Gezamenlijke opleiding UHasselt en KU Leuven

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

6D pose tracking of bins without CAD model using RGBD data: evaluation of the BundleTrack algorithm

Koen Fierens
Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

PROMOTOR :
Prof. dr. ir. Eric DEMEESTER

BEGELEIDER :
Mevrouw Yanming WU

2022
2023

6D pose tracking of bins without CAD model using RGBD data: evaluation of the BundleTrack algorithm

Koen Fierens

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

PROMOTOR :

Prof. dr. ir. Eric DEMEESTER

BEGELEIDER :

Mevrouw Yanming WU

ACKNOWLEDGEMENTS

I would like to acknowledge the following individuals, groups and organizations for their invaluable contributions to my master's thesis:

- Ir. Yanming Wu for her advice and expertise in the field, which were essential to the successful completion of my thesis.

- Prof. Dr. Ir. Eric Demeester for his guidance and insightful feedback throughout the course of my research.

- My parents for their support and encouragement throughout my academic journey.

- My fellow students and friends, who have provided me with valuable insights throughout my studies.

- ACRO for supporting my research.

TABLE OF CONTENTS

TABLES

FIGURES

NOMENCLATURE

Acronyms /	**Abbreviations**
6D pose	six degrees-of-freedom pose
ACRO	Automatization, Computer vision & Robotics
ADD	Average Distance of model points
ADD-S	Average Distance of model points-symmetry
AUC	Area Under Curve
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
CUDA	Compute Unified Device Architecture
DoG	Difference-of-Gaussian
ICP	Iterative Closest Point
LF-Net	Learnable Feature-Net
LIDAR	Light Detection and Ranging
OS	Operating System
ReLU	Rectified Linear Units
RANSAC	Random Sample Consensus
ROI	Region Of Interest
SAM	Segment Anything Model
SIFT	Scale Invariant Feature Transform
T-VOS	Transductive-VOS
ViT	Vision Transformer
WSL	Windows Subsystem for Linux

ABSTRACT

The Automatization, Computer vision and Robotics (ACRO) research group of KU Leuven has developed a robot capable of picking up wooden bins, or palloxes, in an orchard. To automate this process, determining the six degrees-of-freedom (6D) pose of a pallox is crucial. A tracking by detection algorithm is used for the first frame, and a faster temporal tracking method is used for subsequent frames without relying on a Computer Aided Design (CAD) model. This approach enables the algorithm to accurately and efficiently track the palloxes while being computationally inexpensive.

This paper presents a software solution that utilizes BundleTrack in combination with the Segment Anything Model (SAM) and custom code. The presented approach calculates the 6D pose of the object relative to the first frame and uses the RealSense LIDAR L515 camera for capturing the data.

The solution's accuracy is evaluated by comparing the algorithm's output to the ground truth measurements. For obtaining the ground truth, the 6DPoseAnnotator algorithm is used. BundleTrack accurately estimates the 6D pose of the pallox, with an average rotation error of 2.93° and a position error of 7.74 cm. It achieves an Area Under Curve (AUC) of 26.17 measured by the Average Distance of Points (ADD) metric and an AUC of 72.94 measured by the Average Distance of Points Symmetry (ADD-S) metric. The algorithm achieves a real-time performance of 6 Hz. However, further optimization is necessary to achieve full autonomy for the proposed approach.

ABSTRACT IN DUTCH

De onderzoeksgroep Automatisering, Computer vision en Robotica (ACRO) van KU Leuven heeft een heftruck ontwikkeld die houten bakken, of palloxen, in een boomgaard kan oppakken. Om dit te automatiseren, is het belangrijk om de six degrees-of-freedom (6D)-pose van een pallox te bepalen. Voor het volgen van de palloxen werd een tracking by detection-algoritme gebruikt voor het eerste frame, en een snellere temporal tracking-methode zonder een Computer Aided Design (CAD)-model voor de volgende frames. Op die manier kan het algoritme de palloxen nauwkeurig en efficiënt volgen, terwijl het weinig rekenkracht vereist.

Deze paper presenteert een softwareoplossing die gebruik maakt van BundleTrack in combinatie met het Segment Anything Model (SAM) en bijkomende code. Het voorgestelde algoritme berekent de 6D-pose van het object ten opzichte van het eerste frame en gebruikt de RealSense LIDAR L515 camera voor gegevensvastlegging.

De nauwkeurigheid van de oplossing werd geëvalueerd door de output van het algoritme te vergelijken met de ground truth-metingen, verkregen met het 6DPoseAnnotator-algoritme. BundleTrack schat de 6D-pose van de pallox nauwkeurig in, met een gemiddelde rotatiefout van 2,93° en een positiefout van 7,74 cm. Het behaalt een Area Under Curve (AUC) van 26,17 voor de Average Distance of Points (ADD) metriek en een AUC van 72,94 voor de Average Distance of Points Symmetry (ADD-S) metriek. Daarnaast heeft het algoritme een realtimeprestatie van 6 Hz, maar vereist het verdere optimalisatie voor volledige autonomie.

1 INTRODUCTION

1.1 Background and motivation

The emergence of automatic robots brings great advancements to the industry, ensuring better efficiency and cost savings. This applies to the agricultural sector as well. For example, machine learning and computer vision enable the automatic harvesting of fruit and vegetables. Tractors can navigate automatically, crops can be planted through automation and drones can map and spray the fields. By automating these tasks, a company operates much more efficiently and achieves cost savings. That is why the Automatization, Computer vision & Robotics (ACRO) research group of KU Leuven has sought assistance.

ACRO operates within the Faculty of Engineering Technology and has many years of experience in vision-based and model-based automation, human-robot interaction and collaboration, flexible product handling and robotic grippers, collision-free trajectory generation and navigation, semi-autonomous & autonomous (dis)assembly and functional programming for robotics and the cloud.

ACRO has developed a pallox truck capable of picking up and offloading wooden boxes, also called palloxes. These palloxes will be placed in an orchard and may contain fruit or be empty. The pallox truck will drive around the orchard and look for those palloxes in order to move them to another area where the fruit can be extracted.

While manual operation by a person is possible, ACRO aims to automate this process due to several disadvantages associated with manual driving. For instance:

- Operators may not use the optimal routes, which results in delays.
- Human errors such as accidents, collisions or overlooking palloxes.
- Labor costs are significant depending on the size of the orchard.

- Expanding an orchard would result in additional personnel, which means even higher costs.

Fig. 1 visualizes how the pallox truck and the wooden box look.

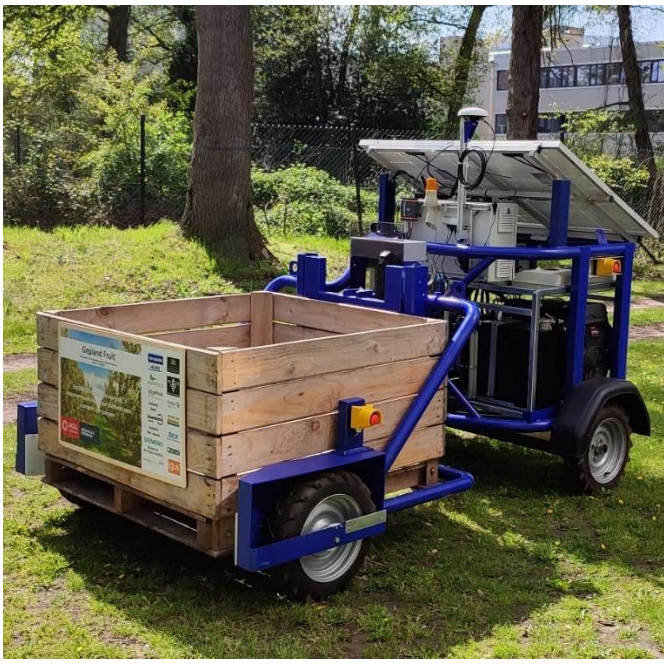


Figure 1: The pallox truck carrying the pallox

1.2 Context

As mentioned earlier, ACRO aims to fully automate the process of picking and placing the palloxes within the orchard. To achieve this goal, the development of several algorithms is required.

Firstly, methods to locate and map the pallox truck's position within the orchard are required. Additionally, algorithms that calculate the most efficient route are needed to efficiently navigate through the orchard. Moreover, the development of algorithms that avoid obstacles is required to ensure the safety of the system. Furthermore, algorithms have to be developed that control the actuators of the pallox truck, based on the various inputs.

Another critical requirement is that the pallox truck should have mechanisms capable of detecting and recovering from failures. This is important since the pallox truck should be able to quickly recover and resume its tasks without compromising efficiency or safety.

There is also a need for monitoring systems, as these will provide oversight of the automated processes. These monitoring systems should also allow for anomaly detection. All the aforementioned methods and algorithms should work together with enough speed to enable real-time performance.

1.3 Problem statement

It is undoubtedly a complex task to fully automate the process of picking and placing the palloxes, given the numerous challenges. Because of the complexity of the task, it has been broken down into smaller parts. One of those parts (which is also the problem this paper aims to solve) is to calculate the position and orientation of the wooden box relative to the camera coordinate system over multiple frames. This calculation is crucial because it directly affects the actions of the pallox truck's actuators.

Determining the position and orientation of an object is commonly referred to as calculating its six degrees-of-freedom (6D) pose. In Fig. 2, the 6D poses of five distinct objects are visualized. The bounding box of each object is determined based on the calculated 6D pose, which includes the position in three dimensions and the orientation around the x, y, and z axes.

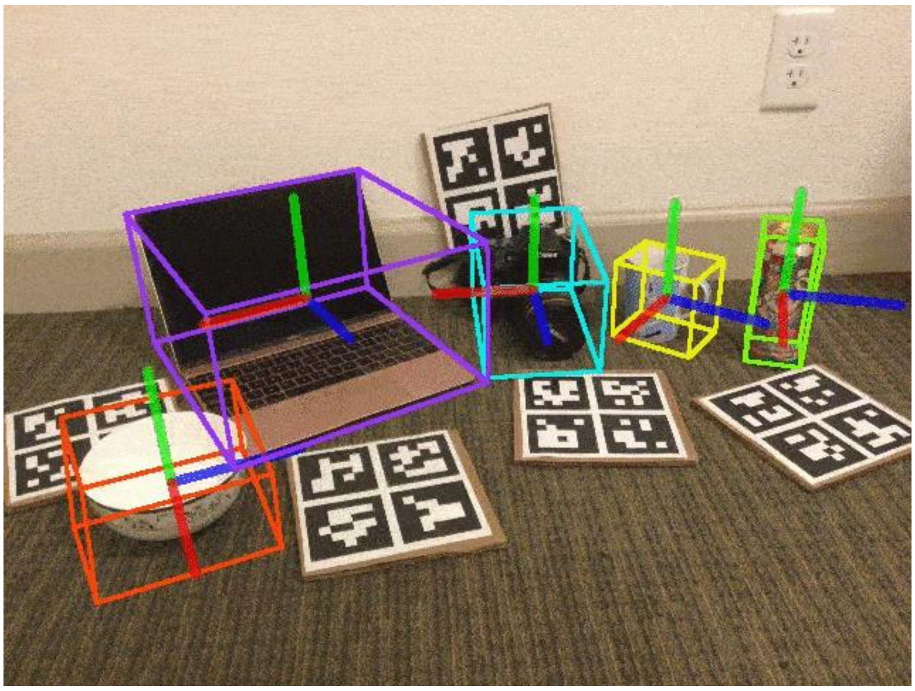


Figure 2: Visualization of 6D pose for five objects: a bowl, a laptop, a camera, a mug and a spray can [1]

1.4 Objectives

The objective of the paper is to present a method that calculates the 6D pose of the object relative to the first frame for multiple frames. Considering the final objective is to automate it so that it works in real-time, the algorithm's speed holds significant importance.

To achieve this objective, the following goals are outlined:

1. Conduct a comprehensive literature review on 6D pose estimation, covering both tracking by detection and temporal tracking approaches. Tracking-by-detection methods detect the object in each frame independently, while temporal tracking aims to obtain the pose of the object using the information from previous frames. Temporal tracking methods, with their ability to utilize temporal information, offer improved speed and efficiency, making them well suited for real-time applications. After thoroughly reviewing the literature, a temporal tracking method, namely BundleTrack, that demonstrated the best trade-off between accuracy and real-time performance, considering the specific requirements of our use case, was selected.
2. Next, implement BundleTrack on the data from the original paper and replicate the results reported in that paper.
3. After that, execute the selected temporal tracking method on the data for the paper's use case and collect ground truth data.
4. Finally, evaluate the performance of BundleTrack using the information acquired in step 3. The evaluation contains information about the algorithm's accuracy and speed.

It is noteworthy that this paper assumes that an alternative method is utilized to calculate the absolute pose of the wooden box in the initial frame. The proposed method focuses solely on calculating the relative pose compared to the first frame.

1.5 Structure of the paper

The paper is divided into several chapters in order to improve its readability. The first section is the literature review. This section explores the various types of 6D pose estimation methods. It makes a clear distinction between each type and discusses the underlying principles. Each method will also be examined in different aspects.

Following the literature study, the methodology chapter introduces and explains the 6D pose estimation method employed in this master's thesis, namely BundleTrack. It covers the prerequisites for running the code and discusses the implementation and customization of the program. Additionally, this section explains the other algorithms that are added to the workflow, which are required to obtain accurate results from the BundleTrack algorithm. Finally, the methodology section includes the different methods used to obtain the ground truth pose of the object and the input modalities considered for the algorithm.

Subsequently, there is a dedicated section that delves into the experimental setup and evaluation metrics. This part elucidates the steps taken to acquire the ground truth pose and clarifies the evaluation metrics used to assess the performance of BundleTrack.

Afterwards, the results section presents and discusses the obtained results, along with potential approaches for further improvement.

Finally, the paper concludes with a section summarizing the main points and outlining future work. This section reiterates the significance of the paper's findings and outlines future directions for utilizing the proposed method in real-world use cases.

2 LITERATURE REVIEW

2.1 Introduction and fundamentals

To better understand the work presented in this paper, it is crucial to have a basic understanding of the fundamental aspects underlying 6D pose estimation. This computer vision technique determines the 6D pose of an object, which refers to the position and orientation of the object. This method uses the features of an object (e.g., corners or edges) to acquire information about the object's position expressed in x, y and z parameters, as well as the orientation of the object expressed in the angle around the yaw, pitch and roll axes [2].

6D pose estimation can be divided in different ways. First of all, 6D pose estimation methods include two categories: tracking by detection methods and temporal tracking methods. Tracking by detection methods involves algorithms that compute the 6D pose of the object on each frame separately. Temporal tracking methods, on the other hand, leverage temporal information from previous frames.

6D object pose estimation can also be classified based on the availability of the object model. There are methods that require CAD models, and there are methods that do not require CAD models. For the methods that require CAD models, they can be further subdivided into instance-level pose estimation and category-level pose estimation. Instance-level pose estimation requires the CAD model of the exact object that needs to be estimated. Category-level pose estimation, on the other hand, requires a CAD model of an object from the same category for training [3].

2.2 Tracking by detection methods

As stated before, 6D pose estimation can be categorized into tracking by detection methods and temporal tracking methods. The former is used to find the pose of the object without relying on previous frames. Tracking by detection methods is divided into two categories: learning-based and model-based pose estimation, with the latter being further subdivided into 2D and 3D model-based pose estimation. Learning-based approaches use machine learning or convolutional neural networks (CNNs) in order to estimate the pose of the object, and there are several kinds of learning-based algorithms available [4].

2D model-based pose estimation estimates the pose of an object through the 2D information in an image and can also be further subdivided into real-image-based approaches and CAD-image-based approaches, the difference being whether the template is a real image or a CAD model. CAD-image-based approaches are often more accurate, but CAD models can be difficult to obtain.

3D model-based pose estimation, on the other hand, uses the 3D information of the object in order to determine its location and orientation. It can also be further divided into two sub-categories: matching-based and local descriptor-based approaches. Matching-based methods directly match the object of interest to a CAD model, while local descriptor-based approaches (also called feature-based approaches) use key-points in order to calculate the object's pose [4]. Fig. 3 shows a summary of tracking by detection methods.

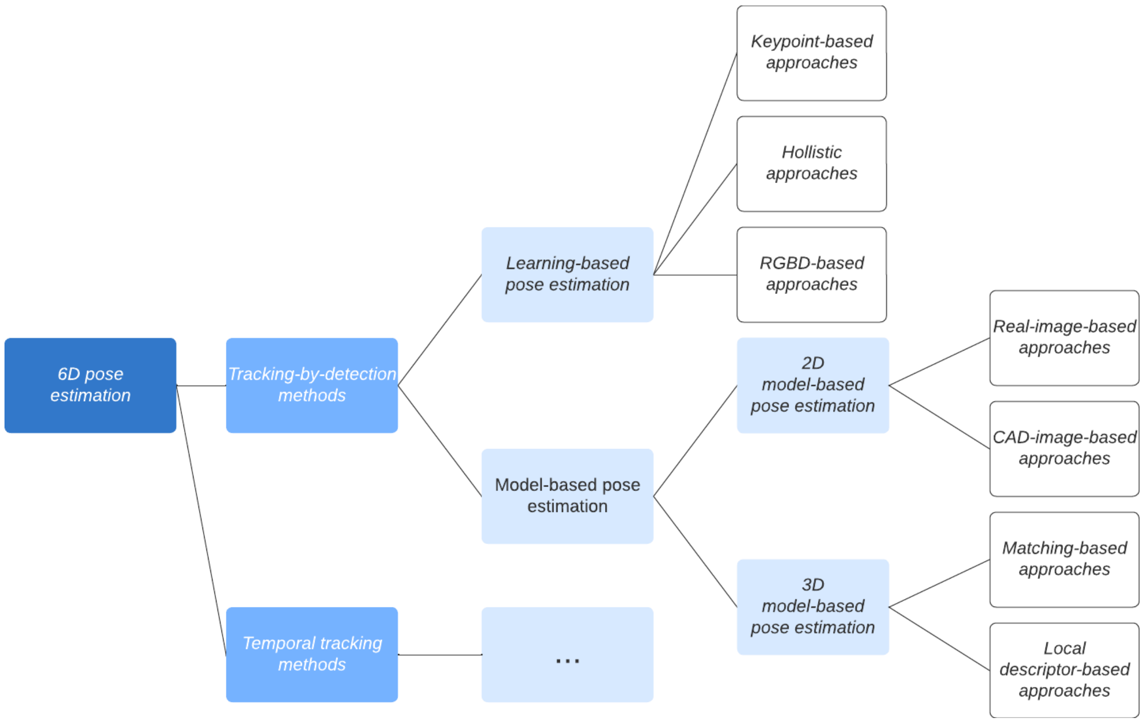

6D pose estimation
Tracking-by-detection methods
Temporal tracking methods
Learning-based pose estimation
Model-based pose estimation
...
Keypoint-based approaches
Hollistic approaches
RGBD-based approaches
2D model-based pose estimation
3D model-based pose estimation
Real-image-based approaches
CAD-image-based approaches
Matching-based approaches
Local descriptor-based approaches

Figure 3: Overview of tracking by detection methods for 6D pose estimation

2.2.1 Learning-based 6D pose estimation

2.2.1.1 Introduction to neural networks

Learning-based pose estimation relies on machine learning to estimate the pose of an object. Machine learning techniques for finding the 6D pose of an object are mostly done using neural networks. This is because neural networks can capture complex relationships between input data (e.g., images) and output predictions (e.g., object poses). The availability of large-scale labeled datasets and advancements in deep learning techniques have further contributed to the widespread use of neural networks for pose estimation.

Artificial Neural Networks (ANNs) consist of layers and nodes. Each node is connected to every node of the previous and next layer of nodes, allowing information to flow through the network. Nodes within each layer are responsible for computations and information transformation. Each node receives input signals from the nodes in the previous layer, and these inputs are weighted and summed. An activation function is then applied to the summed input, introducing non-linearities and enabling the network to model complex relationships and capture intricate patterns in the data [5].

The architecture of the ANN, which refers to the number of layers, nodes and activation functions, is dependent on the specific use case. The number of layers depends on the complexity of the dataset but usually ranges from a few to several dozen. The number of nodes also depends on the complexity of the dataset, but a typical range for the number of nodes is between 64 and a couple hundred. The activation functions used in pose estimation include Rectified Linear Units (ReLU) or variants of the sigmoid or hyperbolic tangent functions.

Neural networks require training and input data. These typically consist of RGB images containing the object. However, RGBD data can also be used to provide additional depth information that can be beneficial for more accurate pose estimation [6], [7]. Using RGBD approaches, the network architecture is designed to process the depth data alongside the RGB data using additional channels within the network. Fig. 4 illustrates an exemplary structure of an Artificial Neural Network.

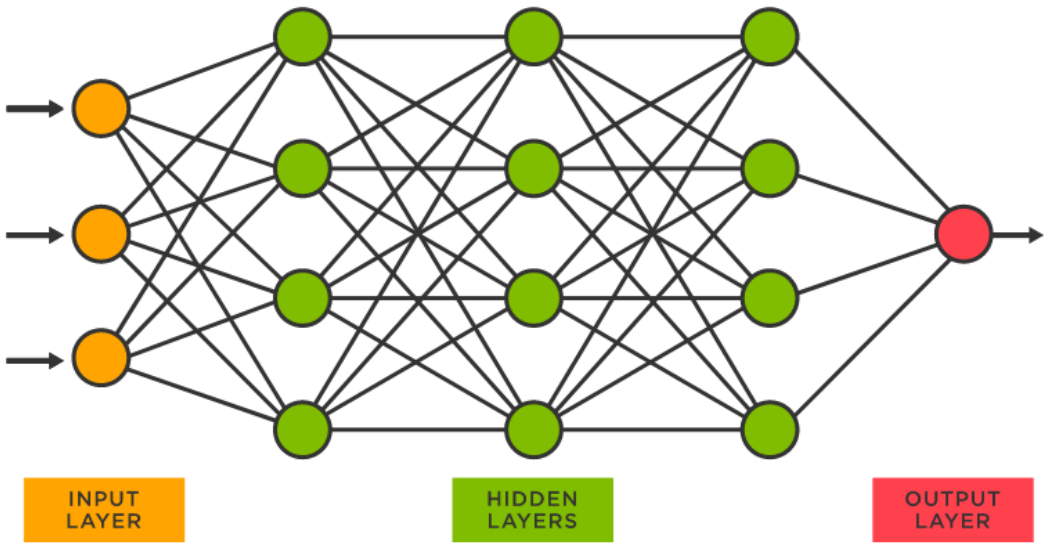


Figure 4: Example of an Aritificial Neural Network [8]

A problem with ANNs is that the input size of the neural network can become large, resulting in high computational power requirements. This is why CNNs are almost always used for computer vision tasks, such as pose estimation. The number of input nodes is reduced using some preliminary steps. These steps are, respectively, convolution, max pooling, and flattening. The convolution step ensures that only the important features (such as edges) of the image are taken as input. Max pooling allows for spatial variance, which means that the CNN can still work for warped images. Finally, flattening reformats the remaining data in order for it to be valid as an input to the ANN. Fig. 5 presents a summary of the preliminary steps taken in CNNs [9].

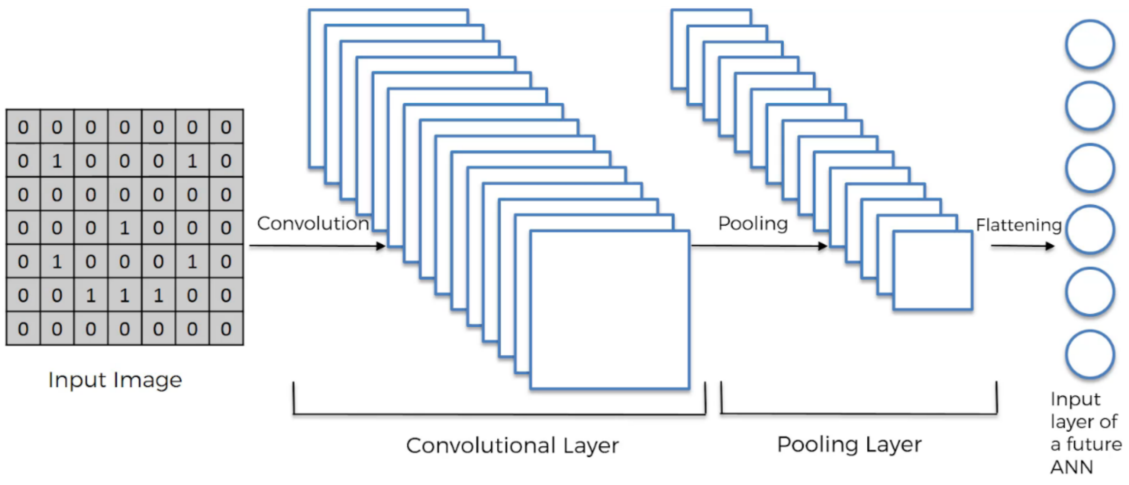


Figure 5: Preliminary steps of a Convolutional Neural Network [10]

2.2.1.2 Deep learning

To train an ANN, initially the weights of the network are given a random value. The training process involves forward propagation and back propagation. The term forward propagation refers to the process of computing the output of a neural network given an input. Then, using all the nodes, layers and activation functions (which are dependent on the specific application of 6D pose calculation), the network can (depending on the specific network) manage the 16 output nodes, which correspond to 9 rotation values, 3 translation values and the homogeneous transformation matrix. Then, the backpropagation step is done. In this step, these output values can be compared to the ground truth. Then, the fault per node is calculated and the weights are changed [10].

After the network is trained and an input is given to it, certain nodes will activate while others do not. For example, if an image of a cardboard box is fed into the ANN with a rotation angle of 30° relative to the camera, the nodes within the network's layers will activate in a way that generates output values representing the rotation matrix. This rotation matrix explicitly indicates that the object is rotated by 30°.

2.2.1.3 Training data and data annotation

Training data is crucial for machine learning-based methods. It consists of a lot of frames that contain the object in them with the corresponding ground truth pose. This data can be acquired using different methods, such as manual annotation, the use of depth sensors or synthetic data.

Manual annotation refers to the time-consuming process of people calculating the pose for each frame. Using the depth data from depth-sensing cameras in combination with known camera poses can also be utilized to generate accurate ground truth poses. Another method would be using synthetic data. This method requires a CAD model and a virtual camera, which are placed in a virtual environment. This approach can be used if a large amount of data should be generated. After obtaining initial training data, a step called data augmentation can be used. Data augmentation refers to the process of generating more training data using transformations such as translations, rotations and scaling on the object within the virtual environment [11], [12]. Fig. 6 shows an example of synthetic and augmented data.

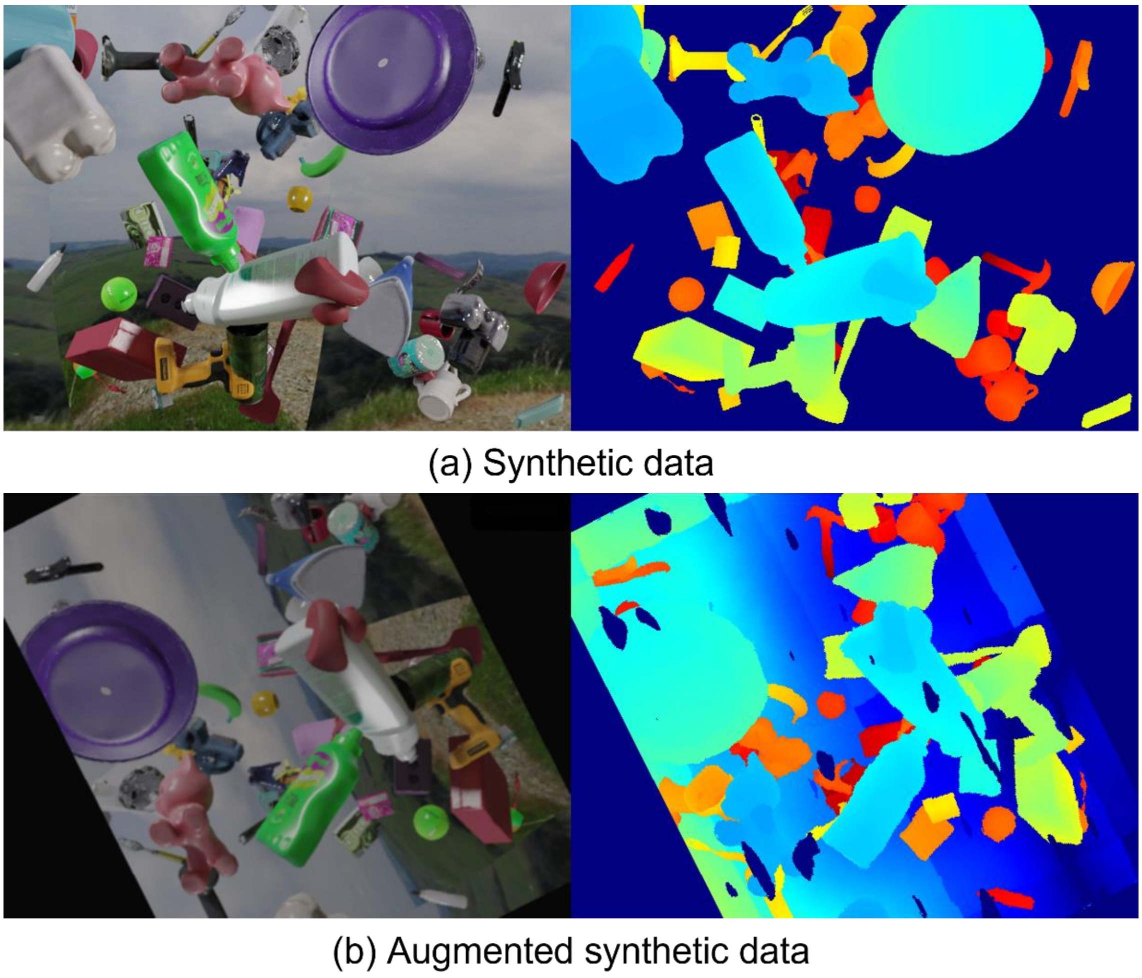


Figure 6: (a) Example of synthetic data (b) Augmented synthetic data to increase the training data [13]

2.2.1.4 Keypoint-based approach

Keypoint-based approaches and holistic approaches are two different ways of implementing a learning-based 6D pose estimation algorithm. Keypoint-based approaches first detect the keypoints on an object, which are then fed into the trained neural network in order to estimate the pose.

Firstly, they are not end-to-end algorithms, which means they require intermediate steps and may not be as well optimized as a single-stage method. Secondly, the loss function or cost function used in these approaches cannot accurately represent the accuracy of 6D pose estimation, which is a significant limitation. A loss function is some sort of measurement of the model's performance during training. Existing algorithms include BB8 and PVNET [4], [14], [15].

2.2.1.5 Holistic approach

The holistic approach aims to eliminate the limitations of keypoint-based approaches by using an end-to-end architecture. This approach is able to calculate the 6D pose of an object without any intermediate steps. As a result, they are faster than key-point-based approaches. Existing algorithms include PoseNet, SSD-6D and PoseCNN [4], [16]–[18].

2.2.1.6 RGBD based approach

Another method for learning-based 6D pose estimation is RGBD-based learning. Algorithms that belong to this approach not only use the color information but also the depth information. Because extra information is provided, RGBD-based learning approaches are generally more accurate than key-point-based or holistic approaches. Existing algorithms that use RGBD-based learning include DenseFusion, G2L-Net and CosyPose [4], [19]–[21].

2.2.2 Model-based 6D pose estimation

Model-based approaches are yet another way of calculating the 6D pose of an object. Algorithms that belong to this group always compare the current image of an object, from which the 6D pose should be calculated, to some sort of template.

2.2.2.1 2D model-based approach

2D model-based approaches only require 2D information about the object. This means that it is easier to obtain the necessary information using cheaper devices. In 2D model-based approaches, the shapes, colors and textures are used for 6D pose estimation [4].

As stated before, the information gathered will be compared to some sort of template. This template can either be a real-life image or a CAD model. In the latter case, the CAD model generates a 2D image, which will be used for comparison. The advantage of using the CAD model as a template is that these generated images are often of higher quality than the real-life images. This is an important property since it improves the accuracy of pose estimation. Using multiple templates can also improve accuracy. Algorithms such as FPM and epipolar geometry methods fall under this category [4], [22], [23].

Real image-based approaches are used when accurate 3D CAD models are not available. HoG and multi-cooperative logos are state-of-the-art algorithms that are commonly used for this approach [24], [25].

2.2.2.2 3D model-based approach

3D model-based approaches require 3D information in order to work, which also implies that they are more robust than the previous method. In matching-based approaches, the current 3D image is compared to either a single CAD model or to multiple CAD models.

When compared to a single CAD model, the 6D pose relative to the CAD model is computed. In contrast, when comparing to multiple models, the one with the highest correlation is chosen as the corresponding model, and the rotation and translation values relative to that model are computed. Algorithms that are used for 3D matching-based approaches include PCOF-MOD in combination with BPT [26].

In local descriptor approaches, two point clouds (a reference and a source) are compared. One of those (the source) is moved around using a transformation matrix. With this transformation matrix and the known pose of the reference, the 6D pose of the object can be found. When using global registration, which means that no manual alignment is required, geometric features are first found in order to find corresponding points between the two point clouds. Algorithms used for a 3D model-based approach are ICP, FPFH and RANSAC [27], [28].

2.2.3 Limitations of existing detection approaches

In this section, the advantages and disadvantages previously discussed will be discussed. While Table 1 illustrates the general performance for each method, it is important to note that the specific algorithm used can impact these values.

Accuracy is a measure of how well a technology can determine the translation and rotation of an object. Storage cost refers to the amount of data required for the method to work. Robustness measures how much performance suffers when noise or environmental changes occur. The time cost is the amount of time and computational resources required to run the algorithm. Range of application refers to the specific scenarios or objects that a 6D pose estimation approach is well-suited for [4].

TABLE 1 Overview of advantages and disadvantages of each approach. 3 represents the best (performance), and 1 represents the worst (performance) [4]

Division	Subdivision	Accuracy	Storage cost	Robustness	Time Cost	Online performance	Range of Application
Learning-based pose estimation	Keypoint-based	2	2	2	1	1	2
	Holistic	1	2	2	2	2	2
	RGBD-based	3	1	3	1	1	1
Model-based pose estimation	2D model-based	2	3	1	3	3	3
	3D model-based	3	2	2	2	2	1

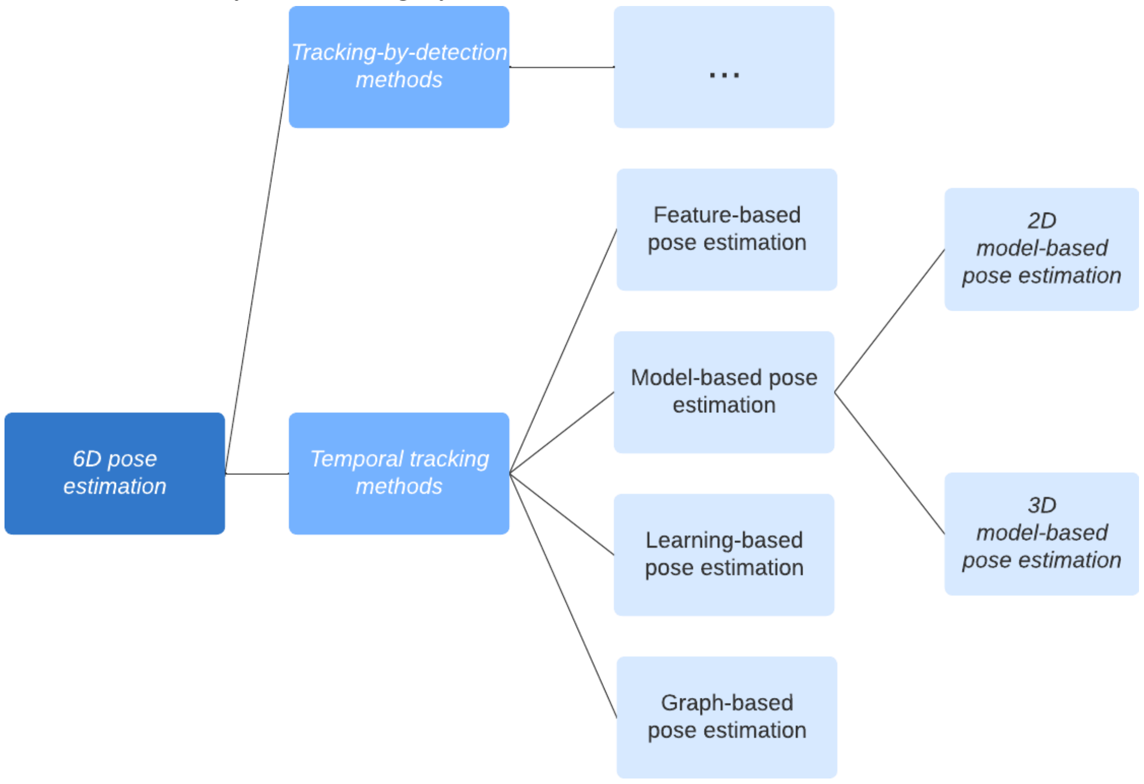
In addition to each approach having its own advantages and disadvantages, there are some general challenges concerning pose estimation. The most important and difficult challenges are [4]:

- Textureless objects do not allow for easy extraction of features since the algorithms that determine those features prefer clear boundaries between edges and corners. The extraction of the 3D image is also more difficult. To overcome the latter, a device that does not rely on texture, such as Light Detection and Ranging (LIDAR), is used for depth calculation.
- Object occlusion presents another challenge for pose estimation. This means that the object being tracked is obscured by other objects. As a result, not all of the features of the object are detected, resulting in a more difficult process of matching to a template when using a model-based approach or less accuracy when using a learning-based approach.
- Reflections of objects are a challenge. High levels of reflection can make it difficult to accurately calculate the object's features. Additionally, objects with low or high reflection can pose difficulties in acquiring depth information.
- Noise in the data can interfere with accurate feature detection on the object.
- Computing power is rather high for tracking by detection algorithms.
- Registration becomes more challenging when dealing with deformable objects such as clothing.

2.3 Temporal tracking methods

The previously discussed methods all belong to tracking by detection algorithms, which means they determine the 6D pose of objects from a single image. Several of those methods could be used in real-time, but their accuracy is limited. Also, because these methods rely on a single image, they ignore the temporal and spatial information across consecutive image frames, which may lead to inconsistent pose estimations across consecutive frames [29].

Temporal tracking methods can be divided into model-based methods, feature-based methods, deep learning-based methods and graph-based methods [30]–[32]. Fig. 7 shows a summary of tracking by detection methods.



6D pose estimation
Tracking-by-detection methods
...
Temporal tracking methods
Feature-based pose estimation
Model-based pose estimation
Learning-based pose estimation
Graph-based pose estimation
2D model-based pose estimation
3D model-based pose estimation

Figure 7: Overview of temporal tracking methods for 6D pose estimation

2.3.1 Feature-based methods

Feature-based algorithms do not require a model. These methods can compute the object's pose by extracting relevant features from a specific area, and then comparing and matching them with features from the previous frame. Leveraging the algorithm's understanding of the known pose in the prior frame and the transformations between the previous and current frame's keypoints, it can accurately determine the object's pose in the current frame. Algorithms used for feature-based approaches are Optical flow, Lucas-Kanade Tracker and Kanade-Lucas-Tomasi Tracker [33]–[35].

2.3.2 Model-based methods

Model-based methods are also an option for temporal tracking methods. These methods can also be further divided into 2D or 3D model-based approaches. However, the algorithms that belong to model-based temporal tracking methods also refine or constrain the estimated pose by leveraging information from previous frames.

Motion estimation is one technique that can be used to improve the speed of this method. As the pose of the previous poses is known, some predictions can be made for the position and rotation of the object. This prediction serves as a starting point for the tracking process, reducing the search space and computational load.

Using the position and rotation of the previous frames, some constraints regarding the object's pose can also be made. If the predicted pose of the object for the current frame is wildly different from the calculated pose of the previous frame, these results can be flagged as inaccurate and recalculated. Particle swarm optimization and genetic algorithms are methods that are commonly used for this approach [36], [37].

2.3.2.1 2D model-based approach

In 2D model-based approaches for temporal tracking methods, the features of the object are extracted from the current image and compared to a set of 2D feature descriptors that originate from either a CAD model or real-life images. However, the estimated pose is refined by leveraging the information from the previous frames.

Temporal information can also be used to improve the speed of the 2D model tracking method. Since the keypoints of the object in the frame are compared to the keypoints of 2D models, a constraint can be used to reduce the number of models it needs to compare to. It relies on spatiotemporal consistency, which means that the object will not move abruptly from one frame to another. This means that it can rule out a large number of models to compare with [30].

2.3.2.2 3D model-based approach

3D model-based approaches use a CAD model in combination with three-dimensional data to determine the 6D pose of the object. The temporal aspect of these methods comes into play by utilizing the pose estimate from the previous frame to initialize and constrain the pose of the object. Local registration methods, such as ICP, are then used to refine the estimated pose [30].

2.3.3 Learning-based methods

These methods use trained neural networks to estimate the 6D pose of the object. Like in the learning-based methods for tracking by detection systems, the architecture of the network depends on the use case. However, in neural networks that are optimized for temporal tracking, the model is adjusted so that previous frames or poses can also act as input for the neural network.

By increasing the amount of information the neural network receives, it is able to output more accurate results. Existing algorithms include VIPose and TrackNet [38], [39].

2.3.4 Graph-based methods

Graph-based methods are methods that extend upon model-based pose estimation or deep learning-based methods. After the pose is estimated using one of those methods, the result is stored in a pose graph. This pose graph consists of nodes and edges. Nodes correspond to a particular pose of the object in a particular frame. The position of these nodes is dependent on the calculated pose. Edges are lines that connect these nodes and represent the difference between the poses of the nodes.

The reason the calculated poses are stored in a pose graph is because pose refinement, also called pose graph optimization for these methods, can be done using these pose graphs. ICP in combination with pose graph optimization is a commonly used approach [40].

2.3.5 Limitations of existing tracking methods

Table 2 highlights some of the advantages and disadvantages of each method for temporal tracking for 6D pose estimation:

TABLE 2 Overview of advantages and disadvantages of each temporal tracking approach [30]

Method	Advantages	Disadvantages
Feature-based	Very fast	Not as accurate
Model-based	High accuracy Can handle occlusions Improved robustness	Requires a model of the object Requires a lot of computational power
Learning-based	Can handle occlusions	Requires large amounts of training data Computationally demanding during training and inference Requires a model of the object
Graph-based	Can handle occlusions	Requires a lot of computational power May require initialization

Pose estimation is a complex task that involves numerous challenges and difficulties. The process of accurately determining the 6D pose of an object requires overcoming various obstacles. Each approach to 6D pose tracking brings its own set of advantages and disadvantages, contributing to the complexity and diversity of this field.

The selection of a particular 6D pose tracking approach involves considering the specific advantages and disadvantages associated with each method. Addressing the challenges of occlusions, real-time performance, and sensor modalities will continue to drive research and innovation in the field of pose estimation, enabling its applications in diverse domains.

2.4 Input modalities

Various algorithms for 6D pose estimation necessitate specific types of input data. Some algorithms solely rely on 2D information, which can be obtained using a simple RGB camera. However, to achieve a more accurate pose estimation, 3D information of the object becomes necessary. Different sensors can be utilized to obtain depth images, with LIDAR and stereo cameras being the most commonly employed options. However, it is important to note that certain algorithms are tailored to work with specific camera types due to the distinct output they provide.

Stereo cameras consist of two separate RGB cameras positioned adjacent to each other at a known distance. These types of cameras use a disparity map to calculate the distance to objects. A disparity map comprises disparity values, which denote the distance between corresponding points on different images. Through a mathematical process known as epipolar geometry, these disparity values can be converted into depth values [41].

For example, when two RGB cameras capture simultaneous images, the corresponding pixels between these images are extracted, and a disparity value is calculated. This disparity value is subsequently translated into a distance measurement.

LIDAR cameras operate based on the principle of time-of-flight. They emit short wave pulses towards a target and calculate the distance by measuring the time it takes for the light to be received back [42]. Both input modalities offer distinct advantages and disadvantages. A comparison between LIDAR and RGBD is found in Table 3.

TABLE 3 Overview of advantages and disadvantages of LIDAR and stereo cameras [43], [44]

Camera type	**Advantages**	**Disadvantages**
LIDAR	High precision High data rate Stable and reliable Not influenced by temperature or light	Adverse weather performance Eye safety regulations limit LIDAR’s signal strength Tied to the reflectivity of the object
Stereo	Low-cost Provides RGB data	Poor long-distance performance Poor performance in low-light environments High computational resource requirements

Typical examples of LIDAR cameras include the well-known Velodyne cameras, as well as the Ouster, RealSense and Livox cameras [45]–[47]. In terms of stereo cameras, widely used options include the ZED stereo camera, Intel's RealSense stereo cameras, and the Bumblebee stereo camera series [48]–[50].

For the specific use case addressed in this paper, a LIDAR camera would be the optimal choice. LIDAR cameras offer distinct advantages, such as being unaffected by lighting conditions, providing high precision, and enabling long-range operation. These characteristics make LIDAR cameras well-suited for the requirements of the paper's application.

The RealSense L515 LIDAR camera is utilized in this master's thesis to test the method employed for 6D pose estimation. However, it is important to note that, because it is designed for indoor applications and only works up to 9 meters, this camera is not suitable for real-world use cases and is primarily intended for testing purposes within the context of the thesis [51].

3 METHODOLOGY

As discussed in the literature review, various methods were explored for calculating the 6D pose of the object. Given that ACRO aims to fully automate the real-time process of picking and placing palloxes, speed becomes a crucial factor. Therefore, the focus of the paper was solely on temporal tracking methods, excluding tracking by detection methods due to their limited potential for real-time implementation.

While model-based methods generally offer improved accuracy, they require computationally intensive comparisons between features of models. Deep learning-based methods provide a potential solution, but acquiring training data with instance-level models can be challenging, and category-level models often yield average results [1]. Additionally, generating custom training data poses laborious challenges.

Feature-based methods have limited accuracy due to their reliance on temporal information from the previous frame, resulting in high drift. On the other hand, graph-based methods are computationally heavy, requiring substantial computational resources for optimizing the pose graph.

However, a solution was devised to mitigate the disadvantages of both feature-based and graph-based methods. By combining these approaches in an algorithm and implementing techniques to optimize the graph-based methods, this approach effectively overcame accuracy limitations and alleviated computational heaviness.

3.1 Introduction to BundleTrack

This paper utilizes the BundleTrack framework, which was introduced in 2021 by B. Wen and K. Bekris. The framework combines various techniques to achieve accurate 6D pose tracking without the need for pre-existing instance- or category-level models. This implies that the method is applicable to novel objects without any training or predefined models. This aspect is significant considering the time-consuming process of creating models for certain objects.

BundleTrack accomplishes this by integrating a feature-based method with pose-graph optimization. Additionally, the framework incorporates algorithms for object segmentation and introduces enhancements to the general pose graph optimization models through the selection of keyframes.

To handle the computational intensity of multi-pair feature matching and pose-graph optimization for 6D object pose tracking, BundleTrack utilizes an efficient CUDA implementation. This allows for parallel processing on a GPU instead of sequential computation on the CPU.

It is important to note that BundleTrack functions as a temporal tracking method, computing the pose relative to the initial frame. Therefore, the accuracy of the initial pose provided to the algorithm greatly influences the output of BundleTrack. Obtaining this initial pose relies on a tracking by detection algorithm, which is used for a limited number of frames. The performance and precision of this supplementary method are crucial, as the overall accuracy of BundleTrack relies on the accuracy of the initial pose estimation method [3].

3.1.1 Working principle of BundleTrack

3.1.1.1 Main components of BundleTrack

BundleTrack serves as a versatile framework designed to track the 6D pose of novel objects. It eliminates the need for instance-level or category-level models. The framework leverages the strengths of advanced segmentation and feature extraction algorithms. Additionally, it integrates memory-augmented pose graph optimization, ensuring consistent spatial and temporal tracking. This powerful combination facilitates resilient and precise tracking, even in the presence of challenges such as occlusions and object movements.

The framework consists of multiple interconnected components, each contributing to the overarching objective of determining the 6D pose of an object. The initial component is the video segmentation network, responsible for generating segmented images that highlight the object of interest from the background. The subsequent component then identifies the object's features, such as distinct spots or edges, and performs data association by comparing these features to the previous frame. This process enables BundleTrack to make an initial estimation of the object's 6D pose based on the feature's relative position to the previous frame.

Furthermore, BundleTrack incorporates a keyframe memory pool, which plays a crucial role in the pose graph optimization algorithm. This algorithm refines the initially estimated pose by utilizing the information stored in the keyframe memory pool. Fig. 8 provides a global overview of the algorithm. Detailed explanations of each individual component will be presented in the subsequent sections.

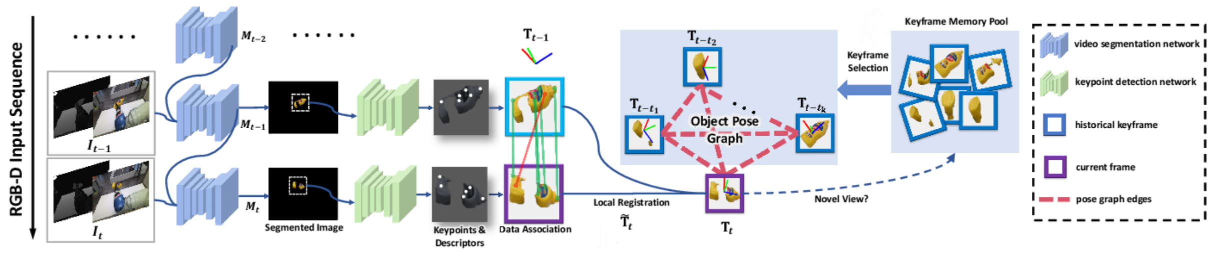


Figure 8: Working principle of BundleTrack [3]

3.1.1.2 Video segmentation network

The initial step of the BundleTrack algorithm involves utilizing a segmentation network to highlight the object of interest for subsequent stages.

To begin, BundleTrack takes RGB and RGBD images as input. Once the images are loaded, a video segmentation network is employed to extract the object of interest as a binary mask from the image. This binary mask represents the region of interest (ROI) by highlighting it in white, while the background is depicted in black.

Multiple algorithms can accomplish this task. Most of these compute the mask for each image independently, such as Mask-RCNN [52]. However, these algorithms are generally less efficient. The approach adopted in this study involves utilizing the Transductive Video Object Segmentation (T-VOS) network.

The T-VOS network is a neural network trained on a large dataset. Through training, the network learns to predict object masks at pixel level based on the input image. Its training on a diverse dataset encompassing various objects, backgrounds, and scenarios enables it to perform effectively on nearly any novel object encountered.

To utilize the trained network, an initial mask is provided, serving as the starting point for the subsequent frame. The network employs the knowledge acquired during training to refine the mask for the new frame by considering the object's appearance within the image. Additionally, the T-VOS network incorporates temporal information from previous frames, utilizing the evolution of prior object masks to ensure consistent segmentation results [53].

3.1.1.3 Keypoint detection and feature descriptors

In the next step of the BundleTrack algorithm, the focus shifts towards detecting the keypoints associated with the object, which constitute its distinctive features within the image. Once the keypoints are detected, a feature vector is generated based on these characteristics. BundleTrack utilizes both the Learnable Feature-Net (LF-Net) and the Scale Invariant Feature Transform (SIFT) algorithms, which are employed for both keypoint detection and feature descriptor extraction [3], [54].

LF-Net consists of two subnetworks: the keypoint subnetwork and the descriptor subnetwork. The keypoint subnetwork is trained to identify potential keypoints within the ROI, while the descriptor subnetwork extracts feature descriptors using the image and the calculated keypoints.

The network was trained using ground truth data containing keypoint locations and feature descriptors. During the training process, the network learned to optimize the parameters of both subnetworks simultaneously. This optimization ensures that the loss function considers both keypoint localization and feature descriptor production, aligning them effectively for accurate matching [55].

Additionally, BundleTrack also leverages the SIFT algorithm to determine feature descriptors for the object. This involves analyzing the image at various scales to identify potential keypoints. This analysis is performed using a Difference-of-Gaussian (DoG) filter. This filter highlights regions in the image where there are noticeable changes in intensity, indicating keypoints.

Subsequently, the positions of these keypoints are further refined by analyzing the local neighborhood surrounding each keypoint. This refinement process involves scrutinizing the intensity values and gradients of nearby pixels, leading to improved accuracy in keypoint localization. Once the keypoints are accurately determined, the SIFT algorithm generates a descriptor for each refined keypoint, capturing important information about its appearance and characteristics [56].

3.1.1.4 Feature matching and pose estimation

When the feature descriptors of two consecutive frames are found, the process of feature matching is initiated. Feature matching involves finding correspondences between the detected features across multiple frames.

To accomplish this, the nearest neighbor matching algorithm is employed. This algorithm compares the feature descriptors of one image with those of another image, aiming to find the closest match based on the total Euclidean distance of all the features in both images [57].

Once the nearest neighbor algorithm performs an initial calculation for feature matches, another algorithm called Random Sample Consensus (RANSAC) is utilized. RANSAC is an iterative algorithm that selects a small subset of matches from the nearest neighbor algorithm. It then estimated a transformation matrix that would convert the features from the subset in one image to the corresponding features from the same subset in the other image. RANSAC evaluates the remaining matched features by applying the same transformation matrix and assessing the distance to the corresponding features in the secondary image. If the distance falls below a certain threshold, the feature is considered an inlier; otherwise, it is ruled an outlier. RANSAC keeps track of the number of inliers obtained for this subset of features.

These preceding steps are repeated for a specific number of iterations. After all the iterations, the transformed model with the highest number of inliers was regarded as the most reliable estimate of the transformation between the two images. Any feature whose distance to the model exceeds a certain threshold is flagged as false and removed [58].

Since RANSAC already calculates a transformation matrix and identifies the best model, an initial estimation of the pose has already been computed. However, this estimated pose might still contain some inaccuracies due to the presence of noise in the data. Therefore, the estimated pose is refined to improve its accuracy.

3.1.1.5 Refinement of the estimated pose

Up to this stage, BundleTrack employs various algorithms to compute the object's pose by leveraging temporal information from the previous frame. However, to minimize drift, BundleTrack also integrates a pose graph optimization step. Consequently, the algorithm not only considers the previous frame but also incorporates additional frames known as keyframes.

Whenever frames are processed by the BundleTrack algorithm, those with minimal motion distortion are stored in the keyframe memory pool. Subsequently, when refining the pose of an object in an image, pose graph optimization is performed using a subset of the stored keyframes from the memory pool. The selection of this subset is based on their mutual viewing overlap with the current frame, ensuring that keyframes capturing similar perspectives of the object are chosen for participation in the pose graph optimization step.

The pose graph optimization step leverages the temporal information from not just the previous frame but several frames sharing similar viewpoints. The algorithm aims to minimize the total energy of the pose graph, which encompasses the feature matching error (E_f) and geometric error (E_g).

To minimize E_f, the algorithm seeks correspondences between features in the current frame and features from the keyframes. For instance, consider a pose graph consisting of three nodes: two keyframes (node A and node B), and one node (node C) representing the current frame requiring refinement. When examining a specific feature, such as a corner, the program utilizes the corresponding feature descriptor of this corner and the pose information from nodes A and B to refine the pose of node C. Meanwhile, to minimize E_g, the program assesses the overall object geometry within each node, facilitating further refinement of the node.

By incorporating not only temporal information from the previous frame but also information from all frames with similar viewpoints, BundleTrack ensures low-drift performance [3].

3.2 Implementation of BundleTrack algorithm

To ensure the correct execution of BundleTrack, certain steps needed to be followed. These steps included fulfilling the prerequisites for running BundleTrack, preprocessing the data, and making specific adjustments to enable BundleTrack to work with the data according to my use case.

3.2.1 Prerequisites

To address the computational complexity of pose graph optimization, as highlighted in Section 2.3.5, a GPU was employed for specific stages of the algorithm. Specifically, BundleTrack optimized its code by implementing a Compute Unified Device Architecture (CUDA) version to execute the multi-pair feature matching and pose-graph optimization for efficient 6D pose tracking. CUDA is a software development platform designed to accelerate parallel computing. It breaks tasks down into multiple threads that are executed separately, allowing the GPU to run numerous threads simultaneously, resulting in improved computational speed. This enables developers to leverage the GPU for general-purpose processing [59].

It is important to note that CUDA is exclusively compatible with NVIDIA GPUs, although not all NVIDIA GPUs support all CUDA versions. To determine compatibility, the compute capability of the GPU must be identified. The compute capability refers to the architectural version of an NVIDIA GPU, which determines the features and capabilities it supports. Reference [60] can be consulted to verify whether a specific compute capability is supported by a certain CUDA version.

BundleTrack heavily depends on external dependencies. Docker images were created to facilitate the installation of required the libraries for running BundleTrack. Docker is a platform that aids in building, sharing, and running applications by packaging them into a single file containing all necessary dependencies. While Docker offers advantages such as readability, version control, and portability, it does not support persistent storage. This means that all program files are stored directly in the operating system (OS) environment. Consequently, Docker images may not always be fully system-independent. For example, while BundleTrack provides a Docker image with most of the necessary libraries, the program relies on storing files on the system, highlighting the importance of selecting the appropriate OS for its successful execution [61], [62].

BundleTrack was originally developed for Linux operating systems. However, it might be possible to run the code in Windows using Windows Subsystem for Linux (WSL), which is a compatibility layer enabling the native execution of Linux binaries on Windows without the need for a virtual machine. With WSL, the Linux file system can be utilized to save files outside the container. Nevertheless, various errors were encountered when attempting this approach, and the forum did not provide clear solutions. As a result, a Linux distribution was installed [63].

For this paper, the BundleTrack program was executed using a GTX 1660S (Super) GPU on Ubuntu 20.04 with CUDA version 10.1. Despite using the Docker images with external dependencies and the aforementioned setup, several errors still arose. This was primarily due to the additional commands required to install the NVIDIA Container Toolkit, which relies on the pre-existing installation of the CUDA runtime and driver program on the system. This program provides the essential components for GPU acceleration, and the NVIDIA Container Toolkit utilizes these components to enable GPU acceleration within Docker containers [64].

Due to the considerable effort invested in resolving issues and running BundleTrack without errors, a comprehensive document was prepared to assist ACRO in the installation and utilization of BundleTrack. The document also includes an explanation and instructions for custom code developed to ensure the correct execution of BundleTrack, which will be further discussed in Section 3.4.

3.3 Initial mask generation

As observed in Section 3.1.1.2, BundleTrack necessitates an initial binary mask to initiate the process of determining the 6D pose for each frame. Since the primary objective is the real-time usage of BundleTrack, automating the creation of the initial mask is essential. Hence, this paper adopts the Segment Anything Model (SAM) developed by Meta AI Research in conjunction with the Grounding DINO algorithm.

SAM is a versatile segmentation system that can handle unfamiliar objects and images without requiring additional training [65]. It is a recently released program, introduced on April 5, 2023. Utilizing SAM, any object within an image can be segmented by providing input prompts such as foreground and background points or bounding boxes. SAM comprises three main components [66]:

- The Image Encoder: This encoder processes the input image. The image undergoes several steps of processing, resulting in a numerical representation that captures crucial features. SAM employs a larger variant of the Vision Transformer (ViT) model called ViT-H.
- The Prompt Encoder: This component changes the input prompt into a numerical representation, commonly referred to as embedding, to improve the model's comprehension of the desired segmentation. Currently, available input options include defining a bounding box or selecting a single point.
- The Mask Decoder: This decoder utilizes the image embedding from the ViT-H encoder and the prompt embedding to predict masks representing the object segmentation within the image.

In terms of SAM's speed, when utilizing an NVIDIA A100 GPU, the image encoder typically requires approximately 150 ms to execute, whereas the prompt encoder and mask decoder take around 50 ms each. Fig. 9 illustrates the overview of the SAM workflow.

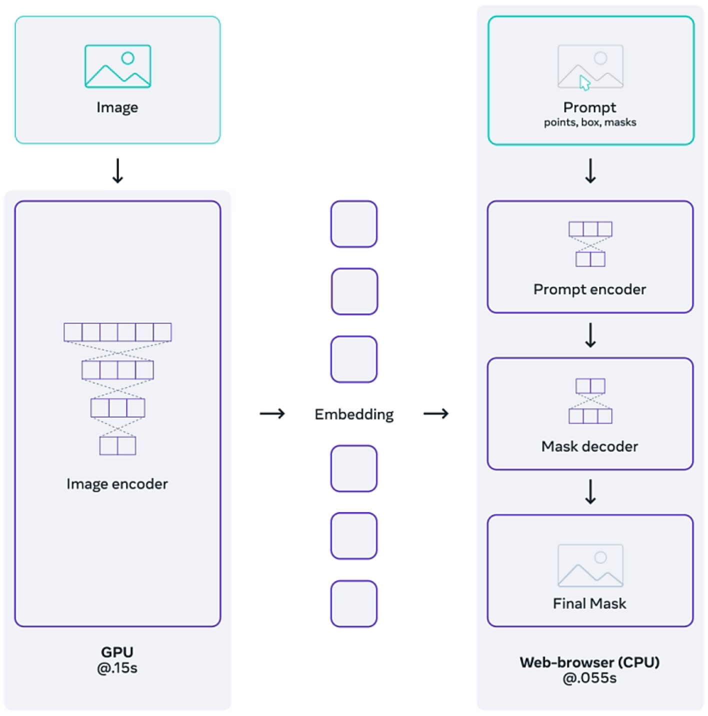


Figure 9: Working principle of SAM [66]

By combining these three components, SAM achieves object segmentation in an image. However, solely relying on this algorithm falls short of enabling real-time functionality for BundleTrack, as the algorithm requires some form of input. One possible approach is to utilize point input prompts. Nevertheless, it has been demonstrated that this approach can yield inaccurate results, as demonstrated in Fig. 10.

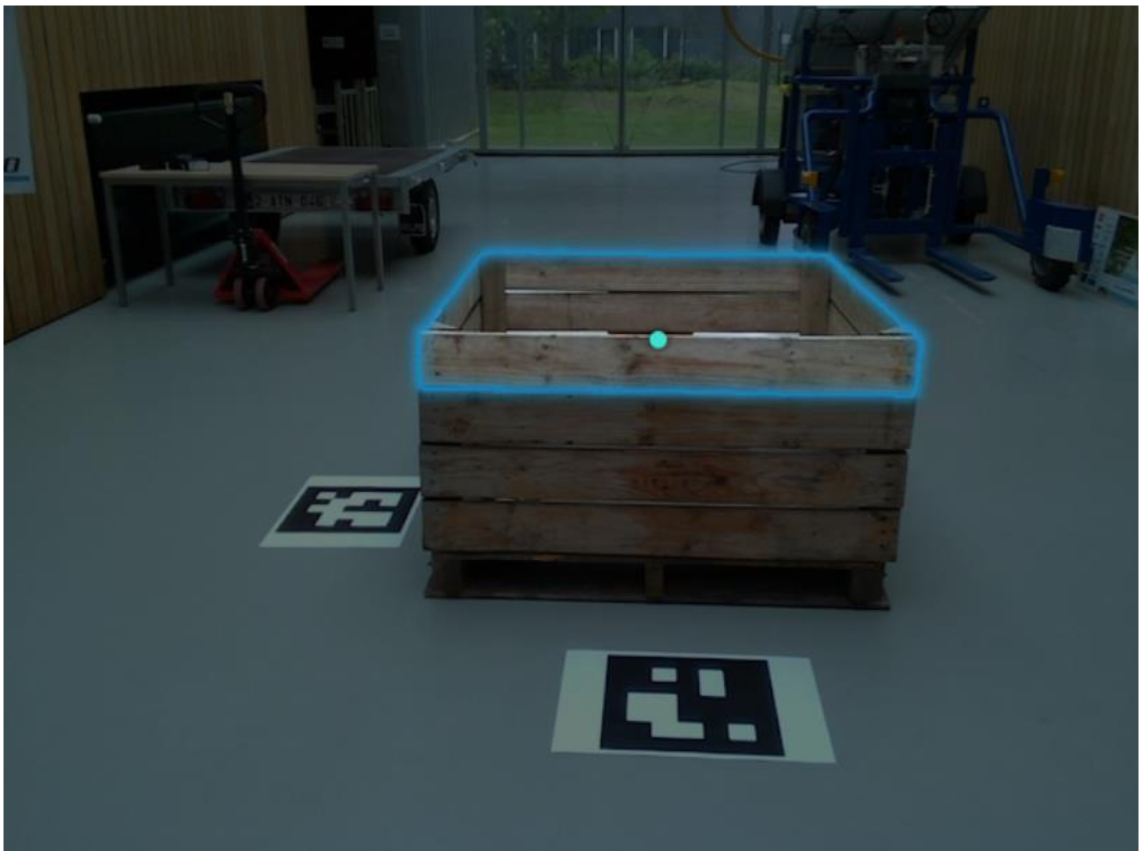


Figure 10: SAM using point input prompt

Another approach would involve using the (inaccurate) bounding box of the pallox. In this case, the coordinates of the upper left corner and the bottom right corner of the wooden box are required. By providing these coordinates to SAM, the algorithm can generate a mask containing the complete wooden box.

3.3.1 Utilizing color-based filtering to obtain the object's bounding box

To find these coordinates, color-based filtering could be employed. Code has been made that processes the frame and identifies regions of brown within a specified color range. It then detects the contours of these brown regions and selects the contour with the largest area. The bounding box of this selected contour is calculated to determine the object's location and size in the image.

Nevertheless, this method is not foolproof, as there is a possibility of encountering a brown object in the frame that is larger than the wooden box. Additionally, the method may fail when the object is in shadow or exposed to excessive light, leading to a change in color and potential a failure in detection. Due to the limitations and potential failures of the color-based filtering method, a different solution was chosen as an alternative. An example of a failure case is shown in Fig. 11.

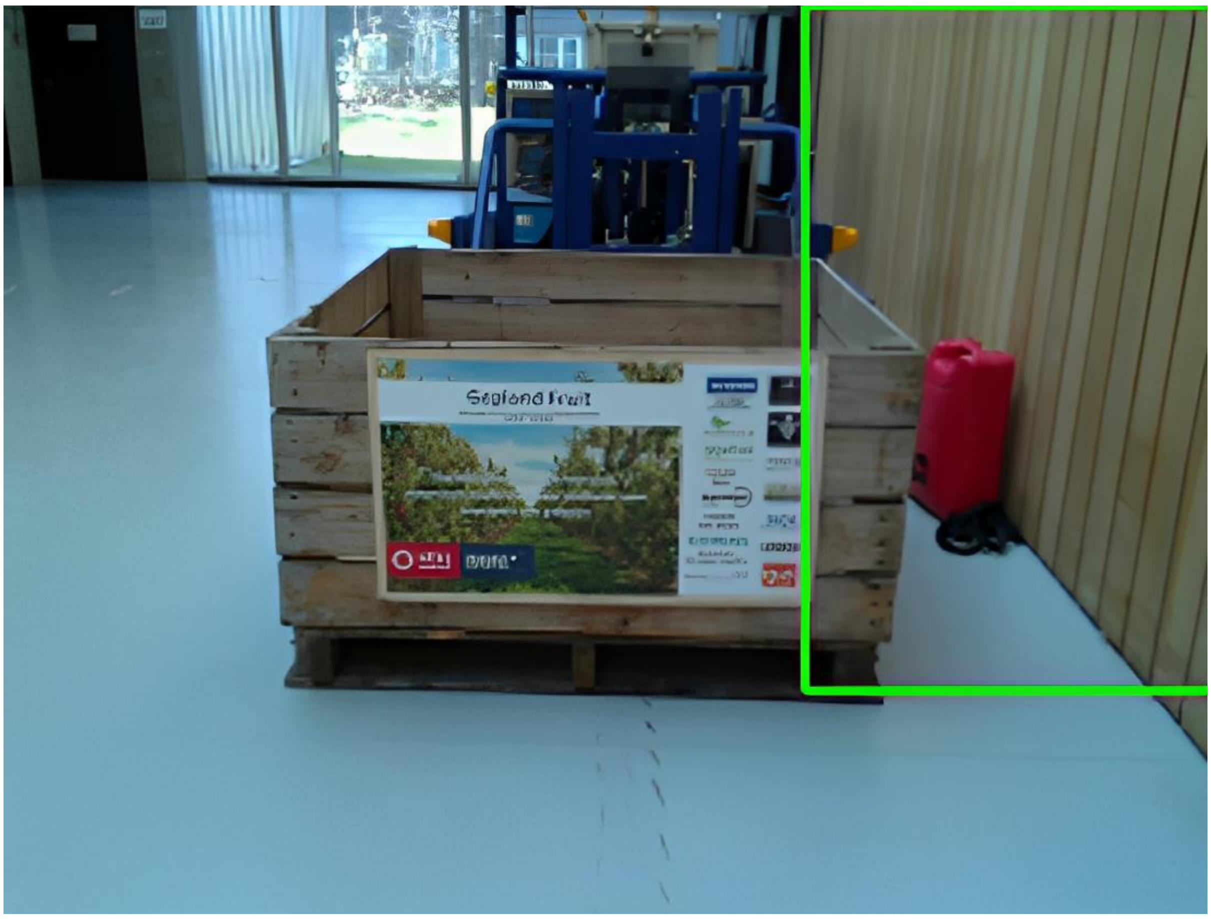


Figure 11: Failure case of using color-based filtering for bounding box extraction

3.3.2 Employing text input prompts for acquiring the object's bounding box

In their paper, SAM also explores the use of text inputs. This approach would enable the program to segment an object immediately based on input such as "wooden box." However, this capability has not been released yet. Therefore, for this paper, a program called Grounding DINO is utilized instead. Grounding DINO requires both an image and text as input. It then calculates several object bounding boxes, each with its own accuracy score. The final output is determined by selecting the bounding box with the highest accuracy score above a certain threshold [67].

Grounding DINO is an object detector specifically trained to handle novel objects by generalizing from the known objects it was trained on. Grounding DINO adopts a dual-encoder-single-decoder architecture. One encoder is responsible for extracting image features, while the other encoder extracts features from the input text.

These features are then passed to the feature enhancer, where cross-modality feature fusion takes place. This process involves combining the extracted features, enabling the algorithm to gain a comprehensive understanding of what to search for in the image.

Subsequently, a query selection module generates queries based on the text prompt. For instance, when the input image contains a pallox and the text prompt is "wooden box," the generated queries may include descriptors like "brown" to correspond with the term "wooden" and "90° edges" to relate to the term "box" within the text prompt. These queries guide the algorithm's attention to relevant features within the image.

The process continues with cross-modality decoding, which involves locating the features corresponding to the generated queries. Finally, the gathered information is utilized to predict the object's location and bounding box [68].

Fig. 12 presents the output of both Grounding DINO and SAM. The output of Grounding DINO includes the object's bounding box. Additionally, it showcases the result of SAM, specifically the generated mask derived from the input provided by Grounding DINO.

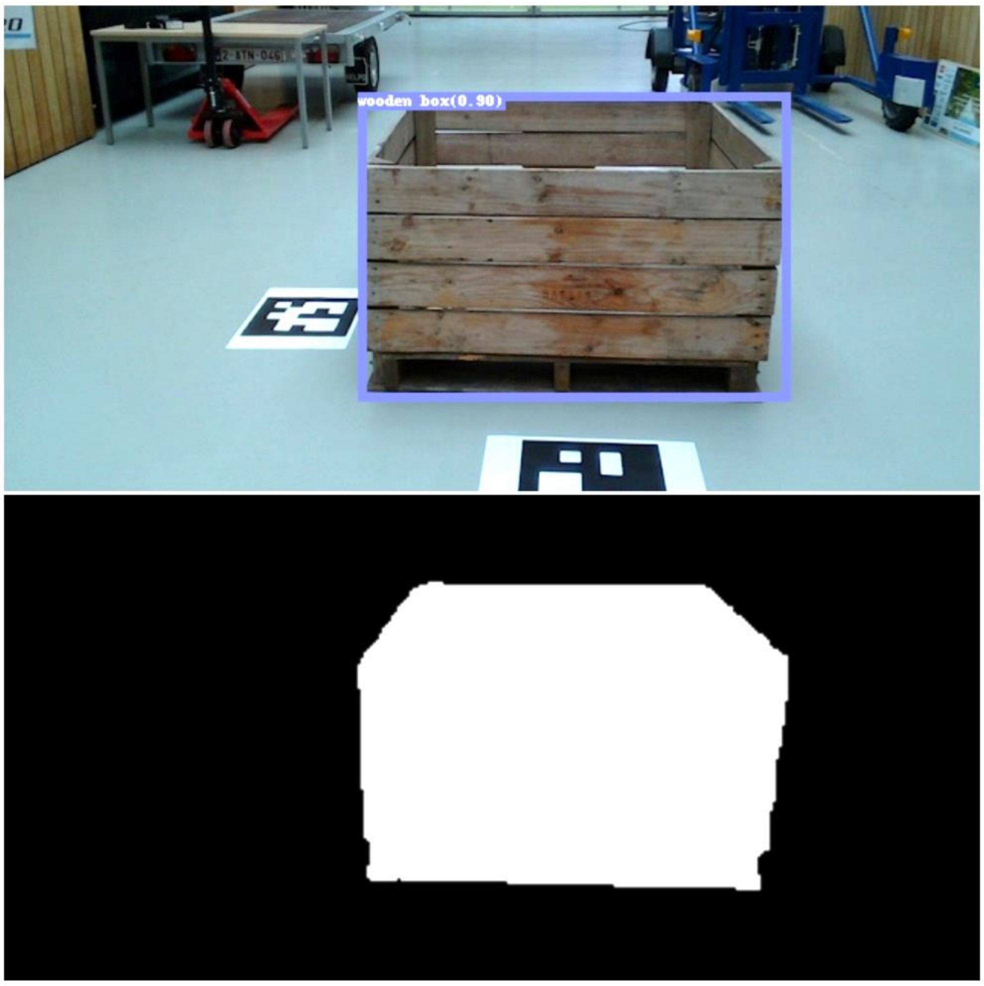


Figure 12: Output of Grounding DINO and SAM

3.4 Customization of BundleTrack

After downloading and installing all the required dependencies mentioned in Sections 3.2 and 3.3, it was crucial to appropriately process the input data to ensure error-free execution of BundleTrack. For this purpose, several Python scripts were developed specifically to preprocess the input data. Each script is accompanied by a requirements.txt file, which specifies the necessary Python version and the required Python libraries along with their corresponding versions.

To ensure that my efforts could be utilized by ACRO, a comprehensive documentation file was also prepared. This document offers detailed explanations of the purpose and usage of each script, along with a recommended sequence for executing them.

Here is a summary of the scripts involved in the data preprocessing stage. These scripts are responsible for preparing the data to be compatible with the BundleTrack algorithm:

- Depth_converter.py: This Python script converts the depth images to the appropriate millimeter scale required by BundleTrack. Since the RealSense L515 camera used in this study produces depth images at a scale of 0.00025 m, this script ensures their conversion to the correct scale.
- Rename_resize_rgb_depth_images.py: This script serves the purpose of renaming the RGB images and depth images while also resizing the RGB images. The resizing is necessary to minimize GPU memory usage during mask calculations by LF-Net. Large image sizes can cause program crashes, so resizing mitigates this issue.
- Inference_on_a_image.py: This script enables the execution of Grounding DINO, allowing for detecting the bounding box of the pallox for the initial image.
- Segmentation.py: This script employs SAM to generate the initial mask required for BundleTrack.

In addition to incorporating the necessary code to enable the functionality of BundleTrack, modifications to the algorithm itself were essential for its successful execution. When applying BundleTrack to the pallox data using the default settings, the algorithm encountered difficulties in accurately tracking the object's pose when the camera movement was relatively fast.

To address this issue, BundleTrack offered a configuration file that allowed users to adjust various algorithmic settings. The configuration file encompassed parameters pertaining to depth images, bundle adjustment, keyframes, the SIFT algorithm, feature correspondences and RANSAC calculations.

To resolve the problem at hand, it was necessary to expand the search space for identifying the nearest neighbor, thereby improving the algorithm's ability to handle faster camera movements.

4 EXPERIMENTAL SETUP AND EVALUATION METRICS

When utilizing BundleTrack to calculate the pose of the wooden box, the algorithm produces the output in the form of text files. This means that in order to accurately assess the performance of BundleTrack, visualizing the results is crucial. This visualization provides valuable insights into how the algorithm performs. However, the paper also included various metrics for a more objective evaluation.

Determining the ground truth is essential for obtaining an objective evaluation. The ground truth poses serve as reference points, allowing for a precise assessment of the algorithm's accuracy. With the ground truth poses established, various metrics can be calculated, including the Area Under Curve (AUC) measured by the Average Distance of Model Points (ADD) score and the Average Distance of Model Points–Symmetry (ADD-S) score. Additionally, the average rotation and position errors are computed, and a graph is generated to visualize these errors per frame. Finally, the time required to compute the pose per frame is recorded for further analysis.

4.1 Camera calibration

To ensure accurate results and evaluate the performance of BundleTrack, it is necessary to calibrate the camera before setting up the experimental setup. The camera used in this study is the Intel RealSense L515, which utilizes LIDAR technology for depth measurements. However, since many applications, including BundleTrack, require depth images rather than point cloud data, typically produced by LIDAR cameras, the RealSense camera automatically converts the acquired data into an RGB stream and depth images.

The RealSense L515 camera is designed for indoor environments and has a limited range of up to 9 meters. It employs a rolling shutter instead of a global shutter, introducing motion blur that can pose challenges for BundleTrack. This camera is suitable for close-range objects and slow movements, making it unsuitable for real-world use cases.

To calibrate the camera, a checkerboard with a known size was created using [69]. Multiple images of the checkerboard were captured at different rotations. Subsequently, the code provided in [70] was utilized to obtain the camera's intrinsic matrix and distortion coefficients.

The camera intrinsics encompass specific parameters such as the focal lengths and the coordinates of the principal point. It is represented by a 3x3 matrix with the following form:

$$\begin{bmatrix} fx & 0 & cx \\ 0 & fy & cy \\ 0 & 0 & 1 \end{bmatrix}$$

Here, fx and fy denote the focal lengths in the x and y directions, while cx and cy represent the pixel coordinates of the principal point. Distortion coefficients are presented as a 1x5 vector with the following general form:

$$[k1 \quad k2 \quad p1 \quad p2 \quad k3]$$

The $k1$,$k2$ and $k3$ parameters define the amount of radial distortion in the captured image, while $p1$ and $p2$ are related to tangential distortion. These parameters are crucial for accurate pose calculation and result visualization. It is important to note that these parameters are specific to the camera being used and not the camera type.

In addition to the camera, the hardware used also influences the speed results of BundleTrack. In this study, the proposed algorithm was tested on an AMD Ryzen 3 3300x CPU and an NVIDIA GeForce GTX 1660 Super.

The object that was used for evaluation was a wooden box with the following dimensions: a width and length of 123 cm, a thickness of 2 cm, a height of 59.5 cm without the lower part and a total height of 74 cm.

4.2 Creating a model of the pallox

As mentioned earlier, BundleTrack does not rely on a model to compute the 6D pose of the object. However, to obtain an objective assessment of BundleTrack's performance, ground truth information is necessary. Various ground truth methods were explored in this study, all of which require a model to function.

To generate a point cloud model of the pallox object, a CAD model is needed. For this paper, FreeCAD was utilized, and the model was exported in STD format. The model is visualized in Fig. 13.

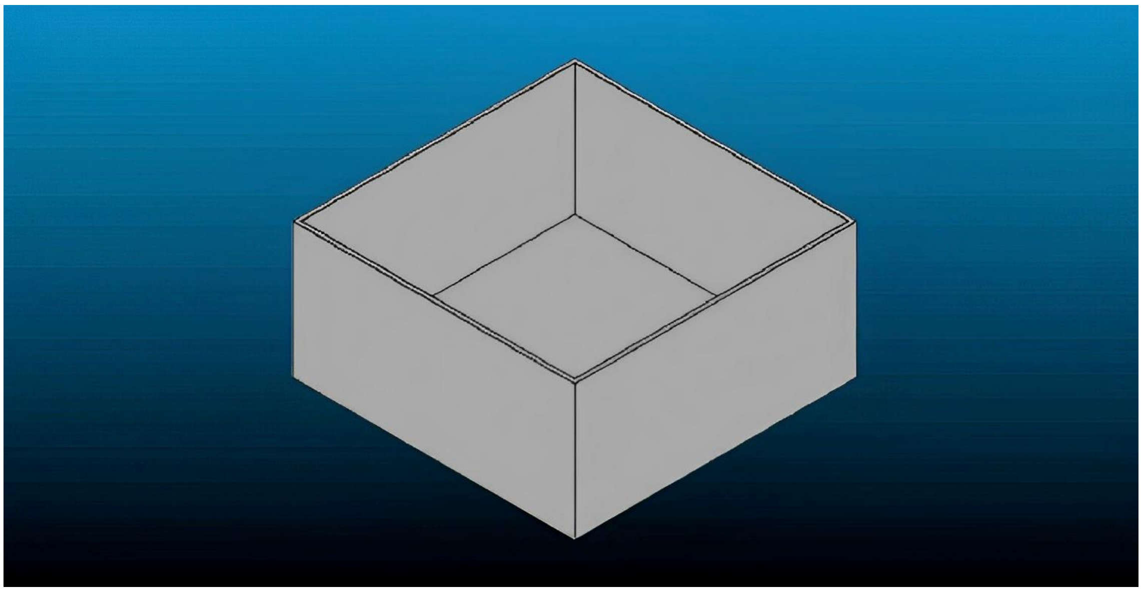


Figure 13: Visualization of model in STD format

To obtain the required ground truth, the model needs to be converted into a point cloud format. For this purpose, a program called CloudCompare is utilized to convert the CAD model from STD format to PLY format. Fig. 14 illustrates the point cloud model of the pallox.

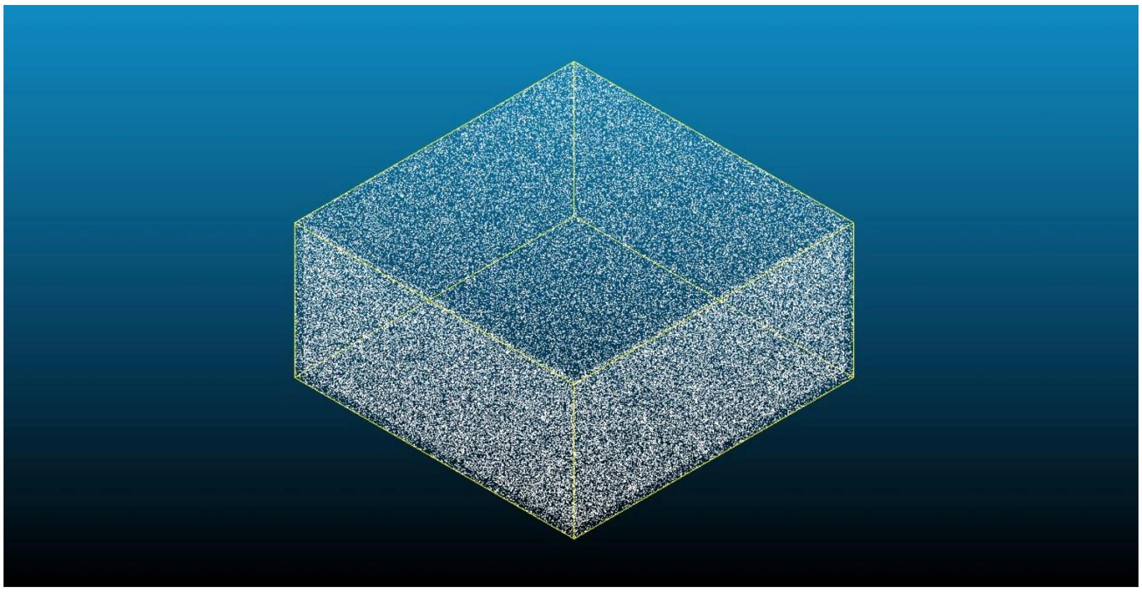


Figure 14: Visualization of model in PLY format

4.3 Obtaining the ground truth pose

4.3.1 Obtaining ground truth pose using 6DposeAnnotator

The ground truth pose is obtained using a program called 6DposeAnnotator by Sakizuki, which serves as an interactive 6D pose annotation tool for point cloud processing [71]. The program allows the user to transform a point cloud model to align it with an RGB image, enabling the extraction of the object's pose for that particular frame.

Upon loading the program, the RGB image, depth image, camera intrinsics, point cloud model, and initial transformation are provided by the user. The depth images are crucial for automatically scaling the point cloud model, based on its position in the image and for facilitating the ICP algorithm, which will be explained later.

Once all the parameters are loaded, the point cloud model is downsampled to reduce its density. The downsampled model is then transformed using the initial transformation, if available. Subsequently, a visualization window is created, displaying the RGB image with the downsampled point cloud model overlaid.

The 6DposeAnnotator awaits user input for further interaction. To perform the translation of the point cloud, the user simply needs to click on the image. The program then retrieves the depth value associated with that position, converting it into a 3D position within the camera coordinate system. This resulting position represents the translation vector. In addition to translation, the user has the capability to rotate the point cloud around the x, y, and z axes. As the user applies rotation, the program updates the rotation matrix accordingly to visualize the new orientation of the point cloud.

Additionally, the program incorporates the iterative closest point (ICP) registration algorithm for pose refinement. ICP is a local registration method used to align two point clouds: the point cloud model and the depth data transformed into a point cloud. It accomplishes this by identifying corresponding points between the two point clouds and estimating a transformation that minimizes the distance between them [28].

Once the user is satisfied with the alignment between the point cloud and the object in the RGB image, the final transformation matrix can be saved. Custom code has been developed to transform this output into the correct format.

However, annotating all frames manually for evaluation purposes can be time-consuming. To address this, modifications have been made to the program, leveraging the principle of spatiotemporal consistency. This means that the object's pose should change gradually over time and follow a realistic trajectory rather than abruptly jumping from one position to another.

As a result, the program has been adjusted to only manually annotate the first pose, while the poses for subsequent frames are automatically refined using ICP on the pose of the previous frame. Fig. 15 shows the first manually annotated pose, as well as the poses for frames 10, 20, and 30, respectively.

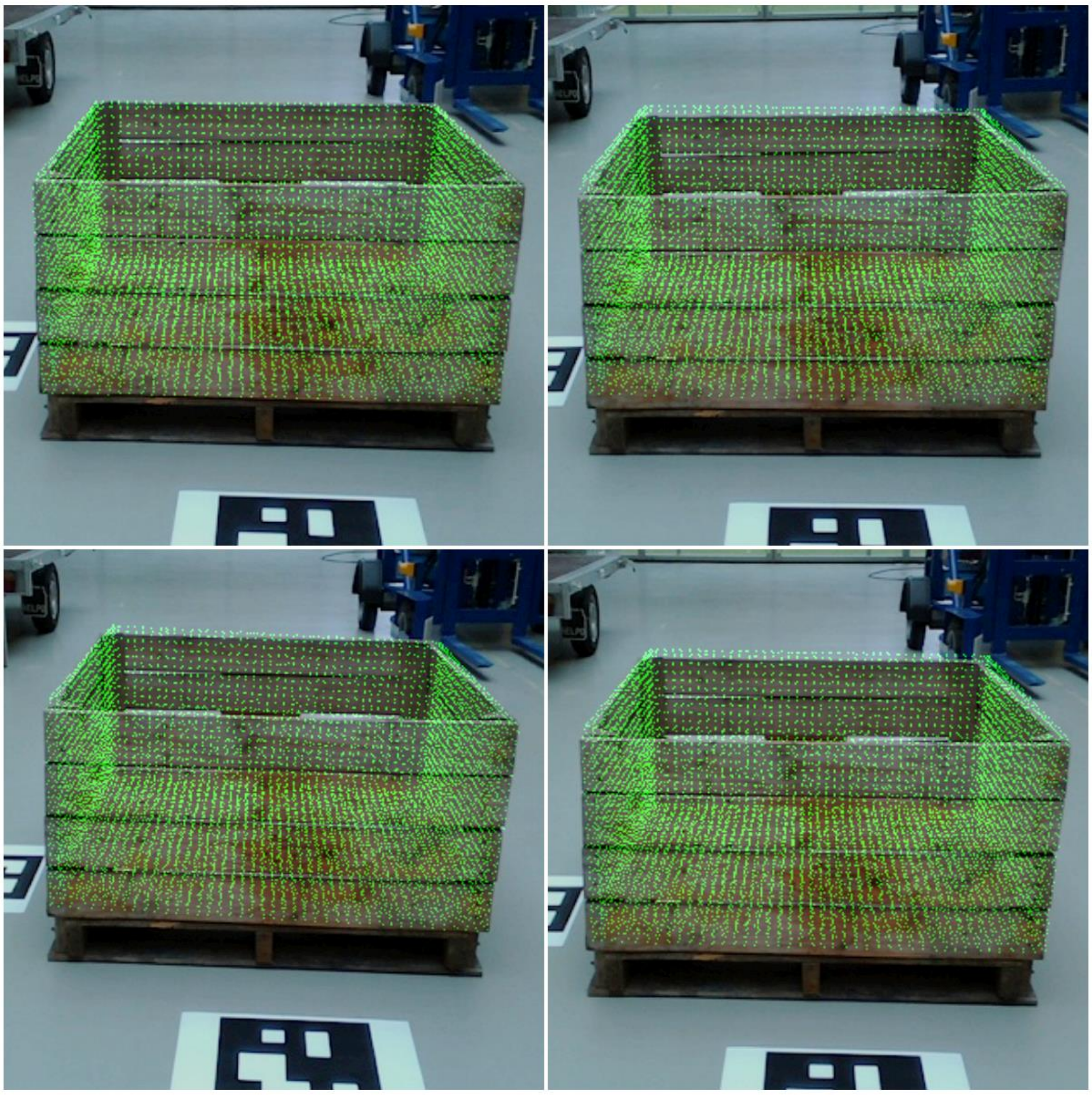


Figure 15: Manually annotated pose for frame 1 and automatically annotated poses for frames 10, 20, and 30 using 6DPoseAnnotator

Fig. 15 demonstrates that manually annotating the pose using 6DPoseAnnotator yields accurate ground truth poses. However, when examining frames that are automatically annotated using ICP (frames 10, 20, and 30), it becomes evident that the ICP calculation is not sufficiently accurate to serve as ground truth.

The reason ICP fails to work effectively in this particular use case is due to the limitations of the RealSense L515 camera, which does not provide depth data with the required level of accuracy for ICP usage. For instance, as depicted in Fig. 16, the RGB image and depth image for frame 10 illustrate that the edges of the wooden box are not distinctly defined in the depth image. When the depth information does not provide distinct object edges, it hinders the accurate identification of points that correspond to the surface of the wooden box.

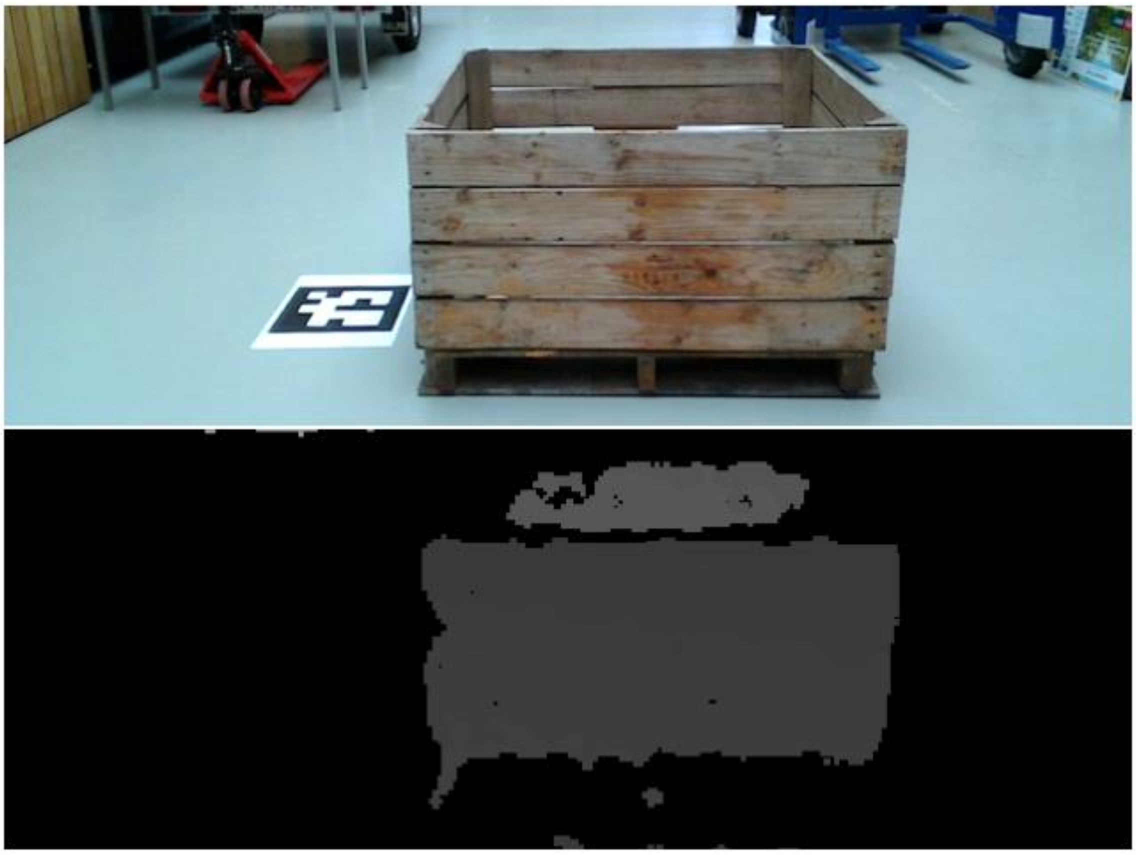


Figure 16: Limitations depth data using L515 camera

4.3.2 Obtaining ground truth pose using markers

An alternative approach for obtaining the ground truth involves the use of markers, specifically ArUco markers in this thesis. ArUco markers consist of black-and-white square grids that encode binary patterns. These markers use a predefined dictionary that specifies the existing patterns and their corresponding IDs. They also come in different sizes, allowing flexibility for different kinds of objects, such as the wooden box.

To calculate the pose of the ArUco markers, a program available on [72] was utilized. This repository contains all the necessary code to generate ArUco tags, detect them within frames, and calculate their poses. The printed ArUco tags were printed on A2-sized paper to ensure sufficient recognition and, hopefully, accuracy.

Regarding marker placement, one option was to position the markers on the pallox itself. However, this approach could potentially impact the performance of BundleTrack, so an alternative was chosen. Instead, the ArUco markers were placed in front of the object, allowing for the determination of their orientations while maintaining the performance of BundleTrack. Fig. 17 illustrates the obtained results for frames 1, 100, 600, and 1100.

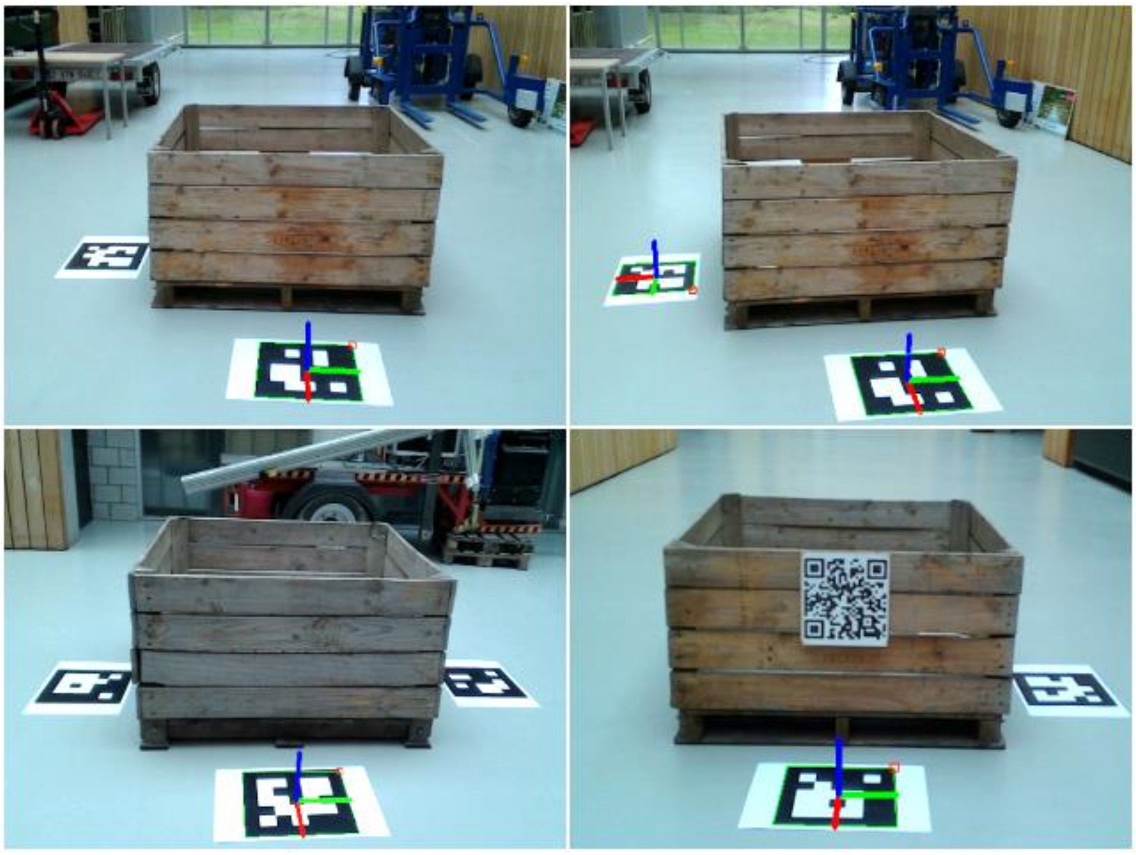


Figure 17: Detection and pose estimation of ArUco markers

The code responsible for calculating the pose of the markers has been modified to extract the pose for each marker individually and convert it to the appropriate format. Additionally, the calculated marker poses need to be transformed into the pose of the wooden box. To achieve this, the pose of the pallox for the first frame was manually determined using 6DposeAnnotator.

By doing this, the necessary information is available to establish the transformation matrix between the wooden box and one of the markers in that frame. With this transformation matrix, the poses of the markers in subsequent frames can be converted to the pose of the wooden box.

For better visualization of the method's output, the bounding box of the pallox has been incorporated. Fig. 18 showcases frames 1, 100, 600, and 1100, along with the wooden box's bounding box, providing a comprehensive view of the results.

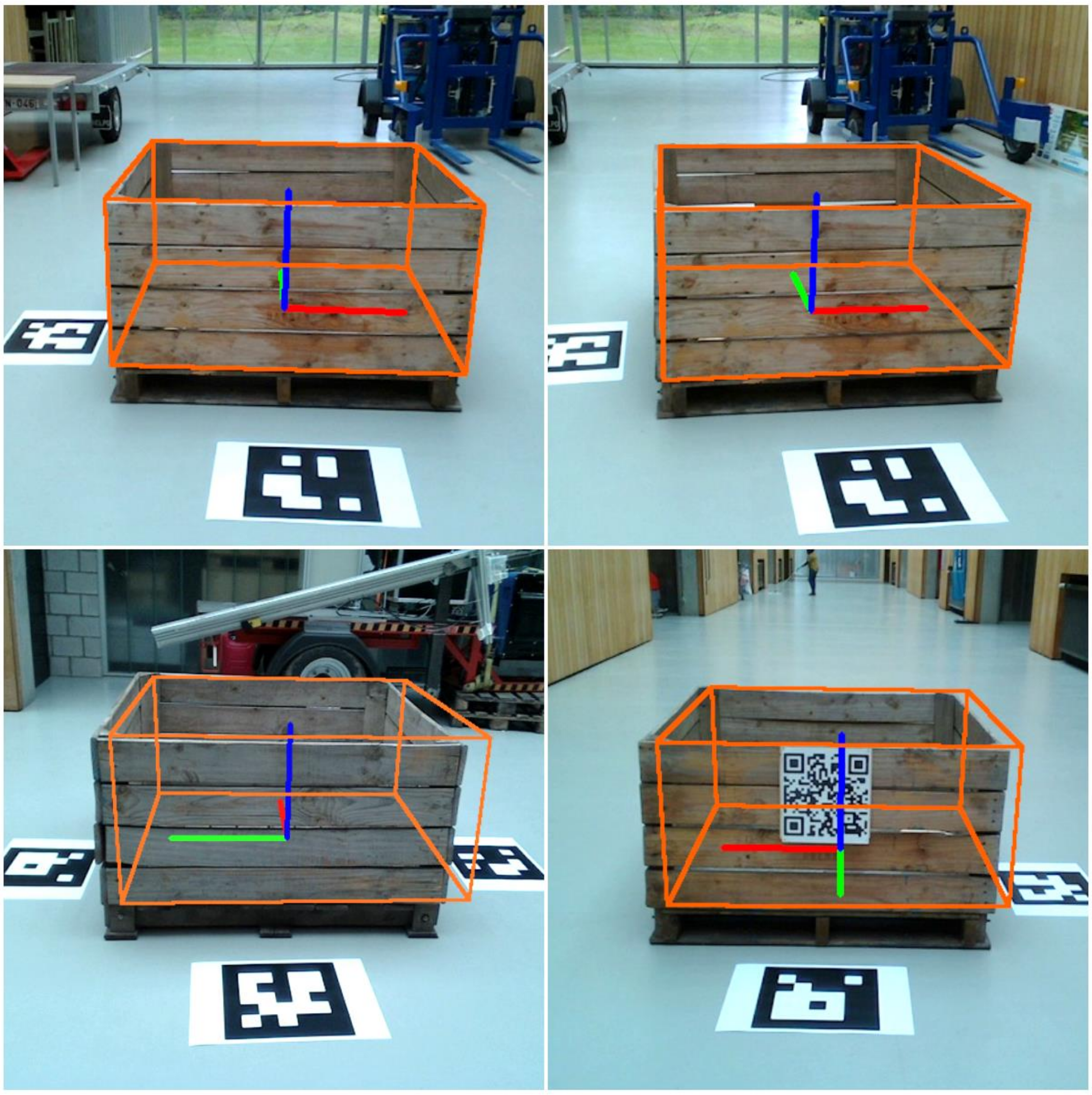


Figure 18: Marker pose converted pallox pose

Initially, the method appears to produce reasonably accurate results for the first few frames. However, subsequent frames reveal that this approach fails to provide poses with the necessary level of accuracy for comparison with BundleTrack. As a result, yet another method had to be employed to obtain poses that met the required accuracy for comparison with BundleTrack's results.

4.3.3 Obtaining ground truth pose manually

Due to the poor accuracy of the previous results, it was necessary to manually calculate the ground truth pose. This task was accomplished using the 6DposeAnnotator tool. However, considering the dataset's size of 1153 images, annotating all frames proved to be inefficient. To ensure a comprehensive representation of the dataset, 50 frames were chosen, evenly distributed across all frames. This selection process guarantees that the annotated images cover a wide range of object angles and perspectives.

Modifications were implemented in the code to enhance the annotation process. Specifically, the code was adjusted to display the object's bounding box instead of the model's point cloud. This modification greatly improved the precision of the annotations, allowing for more accurate ground truth poses. Fig. 19 shows the resulting ground truth poses for frames 1, 368, 759, and 1127.

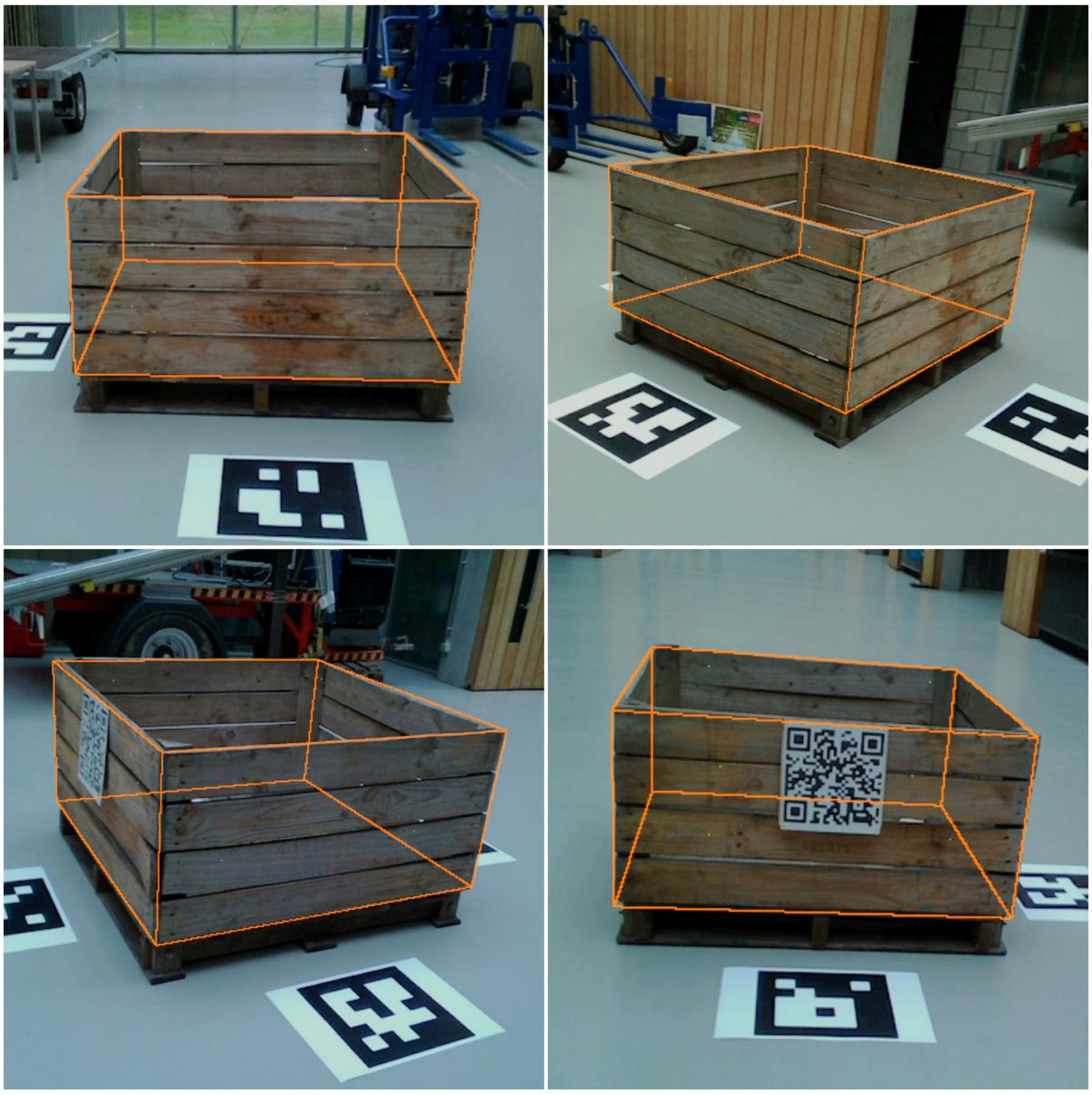


Figure 19: Ground truth pallox poses obtained manually using 6DPoseAnnotator

4.4 Evaluation metrics

The ADD score and ADD-S score have been computed to objectively assess the performance of BundleTrack. The ADD score represents the average distance between the calculated pose and the ground truth pose of an object. This is achieved by transforming the point cloud model of the object using both the calculated 6D pose and the ground truth pose for a single frame. The distance between corresponding points in the two transformed point cloud models is then calculated, resulting in the average distance for that frame. This process is repeated for all frames, and the overall average distance across all frames represents the ADD score. It provides an indication of the overall alignment between the two transformed models.

To address the challenge posed by symmetric objects, the ADD-S score is introduced. Symmetric objects, such as spheres, can lead to inaccurate pose estimation as they appear the same from different perspectives. The ADD-S score tackles this by calculating the distance to the nearest neighbor instead of the corresponding points. This makes it suitable for evaluating the performance of pose estimation methods on symmetric objects, while the ADD score is more appropriate for non-symmetric objects.

The ADD and ADD-S scores are typically represented using the AUC values. AUC quantifies the performance of the pose estimation method by measuring the area under the accuracy-threshold curve. The threshold represents the level of accuracy, and the curve reflects the percentage of frames below that threshold. As the threshold increases, the number of frames with accuracy below the threshold also increases. By calculating the average percentage of frames for different thresholds, the AUC value is obtained and used for comparison.

In addition to the AUC scores, graphs depicting translation error and rotational error are generated to assess BundleTrack's performance on each frame. The translation error is measured by calculating the Euclidean distance between the ground truth translation vector and the calculated translation vector, expressed in meters. As mentioned before, vectors represent points in 3D space, and their Euclidean distance is determined using the formula below:

$$ed = \sqrt{((tc[0] - tg[0])^2 + (tc[1] - tg[1])^2 + (tc[2] - tg[2])^2)}$$

Where ed stands for Euclidean distance, tc stands for translation calculated and tg stands for ground truth translation. To calculate the rotation error, the following formulas have been used:

$$r_{new} = r_{gt.T} * r_{calc}$$

$$r_e = \operatorname{acos}\left(\frac{trace(r_{new}) - 1}{2}\right) * \left(\frac{180}{\pi}\right)$$

In the provided equations, the notation $r_{gt.T}$ denotes the transposed 3x3 ground truth rotation matrix of the object. On the other hand, r_{calc} represents the calculated 3x3 rotation matrix of the object. The matrix r_{new} is obtained by multiplying the aforementioned matrices, representing the difference between the two rotations. Finally, the rotation error in degrees, denoted by r_e, is computed [73].

5 RESULTS

5.1 Results without reinitialization

After recording a video of the wooden box and obtaining ground truth pose data for fifty frames within the video, the results of the BundleTrack program can be analyzed. In the upcoming paragraphs, a comparison will be made between the results of BundleTrack in this dataset and in the dataset of BundleTrack's paper.

The initial focus is on the AUC scores. On the wooden box dataset, BundleTrack achieved an ADD-based AUC of 26.17 and an ADD-S-based AUC of 72.94. In contrast, on the YCBInEOAT dataset, the AUCs based on ADD and ADD-S were 87.34 and 92.53, respectively [3]. These results demonstrate that BundleTrack's performance on the wooden box dataset is significantly lower compared to its performance on the YCBInEOAT dataset.

Furthermore, graphs were generated to depict the rotational error in degrees and the translation error in meters for this particular use case. The x-axis represents the frames, which encompass the selected 50 frames across the entire dataset. Fig. 20 illustrates those errors.

The graphs indicate that the algorithm performs relatively well for the first fifth of the dataset, as the translation error remains below 5 cm and the rotational error remains close to 2°. However, starting from frame 10, the translation error gradually increases. This can be attributed to frames where only the wooden box's edge is visible, leading to lower-quality depth data and hampering the algorithm's accurate pose tracking.

Around frame 34 (approximately), there is another notable increase in rotational and translational errors, coinciding with another edge in the images. Overall, BundleTrack demonstrates satisfactory performance when the front of the wooden box is visible, maintaining relatively stable rotational and translational errors.

However, when the camera rotates around an edge of the wooden box, the algorithm struggles to precisely track the object's pose.

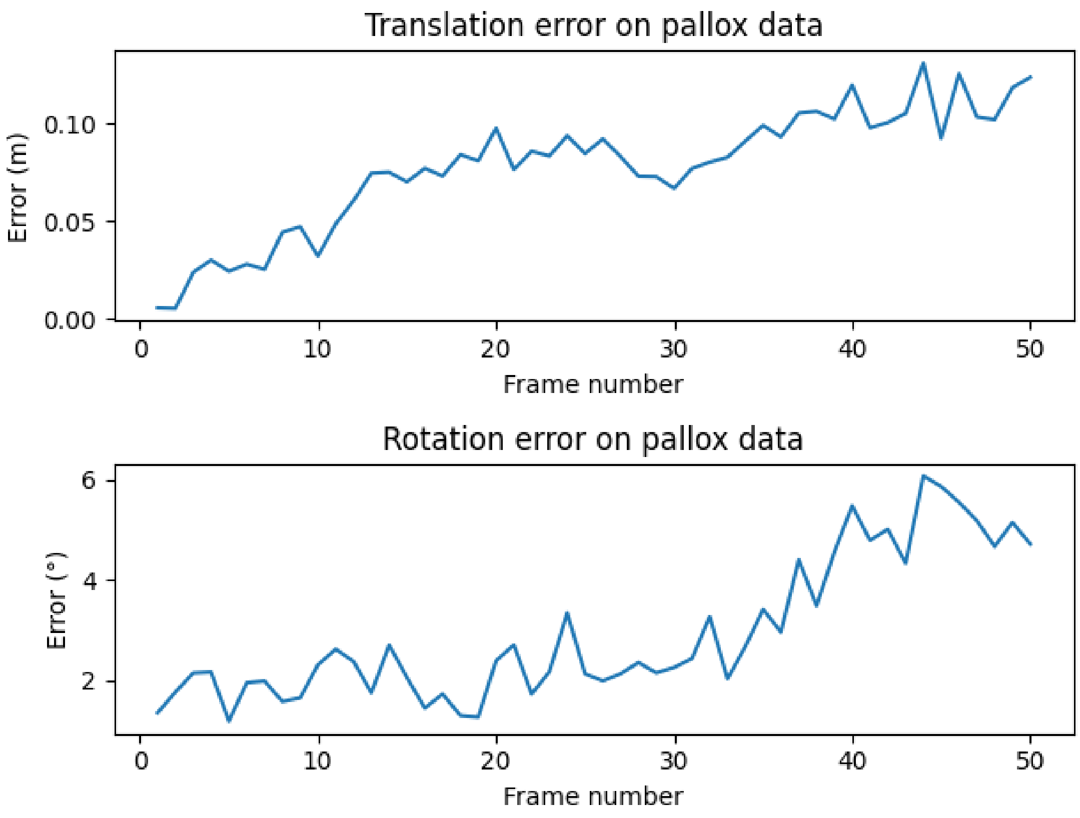


Figure 20: Translation and rotation error on pallox data

The average translation error of BundleTrack on the pallox data is 7.74 cm, and the average rotational error is 2.93°. In comparison, when using BundleTrack on the NOCS dataset, the average translation and rotational errors are 2.1 cm and 2.4°, respectively.

To gain an initial understanding of BundleTrack's performance, a visual representation can be observed in Fig. 21. This image indicates that BundleTrack initially tracks the pose of the wooden box quite accurately. However, as the frames progress, a noticeable drift in the estimated pose becomes apparent. This drift primarily occurs when the camera captures the edges of the wooden box. The BundleTrack algorithm encounters difficulty finding sufficient feature correspondences to precisely estimate the pose. This limitation arises from the depth data's inadequate quality, which only provides depth information for a small portion of the wooden box.

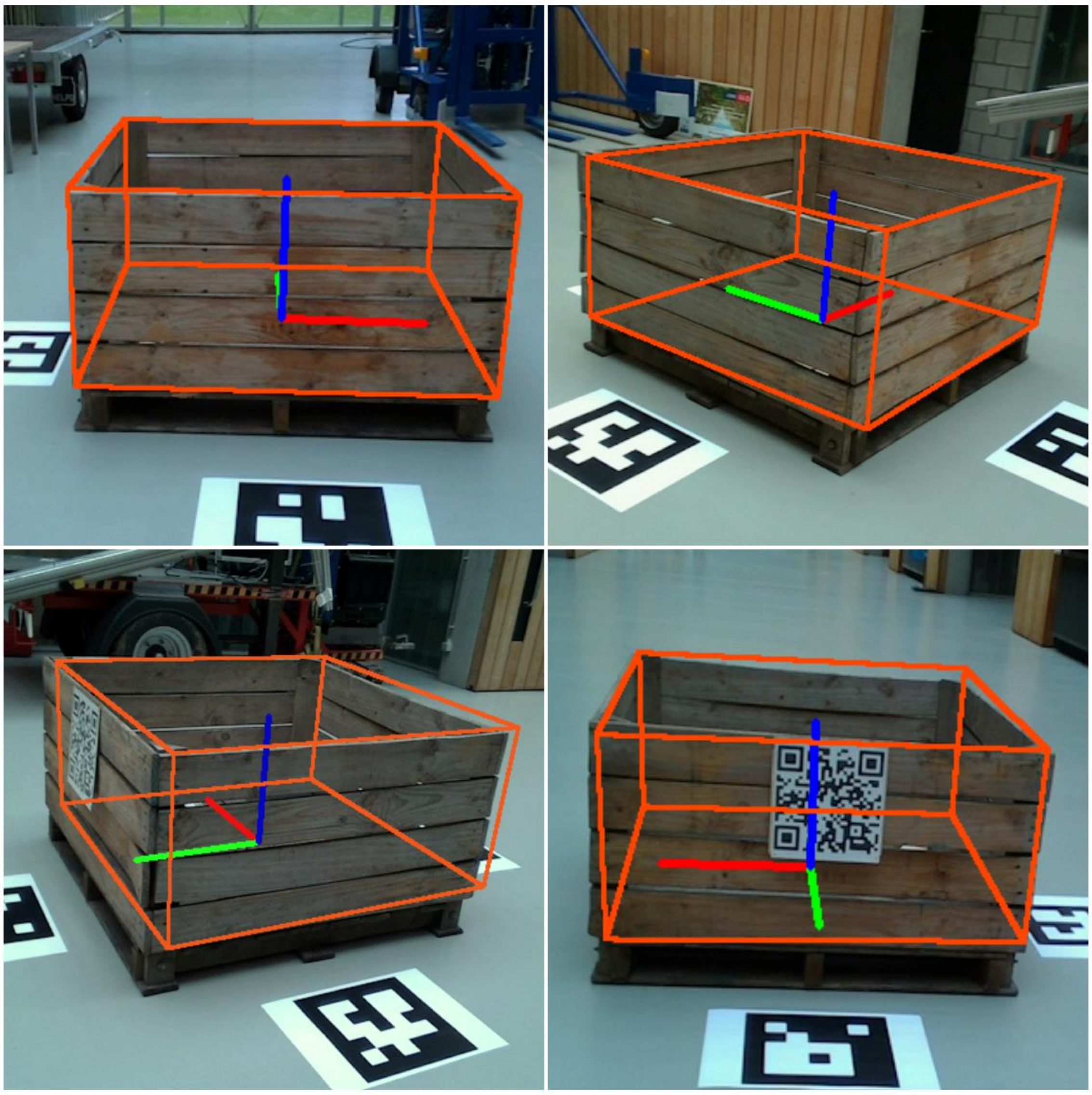


Figure 21: Output of BundleTrack algorithm on pallox data

To enhance BundleTrack's performance for this specific use case, two potential options exist. The first option involves using a camera that offers improved depth imaging of the object during camera rotations. With a broader coverage of surface depth data, the algorithm can identify more object features, thereby reducing drift. Alternatively, reinitializing BundleTrack at regular intervals using a tracking by detection method can be considered. Since a tracking by detection method is already necessary for the first frame, it can be employed periodically to maintain high speed while minimizing drift.

5.2 Results with reinitialization

To simulate the effects of reinitialization using a tracking by detection method, the pose is reset to the ground truth pose every 100 frames. The results of this simulation can be observed in Fig. 22 and Fig. 23.

Notably, the performance of BundleTrack significantly improves, with the translation error remaining below 5 cm and the rotation error hovering around 2°. The average translation error reduces to 1.83 cm, and the average rotation error decreases to 0.93°.

It is important to note that these results represent an ideal scenario with a perfect tracking by detection algorithm, which is not achievable in real-world use cases. Additionally, implementing a tracking by detection algorithm will impact the speed at which the solution operates. One suitable method for tracking by detection is the 3D-model-based approach called Point Pair Feature Matching (PPFM), which eliminates the need for an initial transformation or training. However, it does require a model [74].

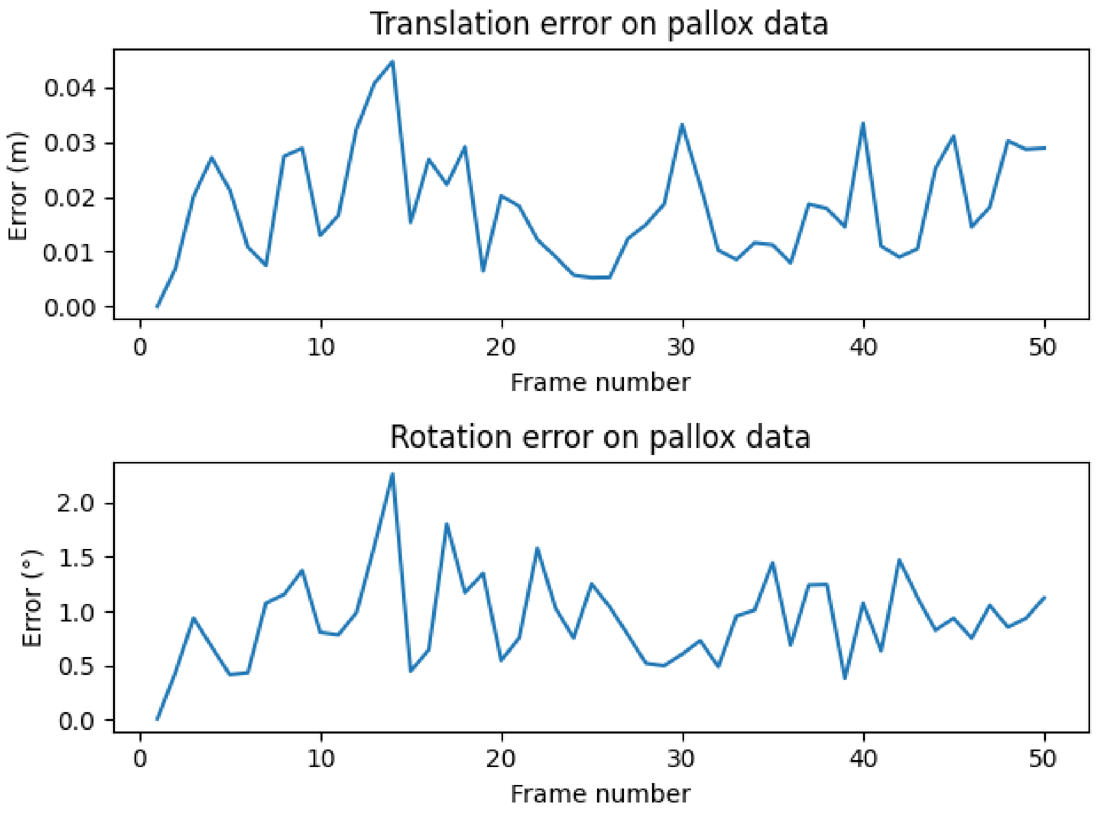


Figure 22: Translation and rotation error on pallox data with reinitialization

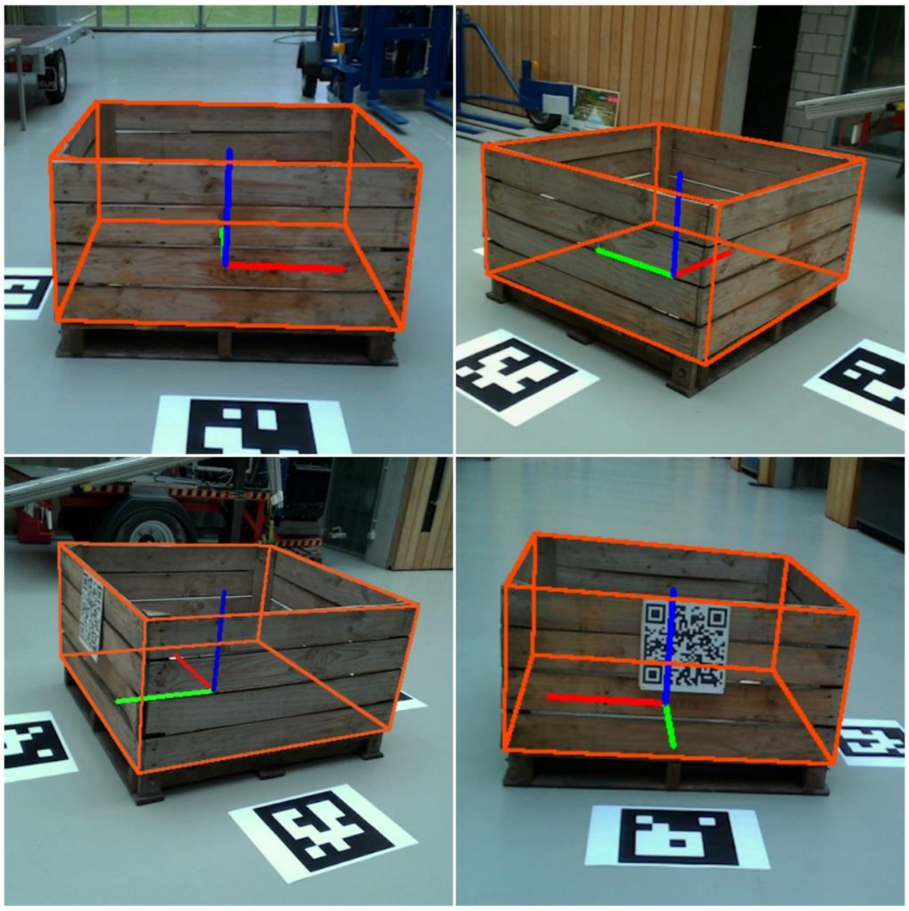


Figure 23: Output of BundleTrack algorithm on pallox data with reinitialization

In terms of speed, BundleTrack currently runs at a frequency of 6 Hz, excluding mask generation. Although this speed is adequate for real-time applications, there is potential for further enhancements by leveraging a more powerful GPU. In the original BundleTrack paper, the algorithm achieved a total speed of 10 Hz utilizing a single NVIDIA RTX 2080 Ti GPU, which possesses considerably greater computing power compared to the GPU employed in this study. Additionally, it is worth noting that the algorithm's speed has not been compromised despite making specific adjustments to accommodate my particular use case. For instance, enlarging the search area to identify corresponding features has no impact on the algorithm's overall speed.

6 CONCLUSION AND FUTURE WORK

The primary objective of this master's thesis was to achieve real-time tracking of the 6D pose of a pallox without relying on a CAD model. To accomplish this, a temporal tracking method called BundleTrack was utilized, along with custom code for data preprocessing and the combined use of Grounding DINO and SAM for initial mask extraction. Various approaches were explored to obtain accurate ground truth data for evaluating the performance of the BundleTrack algorithm.

The algorithm demonstrates an average position error of 7.74 cm and an average rotation error of 2.93°. The AUC values measured by ADD and ADD-S are 26.17 and 72.94, respectively. With the GTX 1660S GPU, the program achieves a real-time speed of 6 Hz. These results indicate that the algorithm has potential for practical applications.

Future enhancements will involve employing a different camera capable of capturing depth images with higher quality than the LIDAR L515 camera, aiming to improve the algorithm's accuracy for this specific use case. Another approach for accuracy improvement will be the implementation of a tracking by detection method at regular intervals to reset the object's pose and mitigate drift, as demonstrated in the obtained results. Finally, further optimization will be required for the proposed method to achieve full autonomy, as certain steps still necessitate manual intervention.

REFERENCES

[1] "wenbowen123/BundleTrack: [IROS 2021] BundleTrack: 6D Pose Tracking for Novel Objects without Instance or Category-Level 3D Models." https://github.com/wenbowen123/BundleTrack (accessed Jun. 03, 2023).

[2] "Real-time tracking and pose estimation for industrial objects using geometric features | IEEE Conference Publication | IEEE Xplore." https://ieeexplore.ieee.org/abstract/document/1242127?casa_token=1gVIS3YEyOoAAAAA:KP1sAlm-ziZzzFBSnH7E5_uMF1kP_Z6Vr9DEP-pHUqKsS2ykAbGk0OOrXD3Ta4IZ5_kPZccEDgnqW9A (accessed Jun. 07, 2023).

[3] B. Wen and K. Bekris, "BundleTrack: 6D Pose Tracking for Novel Objects without Instance or Category-Level 3D Models," p. 2021.

[4] Z. He, W. Feng, X. Zhao, and Y. Lv, "6D Pose Estimation of Objects: Recent Technologies and Challenges," 2020, doi: 10.3390/app11010228.

[5] A. D. Dongare, R. R. Kharde, and A. D. Kachare, "Introduction to Artificial Neural Network," *Certified International Journal of Engineering and Innovative Technology (IJEIT)*, vol. 9001, no. 1, pp. 2277–3754, 2008.

[6] D. Pascual-Hernández, N. Oyaga De Frutos, I. Mora-Jiménez, and J. María Cañas-Plaza, "Efficient 3D human pose estimation from RGBD sensors ☆," 2022, doi: 10.1016/j.displa.2022.102225.

[7] R. Bashirov *et al.*, "Real-time RGBD-based Extended Body Pose Estimation".

[8] "What is a Neural Network? | TIBCO Software." https://www.tibco.com/reference-center/what-is-a-neural-network (accessed Mar. 31, 2023).

[9] K. O'shea and R. Nash, "An Introduction to Convolutional Neural Networks".

[10] "Deep Learning A-Z™ 2023: Neural Networks, AI & ChatGPT Bonus | Udemy." https://www.udemy.com/course/deeplearning/learn/lecture/6761108?start=0#overview (accessed May 19, 2023).

[11] Z. Li *et al.*, "Learning the Depths of Moving People by Watching Frozen People".

[12] G. Gao, M. Lauri, X. Hu, J. Zhang, and S. Frintrop, "CloudAAE: Learning 6D Object Pose Regression with On-line Data Synthesis on Point Clouds", Accessed: May 30, 2023. [Online]. Available: https://github.com/GeeeG/CloudAAE.

[13] H. Cao, L. Dirnberger, D. Bernardini, C. Piazza, and M. Caccamo, "6IMPOSE: Bridging the Reality Gap in 6D Pose Estimation for Robotic Grasping", Accessed: May 30, 2023. [Online]. Available: https://youtu.

[14] 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".

[15] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation", Accessed: Apr. 03, 2023. [Online]. Available: https://zju3dv.github.io/pvnet/.

[16] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes", Accessed: Apr. 03, 2023. [Online]. Available: https://rse-lab.cs.washington.edu/projects/posecnn/.

[17] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "SSD-6D: Making RGB-Based 3D Detection and 6D Pose Estimation Great Again", Accessed: Apr. 03, 2023. [Online]. Available: https://wadimkehl.github.io/

[18] A. Kendall, M. Grimes, and R. Cipolla, "PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization".

[19] Y. Labbé, J. Carpentier, M. Aubry, and J. Sivic, "CosyPose: Consistent multi-view multi-object 6D pose estimation", Accessed: Apr. 03, 2023. [Online]. Available: https://www.di.ens.fr/willow/research/cosypose/

[20] W. Chen, X. Jia, H. J. Chang, J. Duan, and A. Leonardis, "G2L-Net: Global to Local Network for Real-time 6D Pose Estimation with Embedding Vector Features", Accessed: Apr. 03, 2023. [Online]. Available: https://github.com/DC1991/G2L_Net.

[21] C. Wang *et al.*, "DenseFusion: 6D Object Pose Estimation by Iterative Dense Fusion", Accessed: Apr. 03, 2023. [Online]. Available: https://sites.google.com/view/densefusion/.

[22] A. Khosla, A. Torralba, and J. J. Lim, "FPM: Fine Pose Parts-Based Model with 3D CAD Models," *Lecture Notes in Computer Science*, pp. 478–493, 2014, doi: 10.1007/978-3-319-10599-4_31.

[23] J. H. White and R. W. Beard, "An Iterative Pose Estimation Algorithm Based on Epipolar Geometry With Application to Multi-Target Tracking", doi: 10.1109/JAS.2020.1003222.

[24] J. Yang, W. Liang, and Y. Jia, "Face Pose Estimation with Combined 2D and 3D HOG Features", Accessed: Apr. 03, 2023. [Online]. Available: www.baidu.com

[25] "A precision pose measurement technique based on multi-cooperative logo", doi: 10.1088/1742-6596/1607/1/012047.

[26] Y. Konishi, K. Hattori, and M. Hashimoto, "Real-Time 6D Object Pose Estimation on CPU," 2019, Accessed: Apr. 03, 2023. [Online]. Available: http://isl.sist.chukyo-u.ac.jp/archives4/

[27] B. Han *et al.*, "Point Cloud Registration Method for Pipeline Workpieces Based on RANSAC and Improved ICP Algorithms," *IOP Conf Ser Mater Sci Eng*, vol. 612, no. 3, p. 032190, Oct. 2019, doi: 10.1088/1757-899X/612/3/032190.

[28] "Iterative closest point - Wikipedia." https://en.wikipedia.org/wiki/Iterative_closest_point (accessed May 31, 2023).

[29] R. Ge and G. Loianno, “VIPose: Real-time Visual-Inertial 6D Object Pose Tracking”.

[30] H. Uchiyama and E. Marchand, “Object Detection and Pose Tracking for Augmented Reality: Recent Approaches”.

[31] L. Huang, G. N. Mckay, N. J. Durr, and G. N. Mckay, “A Deep Learning Bidirectional Temporal Tracking Algorithm for Automated Blood Cell Counting from Non-invasive Capillaroscopy Videos”.

[32] E. Mendes, P. Koch, and S. Lacroix, “ICP-based pose-graph SLAM,” *SSRR 2016 - International Symposium on Safety, Security and Rescue Robotics*, pp. 195–200, Dec. 2016, doi: 10.1109/SSRR.2016.7784298.

[33] D. J. Fleet and Y. Weiss, “Optical Flow Estimation”.

[34] F. A. Moreno, J. L. Blanco, J. González-Jiménez, F. A. Moreno, J. L. Blanco, and J. González, “An Efficient Closed-Form Solution to Probabilistic 6D Visual Odometry for a Stereo Camera,” 2007, doi: 10.1007/978-3-540-74607-2_85.

[35] R. Hambali, D. Legono, and R. Jayadi, “Jurnal Teknologi Full Paper THE APPLICATION OF PYRAMID LUCAS-KANADE OPTICAL FLOW METHOD FOR TRACKING RAIN MOTION USING HIGH-RESOLUTION RADAR IMAGES,” 2021, doi: 10.11113/jurnalteknologi.v83.14494.

[36] H. Yang, F. Wang, Z. Li, and H. Dong, “Simultaneous Pose and Correspondence Estimation Based on Genetic Algorithm,” 2015, doi: 10.1155/2015/828241.

[37] P. Padeleris, X. Zabulis, and A. A. Argyros, “Head pose estimation on depth data based on Particle Swarm Optimization,” *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 42–49, 2012, doi: 10.1109/CVPRW.2012.6239236.

[38] R. Ge and G. Loianno, “VIPose: Real-time Visual-Inertial 6D Object Pose Tracking,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 4597–4603, Jul. 2021, doi: 10.1109/IROS51168.2021.9636283.

[39] B. Wen and K. Bekris, “CVPR 2021 Workshop on 3D Vision and Robotics Data-driven 6D Pose Tracking by Calibrating Image Residuals in Synthetic Domains Chaitanya Mitash Amazon”, Accessed: May 31, 2023. [Online]. Available: https://github.com/wenbowen123/iros20-6d-pose-tracking

[40] E. Mendes, P. Koch, and S. Lacroix, “ICP-based pose-graph SLAM,” *SSRR 2016 - International Symposium on Safety, Security and Rescue Robotics*, pp. 195–200, Dec. 2016, doi: 10.1109/SSRR.2016.7784298.

[41] A. Dubey, “Stereo vision-Facing the challenges and seeing the opportunities for ADAS applications”.

[42] “Review on LiDAR technology Srushti Neoge · Ninad Mehendale”, Accessed: Jun. 12, 2023. [Online]. Available: https://ssrn.com/abstract=3604309

[43] "A Closer Look at LiDAR and Stereovision - Edge AI and Vision Alliance." https://www.edge-ai-vision.com/2021/07/a-closer-look-at-lidar-and-stereo-vision/ (accessed Jun. 07, 2023).
[44] "A closer look at LiDAR and stereovision. | Ambarella." https://www.ambarella.com/blog/a-closer-look-at-lidar-and-stereovision/ (accessed Jun. 07, 2023).
[45] "Livox LiDAR Sensors - For massive industrial applications." https://www.livoxtech.com/ (accessed Jun. 07, 2023).
[46] "The all-new REV7 sensors powered by the L3 Chip | Ouster." https://ouster.com/products/rev7/ (accessed Jun. 07, 2023).
[47] "Envision the Future | Velodyne Lidar." https://velodynelidar.com/ (accessed Jun. 07, 2023).
[48] "Stereo Imaging Systems | Teledyne FLIR." https://www.flir.eu/support/browse/oem-cameras-components-and-lasers/stereo-imaging-systems (accessed Jun. 07, 2023).
[49] "Stereo Depth Solutions from Intel RealSense." https://www.intelrealsense.com/stereo-depth/ (accessed Jun. 07, 2023).
[50] "Shop the ZED Store | Stereolabs." https://store.stereolabs.com/en-eu?_gl=1*bmpw1u*_ga*MTc0OTg0OTc2LjE2ODYxNjcyNTA.*_ga_LQLT-WBS792*MTY4NjE2NzI1MC4xLjAuMTY4NjE2NzI1MC42MC4wLjA.&_ga=2.101487428.491189895.1686167251-174984976.1686167250 (accessed Jun. 07, 2023).
[51] "LiDAR Camera L515 – Intel® RealSense™ Depth and Tracking Cameras." https://www.intelrealsense.com/lidar-camera-l515/ (accessed Jun. 12, 2023).
[52] "Score-based mask edge improvement of Mask-RCNN for segmentation of fruit and vegetables - ScienceDirect." https://www.sciencedirect.com/science/article/pii/S0957417421015190?casa_token=-94GhTjPoe-QAAAAA:xmG1CFYU6tJFH14nO7KktiIUXtZCFCERGnXtJENmZKLw6LU-aCuLR9rlH1x6keEHnoprwLZdszg4 (accessed Jun. 06, 2023).
[53] Y. Zhang, Z. Wu, H. Peng, and S. Lin, "A Transductive Approach for Video Object Segmentation." pp. 6949–6958, 2020. Accessed: May 31, 2023. [Online]. Available: https://github.com/
[54] "BundleTrack/SIFTImageManager.cpp at master · wenbowen123/BundleTrack · GitHub." https://github.com/wenbowen123/BundleTrack/blob/master/src/cuda/SIFTImageManager.cpp (accessed Jun. 11, 2023).
[55] Y. Ono, E. Trulls, P. Fua, and K. Moo Yi, "LF-Net: Learning Local Features from Images".
[56] "Distinctive Image Features from Scale-Invariant Keypoints | SpringerLink." https://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94 (accessed Jun. 06, 2023).

[57] “K-nearest neighbor - Scholarpedia.” http://scholarpedia.org/article/K-nearest_neighbor (accessed Jun. 07, 2023).
[58] K. G. Derpanis, “Overview of the RANSAC Algorithm,” 2010.
[59] “Understanding NVIDIA CUDA: The Basics of GPU Parallel Computing.” https://www.turing.com/kb/understanding-nvidia-cuda#what-is-cuda? (accessed May 19, 2023).
[60] “CUDA GPUs - Compute Capability | NVIDIA Developer.” https://developer.nvidia.com/cuda-gpus (accessed May 19, 2023).
[61] “Are Docker images OS dependent? - Quora.” https://www.quora.com/Are-Docker-images-OS-dependent (accessed Jun. 07, 2023).
[62] “Docker overview | Docker Documentation.” https://docs.docker.com/get-started/overview/ (accessed Jun. 07, 2023).
[63] “Install WSL | Microsoft Learn.” https://learn.microsoft.com/en-us/windows/wsl/install (accessed Jun. 07, 2023).
[64] “Installation Guide — NVIDIA Cloud Native Technologies documentation.” https://docs.nvidia.com/datacenter/cloud-native/container-toolkit/install-guide.html (accessed Jun. 07, 2023).
[65] “Segment Anything - Wiki | Golden.” https://golden.com/wiki/Segment_Anything-6AY9VXG (accessed May 20, 2023).
[66] “Segment Anything | Meta AI.” https://segment-anything.com/ (accessed Jun. 07, 2023).
[67] “IDEA-Research/GroundingDINO: The official implementation of ‘Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection.’” https://github.com/IDEA-Research/GroundingDINO (accessed Jun. 07, 2023).
[68] S. Liu *et al.*, “Grounding DINO: Marrying DINO with Grounded Pre-Training for Open-Set Object Detection”.
[69] “Camera Calibration Pattern Generator – calib.io.” https://calib.io/pages/camera-calibration-pattern-generator (accessed May 24, 2023).
[70] “Camera Calibration with Python - OpenCV - GeeksforGeeks.” https://www.geeksforgeeks.org/camera-calibration-with-python-opencv/ (accessed May 24, 2023).
[71] “sakizuki/6DPoseAnnotator: An interactive 6 degree-of-freedom (DoF) pose annotation tool using point cloud processings.” https://github.com/sakizuki/6DPoseAnnotator (accessed May 26, 2023).
[72] “GSNCodes/ArUCo-Markers-Pose-Estimation-Generation-Python: Estimating pose using ArUCo Markers.” https://github.com/GSNCodes/ArUCo-Markers-Pose-Estimation-Generation-Python (accessed Jun. 03, 2023).

[73] "linear algebra - Comparing two rotation matrices - Mathematics Stack Exchange." https://math.stackexchange.com/questions/2113634/comparing-two-rotation-matrices (accessed Jun. 09, 2023).

[74] "Point Pair Feature Matching: Evaluating Methods to Detect Simple Shapes | SpringerLink." https://link.springer.com/chapter/10.1007/978-3-030-34995-0_40 (accessed Jun. 09, 2023).