國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

以 RGB-D 相機在書桌環境完成物品建模與搜尋

RGB-D Based Desk Object Modelling and Search

盧恭弘

Kung-Hung Lu

指導教授：王傑智 博士

Advisor: Chieh-Chih Wang, Ph.D.

中華民國 一百零四 年 一 月

January, 2015

國立臺灣大學碩士學位論文
口試委員會審定書

以 RGB-D 相機在書桌環境完成物品建模與搜尋

RGB-D Based Desk Object Modelling and Search

本論文係盧恭弘君（學號 R02922102）在國立臺灣大學資訊工程學系完成之碩士學位論文，於民國 103 年 12 月 24 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：

王傑智

（指導教授）

宋開泰

[illegible]

[illegible]

系 主 任 趙坤茂

誌謝

誠摯的感謝王傑智老師於研究上的指導。與老師的討論總是讓我收穫良多，不僅僅是發現研究上的盲點，還有許多待人接物的道理，特別是老師對於做學問的熱情與實事求是的態度，在我灰心喪志時，提醒著我要向老師看齊，繼續不斷努力。感謝實驗室學長對我的照顧與指導，張峻華學長指點我學術寫作的要領並幫我釐清論文的重點。林坤立學長在程式實作上提供了我莫大的幫助。余宗哲、陳俊甫、周執中、徐慶祥學長引導我研究的方向並分享了許多寶貴的經驗。也感謝彥廷、邦丞、贇珺、泓志、鈞凱、志強與維剛的陪伴與砥礪，一同在實驗室打拼奮鬥。很慶幸能加入 PAL 實驗室，度過了充實且充滿歡笑的碩班生活，也認識了一群一輩子的好夥伴。最後，謹以此文獻給一直默默支持我的家人與女朋友。

中文摘要

在雜亂無章的書桌上尋找物品是一件惱人且瑣碎的工作。許多先前的研究嘗試使用物品辨識技術來解決在真實世界中尋找實體物品的問題。然而，此種方法通常需要事前費時且繁重的機器學習過程，且難以移植到其他情境中使用。在此篇論文中，透過使用 RGB-D 結構光距離感測器，我們提出了一個概念簡單卻效果顯著的方法，利用偵測場景穩定度來區分靜態場景與動態場景來完成書桌物品的建模與搜尋。在靜態場景中，藉由比較當前場景與資料庫中的物品模型來完成物品的分割與建模；而在動態場景中，我們使用多目標追蹤的技術來確認各物品的位置。同時，本論文實作了一個即時書桌物品搜尋系統，同時提供了物品的三維模型與其空間中所處位置的資訊予使用者，讓使用者能夠以簡單且直觀的方式來了解書桌上各物品的位置並進而完成尋找物品的任務。最後，在實驗中我們測試了各種書桌環境中常見的物品與不同的操作方式來展示此方法的可行性與有效性。

RGB-D BASED DESK OBJECT MODELLING AND SEARCH

Kung-Hung Lu

Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan

January 2015

*Submitted in partial fulfilment of
the requirements for the degree of
Master of Science*

Advisor: Chieh-Chih Wang

Thesis Committee:
Chieh-Chih Wang
Kai-Tai Song
Jen-Hwa Guo
Pei-Chun Lin

ABSTRACT

FINDING desk objects could be annoying with a messy desk. Many previous works used object recognition technique to deal with this object searching problem. However, it usually requires massive pre-training process, which has much more effort when it is transferred to another scenario. In this work, we propose a simple yet effective approach to accomplish desk object modelling and search using a static RGB-D camera. Assume that the desk could be monitored over time, the concept of scene stability is proposed to distinguish between stable and dynamic scene. Object segmentation and modelling are done concurrently by differentiating the current stable scene state and the model of object in database so the new object; while multiple objects tracking is adopted to find the locations of objects in dynamic scene. A user interface is designed in which both locations and appearances of the modelled objects are provided. It is easy for the user to have an understanding of the objects on the desk and the minimum effort is required to find a specific object. A variety of the desk objects with different sizes and thickness are tested, even the objects with indistinguishable volume. We also test the proposed approach for various manipulations. The experimental results demonstrate the feasibility and effectiveness of the proposed desk object modelling and search system.

Keywords: RGB-D video; Segmentation; Multi-Objects Tracking;

TABLE OF CONTENTS

LIST OF FIGURES

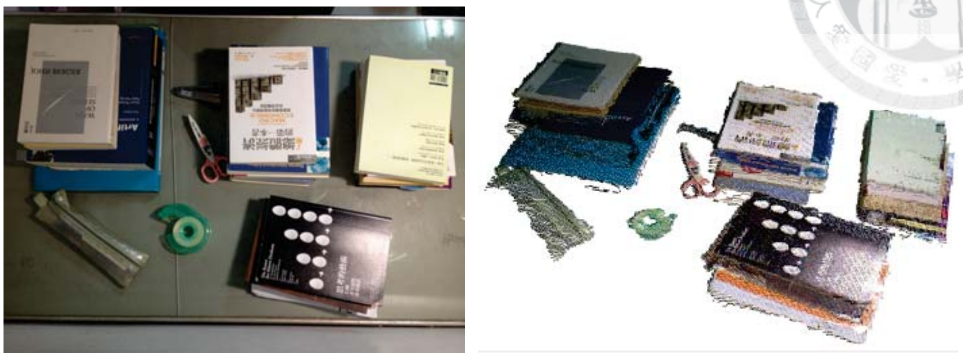
LIST OF TABLES

CHAPTER 1

Introduction

ALTHOUGH it is still debatable if a person is more productive using a messy desk, there is no doubt that it is not always easy to find an object or accessory on a cluttered desk as illustrated in Fig. 1.1(a). With the advance of technology, if the computer could help us remember where the object is and what object is on the desk, it will save much time for people to find their stuff in daily life. A reliable object finding system provides both the location of objects and a straightforward representation of objects which people can easily understand. One possible solution for the object finding system is to equip objects with extra sensors or tags (Pederson, 2001), (Ljungstrand et al., 2000), (Butz et al., 2004). However, as the equipment should be accomplished in advance, this solution could be intrusive for users and is not scalable. Another possible solution is to apply the object recognition technique to detect and localize objects in cluttered scenes (Lai and Fox, 2010) (Lai et al., 2011a) (Lai et al., 2011b) (Attamimi et al., 2010). Nevertheless, the main disadvantage is that training is needed to generate object models before applying the system.

In this thesis, we propose an effective object finding system in which a priori preparation is not required. With the use of the RGB-D camera, the 3D models of objects and their positions are inferred online. To build the 3D models for unseen objects, the main issue arises from the occlusion caused by the hand while objects are moving by people as illustrated in Fig. 1.2(a). Although it is possible to integrate skin detection to separate the hand and the object, in our work, a simple but effective approach is proposed based on the scene stability. The main observation is that in the desk scenario, there is often a period of time that the scene is stable, i.e. there are no moving hands or objects, as shown in



(a) A cluttered desk. (b) Object indexing from the proposed approach. The observed objects are modeled.

Figure 1.1. A motivating example of the proposed solution in which the object indexing is constructed over time. Search of object and accessary can be accomplished accordingly.

Fig. 1.2(b). Accordingly, by detecting the time periods that the scenes are stable, models of unseen objects can be built without the influence of hands.

In addition to generating models for unseen objects, as objects can be moved by people, moving object tracking is also integrated to update the positions of objects. However, as thin objects, such as papers and letters, can be commonly seen in the desk scenes, the main challenge is that it is difficult to differentiate stacked thin objects due to the depth uncertainty of the RGB-D camera. Therefore, we propose to infer the spatial relation between objects to assist data association in tracking, in which the order of the objects is constructed.

Based on the estimation results, our system also provides a friendly user interface in which the stable locations of the objects along with their pictures taken when appearing are shown. Without extra sensors on objects or specific operations, the user can easily retrieve the location and appearance information of all detected objects through our system. The experiments demonstrate that our system can manage desk objects ranging from large books to small accessories. The proposed system can run in real-time at 5Hz.

The rest of the thesis is organized as the follows. In Chapter 2, related works on the problem of object finding are reviewed. The proposed method is described in detail in Chapter 3. Chapter 4 demonstrates the experimental results, in which both successful and failed cases are discussed. In Chapter 5, the conclusion and the future work are addressed.

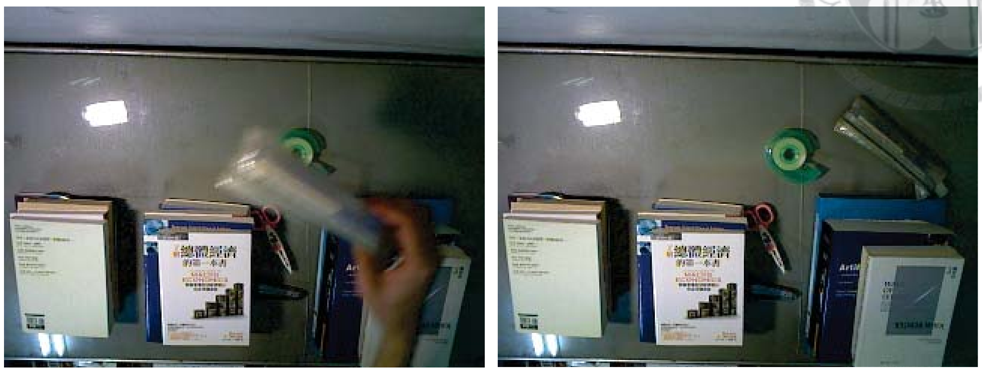


Figure 1.2. (a) An example of stable scene while an example of dynamic scene is showed in (b).

CHAPTER 2

Related Work

THERE are a number of approaches to tackle the problem of finding object in a physical world. In the recent research, the possible ways to search for real world entity such as objects or people has been surveyed in (Romer et al., 2010) and (Koyuncu and Yang, 2010). One of the core ideas of these researches is to make the search for real-world entities equally feasible as the search for files such as Web pages, images and videos in the virtual space. The challenges for object finding system are (1) detecting an object and retrieving its location, and (2) representing the information in a manner the user can easily understand. In the early research, most popular approaches equip objects with sensors or visual tags to make them searchable. The sensor used in previous works include the active sensors such as position transmitters or the passive sensors such as RFID tags. For example, (Pederson, 2001) let a user be equipped with a wearable RFID reader that sends the location of an object to a system whenever a tagged object is within the sensor range of reader. The main disadvantage of this approach is that the system is too expensive for practical use. Besides, it is too intrusive for user and not scalable as each object has to be equipped with a sensor or tag. On the other hand, (Ljungstrand et al., 2000) and (Butz et al., 2004) utilized a camera to visually detect the markers on the target objects which are attached with the AR-markers and barcodes respectively. The location of an object is inferred and calculated through analysis of the color image. In this case, tags are relatively cheaper compared to electronic sensors as it is possible to print them out from a common printer. The defects include that there must be free line-of-sight between the camera and the tags on the objects and that the tags should be fairly large. All of these approaches mentioned above need much preparation in advanced. It is so consuming and inconvenient for people to use.

Another possible solution to find objects in physical world is to apply the object recognition techniques to find and localize objects in cluttered scenes. With the use of powerful RGB-D cameras such as Microsoft Kinect and ASUS Xtion, a number of works have demonstrated that it is feasible to accomplish object recognition in indoor scenes (Lai and Fox, 2010) (Lai et al., 2011a) (Lai et al., 2011b) (Attamimi et al., 2010). Training is needed in these approaches to generate models of the objects and testing is then performed using RGB-D scans. Although the number of trained objects can be large, e.g. 300 objects, 51 categories in (Lai and Fox, 2010), it is still not guaranteed that a target on the desk is learned in advance. Asking a user to collect and annotate data for training should not be a preferred option. In addition, one challenge to recognize and localize an object on a messy desk is occlusion. For instance, it would be infeasible to localize a paper under a book given the current view of the desk. Moreover, the number of objects around us is not very much in daily life so using a lot of data seems to be unnecessary.

Given that objects usually are moved by human, to detect and track hands is another feasible solution to understand where the objects are. The hand is a complex object. It has 14 joints which means that the number of possible configurations, make its detection a challenged task. In the early research, many researchers have constrained the problem through the use of coloured gloves (Starner and Pentland, 1997),fixed or known backgrounds (Wren et al., 1997), limited movement or markers (Lockton and Fitzgibbon, 2002). To discard such constraints, an intuitive method is color-based hand detection (Zhu et al., 2000) which relies on the fact that human skin is relatively uniform. However, this is not a reliable approach since hands tend to be confused with other skin-colored objects and there are issues of insufficient lighting conditions. With the depth sensors becomes ubiquitous, hands can be effectively detected by Kinect sensor (Shotton et al., 2013) and prior work on articulated hand pose estimation for RGB-D sensors (Qian et al.) (Oikonomidis et al., 2011) (Tang et al., 2013). It performs well for detecting hands though they all need complicated and annoying pre- training process. One problem of these methods is that it can not robustly handle the situation that hands interact with objects since the occlusion and the variety of manipulation. However, the objects must be put on somewhere after the interaction with human. Our approach utilizes this idea to achieved objects modeling and searching by focusing on stable scenes. So we can avoid doing object recognition or analyzing complicated human-object interaction in dynamic scenes. The experiment can show that the robustness and adaptability for various objects of our method.

CHAPTER 3

Approach

THIS chapter will discuss the detail of the proposed desk object modeling method. For achieving usability, it is critical to show all the occluded parts of the scene as illustrated in Fig. 3.1. Once a new object is put on the desk, its model will be extracted through proposed method. Even if it is occluded by other objects in the subsequent frames, the object will be showed by the model extracted before.

In order to provide a straightforward representation of objects to users, the model of objects should be clean without the influence of hands. As mentioned before, we could extract the models of object correctly in the stable scene. Object modeling is processed in the stable scenes while the tracking is activated to find out the moving objects and to record the trajectories of moving objects in the dynamic scene. For the purpose of determining the scene is stable or not, the concept of stability of scene is proposed to as a criterion. Through analyzing the difference of two consecutive RGB-D frames over a period of time, we can

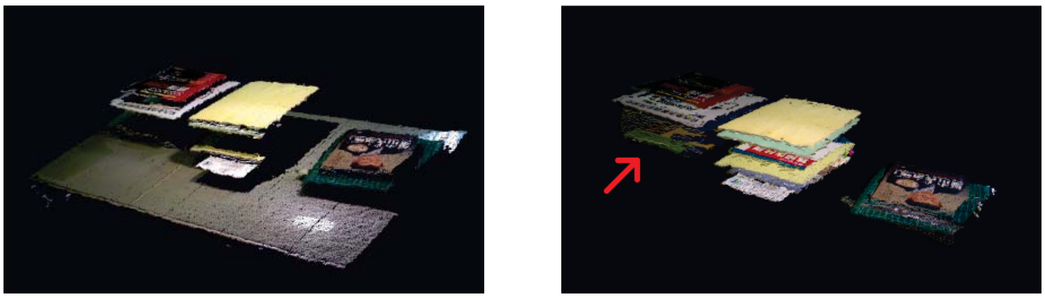

(a) Raw point cloud that many objects are occluded. (b) The scene reconstructed by proposed method.

Figure 3.1. The occluded parts of the scene are indicated by the red arrow.

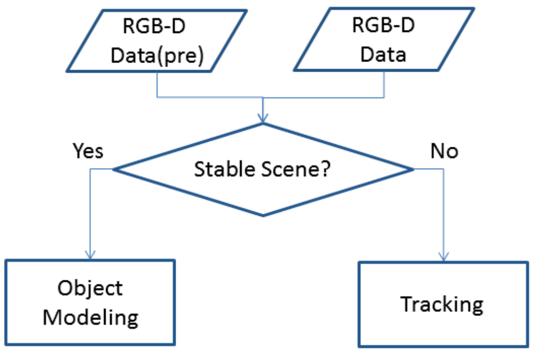


Figure 3.2. System overview.

determine a scene is stable or not. In the stable scene, the volume and appearance difference could be extracted by comparing the current scene with each object models. If the difference is beyond a range which is set as 0.01m in this thesis, this point is considered to be a part of changed area. Then, the clustering method is used to group changed parts into several clusters which represents an object independently. In the dynamic scene, moving objects are identified by tracking and their trajectories are recored respectively. Combined with the trajectories from the tracking module, if there is a trajectory ends around a cluster, a new object will be modeled with the cluster in stable scene and the spatial relation will be updated through the calculation of physical support. Meanwhile, objects which has been moved could be updated to the new place with the end position of its trajectory and the inference of spatial relation.

In this section, we introduce the framework of our system as shown in Figure 3.2. At the beginning, two consecutive RGB-D frames are compared to determine that whether the scene is stable or not. If the scene is unstable, the tracking module will be activated and all the valid trajectories will be recorded. On the other hand, if the scene is stable, the system will switch to the object modeling stage and the scene will be updated. In the following sections, we will go through the details of each part.

3.1. The Scene Stability

To generate the clean model of each object, hands or other manipulators like robot arms attached on objects must be discarded. Removing these redundant parts is complicated and daunting when skin detection or hand detection are integrated. Through observing the way that people interact with objects in real world, we found that there must be some stable moment in which people do not affect the environment. For example, after we take away a book from the bookshelf, the bookshelf would be stable for a while until someone interacts with it again. We found that objects in the stable scene is clean enough to be extracted as the model. So if we can detect the stable scene, the model of objects could be extracted properly in the stable scene. Based on this discussions, we proposed a new idea called ”scene stability” to determine the stable scenes.

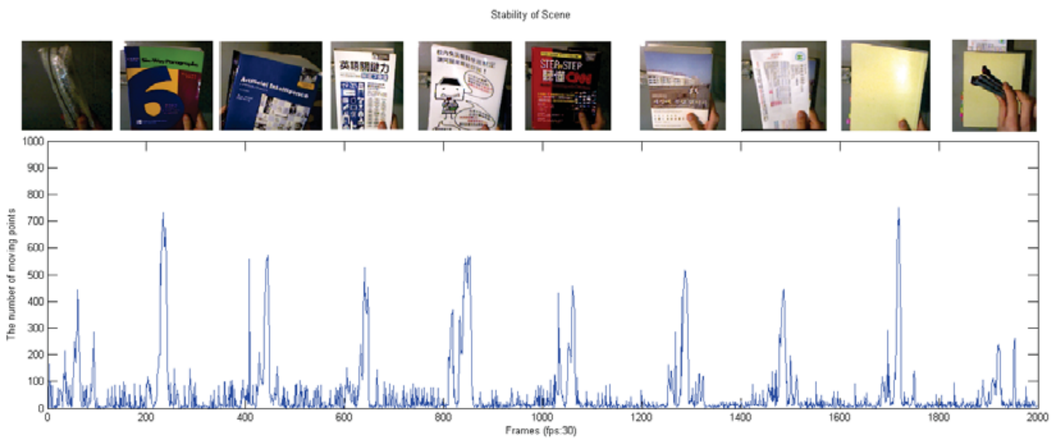


Figure 3.3. This is one of chart about stability of scene from our experiments, in which x-axis represents frames and Y-axis represents the number of moving points. It could be found that a book usually has a high peak; while a small accessory like stapler(the one from far right in the chart) has a low peak.

The scene stability is a metric to measure the variation of a scene in a period of time. A scene is stable means that objects in the scene are not moved. Once the stable scenes could be identified, we can focus on these stable scenes to extract the new objects without influence of hands or other manipulators. The depth difference between two consecutive point clouds is computed to determine the scene stability. The points in previous point cloud are set as reference to find a corresponding point in the current point cloud which has the minimum distance from it. To take the sensor noise into account, the distance of the point pairs should be under 0.01 m according to our observation if that object is static. Otherwise, the point pairs should be a part of a moving object.

Instead of analyzing the dynamic scene, the stable periods/states of the monitored scene are detected and used for accomplishing the task. In a stable state, the objects on the desk are stationary where there should be no external force. The positions, shapes and appearances of the objects should be stationary enough for modeling. The stability of the scene is computed based on the assumption that there is a short time period between every move, where each apparent peak represents that there is an object moved by the user. The hard threshold is set as 100 moving points based on the analysis as illustrated in Fig.3.3. If the moving points in a sequence of frames are less than threshold, it is considered in stable state. Practically, we use time window of 2 sec to make sure that the scene is really stable instead of only considering a single frame to enhance the robustness.

3.2. Objects Modeling in Stable Scene

Based on the discussion above, the object modeling should be processed in the stable scene to get a clean model without the occlusion. The task of object modeling includes two things, one is to model the new object and another is to update the objects which have been moved. Once a new stable scene is detected, the changed parts could be found by comparing the differences in volume and appearance between the current stable scene and all the objects which have been modeled. If the depth information is same as previous stable state at the same location, we can not determine whether there changes or not. For instance, if an envelope is put on a book, the depth information would not change because the resolution of the sensor could not afford to detect. Therefore, appearance should be considered to extract these kind of objects whose volume is indistinguishable. After all the changed parts are found, they will be clustered to several clusters. A cluster which is indicated by the end position of a trajectory without associating to existing object will be the model of new object. On the other hand, an object which has been moved is updated by translating its model to the new place which is indicates by an associated trajectory. For both cases, the spatial relation between an object and its supporter will be updated according to the method in 3.3.3.

The clustering module for handle noise and group points into meaningful object is introduced. After we got the volumetric difference and appearance difference, we merge

(a) The color image.

(b) The depth image.

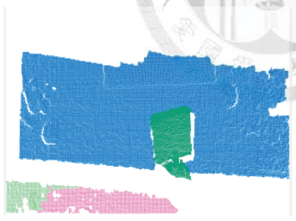

(c) The result of clustering.

Figure 3.4. The result of clustering which each object is marker with different color.

them into one point cloud and then use the clustering method to split it into many individual objects as illustrated in Fig. 3.4. The metric of clustering is Euclidean distance which we set 0.01 meter in this work. Since the RGB-D camera measurements often contain noises around the object boarders/boundaries, even two consecutive stable frames may have depth differences around the object boundaries. Thus the result from comparing two stable scenes does not only contain the points of new object but also noises. To cope with noise problem, we cluster points in the same group if they are close enough in Euclidian distance (Rusu, 2010). In this work, the value for clustering is set to 0.01 meter. So there should be several groups $c_1, c_2 \ldots c_m$. According to our observation, the result from noise is not centralized in a region but fairly well distributed. So the size of cluster from noise is relatively smaller than real moving objects. We can discard the noise parts by checking its size effectively.

Algorithm 1 Clustering Algorithm.

```
Require: Given an point cloud $p_1, p_2 \ldots p_n \in P$;
 1: Create a Kd-tree for the input point cloud $P$;
 2: Create an empty list of clusters $C$ and a queue of the points which need to be checked
    $Q$;
 3: for $i = 1, \ldots, N$ do
 4:     Add $p_i$ to the $Q$;
 5:     for $p_i \in Q$ do
 6:         Search for the set $K$ of point neighbors of $p_i$ in a sphere with radius r $\prec d_{th}$;
 7:         For every neighbor point of $K$, check if the points has already been processed , and
            if not add it to $Q$;
 8:     end for
 9:     If all points in $Q$ has been processed , add $Q$ to the list of cluster $C$, and reset $Q$ to an
        empty list;
10: end for
```

3.2.1. Volumetric difference

Most objects have obvious volumes that can be detected easily by ASUS Xtion Pro sensor. The segmentation method for this kind of objects is intuitive, whose main concept is comparing current 3D stable scene with all stored models of objects by their spatial difference. Practically, the whole first-come-in frame would be set as the first model. The current stable scene is retrieved from the sensor, while the reference scene is represented by several object models which has been modeled before. The segments of object in point cloud format represent the static objects in current scene, which are taken as a reference for the following segmentation process. In other words, all the objects which have been modeled compose the previous stable scene. Every time a stable scene is detected, the detected scene would be compared with every existing object model sequentially. If a point in detected scene is close to a point in one of models, it will be considered a part of existing static objects and discarded. Every point in detected scene will be compared with every model iteratively to found the volumetric difference. The reason that not comparing two stable scene but comparing scene with models is that if the current scene has been moved so that the previous occluded parts could be disclosed, and the occluded parts would be taken as difference since the previous scene don't cover this parts. By this scene-to-model comparing, we can make sure that every found difference is real changed parts.

3.2.2. Appearance difference

As mentioned before, our proposed method depends on volumetric difference between two stable scene(the reference scene is represented by several object models). The main limitation is that if the volumetric difference is not significant(e.g., an envelope on a book), it can not be distinguished only by comparing volume. So to cope with this problem, appearance information should be taken into account. To enhance robustness to lighting changes, we use HSV color space instead of original RGB space. Because HSV separate image intensity and color information, the hue component which represents only color information in HSV space is used as the metric to comparing appearance.

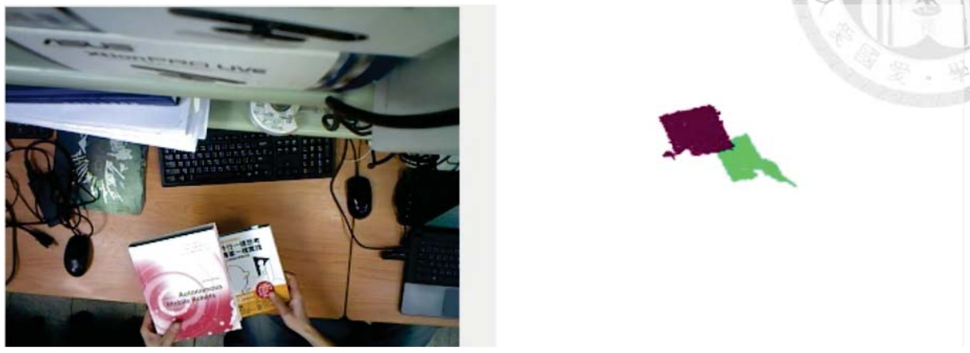
The detail of comparing appearance is described below. The current stable scene is split into several clusters by clustering method mentioned before. The ICP algorithm would be adopted to estimate the translation between these clusters and existing objects. The existing objects which has the max Fitness score is considered the same as that cluster.

Then, if the difference in hue component for each point pair exceeds a threshold, the point is thought a part of appearance differences. Finally, the result from appearance comparison is filtered by clustering method, so the points from object with undistinguished volume could be grouped into a meaningful object.

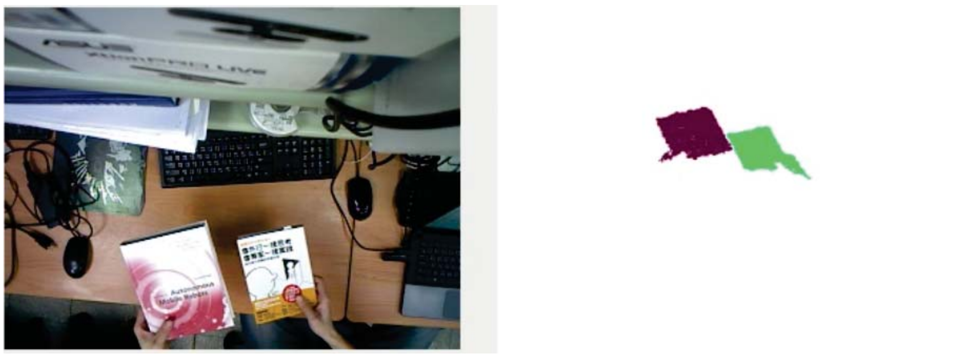
3.3. Objects Tracking in Dynamic Scene

With the concept of stable scene, most objects can be extracted from cluttered scene easily and efficiently if the volume or appearance of them could be differentiated. However, there are some objects which are hard to be distinguished only by comparing two stable scene. For such object, the difference is little in stable scene but the motion is obvious while people are moving it. So we should use this motion cue to help us deal with such objects. Besides, we not only handle putting a new object on the desk but also moving the objects which are already on the desk. Using motion cue is also important and helpful for understanding which object is moved and where it is moved. Therefore, the tracking module is activated in dynamic scene to trace locations of each moving object in scene. The goal is to find the correct moving object and its trajectory. The end position of trajectory indicates the destination of an object. This information is used at the object modeling stage to model the objects with proper position. As mentioned before, it is hard to differentiate the objects without obvious volume. The spatial relation is used to help find the correct data association when an object is moved. Assume that the actions which could collapse the structure of objects are not considered, an object could not be moved until the object above it is moved. In other words, the object on top has high priority to be moved. We can find the correct moving object by checking whether there is another object is on it.

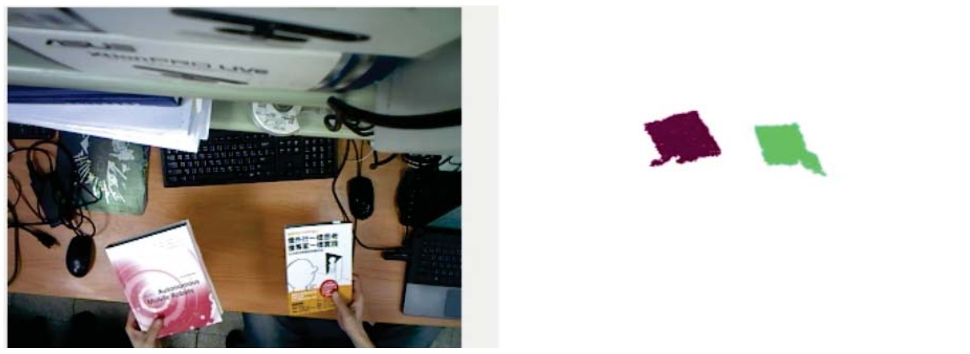
In this work, the tracking algorithm in object level is adopted to establish trajectories for moving objects. Fig 3.5 shows one of the tracking example which different moving objects are represented by different colors. After the raw data is transformed to object level by clustering, we can use the centroid of objects as representative. The tracking is solved by Global Nearest Neighbor Method (GNN). The GNN method is thought as assignment problem which aims to find a minimum distance matching in a bipartite graph that has two disjoint sets: objects in the previous frame and observations in the current frame. The classic Hungarian algorithm is adopted to get the optimal solution here. The effectiveness of this method is demonstrated in the later experiment part.



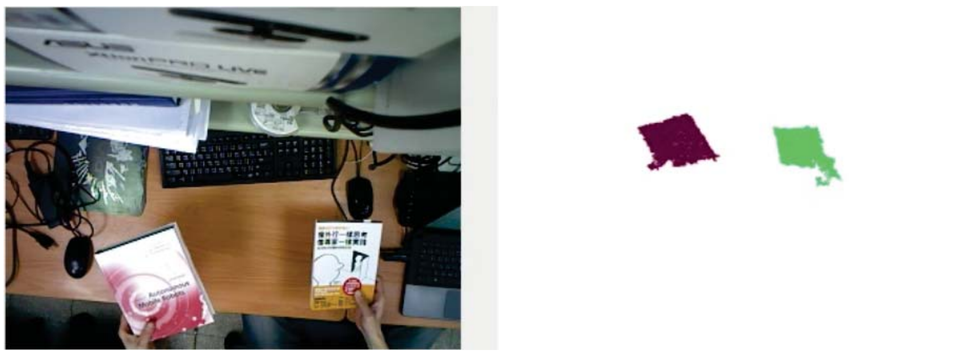
(a)



(b)



(c)



(d)

Figure 3.5. A sequence of tracking result is illustrated from (a) to (d). The left image is color image while the tracking result is showed on the right.

3.3.1. Filtering False Matching

Since the tracking is doing in the object level, the representative of objects in computation is a critical issue here. An intuitive choice is the centroid of object. However, there is a problem for using the centroid of object, which the centroid of object is not only changed by motion but also the variation of size. According to this attribute, we should differentiate whether a change of centroid is really caused by motion. In order to solve this problem, we observe the data and conclude the causes of the variation of object size. First one is the limitation of view field. For instance, an observation would be broken when it is sensed by sensor first time and become complete when it moving to the ideal sensing range. The variation of size caused by this reason is so abrupt that we can design a new cost function which involves the size constraint to solve this problem. The second reason is caused by occlusion. There is a strong possibility that the static objects will be occluded by moving object in the desk environment. The size of object will be changed caused by occlusion so it may be mistaken for moving objects. The variation of size owing to occlusion is too smooth to solve by designed cost function. To avoid the false detection for static object as moving object, the static objects will be checked before doing the tracking module. All points in each observation will find a closest point from all existing objects as its corresponding point. If a point is close to its corresponding point, which means that the Euclidean distance between this point pair is within a threshold(0.01 meter in this work), it is regarded as a static point. An observation whose proportion of static points is over 70 % among all points will be considered a static object and will not be involved in tracking process.

3.3.2. Global Nearest Neighbor Method

The Global Nearest Neighbor method (GNN) is a simple but effective method for data association. This method handles the input data in a sequential way and only maintains a single hypothesis. When new data are coming, the goal is to find the most likely assignment of observations to existing tracks and to the new tracks created in that moment. The definition of a solution for observation-to-track association is the critical issue in GNN method. It is typically addressed through the solution of an assignment matrix. An example of the assignment matrix is illustrated in Table 3.1. Note that it not only includes the assignment of observations to tracks but also assignment of observations to the new tracks.

The value in the matrix is computed by cost function, while the prohibited assignments, which is failed the gate test, are denoted by X.

Table 3.1. An Example of Assignment Matrix

	Observations		
Tracks	O1	O2	O3
T1	15	X	X
T2	3	8	X
New track 1	X	X	O

The trivial cost function only include the distance between observations and tracks. As the hands are involved in the moving process, an unwilling situation may happen, which a detection of hand holding object is linked to an empty hand in the previous frame. In this case, the position at which the hand showed at the first time is regarded as the start position of the object. To solve this problem, the cost function should not only include the distance but also the difference of the size between observations and tracks. The proposed cost function in (3.1) contains two terms. The term d_{ij} represents the distance while the term S_{ij} stands for differences of sizes. The idea is that the size of observations between two consecutive frames which are regarded as the same object must be almost equal. One of the results which separates the empty hand and hand holding objects into different trajectories is showed in Fig. 3.6.

$$C_{ij} = \alpha \times d_{ij} + \beta \times S_{ij} \tag{3.1}$$

Then, the goal is to find the solution which achieve the maximum number of possible assignments and minimizes the total cost, given the maximum number of assignments. The assignment problem can be represented as follows:

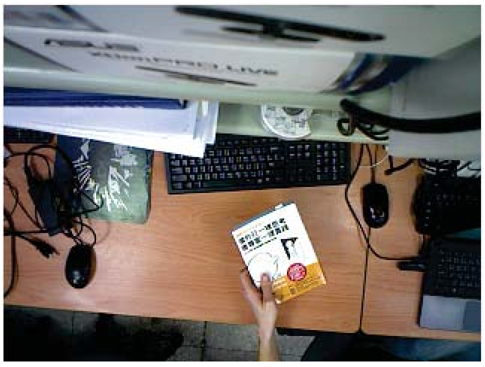
Given the matrix of elements w_{ij}, find $X = \{x_{ij}\}$ such that

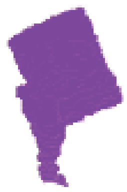
$$C = \sum_{i=1}^{n} \sum_{J=1}^{n} a_{ij} x_{ij} \tag{3.2}$$

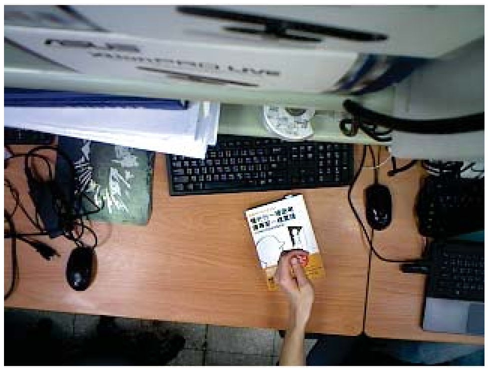
which is minimized subject to:

$$\sum_{i} x_{ij} = 1, \forall j \tag{3.3}$$

$$\sum_{j} x_{ij} = 1, \forall i \tag{3.4}$$


(a) The color image before the book is put down.


(b) The tracking result before the book is put down.


(c) The color image after the book is put down.

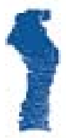

(d) The tracking result after the book is put down.

Figure 3.6. An example of the case which successfully splits the empty hand and hand with object into different tracks.

An optimal solution allows the x_{ij} to be either 0 or 1. In the tracking, the x_{ij} elements are the observations-to-tracks pair to be found while the w_{ij} elements stand for the cost of every observation-to-track pair. The total cost is that which is to be minimized. One possible solution can be found by trivial enumeration but there are other more efficient methods, such as Hungarian algorithm which was introduced by Harold Kuhn in 1955 (Kuhn, 1955). The Hungarian algorithm is a combinatorial optimization algorithm in polynomial time. It has been proved that the solution is optimal. The procedure of explanation for this algorithm is as follows:

Step1: Find out the minimum element in every row and subtract this minimum value from all the elements in the respective row.

Step2: Like Step1, subtract the column minimum from all the elements in the respective column.

Step3: Cover the zero elements with the minimum number of lines.

Step4: Add the minimum uncovered element to every covered element. If an element is covered twice, add the minimum element to it twice.

Step5: Subtract the minimum element from every element in the matrix.

Step6: Cover the zero elements with the minimum number of lines. If the number of lines is not equal to number of rows, go to step 4.

Step7: Choose the zero element where each row or column has only one selected as the assignment.

Through solving the assignment problem in each frame, we can get an optimal solution for observation-to-track matching. The observation which can not be assigned to existing tracks will be regarded as a new track and used in the subsequent frames. And the existing tracks which is not matched with any observation will be regarded as missing. In our system, if a track is missing continuously for 2 seconds, it will be removed from the tracking system.

3.3.3. Establish Spatial Relation

As we stated before, People could not only put a new object on the desk but also move an object which is already on the desk. If we take this situation into account, there are more issues should be concerned. One of the issues is that how to know that which objects have been moved. An intuitive solution is using object recognition technique to identify each object and update its location in database but object recognition needs priori preparation and occlusion may cause mistake. Besides, objects with similar appearance are hard to be distinguished by recognition technique. In our scenario, the desk environment will be kept under surveillance so each location of object can be tracking continuously by 3D multiple object tracking. Nevertheless, as the data retrieved by RGB-D camera presents in surface, the position in z-direction tends to be confused easily. An evident example is that if we put a sheet of paper above another paper, it is hard to differentiate them with depth information. So indistinguishable depth data could cause the wrong data association for multiple objects tracking. The challenge here is how to differentiate the objects with insufficient depth data. To solve this problem, we assume that object could not be drawn

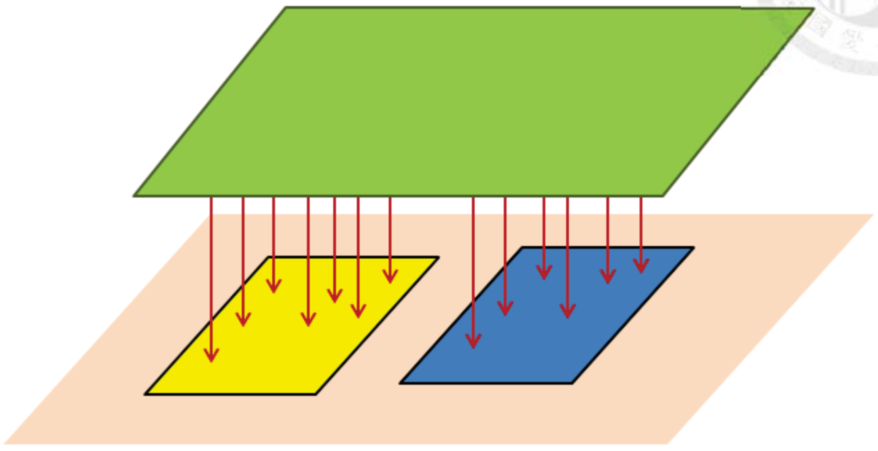


Figure 3.7. The figure is to show how to find the objects under the green object.

directly if there are something above it, which means that people must move the object on top first. Based on this assumption, the relation between objects like object A is above object B has to be maintained over time. In stead of maintaining this relation during tracking, a more efficient method is establish this relation in the stable scene when the location of object is confirmed.

There is a simple example to explain that how to establish the spatial relation between objects in Fig. 3.7. The green object is processed to update its location as well as the relation between itself and others. The new location is updated according to the result from tracking. Once the new location of green object is confirmed, we have to find the objects below it to establish the spatial relation for green object. Several points of green object are sampled. Each point searches its closest point along the normal of ground toward the bottom as illustrated in Fig 3.7 where the red arrow indicates the searching direction. Then the objects which involves these found points are regarded as the objects below green object. In this example, the yellow and the blue object are the objects which are found below the green object while the orange object represents desk which could not be moved. In the beginning of an object being moved, the possible object in the database is first found according to its location. Then if there is nothing above the found object, it will be initialized as a new tracker. On the other hand, based on our previous assumption, we will look up upward until the top object and it will be the new tracker. The spatial relation is proved to be effective for establishing correct data association in the experiment section.

3.4. Desk Object Search

Compared to the standard search problems, our search problem is easy in terms of the number of objects to search. In this work, we focus on designing a user graphical user interface in which a user can browse all modeled objects and query object's spatial and appearance information. The interface of our desk object search system is as illustrated in Fig. 3.8. There are three basic parts in our designed interface. The models of the objects on the desk are shown in which the current configuration of the desk is displayed. The object list is represented using the photos of the objects at the bottom. The items in the object list are allocated dynamically when a new object enters the scene. The top left part shows the RGB image which Xtion sensor is capturing currently while the current 3D model of the scene visualize at the top right. Once the scene is stable, the new object is segmented, modeled and showed on the top right part of the window. For querying, users just need to click the "Find" button of the object they want to search. The 3D model of the selected object is highlighted with yellow color, which provides an intuitive and noticeable display. Furthermore, the visualization of the models of the objects are done using a 3D point cloud viewer based on the Point cloud library (PCL), which could be zoomed in, zoomed out and change the perspective.

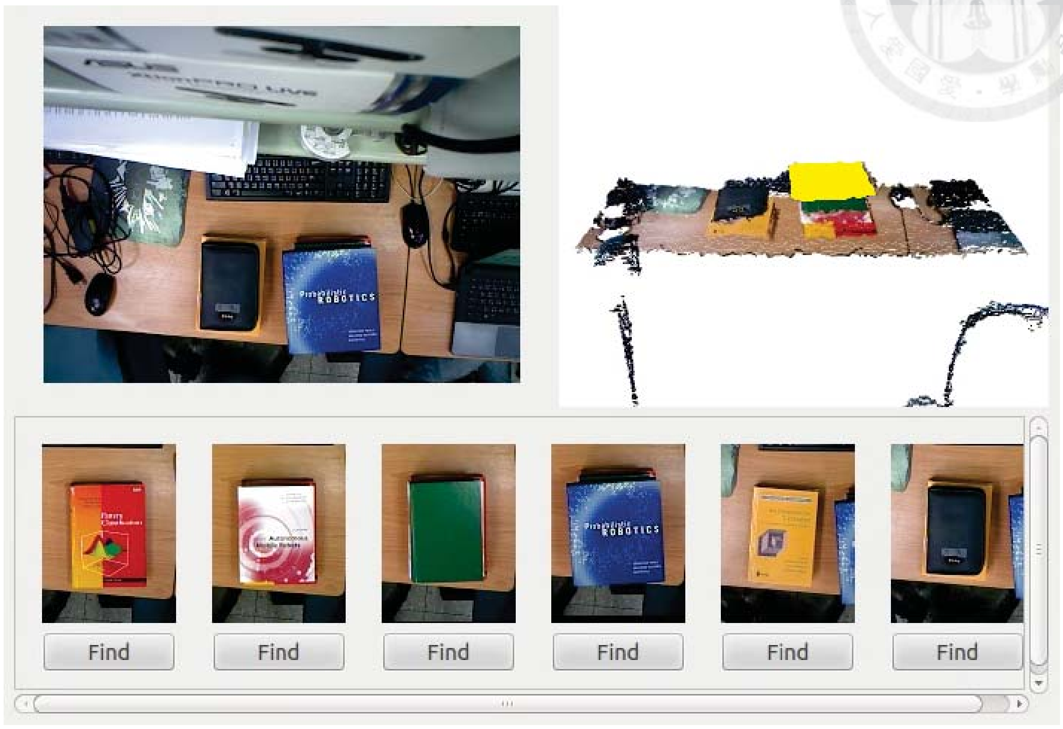


Figure 3.8. The interface of the desk object search system. A user can click the "Find" button with the photos at the bottom which corresponds to an object on the desk. The top left is the RGB image of current scene; while the top right is the visualizer that shows the 3D models in which the user's query object is indicated with yellow color.

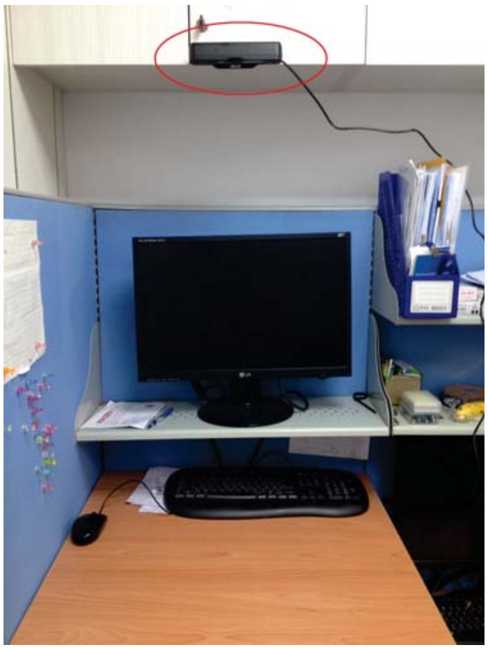
CHAPTER 4

Experiments

IN this section, the experiment setting, the test scenarios and the results of the proposed approach are described. The system is run real time which could be demonstrated as a feasible application. A variety of the desk objects such as books, envelopes, pencil boxes,scissors, and staplers are tested and many different interaction way with objects are tested in our experiments too. Both the successful and failure cases are analyzed and explained objectively and clearly.

4.1. Experiment Setting

An Asus Xtion PRO LIVE camera is placed on the bookshelf to monitor the desk as illustrated in Fig. 4.1(a). The captured frame peR second is around 30. The resolution of a RGB-D image is 320 240. The detected depth resolution is 100 um. The horizontal and vertical field of view for converting a RGB-D image to a point cloud are 1.02259993 and 0.79661566 respectively. The distance between the RGB-D camera and desk is about 0.8 meter. As the closest sensing range of Xtion is around 0.5 meter, the upper bound of the depth information is about 0.3 meter above the desk. The input data of the proposed system are a series of RGB-D images ordered by time, and all the RGB-D images are transformed to 3D point clouds with RGB. The different objects with various sizes and thicknesses are used to test the capability of our approach. In addition, the position of the objects are decided arbitrary while the occlusion is created intentionally for demonstrating the feasibility of the proposed method.


(a) Setting of experiment.

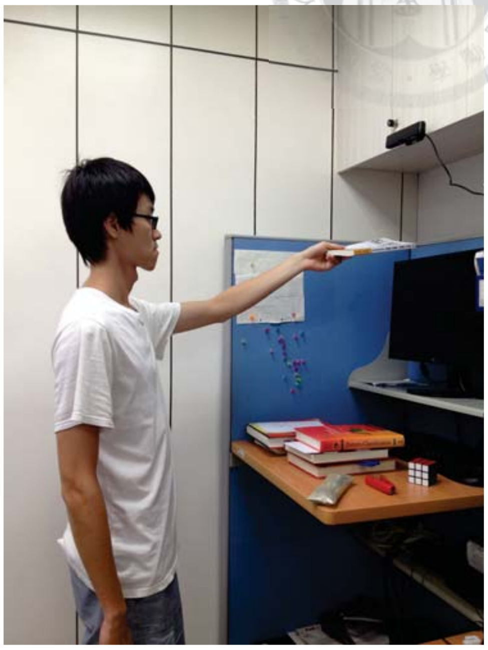

(b) Scenario of use.

Figure 4.1

4.2. Testing Scenario

In our test scenario, the scene which is captured when the system turns on is treated as the background which means that all objects in the scene will be regarded as one object. Then users can put a new object into the sensing range from arbitrary direction as usual way like Fig. 4.1(b). After the object is placed in the scene, its photo will be taken when scene is stable. The photo of the object is used as the icon in our desk object search user interface. There are two limitation for our system. First is that the motion should be slower than 5 cm per second because the hardware limitation when we applied it real time. Second, both hand can manipulate objects simultaneously but a hand is restricted to take one thing at a time. Because when holding multiple objects, there are too much occlusion to distinguish how many objects are moved. It is a difficult case that we could not handle well yet. Furthermore, moving an object on the desk to a new position is allowed in our system.

4.3. Result and Analysis

One of our testing dataset is shown in Fig. 4.2. The whole process from the first object appearing to the last object put on the desk is depicted. The figure is shown from top to down and left to right, each column includes a photo of the new-coming object below and the corresponding model when it is put on the desk.

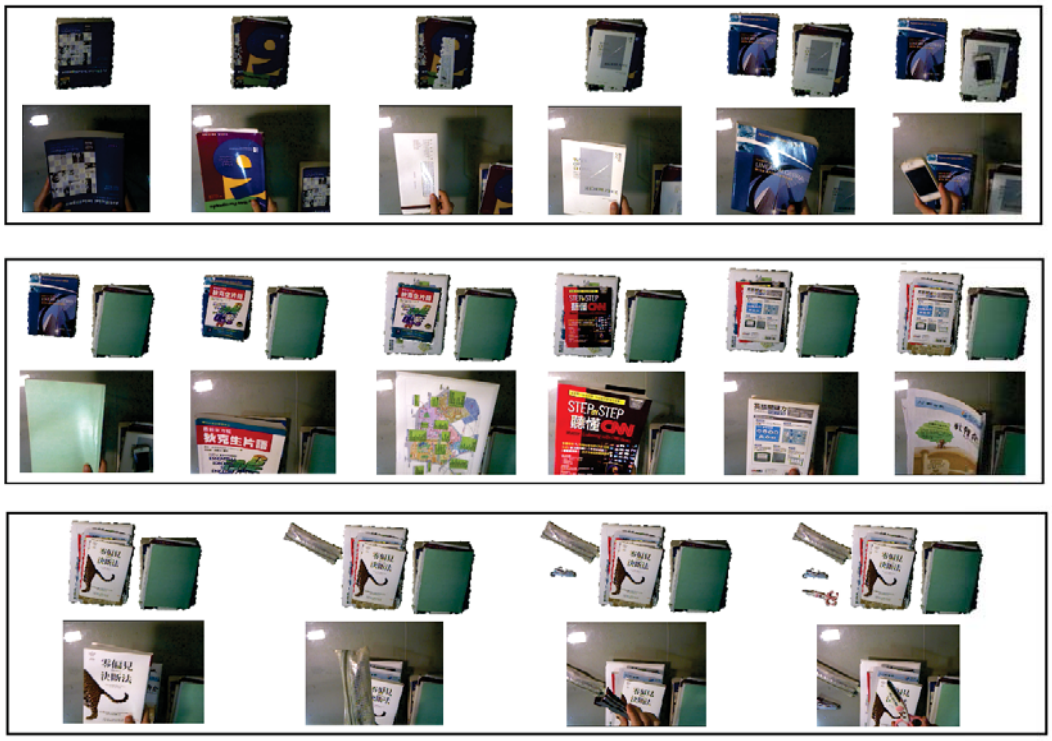
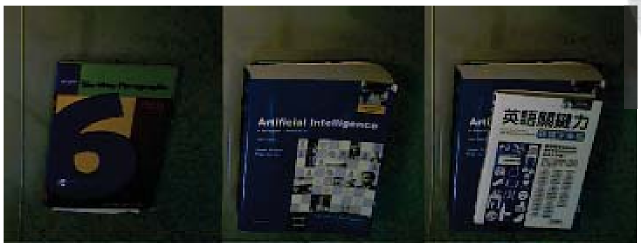


Figure 4.2. One of our testing data set includes 16 different objects. The order of objects coming into scene is from top to down and left to right. Each column represents an object with its corresponding model. The third of the first row is the case of a letter/envelope; while the rightmost of the third row is the case of a scissor.

The obviously simple cases are rigid objects with large volumes. The common examples are books which can be simply differentiated using depth measurements. Background subtraction achieves excellent segmentation as depicted in Fig. 4.3. The dense point cloud with RGB provides users a good connection between the real world and the virtual models. Even though the books are stacked up, the segmentation of each book is clear and without affection by each other.


(a) RGB images of objects

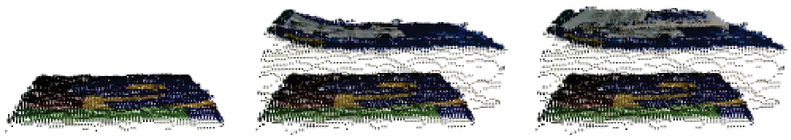

(b) The process of stacking books from left to right

Figure 4.3. The RGB images of the book stack in different times are shown in (a) from left to right. The corresponding models are shown in (b). The occlusion part is still clear in our models.

In the following two cases where the segmentation performance degenerates significantly due to the object without evident volume or thickness are discussed. The volumes of common accessories are not as large as books. Their appearances and shapes are usually irregular. An example of a scissor is shown on the far right of the third row in Fig. 4.2. The challenge here is that the depth information of the small object is not as obvious as books for splitting it from background. However, the proposed method still provides a satisfying segmentation result. In spite of the fraction of boarder, the whole appearance is distinct enough for identification. It shows that our approach is not only for the objects with remarkable volume but also for small objects. The experiment constructed the thin object case as illustrated at third one from left of the first row in Fig. 4.2. There are a letter/envelope moved by a user and one stack of books above the desk. When the letter was put on the books, the depth information is not sufficient to segment it. So the background subtraction in 3D could not properly split them with depths. However, combined with the concept of appearance difference ,the issue is solved using the approach described in the previous section.

The applicability for different size object of proposed method has been show above. Beside the experiment to show the utilities of different size of object, multiple interaction

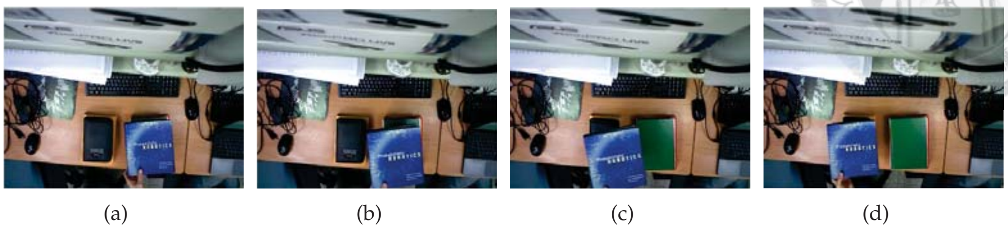


Figure 4.4. A sequential color images of an example which an object is moved at a time.

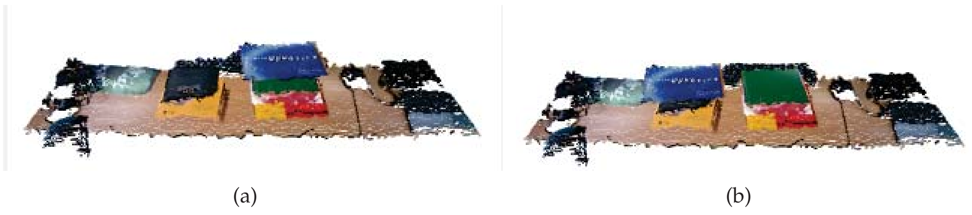


Figure 4.5. The result of experiment for moving an object. (a) is the whole scene before moving, while the outcome after moving is showed in (b).

manners with objects are also discussed in the following paragraph. The proposed method can not only deal with putting an object on the desk but also moving the object on the desk to a new position. As showed in Fig 4.4, it is an example that moving an object at a time. The blue book on the top of stack is moved from the right side to the left side and is put on the black cardcase finally. Each point cloud of object is segmented when they are first place on the desk. Fig. 4.5(a) shows the 3D point cloud before the blue book is moved while the 3D point cloud before moving is showed in Fig. 4.5(b). The point cloud of blue book is translated above the black cardcase precisely from the top of right stack after the whole moving course. The tracking module traces the trajectory accurately and excludes the influence of hand. This is a simple case that moving only an object without any occlusion on target object, which is one of the common actions in our daily life. It proves that the proposed method can handle the moving case correctly.

Next, we will show some more challenged cases like moving multiple objects at a time. It is more difficult because the serious occlusion may happened like fig. 4.8(c) and the top and down relation is more complicated as multiple objects should be considered simultaneously. Based our observation, there are two common actions of moving multiple objects at a time happened in the desk environment. The first case is that put an object to

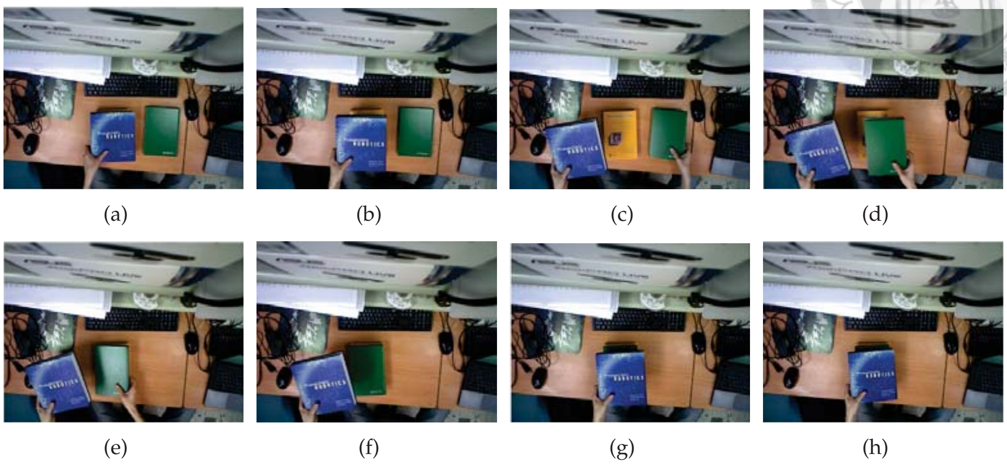


Figure 4.6. A sequential color images of an example which an object is put to the middle of a stack.

the middle of a stack. As illustrate in Fig. 4.6, the goal is to put the green book on the right side to the middle of stack on the left side. To achieve this purpose, the blue book must be picked up before the green book is placed to its destination. The final and the original state are showed in Fig. 4.7 respectively. The tracking module will trace all the moving objects and record their trajectory while the objects are being moved. When the scene is stable, which objects are moved is confirmed according the information of trajectory and the system could understand that there is nothing new to be put in the scene although there are differences in volume and appearance. We can find that the green book is under the blue book correctly after processed by our system and the number of objects are the same as original state.

Another case of moving multiple object is taking out an object which is under another object. The green book is taken out from the bottom of the blue book to the right hand side as illustrated in Fig. 4.8. The proposed tracking combined with the concept of top down relation could effectively help find the correct target object under the severe occluded condition. The 3D point clouds before and after moving are showed in 4.8 respectively. It demonstrates that the proposed method can handle the occlusion by the spatial cue.

One of the challenged case which is mentioned before is about objects which have less information in depth such as paper. The difficulty is that if two paper is placed together(one is above another), the pure information from z-direction is not sufficient for

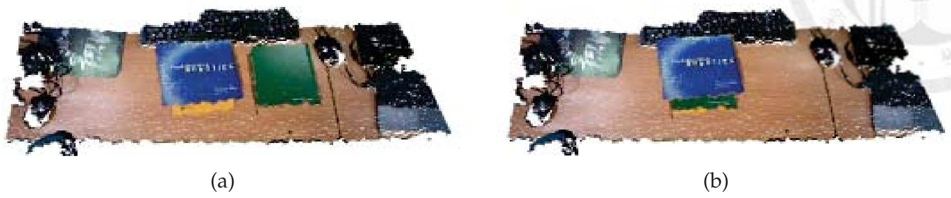


Figure 4.7. The result of experiment for moving a book to the middle of a stack. (a) is the scene before moving, while the outcome after moving is showed in (b).

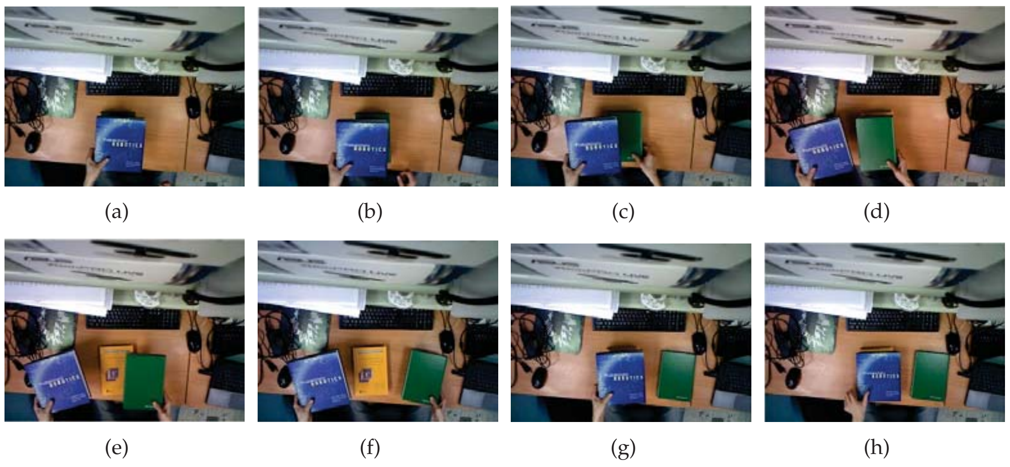


Figure 4.8. A sequential color images of an example which an object is taken out from the bottom of another object.

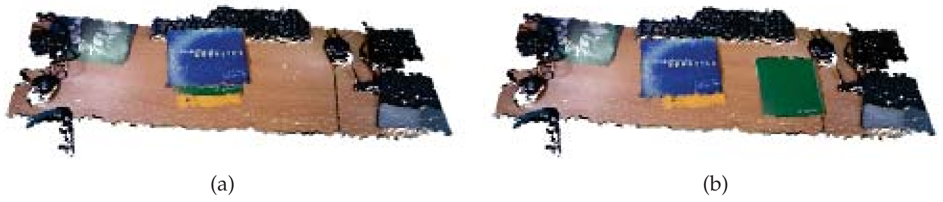


Figure 4.9. The result of experiment for taking out an object from the bottom of another object. (a) is the scene before moving, while the outcome after moving is showed in (b).

differentiating which one is moved. An experimental result is showed in Fig. 4.10(b). In the original state, there are two sheets of paper are placed together and their 3D point clouds are segmented when they are first place on the desk as illustrated in Fig. 4.10(a). In this experiment, the paper at the bottom is moved to the left hand side as illustrate in

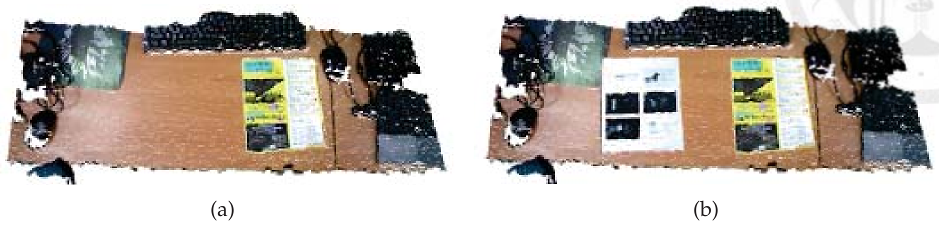


Figure 4.10. The result of experiment about paper. (a) is the scene before moving, while the outcome after moving is showed in (b).

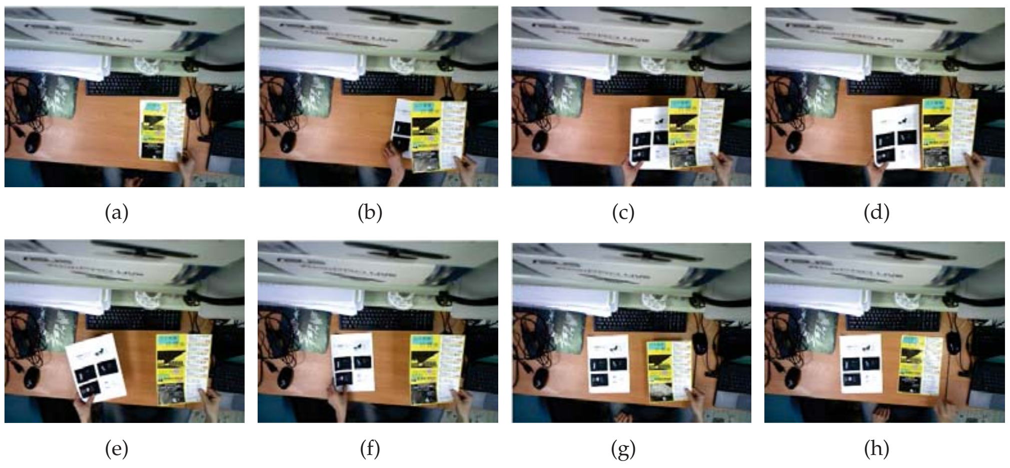
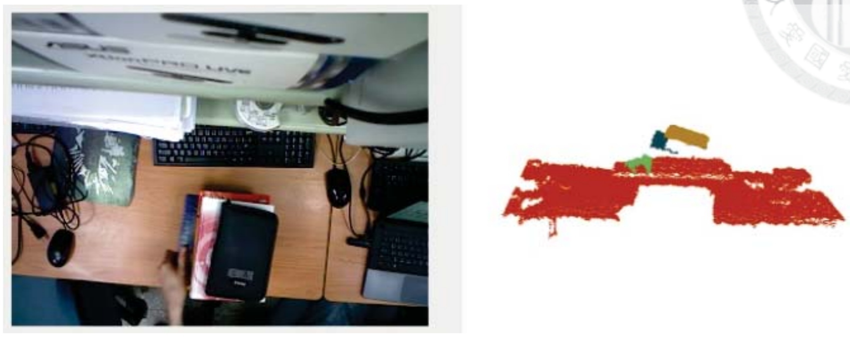
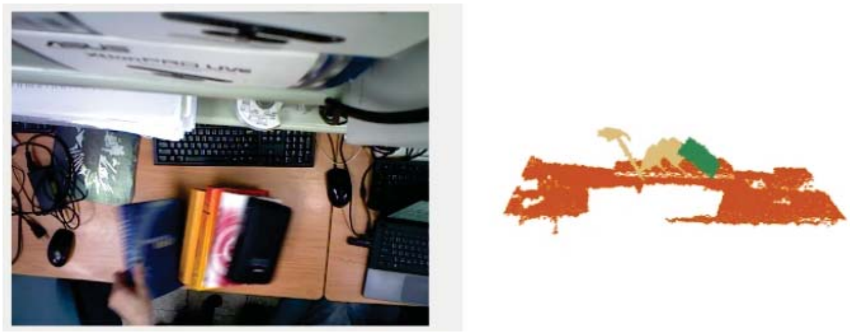


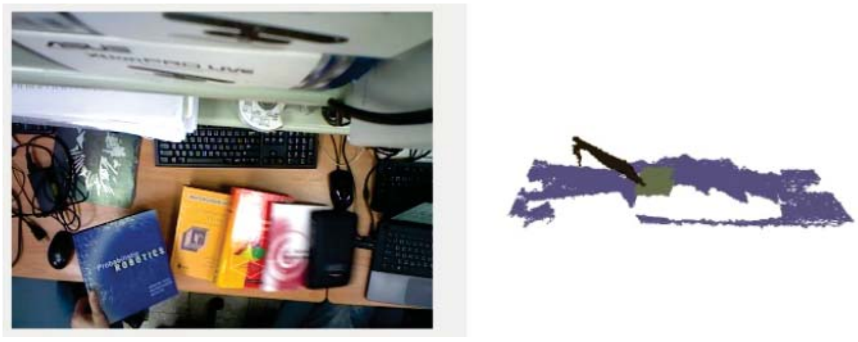
Figure 4.11. A sequential color images of an example of moving two sheets of paper.

Fig. 4.11. It is also a case of moving multiple objects since the paper on top must be moved first. Owing to the up an down relation between paper, we can understand that the yellow paper should be taken before the paper at the bottom. With this prior knowledge, each paper could be tracked properly.

There are some much difficult cases that we still can not solve with the proposed method. One is the appearance of object is changed such as opening a closed box and putting a ball in it. Since the information of the resolution of depth is not sufficient for spiting the box and the ball in it into two objects. It is hard to check the ball is in the box or not. In addition, to model the in and out relation is much more complicated, so it is still a limitation for our system. Another difficult case is that directly drawing out a book from the middle of a stack without moving the books on top in advance as illustrated in the Fig. 4.12. In each image, different color represents an independent object from the result of clustering. We can find that almost all objects are clustered to one because they are indeed connected in some parts. Moreover, the collapse happens too fast to be tracked in the object level in real time. A possible solution is ICP with the color information which we may try in the future work.


(a)


(b)


(c)

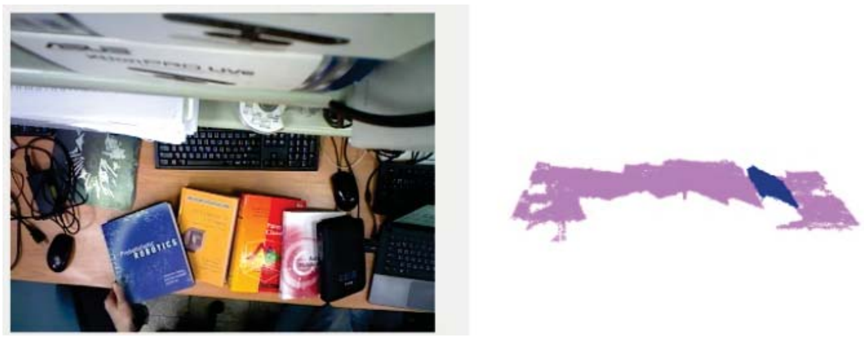

(d)

Figure 4.12. A failed case which a book is directly drawn from the middle of stack. The left side is the color image while the point cloud is depicted right side. Each object from clustering is marked by one color.

CHAPTER 5

Conclusion and Future Work

IN conclusion, to solve the problem of finding objects on the desk, we have proposed a approach to effectively model objects with its location and appearance with a static RGB-D camera based on the concept of scene stability. The inference of spatial relation is proposed to handle the data association for thin objects. In addition, a variety of the desk objects ranging from large objects such as books to small or thin objects such as envelopes and scissor are successfully detected, segmented and modeled as well as many different ways to manipulate objects are tested for the feasibility. In the end, a system with friendly and intuitive user interface is implemented in real time and it has been demonstrated to be suitable for practical use. For the future work, more manipulative ways would be considered and solved in order to achieve a better performance and more practical in real life. We will try to exploit and enhance the performance in real time to loose the constraint on utilization. Furthermore, with the rise of wearable device, it is in our interests to extend this work to other scenarios using moving or egocentric RGB-D cameras.

BIBLIOGRAPHY

Point cloud library. `http://pointclouds.org/`.

Attamimi, M., Mizutani, A., Nakamura, T., Nagai, T., Funakoshi, K., and Nakano, M. (2010). Real-time 3d visual sensor for robust object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS),*, pages 4560–4565.

Butz, A., Schneider, M., and Spassova, M. (2004). Searchlight–a lightweight search function for pervasive environments. In *Pervasive Computing*, pages 351–356. Springer.

Koyuncu, H. and Yang, S. H. (2010). A survey of indoor positioning and object locating systems. *International Journal of Computer Science and Network Security(IJCSNS)*, 10(5):121–128.

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, 2:83–97.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011a). A large-scale hierarchical multi-view rgb-d object dataset. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1817–1824.

Lai, K., Bo, L., Ren, X., and Fox, D. (2011b). Sparse distance learning for object recognition combining rgb and depth information. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4007–4013.

Lai, K. and Fox, D. (2010). Object recognition in 3d point clouds using web data and domain adaptation. *The International Journal of Robotics Research*, 29(8):1019–1037.

Ljungstrand, P., Redström, J., and Holmquist, L. E. (2000). Webstickers: using physical tokens to access, manage and share bookmarks to the web. In *Proceedings of DARE 2000 on Designing augmented reality environments*, pages 23–31.

Lockton, R. and Fitzgibbon, A. W. (2002). Real-time gesture recognition using deterministic boosting. In *BMVC*, volume 2002, pages 1–10.

Oikonomidis, I., Kyriazis, N., and Argyros, A. A. (2011). Efficient model-based 3d tracking of hand articulations using kinect. In *BMVC*, volume 1, page 3.

Pederson, T. (2001). Magic touch: A simple object location tracking system enabling the development of physical-virtual artefacts in office environments. *Personal and Ubiquitous Computing*, 5(1):54–57.

Qian, C., Sun, X., Wei, Y., Tang, X., and Sun, J. Realtime and robust hand tracking from depth.

Romer, K., Ostermaier, B., Mattern, F., Fahrmair, M., and Kellerer, W. (2010). Real-time search for real-world entities: A survey. *Proceedings of the IEEE*, 98(11):1887–1902.

Rusu, R. B. (2010). Semantic 3d object maps for everyday manipulation in human living environments. *KI-Künstliche Intelligenz*, 24(4):345–348.

Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.

Starner, T. and Pentland, A. (1997). Real-time american sign language recognition from video using hidden markov models. In *Motion-Based Recognition*, pages 227–243. Springer.

Tang, D., Yu, T.-H., and Kim, T.-K. (2013). Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3224–3231.

Wren, C. R., Azarbayejani, A., Darrell, T., and Pentland, A. P. (1997). Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785.

Zhu, X., Yang, L., and Waibel, A. (2000). Segmenting hands of arbitrary color. In *IEEE International Conference on Automatic Face and Gesture Recognition*, pages 446–453.

Document Log:

Manuscript Version 1.0 — 27 January 2015

Typeset by $\mathcal{AMS}$-LaTeX — 27 January 2015

Kung-Hung Lu

The Robot Perception and Learning Lab., Department of Computer Science and Information Engineering, National Taiwan University, No.1, Sec. 4, Roosevelt Rd., Da-an District, Taipei City, 106, Taiwan, *Tel.* : (+886) 2-3366-4888 ext.407

E-mail address: `r02922102@ntu.edu.tw`

Typeset by $\mathcal{AMS}$-LaTeX