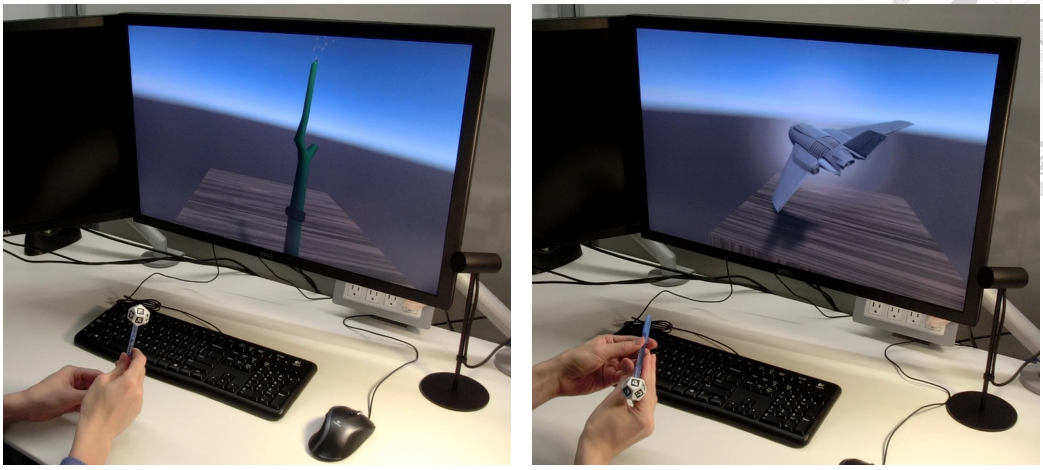
sculpting tool [193]. For demonstration purposes, we use the spacebar to emit 3D ink, as shown in Figure 6.24(b). The DodecaPen can also be used to digitize real 3D objects by specifying the 3D points of a surface (e.g., Ivan Sutherland's Volkswagen [194]) rather than scanning and then re-meshing [195].

6.8.3 General 6DoF Object Tracking

Although we focused on the specific application of tracking a pen, the dodecahedron can be used as a general 6DoF tracked object. We can use the dodecahedron to enable tangible input [196], either as a proxy for virtual 3D objects or to bring in other physical devices into VR. The form of the pen lends itself to represent cylindrical objects such as a VR wand or baton, as shown in Figure 6.25(a). Additionally, it can represent more general objects to be inspected for educational or industrial (e.g., CAD models) purposes, as shown in Figure 6.25(b). Furthermore, the proposed system can serve as a low-cost motion capture system for digital puppetry [197].

The tracked dodecahedron can be attached to physical objects other than a pen.

doi:10.6342/NTU201800854



(a) Cylindrical object (b) General object

Figure 6.25: The DodecaPen can (a) double as other cylindrical objects such as a VR wand or (b) provide general 6DoF object tracking.

In Figure 6.26(a), we attach the dodecahedron to a physical keyboard to display in VR. The dodecahedron itself could be a tangible 12-sided VR die for use in a board game, as shown in Figure 6.26(b).

6.9 Summary

We have demonstrated a system for sub-millimeter-accurate 6DoF tracking using a set of readily available and easy-to-assemble components. Through design choices around the shape and appearance of the tracking fiducial as well as careful the application of computer vision algorithms for calibration and pose estimation, we show that single camera pose estimation can be fast enough and robust enough for drawing in 2D, 3D and in VR.

We have systematically validated each design decision of the system. We show that marker corner alignment is insufficient for robust and accurate tracking. A combination of inter-frame alignment and dense pose refinement is needed to achieve sufficient accuracy and robustness. A straightforward application of the Lucas and Kanade method is improved by adapting the step size with a backtrack-

doi:10.6342/NTU201800854

(a) DodecaKeyboard (b) DodecaDie

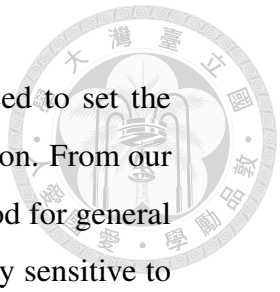
Figure 6.26: The dodecahedron can (a) be attached to physical objects such as a keyboard for tracking in VR or (b) be used as a simple 12-sided VR die.

ing line search. We show empirically that the algorithm can be accelerated by considering only the most relevant parts of the square marker for direct alignment. We also show that the bundle adjustment calibration of the handmade dodecahedron is essential and effective at correcting systematic errors in the model. Through a combination of simulation and experimentation, we characterize the system's sensitivity to shot noise, spatial blur, and image resolution to provide practitioners a useful guide for evaluating its applicability.

6.9.1 Limitations and Future Work

Despite the ease-of-construction and setup of our proposed system, it has some significant drawbacks. The proposed computer vision algorithm is slow by the standards of Lumitrack [154] or motion capture systems which can achieve a throughput of 300-800Hz. Because the algorithm is run on a PC, it incurs the latency of transferring the image to the host in addition to processing time. Although we show graceful degradation of the algorithm accuracy with camera resolution, the accuracy and the working volume of the system is ultimately limited by the angular resolution of the chosen camera system and the robustness of the binary square fiducial marker recognition software.

doi:10.6342/NTU201800854



Since the tracking accuracy suffers from motion blur, we need to set the exposure time of the camera to a reasonable value for the application. From our experiments, we find that a maximum exposure time of 4ms is good for general writing or drawing. Therefore, if the imaging system is sufficiently sensitive to produce bright enough images in 4ms to detect markers, our tracking system works properly. If the input frame is too dark for the ArUco marker detector to detect markers, our system will not work. In this case, we need to either add more light or improve the imaging system (with a better sensor or a faster lens).

Our presented stylus contains no electronic components, but the proposed computer vision system can easily be augmented with buttons for discrete input and an inertial measurement unit to reduce latency and increase throughput. To simplify the VR setup, we could attach the DodecaPen camera to the headset instead of setting it on a desk, since the headset is also tracked. Although we have demonstrated that only part of the binary square fiducial marker is useful for dense alignment, we still transfer the entire image from the camera to the host. Integrating on-camera compute or new sensing modalities such as event cameras may further reduce latency and improve throughput. The proposed system cannot handle occlusion because it relies on a single camera, but occlusion can be addressed with the addition of more cameras at the cost of additional setup complexity and calibration.

doi:10.6342/NTU201800854

Chapter 7

Conclusion

In this dissertation, we have interpreted the formulation of the 6DoF object pose recovering problem in Chapter 2 and given a comprehensive introduction to the related work of this topic in Chapter 3. Previous approaches have been thoroughly analyzed and a number of new techniques have been presented in Chapter 4 to Chapter 6. The main achievements are:

- A large-scale object pose tracking benchmark dataset consisting of RGB-D video sequences of 2D and 3D targets with ground-truth information (Chapter 4). The videos are recorded under various lighting conditions, different motion patterns and speeds with the help of a *programmable robotic arm*.

- A novel and robust scheme to annotate the ground-truth poses by leveraging the clear infrared images recorded by the *global-shutter* infrared camera with fast shutter speed from the *Kinect V2* sensor, which enables us to record sequence even under rapid motions (Chapter 4).

- An efficient *direct approach* of approximate pose estimation for planar objects which is posed as a template matching problem (Chapter 5). The proposed method performs robustly even when the target images contain less textured surfaces or motion blurs.

doi:10.6342/NTU201800854

- An accurate pose refinement strategy using a *Lucas-Kanade dense alignment* scheme (Chapter 5,6). In this approach, the image changes with respect to the 6DoF pose, which is parameterized as a 6D vector consisting of the 3D *rotation vector* and the 3D *translation vector*, are approximated using the first-order Taylor series. In addition, a *backtracking line search* is performed to ensure the convergence of Gauss-Newton iteration within the pose refinement technique.

- A practical method for planar object *pose disambiguation*, which find all possible poses first and then the one with smallest *appearance distance* between the camera image and the target image is considered as the estimated pose Chapter 5.

- An accurate 6DoF pose tracking solution by leveraging both the binary square fiducial marker toolkit and the proposed pose refinement scheme Chapter 6. As each fiducial marker has many sharp edges and corners, it is well-suited for providing a precise pose using dense alignment methods. In addition, since there are large portions of the fiducial marker that do not significantly contribute to the pose estimation procedure, we can take advantage of this by selectively masking out flat regions ahead of time on markers. A significant acceleration of the algorithm can be achieved without compromising tracking quality using this masking technique.

- A one-time *model calibration* procedure using bundle adjustment based on the proposed pose refinement algorithm Chapter 6. This technique can be employed not only for a dodecahedron but also for any 3D model composed of planes.

- Extensive experimental evaluations of the proposed method as well as previous approaches on both synthetic data and real data (Chapter 4,5,6).

- The implementation of an accurate pen-trajectory tracking solution, which

doi:10.6342/NTU201800854

is comparable to state-of-the-art professional motion capture systems Chapter 6.

- Demonstrations of the proposed accurate and easy-to-setup 6DoF stylus tracking system for the application of drawing in 2D and 3D as well as object manipulation in a virtual reality environment.

7.1 Discussion and Future Work

Recently, object pose estimation has benefitted from the advent of deep learning based approaches and the possibility of using large datasets for training such methods, and we believe this trend will continue in the future. But because these deep learning based approaches generally cannot give pretty accurate results, another pose refinement procedure is preferable to use. Traditionally, the iterative closest point algorithm is commonly applied to accomplish getting a more accurate pose. However, since the depth image obtained by present sensors is still noisy (as we have discussed in Section 4.2), the pose recovering results may not be entirely satisfactory. In contrast, because the proposed pose refinement approach has been demonstrated to improve the accuracy of the estimated pose significantly with RGB images (which can be captured clearly by present sensors), it is suitable for being augmented by any object pose estimation method especially when the object is composed of planes. But one should still be careful with the appearance consistency between the targets in the camera image and its original representation when using the dense alignment strategy.

There are many object pose estimation and tracking datasets which use fiducial markers to establish ground-truth poses. Nonetheless, as we have presented in Chapter 6, the marker corner alignment is insufficient for robust and accurate tracking. Therefore, the pose annotation process in the previous benchmark dataset may not be reliable. By employing the pose tracking solution proposed in Chapter 6, it is conceivable to achieve more accurate and robust pose annotation results.

doi:10.6342/NTU201800854

We also encourage everyone to utilize the dense pose refinement technique proposed in Section 6.4 when using any fiducial marker toolkit as it can phenomenally enhance the tracking quality with slight additional computational cost.

Pose estimation for planar objects can explicitly take advantage of depth images as the estimated pose can, therefore, be unambiguous. The candidate poses in the approximate pose estimation step presented in Section 5.1 can also be significantly reduced. If there is no depth information, we can still use some temporal filtering strategy to not only disambiguate the estimated pose but further improve the result accuracy as well.

doi:10.6342/NTU201800854

Reference

[1] E. Lachat, H. Macher, M. Mittet, T. Landes, and P. Grussenmeyer, "First Experiences with Kinect V2 Sensor for Close Range 3d Modelling," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences (ISPRS Archives)*, vol. 40, no. 5, p. 93, 2015. ix, 51, 52

[2] R. Mur-Artal and J. D. Tardós, "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," *IEEE Transactions on Robotics (T-RO)*, vol. 33, no. 5, pp. 1255–1262, 2017. xi, 6, 9, 46, 65, 66, 68, 82

[3] H.-Y. Tseng, P.-C. Wu, M.-H. Yang, and S.-Y. Chien, "Direct 3D Pose Estimation of a Planar Target," in *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2016, pp. 1–9. xiii, 4, 23, 35, 65, 66, 86, 100, 125

[4] S. Gauglitz, T. Höllerer, and M. Turk, "Evaluation of Interest Point Detectors and Feature Descriptors for Visual Tracking," *International Journal of Computer Vision (IJCV)*, vol. 94, no. 3, pp. 335–360, 2011. xiii, xiv, xx, 41, 43, 86, 99, 100, 108, 109, 110, 111, 112, 113, 114

[5] S. Hinterstoisser, V. Lepetit, S. Ilic, S. Holzer, G. Bradski, K. Konolige, and N. Navab, "Model Based Training, Detection and Pose Estimation of Texture-Less 3D Objects in Heavily Cluttered Scenes," in *Proceedings of*

doi:10.6342/NTU201800854

Asian Conference on Computer Vision (ACCV), 2012, pp. 548–562. 4, 5, 23, 35, 41, 43, 54

[6] V. Lepetit, J. Pilet, and P. Fua, "Point matching as a classification problem for fast and robust object pose estimation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2004, pp. II–244–II–250. 4

[7] A. Collet, D. Berenson, S. S. Srinivasa, and D. Ferguson, "Object Recognition and Full Pose Registration from a Single Image for Robotic Manipulation," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2009, pp. 48–55. 4, 35

[8] J. Tang, S. Miller, A. Singh, and P. Abbeel, "A Textured Object Recognition Pipeline for Color and Depth Image Data," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3467–3474. 4, 35

[9] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, 2004. 4, 26, 65, 73, 98

[10] G. Yu and J.-M. Morel, "ASIFT: An Algorithm for Fully Affine Invariant Comparison," *Image Processing On Line (IPOL)*, vol. 1, pp. 11–38, 2011. 4, 65, 99

[11] M. A. Fischler and R. C. Bolles, "RANdom SAmple Consensus: A Paradigm for Model Fitting With applications to Image Analysis and Automated Cartography," *Communications of the ACM (CACM)*, vol. 24, no. 6, pp. 381–395, 1981. 4, 26, 65, 99

[12] V. Lepetit, F. Moreno-Noguer, and P. Fua, "EPNP: An Accurate O(n) Solution to the PnP Problem," *International Journal of Computer Vision (IJCV)*, vol. 81, no. 2, pp. 155–166, 2009. 4, 27, 99

doi:10.6342/NTU201800854

[13] Y. Zheng, Y. Kuang, S. Sugimoto, K. Astrom, and M. Okutomi, "Revisiting the PnP Problem: A Fast, General and Optimal Solution," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 2344–2351. 4, 27, 38, 53, 65, 99, 106

[14] A. Tejani, D. Tang, R. Kouskouridas, and T.-K. Kim, "Latent-Class Hough Forests for Object Detection and Pose Estimation," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2014, pp. 462–477. 4, 5, 35, 41, 43, 54

[15] E. Brachmann, A. Krull, F. Michel, S. Gumhold, J. Shotton, and C. Rother, "Learning 6D Object Pose Estimation Using 3D Object Coordinates," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2014, pp. 536–551. 4, 5, 35, 41, 43, 54

[16] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, and C. Rother, "Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3364–3372. 4, 24, 35, 36, 65, 66, 67, 82

[17] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep Learning of Local RGB-D Patches for 3D Object Detection and 6D Pose Estimation," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016, pp. 205–220. 4, 36

[18] Y. Park, V. Lepetit, and W. Woo, "Texture-Less Object Tracking with Online Training using An RGB-D Camera," in *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2011, pp. 121–126. 5, 39

doi:10.6342/NTU201800854

[19] V. A. Prisacariu and I. D. Reid, "PWP3D: Real-Time Segmentation and Tracking of 3D Objects," *International Journal of Computer Vision (IJCV)*, vol. 98, no. 3, pp. 335–354, 2012. 5, 39, 65, 66

[20] C. Choi and H. I. Christensen, "RGB-D Object Tracking: A Particle Filter Approach on GPU," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 121–126. 5

[21] H. Tjaden, U. Schwanecke, and E. Schömer, "Real-Time Monocular Segmentation and Pose Tracking of Multiple Objects," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016, pp. 423–438. 5, 39, 78, 83

[22] E. Marchand, H. Uchiyama, and F. Spindler, "Pose Estimation for Augmented Reality: A Hands-On Survey," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 22, no. 12, pp. 2633–2651, 2016. 5

[23] M. Billinghurst, A. Clark, G. Lee *et al.*, "A Survey of Augmented Reality," *Foundations and Trends® in Human-Computer Interaction*, vol. 8, no. 2-3, pp. 73–272, 2015. 5

[24] C. Choi and H. I. Christensen, "RGB-D Object Tracking: A Particle Filter Approach on GPU," in *Proceedings of IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2013, pp. 1084–1091. 5, 39, 42, 43

[25] A. Krull, F. Michel, E. Brachmann, S. Gumhold, S. Ihrke, and C. Rother, "6-DOF Model Based Tracking via Object Coordinate Regression," in *Proceedings of Asian Conference on Computer Vision (ACCV)*, 2014, pp. 384–399. 5, 39, 41, 43

[26] C. Rennie, R. Shome, K. E. Bekris, and A. F. De Souza, "A Dataset for Improved RGBD-based Object Detection and Pose Estimation for Warehouse Pick-and-Place," *IEEE Robotics and Automation Letters (RA-L)*, vol. 1, no. 2, pp. 1179–1185, 2016. 5, 41, 43

doi:10.6342/NTU201800854

[27] H. Durrant-Whyte and T. Bailey, "Simultaneous Localization and Mapping: Part I," *IEEE Robotics and Automation Magazine (RAM)*, vol. 13, no. 2, pp. 99–110, 2006. 6

[28] T. Bailey and H. Durrant-Whyte, "Simultaneous Localization and Mapping: Part II," *IEEE Robotics and Automation Magazine (RAM)*, vol. 13, no. 3, pp. 108–117, 2006. 6

[29] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," *IEEE Transactions on Robotics (T-RO)*, vol. 31, no. 5, pp. 1147–1163, 2015. 6

[30] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-Time Single Camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 29, no. 6, pp. 1052–1067, 2007. 6

[31] G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," in *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2007, pp. 225–234. 6

[32] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2320–2327. 6

[33] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-Scale Direct Monocular SLAM," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2014, pp. 834–849. 6

[34] D. Nistér, O. Naroditsky, and J. Bergen, "Visual Odometry," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2004, pp. 652–659. 6

doi:10.6342/NTU201800854

[35] D. Scaramuzza and F. Fraundorfer, "Visual Odometry [Tutorial]," *IEEE Robotics and Automation Magazine (RAM)*, vol. 18, no. 4, pp. 80–92, 2011. 6

[36] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast Semi-direct Monocular Visual Odometry," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 15–22. 6

[37] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, "An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics," *Intelligent Industrial Systems*, vol. 1, no. 4, pp. 289–311, 2015. 6

[38] J. Engel, V. Koltun, and D. Cremers, "Direct Sparse Odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 40, no. 3, pp. 611–625, 2018. 6

[39] H. C. Longuet-Higgins, "A Computer Algorithm for Reconstructing a Scene from Two Projections," *Nature*, vol. 293, no. 5828, pp. 133–135, 1981. 6

[40] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski, "Building Rome in a Day," *Communications of the ACM (CACM)*, vol. 54, no. 10, pp. 105–112, 2011. 6

[41] O. Özyeşil, V. Voroninski, R. Basri, and A. Singer, "A Survey of Structure from Motion," *Acta Numerica*, vol. 26, pp. 305–364, 2017. 6

[42] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle Adjustment in the Large," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2010, pp. 29–42. 7

[43] N. Snavely, S. M. Seitz, and R. Szeliski, "Skeletal Graphs for Efficient Structure from Motion," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2008, p. 2. 7

doi:10.6342/NTU201800854

[44] M. Havlena, A. Torii, and T. Pajdla, "Efficient Structure from Motion by Graph Optimization," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2010, pp. 100–113. 7

[45] A. Irschara, C. Zach, J.-M. Frahm, and H. Bischof, "From Structure-from-Motion Point Clouds to Fast Location Recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 2599–2606. 7

[46] T. Sattler, B. Leibe, and L. Kobbelt, "Improving Image-Based Localization by Active Correspondence Search," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2012, pp. 752–765. 7

[47] X. Sun, Y. Xie, P. Luo, and L. Wang, "A Dataset for Benchmarking Image-based Localization," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7436–7444. 7

[48] G. Schindler, M. Brown, and R. Szeliski, "City-scale location recognition," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2007, pp. 1–7. 7

[49] N. Snavely, S. M. Seitz, and R. Szeliski, "Photo Tourism: Exploring Photo Collections in 3D," in *ACM Transactions on Graphics (TOG)*, vol. 25, no. 3, 2006, pp. 835–846. 7

[50] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, "Worldwide Pose Estimation Using 3D Point Clouds," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2012, pp. 15–29. 7

[51] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 22, no. 11, pp. 1330–1334, 2000. 7

doi:10.6342/NTU201800854

[52] C.-K. Liang, L.-W. Chang, and H. H. Chen, "Analysis and Compensation of Rolling Shutter Effect," *IEEE Transactions on Image Processing (TIP)*, vol. 17, no. 8, pp. 1323–1330, 2008. 8

[53] J. Sun, N.-N. Zheng, and H.-Y. Shum, "Stereo matching using belief propagation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 25, no. 7, pp. 787–800, 2003. 8

[54] C. Shi, G. Wang, X. Yin, X. Pei, B. He, and X. Lin, "High-Accuracy Stereo Matching Based on Adaptive Ground Control Points," *IEEE Transactions on Image Processing (TIP)*, vol. 24, no. 4, pp. 1412–1423, 2015. 8

[55] Z. Zhang, "Microsoft Kinect Sensor and Its Effect," *IEEE Multimedia*, vol. 19, no. 2, pp. 4–10, 2012. 8

[56] L. Yang, L. Zhang, H. Dong, A. Alelaiwi, and A. El Saddik, "Evaluating and Improving the Depth Accuracy of Kinect for Windows V2," *IEEE Sensors Journal*, vol. 15, no. 8, pp. 4275–4285, 2015. 8

[57] J. Davis, R. Ramamoorthi, and S. Rusinkiewicz, "Spacetime Stereo: A Unifying Framework for Depth from Triangulation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2003, pp. 359–366. 8

[58] D. Scharstein and R. Szeliski, "High-Accuracy Stereo Depth Maps Using Structured Light," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2003, pp. 195–202. 8

[59] S. B. Gokturk, H. Yalcin, and C. Bamji, "A Time-Of-Flight Depth Sensor – System Description, Issues and Solutions," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW)*, 2004, pp. 35–43. 8

doi:10.6342/NTU201800854

[60] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, "ElasticFusion: Real-Time Dense SLAM and Light Source Estimation," *International Journal of Robotics Research (IJRR)*, vol. 35, no. 14, pp. 1697–1716, 2016. 9, 46, 65, 66, 82

[61] P.-C. Wu, Y.-Y. Lee, H.-Y. Tseng, H.-I. Ho, M.-H. Yang, and S.-Y. Chien, "A Benchmark Dataset for 6DoF Object Pose Tracking," in *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, 2017, pp. 186–191. 11

[62] P.-C. Wu, H.-Y. Tseng, M.-H. Yang, and S.-Y. Chien, "Direct Pose Estimation for Planar Objects," *Computer Vision and Image Understanding (CVIU)*, 2018. 11

[63] P.-C. Wu, R. Wang, K. Kin, C. Twigg, S. Han, M.-H. Yang, and S.-Y. Chien, "Dodecapen: Accurate 6dof tracking of a passive stylus," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, 2017, pp. 365–374. 11

[64] F. S. Grassia, "Practical Parameterization of Rotations Using the Exponential Map," *Journal of Graphics Tools (JGT)*, vol. 3, no. 3, pp. 29–48, 1998. 14

[65] D. Eberly, "Euler Angle Formulas," Geometric Tools, LLC, Tech. Rep., 2008. 17, 87

[66] Wikipedia contributors, "Rodrigues' rotation formula — Wikipedia, the free encyclopedia," 2018, [Online; accessed 21-April-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Rodrigues_rotation_formula&oldid=822424523 18

[67] ——, "Axis–angle representation — Wikipedia, the free encyclopedia," 2017, [Online; accessed 21-April-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Axis_angle_representation&oldid=806689583 18

doi:10.6342/NTU201800854

[68] E. Eade, "Lie Groups for 2D and 3D Transformations," 2013, [Online; accessed 21-April-2018]. [Online]. Available: http://ethaneade.com/lie.pdf 19

[69] J. B. Kuipers, *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality*. Princeton University Press, 2011. 20

[70] J. Van Waveren, "From quaternion to matrix and back," *Id Software, Inc*, 2005. 21

[71] J. Shotton, B. Glocker, C. Zach, S. Izadi, A. Criminisi, and A. Fitzgibbon, "Scene Coordinate Regression Forests for Camera Relocalization in RGB-D Images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 2930–2937. 23

[72] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-Up Robust Features (SURF)," *Computer Vision and Image Understanding (CVIU)*, vol. 110, no. 3, pp. 346–359, 2008. 26

[73] P. F. Alcantarilla, A. Bartoli, and A. J. Davison, "KAZE Features," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2012, pp. 214–227. 26

[74] P. F. Alcantarilla, J. Nuevo, T. Solutions, and A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces," in *Proceedings of British Machine Vision Conference (BMVC)*, 2013. 26

[75] J. Weickert, B. T. H. Romeny, and M. A. Viergever, "Efficient And Reliable Schemes For Nonlinear Diffusion Filtering," *IEEE Transactions on Image Processing (TIP)*, vol. 7, no. 3, pp. 398–410, 1998. 26

doi:10.6342/NTU201800854

[76] E. Rosten and T. Drummond, "Machine Learning for High-Speed Corner Detection," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2006, pp. 430–443. 26, 72

[77] E. Mair, G. D. Hager, D. Burschka, M. Suppa, and G. Hirzinger, "Adaptive and Generic Corner Detection Based on the Accelerated Segment Test," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2010, pp. 183–196. 26

[78] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary Robust Independent Elementary Features," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2010, pp. 778–792. 26

[79] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary Robust Invariant Scalable Keypoints," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2548–2555. 26

[80] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An Efficient Alternative to SIFT or SURF," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 2564–2571. 26, 72

[81] A. Alahi, R. Ortiz, and P. Vandergheynst, "FREAK: Fast Retina Keypoint," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 510–517. 26

[82] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete Solution Classification for the Perspective-Three-Point Problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 25, no. 8, pp. 930–943, 2003. 26

[83] L. Kneip, D. Scaramuzza, and R. Siegwart, "A Novel Parametrization of the Perspective-Three-Point Problem for a Direct Computation of Absolute Camera Position and Orientation," in *Proceedings of IEEE Conference on*

doi:10.6342/NTU201800854

Computer Vision and Pattern Recognition (CVPR), 2011, pp. 2969–2976. 26

[84] T. Ke and S. I. Roumeliotis, "An Efficient Algebraic Solution to the Perspective-Three-Point Problem," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7225–7233. 26

[85] O. Chum and J. Matas, "Matching with PROSAC-PROgressive SAmple Consensus," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 220–226. 26

[86] V. Fragoso, P. Sen, S. Rodriguez, and M. Turk, "EVSAC: Accelerating Hypotheses Generation by Modeling Matching Scores with Extreme Value Theory," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 2472–2479. 26

[87] C.-P. Lu, G. D. Hager, and E. Mjolsness, "Fast and Globally Convergent Pose Estimation from Video Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 22, no. 6, pp. 610–622, 2000. 27, 38

[88] G. Schweighofer and A. Pinz, "Robust Pose Estimation from a Planar Target," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 28, no. 12, pp. 2024–2030, 2006. 27, 38, 85, 99

[89] J. A. Hesch and S. I. Roumeliotis, "A Direct Least-Squares (DLS) Method for PnP," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 383–390. 27

[90] S. Li, C. Xu, and M. Xie, "A Robust O(n) Solution to the Perspective-n-Point Problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 34, no. 7, pp. 1444–1450, 2012. 27

doi:10.6342/NTU201800854

[91] L. Kneip, H. Li, and Y. Seo, "UPnP: An Optimal O(n) Solution to the Absolute Pose Problem with Universal Applicability," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2014, pp. 127–142. 27, 99

[92] L. Ferraz, X. Binefa, and F. Moreno-Noguer, "Very Fast Solution to the PnP Problem with Algebraic Outlier Rejection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 501–508. 27

[93] ——, "Leveraging Feature Uncertainty in the PnP Problem," in *Proceedings of British Machine Vision Conference (BMVC)*, 2014, pp. 1–13. 27

[94] W. Kabsch, "A Solution for the Best Rotation to Relate Two Sets of Vectors," *Acta Crystallographica*, vol. 32, no. 5, pp. 922–923, 1976. 27, 35

[95] Wikipedia contributors, "Singular-value decomposition — Wikipedia, the free encyclopedia," 2018, [Online; accessed 26-April-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Singular-value_decomposition&oldid=837622148 28

[96] B. D. Lucas and T. Kanade, "An Iterative Image Registration Technique with an Application to Stereo Vision," in *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, vol. 81, 1981, pp. 674–679. 28, 53

[97] E. W. Weisstein, "Normal Equation," 2018, from MathWorld–A Wolfram Web Resource. [Online]. Available: http://mathworld.wolfram.com/NormalEquation.html 30

[98] Wikipedia contributors, "Matrix calculus — Wikipedia, the free encyclopedia," 2018, [Online; accessed 30-April-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Matrix_calculus&oldid=838546380 30

doi:10.6342/NTU201800854

[99] K. B. Petersen, M. S. Pedersen *et al.*, "The Matrix Cookbook," *Technical University of Denmark*, vol. 7, no. 15, p. 510, 2008. 30

[100] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," *International Journal of Computer Vision (IJCV)*, vol. 56, no. 3, pp. 221–255, 2004. 31

[101] H.-Y. Shum and R. Szeliski, "Construction of Panoramic Image Mosaics with Global and Local Alignment," *Panoramic Vision*, pp. 227–268, 2001. 31

[102] G. D. Hager and P. N. Belhumeur, "Efficient Region Tracking with Parametric Models of Geometry and Illumination," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 20, no. 10, pp. 1025–1039, 1998. 31

[103] S. Baker and I. Matthews, "Equivalence and Efficiency of Image Alignment Algorithms," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2001, pp. 1090–1097. 31

[104] S. Benhimane and E. Malis, "Homography-based 2D Visual Tracking and Servoing," *International Journal of Robotics Research (IJRR)*, vol. 26, no. 7, pp. 661–676, 2007. 31

[105] A. Crivellaro, P. Fua, and V. Lepetit, "Dense Methods for Image Alignment with an Application to 3D Tracking," EPFL, Tech. Rep., 2014. 31

[106] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 14, no. 2, pp. 239–256, 1992. 31

[107] F. Pomerleau, F. Colas, R. Siegwart *et al.*, "A Review of Point Cloud Registration Algorithms for Mobile Robotics," *Foundations and Trends® in Robotics*, vol. 4, no. 1, pp. 1–104, 2015. 31

doi:10.6342/NTU201800854

[108] Y. Chen and G. Medioni, "Object ModelLing by Registration of Multiple Range Images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992. 32

[109] J. Nocedal and S. J. Wright, "Numerical Optimization," *Springer*, 2006. 32, 35

[110] S. Rusinkiewicz and M. Levoy, "Efficient Variants of the ICP Algorithm," in *Proceedings of IEEE International Conference on 3-D Digital Imaging and Modeling (3DIM)*, 2001, pp. 145–152. 32

[111] H. Pottmann, S. Leopoldseder, and M. Hofer, "Registration Without ICP," *Computer Vision and Image Understanding (CVIU)*, vol. 95, no. 1, pp. 54–71, 2004. 32

[112] S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, and V. Lepetit, "Multimodal Templates for Real-Time Detection of Texture-less Objects in Heavily Cluttered Scenes," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 858–865. 35

[113] A. Krull, E. Brachmann, F. Michel, M. Ying Yang, S. Gumhold, and C. Rother, "Learning Analysis-by-Synthesis for 6D Pose Estimation in RGB-D Images," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 954–962. 35, 36

[114] A. Doumanoglou, R. Kouskouridas, S. Malassiotis, and T.-K. Kim, "Recovering 6D Object Pose and Predicting Next-Best-View in the Crowd," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3583–3592. 35, 83

[115] Y. Konishi, Y. Hanzawa, M. Kawade, and M. Hashimoto, "Fast 6D Pose Estimation from a Monocular Image Using Hierarchical Pose Trees," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016, pp. 398–413. 35, 83

doi:10.6342/NTU201800854

[116] A. Krull, E. Brachmann, S. Nowozin, F. Michel, J. Shotton, and C. Rother, "PoseAgent: Budget-Constrained 6D Object Pose Estimation via Reinforcement Learning," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2017. 36

[117] F. Michel, A. Kirillov, E. Brachmann, A. Krull, S. Gumhold, B. Savchynskyy, and C. Rother, "Global Hypothesis Generation for 6D Object Pose Estimation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 462–471. 36

[118] P. Wohlhart and V. Lepetit, "Learning Descriptors for Object Recognition and 3D Pose Estimation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3109–3118. 36, 82

[119] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587. 36

[120] R. Girshick, "Fast R-CNN," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448. 36

[121] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 39, no. 6, pp. 1137–1149, 2017. 36

[122] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988. 36

[123] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of IEEE Conference*

doi:10.6342/NTU201800854

on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779–788. 36

[124] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6517–6525. 36

[125] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2016, pp. 21–37. 36

[126] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1521–1529. 36

[127] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes," in *Proceedings of Robotics: Science and Systems (RSS)*, 2018. 36, 41, 43

[128] M. Rad and V. Lepetit, "BB8: A Scalable, Accurate, Robust to Partial Occlusion Method for Predicting the 3D Poses of Challenging Objects without Using Depth," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3848–3856. 36

[129] B. Tekin, S. N. Sinha, and P. Fua, "Real-Time Seamless Single Shot 6D Object Pose Prediction," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 36

[130] M. Rad, M. Oberweger, and V. Lepetit, "Feature Mapping for Learning Fast and Accurate 3D Pose Inference from Synthetic Images," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 36

doi:10.6342/NTU201800854

[131] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-DoF Object Pose from Semantic Keypoints," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 2011–2018. 36

[132] P.-C. Wu, J.-H. Lai, J.-L. Wu, and S.-Y. Chien, "Stable Pose Estimation with a Motion Model in Real-Time Application," in *Proceedings of IEEE International Conference on Multimedia and Expo (ICME)*, 2012, pp. 314–319. 37

[133] P.-C. Wu, Y.-H. Tsai, and S.-Y. Chien, "Stable Pose Tracking from a Planar Target With an Analytical Motion Model in Real-Time Applications," in *Proceedings of IEEE International Workshop on Multimedia Signal Processing (MMSP)*, 2014, pp. 1–6. 37, 85

[134] Z. Kukelova, M. Bujnak, and T. Pajdla, "Automatic Generator of Minimal Problem Solvers," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2008, pp. 302–315. 38, 94

[135] T. Collins and A. Bartoli, "Infinitesimal Plane-Based Pose Estimation," *International Journal of Computer Vision (IJCV)*, vol. 109, no. 3, pp. 252–286, 2014. 38, 65, 99

[136] V. Lepetit, P. Fua *et al.*, "Monocular Model-Based 3D Tracking of Rigid Objects: A Survey," *Foundations and Trends® in Computer Graphics and Vision*, vol. 1, no. 1, pp. 1–89, 2005. 38

[137] Y. Park, V. Lepetit, and W. Woo, "Multiple 3D Object Tracking for Augmented Reality," in *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2008, pp. 117–120. 39

[138] C. Schmaltz, B. Rosenhahn, T. Brox, and J. Weickert, "Region-Based Pose Tracking with Occlusions Using 3D Models," vol. 23, no. 3, pp. 557–577, 2012. 39

doi:10.6342/NTU201800854

[139] J. Hexner and R. R. Hagege, "2D-3D Pose Estimation of Heterogeneous Objects Using a Region Based Approach," *International Journal of Computer Vision (IJCV)*, vol. 118, no. 1, pp. 95–112, 2016. 39, 83

[140] O. Korkalo and S. Kahn, "Real-Time Depth Camera Tracking with CAD Models and ICP," *Journal of Virtual Reality and Broadcasting (JVRB)*, vol. 13, no. 1, 2016. 39

[141] W. Kehl, F. Tombari, S. Ilic, and N. Navab, "Real-Time 3D Model Tracking in Color and Depth on a Single CPU Core," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 745–753. 39

[142] D. J. Tan, F. Tombari, S. Ilic, and N. Navab, "A Versatile Learning-Based 3D Temporal Tracker: Scalable, Robust, Online," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 693–701. 39

[143] D. J. Tan, N. Navab, and F. Tombari, "Looking Beyond the Simple Scenarios: Combining Learners and Optimizers in 3D Temporal Tracking," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 23, no. 11, pp. 2399–2409, 2017. 39

[144] M. Garon and J.-F. Lalonde, "Deep 6-DOF Tracking," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 23, no. 11, pp. 2410–2418, 2017. 39, 41, 43

[145] Y. Li, G. Wang, X. Ji, Y. Xiang, and D. Fox, "DeepIM: Deep Iterative Matching for 6D Pose Estimation," *arXiv preprint arXiv:1804.00175*, 2018. 39

[146] H. Kato and M. Billinghurst, "Marker Tracking and HMD Calibration for a Video-based Augmented Reality Conferencing System," in *Proceedings*

doi:10.6342/NTU201800854

of IEEE and ACM International Workshop on Augmented Reality (IWAR), 1999, pp. 85–94. 39

[147] D. Wagner and D. Schmalstieg, "ARToolKitPlus for Pose Tracking on Mobile Devices," in *Proceedings of ComputerVisionWinterWorkshop (CVWW)*, 2007, pp. 139—-146. 39

[148] M. Fiala, "ARTag, a Fiducial Marker System Using Digital Techniques," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, 2005, pp. 590–596. 39

[149] ——, "Designing highly reliable fiducial markers," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, vol. 32, no. 7, pp. 1317–1324, 2010. 39, 125

[150] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014. 40, 126

[151] S. Garrido-Jurado, R. Munoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer, "Generation of Fiducial Marker Dictionaries using Mixed Integer Linear Programming," *Pattern Recognition*, vol. 51, pp. 481–491, 2016. 40, 126

[152] S. Heo, J. Han, S. Choi, S. Lee, G. Lee, H.-E. Lee, S. Kim, W.-C. Bang, D. Kim, and C. Kim, "IrCube Tracker: An Optical 6DOF Tracker based on LED Directivity," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, 2011, pp. 577–586. 40

[153] J. Han, S. Heo, H.-E. Lee, and G. Lee, "The IrPen: A 6-DOF Pen for Interaction with Tablet Computers," *IEEE Computer Graphics and Applications (CG&A)*, vol. 34, no. 3, pp. 22–29, 2014. 40

doi:10.6342/NTU201800854

[154] R. Xiao, C. Harrison, K. D. Willis, I. Poupyrev, and S. E. Hudson, "Lumitrack: Low Cost, High Precision, High Speed Tracking with Projected m-Sequences," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, 2013, pp. 3–12. 40, 155

[155] V. Bubník and V. Havran, "Light Chisel: 6DOF Pen Tracking," *Computer Graphics Forum (CGF)*, vol. 34, no. 2, pp. 325–336, 2015. 40

[156] J. Tompkin, S. Muff, J. McCann, H. Pfister, J. Kautz, M. Alexa, and W. Matusik, "Joint 5D Pen Input for Light Field Displays," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, 2015, pp. 637–647. 40

[157] HTC, *HTC Vive*, Accessed: 2018-05-07. [Online]. Available: https://www.vive.com/ 40

[158] Oculus, *Oculus Touch*, Accessed: 2018-05-07. [Online]. Available: https://www.oculus.com/rift/ 40

[159] Sony, *PlayStation Move Motion Controller*, Accessed: 2018-05-07. [Online]. Available: https://www.playstation.com/en-us/explore/accessories/vr-accessories/playstation-move/ 40

[160] Razer, *Razer Hydra*, Accessed: 2018-05-07. [Online]. Available: https://www2.razerzone.com/au-en/gaming-controllers/razer-hydra-portal-2-bundle 40

[161] NaturalPoint, *OptiTrack*, Accessed: 2018-05-07. [Online]. Available: http://optitrack.com/ 40, 132

[162] Vicon, *Vicon*, Accessed: 2018-05-07. [Online]. Available: https://www.vicon.com/ 40

[163] Qualisys, *Qualisys*, Accessed: 2018-05-07. [Online]. Available: http://www.qualisys.com/ 40

doi:10.6342/NTU201800854

[164] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab, "A Dataset and Evaluation Methodology for Template-Based Tracking Algorithms," in *Proceedings of IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, 2009, pp. 145–151. 41, 43, 54, 100

[165] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-LESS: An RGB-D Dataset for 6D Pose Estimation of Texture-less Objects," in *Proceedings of IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 880–888. 41, 43

[166] S. Akkaladevi, M. Ankerl, C. Heindl, and A. Pichler, "Tracking Multiple Rigid Symmetric and Non-symmetric Objects in Real-Time Using Depth Data," in *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 5644–5649. 42, 43

[167] ReconstructMe, *ReconstructMe*, Accessed: 2018-05-07. [Online]. Available: http://reconstructme.net 42

[168] J.-Y. Bouguet, "Camera Calibration Toolbox for Matlab," *MATLAB*, 2004. 50, 53

[169] J. Shi and C. Tomasi, "Good Features to Track," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1994, pp. 593–600. 53

[170] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004. 65

[171] B. Glocker, J. Shotton, A. Criminisi, and S. Izadi, "Real-Time RGB-D Camera Relocalization via Randomized Ferns for Keyframe Encoding," *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, vol. 21, no. 5, pp. 571–583, 2015. 67

doi:10.6342/NTU201800854

[172] Y. Wu, J. Lim, and M.-H. Yang, "Object tracking benchmark," vol. 37, no. 9, pp. 1834–1848, 2015. 69

[173] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang, "Semantic Image Segmentation via Deep Parsing Network," in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1377–1385. 81

[174] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3431–3440. 81

[175] D. Oberkampf, D. F. DeMenthon, and L. S. Davis, "Iterative Pose Estimation Using Coplanar Points," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1993, pp. 626–627. 85

[176] S. Li and C. Xu, "Efficient Lookup Table Based Camera Pose Estimation for Augmented Reality," *Computer Animation and Virtual Worlds*, vol. 22, no. 1, pp. 47–58, 2011. 85

[177] S. Korman, D. Reichman, G. Tsur, and S. Avidan, "FasT-Match: Fast Affine Template Matching," *International Journal of Computer Vision (IJCV)*, vol. 121, no. 1, pp. 111–125, 2017. 86

[178] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2009. 86, 88

[179] Wikipedia contributors, "Delone set — Wikipedia, the free encyclopedia," 2017, [Online; accessed 8-May-2018]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Delone_set&oldid=795315991 87

[180] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H.-T. Lin, *Learning From Data*. AMLBook, 2012. 92

[181] M. J. Kearns and U. V. Vazirani, *An Introduction to Computational Learning Theory*. MIT Press, 1994. 93

doi:10.6342/NTU201800854

[182] G. Gallego and A. Yezzi, "A Compact Formula for the Derivative of a 3-D Rotation in Exponential Coordinates," *Journal of Mathematical Imaging and Vision (JMIV)*, vol. 51, no. 3, pp. 378–384, 2015. 95, 129

[183] H.-Y. Tseng, P.-C. Wu, Y.-S. Lin, and S.-Y. Chien, "D-PET: A Direct 6 DoF Pose Estimation and Tracking System on Graphics Processing Units," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS)*, 2017, pp. 1–4. 98

[184] H. Jegou, M. Douze, and C. Schmid, "Hamming Embedding and Weak Geometric Consistency for Large Scale Image Search," in *Proceedings of European Conference on Computer Vision (ECCV)*, 2008, pp. 304–317. 101

[185] T. Ha and W. Woo, "An Empirical Evaluation of Virtual Hand Techniques for 3D Object Manipulation in a Tangible Augmented Reality Environment," in *Proceedings of IEEE Symposium on 3D User Interfaces (3DUI)*, 2010, pp. 91–98. 125

[186] T. Petersen, "A Comparison of 2D-3D Pose Estimation Methods," *Aalborg University*, 2008. 126

[187] J.-Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade Feature Tracker: Description of the Algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001. 127

[188] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *Automatica*, vol. 11, no. 285-296, pp. 23–27, 1975. 143

[189] Wacom, *Wacom*, Accessed: 2018-05-12. [Online]. Available: https://www.wacom.com/ 152

[190] Anoto, *Anoto*, Accessed: 2018-05-12. [Online]. Available: http://www.anoto.com/ 152

doi:10.6342/NTU201800854

[191] T. Grossman, K. Hinckley, P. Baudisch, M. Agrawala, and R. Balakrishnan, "Hover Widgets: Using the Tracking State to Extend the Capabilities of Pen-operated Devices," in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems (CHI)*, 2006, pp. 861–870. 152

[192] S. Subramanian, D. Aliakseyeu, and A. Lucero, "Multi-layer Interaction for Digital Tables," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, 2006, pp. 269–272. 152

[193] T. A. Galyean and J. F. Hughes, "Sculpting: An Interactive Volumetric Modeling Technique," in *ACM SIGGRAPH Computer Graphics*, vol. 25, no. 4, 1991, pp. 267–274. 153

[194] Computer History Museum, *Mapping Sutherland's Volkswagen*, Accessed: 2018-05-12. [Online]. Available: http://www.computerhistory.org/revolution/computer-graphics-music-and-art/15/206/560 153

[195] P. Alliez, D. Cohen-Steiner, O. Devillers, B. Lévy, and M. Desbrun, "Anisotropic Polygonal Remeshing," in *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3, 2003, pp. 485–493. 153

[196] H. Ishii and B. Ullmer, "Tangible Bits: Towards Seamless Interfaces Between People, Bits and Atoms," in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems (CHI)*, 1997, pp. 234–241. 153

[197] R. Held, A. Gupta, B. Curless, and M. Agrawala, "3D Puppetry: A Kinect-based Interface for 3D Animation," in *Proceedings of ACM Symposium on User Interface Software and Technology (UIST)*, 2012, pp. 423–434. 153

doi:10.6342/NTU201800854