Coded Beacons for Localization, Object Tracking and SLAM Augmentation

Roman Rabinovich
Coded Beacons for Localization, Object Tracking and SLAM Augmentation

Research Thesis

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Autonomous Systems and Robotics

Roman Rabinovich

Submitted to the Senate of the Technion — Israel Institute of Technology
IYAR 5780 Haifa May 2020
This research was carried out under the supervision of Prof. Ron Kimmel, in the The Technion Autonomous Systems Program.

Acknowledgements

I would like to thank my advisor Professor Ron Kimmel, my colleague Ibraim Jubran and my family Ortal Ethan and Lucy.

The generous financial help of the Technion is gratefully acknowledged.
Contents

List of Figures

Abstract 1

1 Introduction 3
  1.1 Tracking methods overview  3

2 Coded beacons 11
  2.1 Related work  11
  2.2 Our contribution and novelty  12
  2.3 Coded light codes  13
    2.3.1 Transmitter properties  13
    2.3.2 Robust cyclic binary codes for unique identity detection  13
  2.4 Positional Tracking System  17
    2.4.1 System overview  19
    2.4.2 Bit value extraction  20
    2.4.3 Design considerations  21
    2.4.4 System implementation  22
  2.5 Real-World Experiments  25
    2.5.1 Accuracy  25
    2.5.2 Handling fast moving objects and occlusions  28
    2.5.3 Challenging environments  30
    2.5.4 Potential Use-Cases  30

3 Conclusion and future work 33
  3.1 Future Work  33

Hebrew Abstract i
List of Figures

1.1 Illustration of an IMU with three accelerometers and three gyroscopes measuring 6-DoF relative to the initial position and orientation ............................................................... 4
1.2 Magnetic tracking ........................................................................................................... 5
1.3 Oculus tracking system .................................................................................................. 7
1.4 HTC tracking system ..................................................................................................... 8
1.5 Robotic tracking with the HTC system .......................................................................... 9
1.6 External tracking systems ............................................................................................. 10

2.2 Beacon deployment. (Left) Internal tracking: beacons are deployed around an office environment. (Right) External tracking: beacons are deployed on a toy car ........................................... 19
2.3 .................................................................................................................................... 23
2.4 System components. (Left) our homemade CoBe (coded beacons) consisting of a microcontroller, a LiPo battery, and an IR LED. (Right) An IoT board and a cheap webcam; see Section 2.4.4. 23
2.5 beacon detection and decoding: (a) image received from the sensor, (b) beacons are detected and tracked (c) beacons are decoded after n consecutive frames .................................................. 25
2.6 Systems used as ground truth in our accuracy testing experiments. (Left) Two axes rail system. (Right) A servo ......................................................... 26
2.7 Rail test accuracy .......................................................................................................... 27
2.8 Rails test cumulative position error compared to the ground truth, from the beginning of the test till each frame. Using our markers, the total sum of errors throughout the entire experiment was roughly 0.6 · 10^4, as compared to 3.5 · 10^4 when using the chessboard pattern .................................................. 28
2.9 Servo test orientation accuracy and stability ................................................................. 29
2.10 External tracking of a fast toy race car ....................................................................... 29
2.11 A staircase with a lack of texture where SLAM algorithms fail. Our beacons are clearly visible and can help SLAM algorithms...
Abstract

In this work we present a novel beacon light coding protocol, which enables fast and accurate identification of the beacons in an image. The protocol is provably robust to a predefined set of detection and decoding errors and does not require any synchronization between the beacons themselves and the optical sensor. A detailed guide is then given for developing an optical tracking and localization system, which is based on the suggested protocol and readily available hardware. Such a system operates either as a standalone system for recovering the six degrees of freedom of fast moving objects, or integrated with existing SLAM pipelines providing them with error-free and easily identifiable landmarks. Based on this guide, we implemented a low-cost positional tracking system which can run in real-time on an IoT board.

We evaluate our system’s accuracy and compare it to other popular methods which utilize the same optical hardware, in experiments where the ground truth is known. A companion video containing multiple real-world experiments demonstrates the accuracy, speed, and applicability of the proposed system in a wide range of environments and real-world tasks. Open source code is provided to encourage further development of low-cost localization systems integrating the suggested technology at its navigation core.
Chapter 1

Introduction

Numerous situations exist in the modern world where accurate knowledge of the position and orientation of an object or person are required in real time (> 20 frames per second) with cm to sub-mm level accuracy. This problem is known as positional or spatial localization (includes orientation) and is necessary for many applications including tracking in Virtual Reality, autonomous vehicle and robot navigation, warehouse automation, to name just a few.

1.1 Tracking methods overview

Methods for positional localization are generally organized into several categories: inertial, magnetic, wireless, acoustic, external and internal optical tracking. We will briefly review these methods, provide examples, and discuss their advantages and disadvantages. We will mention some of the leading commercial systems.

Inertial tracking

The inertial tracking method relies on data received from accelerometers and gyroscopes which are mounted on the tracked object. Accelerometer measure linear acceleration. By integrating twice the acceleration data it is possible to find the position of a body relative to some initial point. Gyroscope measure angular velocity. By integrating this velocity it is possible to calculate the relative angular position to some initial point. A tracking unit which is based on the inertial data is called an Inertial Measurement Unit (IMU). The unit can provide six degrees of freedom by using three accelerometers, one for each axis, and three gyroscopes, one for each rotation. See Figure 1.1 for an illustration.

The advantages of this method are high update rates with minimal latency, unlimited working area, small device size, and low price. The disadvantage of the
method is the large drifts that occur over time due to the need to integrate the incoming data and the lack of reference points for re-localization. These drifts are translated to large error in the positional data. The IMU is widely used today in most modern tracking systems as an add-on to other methods and not as a stand-alone system. For more information see [DAK19].

Electromagnetic tracking

Electromagnetic tracking systems rely on measuring magnetic field with electromagnetic sensors. A static transmitter generates electromagnetic fields in the three spatial directions which are generated sequentially at high rates. A sensor, located on the tracked object, measures the attenuation and level of the signal in each axis and the six degrees of freedom can be calculated. The transmitter and the sensors are connected to the same base station.

The advantages of this method are precise tracking data at high frame rates and the ability to work without a direct line of sight. The disadvantages of the method are the sensitivity to environmental noise (magnetic interference), low working range, wire connectivity to a large base, and the relatively high price. This method is mainly being used when there is a need for very high precision and short working range, for example, finger tip tracking and medical needle tip tracking.
There are many commercial systems available, for example the TrakStar system shown in Figure 1.2. For more information about this method see [RDB01].

(a) TrakStar system components. Picture taken from [NDI]

(b) Illustration of the working system.

(c) Finger tip tracking setup using magnetic tracking as shown in [WSK15].

Figure 1.2: Magnetic tracking

**Wireless tracking**

Wireless tracking methods rely on wireless communication protocols such as WiFi, Bluetooth, and Ultra-wideband (UWB) to determine the position of an object. In these methods a set of fixed wireless transmitters are placed around the working environment and the tracked object receives their signal. By measuring the signal strength or by measuring the Time of Flight, depending on the protocol, the system can triangulate the location of the tracked object.

The main advantages of this method are the low price, scalability, and lightweight hardware. In addition, when using WiFi and Bluetooth protocols, the tracked object can be a standard smartphone and there is no need for direct line of sight. There are several disadvantages for these methods. The first is that they only provide location and not orientation (3DoF). In addition the accuracy of the tracking data is low. Improved accuracy can be achieved with the UWB which requires using specialized hardware, restricted frequencies, and direct line of sight.
There are many cheap products for WiFi and Bluetooth trackers. There are also a few commercially available UWB products which are mainly used in industrial applications. For more information about this method see [LDBL07].

**Acoustic tracking**

Acoustic trackers use ultrasonic waves to determine the position and orientation of the tracked object. At least three emitters and three receivers are needed for 6 DoF calculation. Either the emitters or receivers can be on the tracked object and in the environment. The system uses the Time Of flight to measure short ultrasonic pulses from the source to the sensors.

The main advantages of this method are that it can work in challenging environments such as underwater and low visibility environments, the hardware is relatively cheap and light, and it can be scaled to large environments. The main disadvantages are the low update rates, sensitivity to environmental changes, noise and echoes, and the direct line of sight restriction. These systems are not widely used and are useful in a low verity of use-cases. For more information see [RDB01].

**Optical tracking**

Optical tracking offers a variety of solutions for the localization challenge. Each solution can overcome different limitations depending on the use case. These optical tracking systems are generally divided into two approaches, internal and external (also known as inside-out and outside-in tracking). In the internal approach the optical sensor is mounted on the tracked object along with a processing unit. The sensor perceives the environment and the processor tries to calculate its pose. On the other end, the external approach relies on an optical sensor which is usually in a stationary location along with its processing unit. The tracked object (or objects) need to be in the sensor’s field of view.

The main challenge in both internal and external approaches is the extraction of features or landmarks that can (i): be easily identified and tracked (in the internal approach, those features are landmarks in the observed scene, and in the external approach its the object itself), and (ii): provide information about the spatial position and orientation of the observer. Extensive research has been done to address this challenge. The solutions can be roughly divided into two categories: marker-based and marker-less solutions. Marker-based methods deploy known objects in the observed scene (internal approach) or mount them on the
tracked object (external approach). Marker-less solutions try to extract (previously unknown) features from its unknown environment.

An example for a marker-less internal tracking system is the Oculus Insight head tracking system [Ocu], see Figure 1.3. The system uses four cameras that are mounted on the head unit and track the environment using SLAM algorithm (see Section 2.1 for details on SLAM). In addition, the same system also uses a marker-based external approach to track the hand controllers relative to the head mount. The marker-less internal approach allows the tracked object to be independent from external equipment and free from spatial limitations. The Oculus tracking system also fuses data received from IMUs to improve the tracking quality and speed.

![Oculus Quest VR headset with four cameras for internal tracking](image1)

![Illustration of the system tracking the environment while also tracking the hand controllers](image2)

Figure 1.3: Oculus tracking system

Another example for a VR optical tracking system is the HTC Lighthouse [HV]. The system is an internal tracking system with active markers. The active markers, base stations, contain a spinning motor with a line laser emitting infra-red light. While this is an optical system the input sensor is not a camera but a set of simple one pixel infra-red sensors in a specific geometry. Both head unit and the hand trackers are equipped with these sensors. See Figure 1.4. Similarly to the Oculus system, this system also fuses IMU data for improved tracking performance. This system is also used for robotic tracking by placing the hand tracker on the robot or creating a custom sensor geometry. See Figure 1.5 for examples.

An example for external systems are the OptiTrack [Opt] and the Vicon [Vic] which are marker-based. These systems use multiple high speed cameras, each equipped with an infra-red LED ring, which are spread around the working
Figure 1.4: HTC tracking system
(a) An example for a wheeled robot that uses the HTC hand tracker for its localization.
(b) A drone with a custom board with four IR sensors used for tracking with the HTC base station. Picture taken from [Bit].

Figure 1.5: Robotic tracking with the HTC system

environment. Multiple objects can be tracked, each object is equipped with a set of reflective markers in a certain geometry. These systems are widely used for body motion capture and as ground truth for lab experiments. See Figure 1.6

Since our main focus in this work is on optical systems, we will expand on available types of solutions and their advantages and disadvantages in Section 2.1.
(a) Illustration of a system setup using OptiTrack. A room with multiple cameras. on the left is a man with markers for body motion capture. On the right a toy car with markers for object tracking.

(b) A drone with reflective markers. Picture taken from [Bit].

Figure 1.6: External tracking systems
Chapter 2

Coded beacons

2.1 Related work

In this section we describe the main related optical-based methods that have been used for positional tracking in recent years.

In fields such as mobile robotics, it is common to use internal tracking solutions (ego-motion). For this goal, many marker-based systems were suggested which utilize known markers for localization, e.g., a chessboard, ARTag [Fia05], arUco [GJMSMCMJ14], RUNE-tag [BART11] and [BHHN00], which are also known as fiducial markers. Generally, such techniques rely on the known geometry of these patterns, which are distributed around the working environment in advance; These techniques are susceptible to illumination variations and partial occlusions, they do not scale up to larger environments due to the need for close visual views of the patterns (see Section 2.5), and require the ability to distinguish between the different detected patterns.

To this end, an alternative marker-less technique, called simultaneous localization and mapping (SLAM), is widely used [TUI17, MAMT15]. This technique uses a digital sensor coupled with a processing unit which detects, identifies, and tracks 2D image points in order both to estimate the sensor motion and to reconstruct, in real-time, the structure of the unknown environment. SLAM systems do not usually require prior knowledge and make use of readily available camera technologies. However, SLAM systems suffer from issues of drift, relocalization (see Section 2.5), as well as sensitivity to environment variation due to its dependency on natural features. In addition, those methods usually require a strong processor since they apply computationally heavy heuristics, such as bundle adjustment [TMHF99], to compensate for their inaccuracies and uncertainties. To overcome the drift and relocalization problems, hybrid systems were developed.
which also utilize predefined markers in this SLAM pipeline, e.g., [MSMC20] which use arUco markers to improve robustness. However, these hybrid systems still suffer from the same problems as the marker based systems described above. Another internal localization system, which is based on active markers, is the HTC VIVE lighthouse [HV], mainly used for virtual environment systems and indoor localization; see [NLL]. This system can provide a precise pose information at high speeds. However, it is limited to a small room scale due to the need to synchronize the lighthouse(s) with the IR receivers deployed on the object. It also includes sophisticated and costly moving mechanical parts which are sensitive to environment noise and are more prone for failure.

The external optical tracking approached is used when there are limitations on the payload that the object can carry, or when multiple objects need to be tracked in the same reference frame. OptiTrack [Opt] and the Vicon [Vic] systems are arguably amongst the most popular commercial marker-based systems which apply the external approach. These systems can produce high precision positional data at high frame rates but are expensive and require a long calibration procedure and high computational power. Some alternative low-cost systems, which use different variations of known markers, were suggested in [FKC+13, NJF15]. In addition the fiducial markers mentioned above are also commonly used for external tracking. These are usually non-robust due to their dependency on the 2D view of the predefined geometric pattern placed on the tracked object. In addition the tracked objects must be in the view of the sensor, which restricts their operating range. In some cases a wireless connection might be required between the external monitoring device and the tracked object to send location-based control commands.

2.2 Our contribution and novelty

Our work introduces the following contributions:

(i) We provide a code-book containing code-words which are provably error resistant and have an efficient decoding process.

(ii) A novel coded beacons (markers) which are easy to extract and identify, based on our code-book. These markers, which we call CoBe (coded beacons), flash a unique cyclic binary code at high speeds, and do not require a shared clock signal for synchronization; see Section 2.3 and patent [WK18].

(iii) Based on our CoBe, we propose a general pipeline for a positional tracking system which can function in both internal and external approaches; see
Section 2.4.1. We then implement our own low-cost off-the-shelf infra-red system based on this pipeline; see Section 2.4.4.

(iv) Extensive real-world tests demonstrate our system’s practical applications, real-time performance, and accuracy; see Section 2.5.

(v) A full open source code for our system along with a simple guide for its reproduction. We hope such an open source code and guide will lead to the development of more homemade tracking systems based on the proposed technology.

2.3 Coded light codes

In this section we describe our key contribution: a coded light sequences scheme that can be used as robust location identifiers. We start by briefly describing the physical transmitter properties that we leverage to generate the coded sequences. We then describe a simple coded light protocol to demonstrate the main concept, followed by the more involved one we used. The protocol was first patented in [WK18].

2.3.1 Transmitter properties

We use optical beacons that emit a cyclic binary sequence of light. This can be done either with a low-high brightness sequence using near-visible (infra-red) illumination or a red-blue visible light sequence, depending on the specific use and environment. The flashing light sequence is interpreted as a binary code by extracting the bit value at each time interval that corresponds to single bit in the sequence. The bit value extraction method we used is detailed in Section 2.4.4.

2.3.2 Robust cyclic binary codes for unique identity detection

The use of coded light pulses for identifying light beacons has a long history in the field of navigation, manifesting itself in lighthouses and way-point buoys among others; see for example [USC19]. We wish to continuously transmit a binary light code that can be used to uniquely identify a beacon.

In order to make the coded-light scheme robust and useful for an automated detection and decoding pipeline, we wish to design a code-book containing codewords that satisfy the following three properties:
- fast lock-on and decoding time that is close to a single full code cycle,
- robustness to single bit-shift and bit-flip errors,
- code-book size at least linear in code-word length.

It is important to note that realizing only one or two of these properties without the other is considerably simpler than satisfying all three together.

**Initial approach - no error robustness**

To design a code-book with code-words that can be detected within one full code cycle without the need to use a synchronization comma we initially choose the set $C_n$ of complete cyclic equivalence classes [Pea03] using n-bit binary codes. Every element $c \in C_n$ is defined as the set of the $n$ cyclic shifts of the $n$-bit sequence $(b_1, b_2, ..., b_n)$. In other words $c$ has the property that if $(b_1, b_2, ..., b_n) \in c$ then $(b_i, b_{i+1}, ..., b_n, b_1, ...b_{i-1}) \in c$ for every $i \in \{1, \cdots, n\}$. We define $c_0$ to be the number in $c$ whose decimal value is the smallest. We denote by $c_i$ the code-word in $c$ with a cyclic shift left of $i$ bits relative to $c_0$. Let $c_0$ be the representative of the set $c$ and consider its decimal value to be the representative identifier $id$. These codes are particularly useful in our case because the same code-word is continuously transmitted by a specific beacon. Assuming there are no errors and that the optical sensor receives at least $!n$ consecutive bits from a given beacon, we can unambiguously decode (identify) the code-word irrespective of the observation starting point in the bit stream. The most trivial cyclic codes one could think of are $00\cdots0$ and $11\cdots1$, and in our case are not useful, so we ignore them. An example of a single element in $C_4$ is the set $c = \{(0001), (0010), (0100), (1000)\}$. The total number of codes available in $C_n$ is exponential in code-word length.

**Creating the code-book** In order to construct the code-book we need to find all the complete cyclic equivalence classes. We store a list of the $c_0$ code-words for every such class $c$, for example, via simple exhaustive search. We also maintain a lookup table that stores every possible $n$-bit sequence $c_0, \cdots, c_n$ together with the corresponding $c_0$ code-word from which it can be generated. The algorithm to generate these data structures is presented in Algorithm 2.1.

**Encoding** We choose an identifier from $0$ to $|C_n| - 1$ to represent a beacon. To encode this identifier we use the identifier as an index into the code-book and take the $c_0$ stored at that location. The process of transmission is now trivial. We
Procedure 2.1 Generating all n-bit cyclic equivalence classes

| Input: | n | \(\triangleright\ n\) code bits |
| Output: | \(D_n, C_n\) | \(\triangleright\) look-up table, code-book |

1: \(D_n \leftarrow \text{ZEROS}(1, 2^n)\) \(\triangleright\ \text{look-up table}

2: \(C_n \leftarrow \emptyset\) \(\triangleright\ \text{code-book}

3: \(k \leftarrow 0\) \(\triangleright\ \text{code counter}

4: foreach \(c_0 \in (0, \ldots, 2^n - 1)\) do

5: \(\text{inC} \leftarrow \text{false}\)

6: \(\hat{c}_0 \leftarrow \text{DEC2BIN}(c_0, n)\) \(\triangleright\ \hat{c}_0\) is n-bit binary

7: \(Id \leftarrow \text{ZEROS}(1, n)\) \(\triangleright\ \text{look-up table indices}

8: foreach \(i \in (0, \ldots, n - 1)\) do

9: \(\hat{c}_i \leftarrow \text{CIRCULARSHIFT}(\hat{c}_0, i)\)

10: \(Id(i) \leftarrow \text{BIN2DEC}(\hat{c}_i)\) \(\triangleright\ \text{index into } D_n\)

11: \(\text{inC} \leftarrow (\text{inC}) \vee (D_n(\text{Id}(i) + 1) > 0)\)

12: if \(\neg\text{inC}\) then

13: \(k \leftarrow k + 1\) \(\triangleright\ \text{increase code counter}

14: \(C_n \leftarrow \{C_n, c_0\}\) \(\triangleright\ \text{add } c_0\) to code-book

15: foreach \(i \in (1, \ldots, n)\) do

16: \(D_n(\text{Id}(i)) \leftarrow c_0\) \(\triangleright\ \text{... stores } c_0\) (identifier) in \(D_n\)

Transfer the code to a beacon and periodically transmit the bits at a known bit rate using either the intensity or hue based methods described in Section 2.3.1.

Decoding Once the optical sensor has detected and recorded at least \(n\) consecutive bits from a given beacon, decoding the received code-word to obtain the beacon identity is performed. Only the last \(n\) bits are processed simply by accessing a (pre-computed) look-up table that maps the bit sequence into a representative beacon identity.

<table>
<thead>
<tr>
<th>(c_0)</th>
<th>0001</th>
<th>0011</th>
<th>0101</th>
<th>0111</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>code</td>
<td>id</td>
<td>code</td>
<td>id</td>
<td>code</td>
</tr>
<tr>
<td>0000</td>
<td>0</td>
<td>0100</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>0001</td>
<td>1</td>
<td>0101</td>
<td>5</td>
<td>1001</td>
</tr>
<tr>
<td>0010</td>
<td>1</td>
<td>0110</td>
<td>3</td>
<td>1010</td>
</tr>
<tr>
<td>0011</td>
<td>3</td>
<td>0111</td>
<td>7</td>
<td>1011</td>
</tr>
</tbody>
</table>

Table 2.1: \(C_4\) code-book and \(D_4\) look-up table mapping every code-word to its decimal value identifier in the non-robust manner.

As an example, a look-up table for \(C_4\) can be seen in Table 2.1. If the following sequence was received (left to right) 1101110111, it would have been decoded in consecutive subsets of \(n = 4\) bits as \((1101) \rightarrow 7, (1011) \rightarrow 7, (0111) \rightarrow 7\) and so
Robustness to noise

The approach to generate and decode binary codes described above is straightforward and over-simplistic. However, it lacks any ability to detect and correct errors. Typical error correcting approaches encode bit sequences with parity bits, Hamming codes, CRC codes and others. These methods measure the distance between two code words using the Hamming metric, and ensure a minimal Hamming distance between all code-words.

Our experiments demonstrated that we should consider the three following types of bit errors that can occur with the same probability:

- Bit-flip, where a bit is interpreted incorrectly (1 as 0 or vice versa).
- Bit-miss, where a bit was not received.
- Bit-insertion, where an additional bit was falsely added to the sequence.

The Hamming metric simply counts the number of bits which are different between two code-words, i.e., bit-flips. Unfortunately, when also bit-misses and bit-insertions occur, the Hamming distance is insufficient. A considerably more appropriate metric is the Levenshtein distance [Lev66]. This measures how many addition/deletion/flip operations on one of the sequences is needed to reach the other. A bit-flip error would result in a Levenshtein distance of 1. A bit-miss and bit-insertion errors are considered as a chained insertion and deletion operations (or vice-versa) on the received sequence and has a Levenshtein distance of 2, since we always compare the last $n$ seen bits to some code of length also $n$. However, two strings from the same cyclic equivalence class (the same string up to some cyclic shift), which basically represent the same string, may have a very large Levenshtein distance. So even though the Levenshtein distance is more appropriate for error detection we will not use it directly.

Instead, in order to construct a robust code-book for every cyclic equivalence class $c$, we would like to find all code-words that can be obtained from some $c_i \in c$ by exactly one error from the above permissible errors, and map all these code-words via our look-up table to the same identifier (representative) $c_0$ of $c$. We found that a simple greedy algorithm was sufficient to compute the maximal set of non-overlapping classes as explained above and produce a code-book with size exponential in the number of bits; see Fig. 2.1a. Besides the cyclic variations (equivalences) of $c_0$, the suggested greedy algorithm also enumerates and stores all
error variants of these cyclic variations in a look-up table; see Algorithm 2.2. The number of codes produced for both this method and the non-robust initial method can be seen in Fig. 2.1a. **Encoding** and **Decoding** are exactly as explained in the non-robust approach suggested above.

**Procedure 2.2** Generating robust $n$-bit cyclic equivalence classes

<table>
<thead>
<tr>
<th>Input:</th>
<th>$n$ code bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output:</td>
<td>$D_n$, $C_n$</td>
</tr>
</tbody>
</table>

1. $D_n \leftarrow \text{ZEROS}(1, 2^n)$  
   ▷ look-up table
2. $C_n \leftarrow \emptyset$  
   ▷ code-book
3. $k \leftarrow 0$  
   ▷ code counter
4. foreach $c_0 \in (0, .., 2^n - 1)$ do
5.   $inC \leftarrow \text{false}$  
   ▷ $c_0$ is decimal
6.   $\hat{c}_0 \leftarrow \text{DEC2BIN}(c_0, n)$  
   ▷ $\hat{c}_0$ is n-bit binary
7.   $Id \leftarrow \emptyset$  
   ▷ look-up table set of indices
8.   foreach $i \in (0, .., n - 1)$ do
9.     $\hat{c}_i \leftarrow \text{CYCLICSHIFT}(\hat{c}_0, i)$
10.    $Idx \leftarrow \{Idx, \text{BIN2DEC}(\hat{c}_i)\}$
11.    $V \leftarrow \text{NOISIFY}(\hat{c}_0)$  
   ▷ insertions, deletion, flips
12.   foreach $\hat{v} \in V$ do
13.     $id \leftarrow \text{BIN2DEC}(\hat{v})$
14.     $Id \leftarrow \{Id, id\}$
15.     $inC \leftarrow (inC) \lor (D_n(idx + 1) > 0)$
16.   if $\neg inC$ then  
   ▷ if $c_0$ not overlaps with $C_n$
17.     $k \leftarrow k + 1$  
   ▷ increase code counter
18.     $C_n \leftarrow \{C_n, c_0\}$  
   ▷ add $c_0$ to code-book
19.   foreach $id \in Id$ do  
   ▷ for all noised $c_i$ ...
20.     $D_n(id) \leftarrow c_0$  
   ▷ ... stores $c_0$ (identifier) in $D_n$

### 2.4 Positional Tracking System

In this section, we specify both the hardware and the software procedures required to construct and operate a full six degrees of freedom (6DoF) positional tracking system which is based on the coded-light scheme presented in Section 2.3. We then demonstrate in detail our low-cost implementation for such a system in Section 2.4.4. The system developed in this paper is used in both the internal and external vision-based approaches. We demonstrate in Section 2.5 our system’s applicability, performance, and accuracy in real-world tasks.
<table>
<thead>
<tr>
<th>code</th>
<th>mapped id</th>
<th>error</th>
</tr>
</thead>
<tbody>
<tr>
<td>00010100</td>
<td>7</td>
<td>2 bit cyclic shift + bit-flip</td>
</tr>
<tr>
<td>00001101</td>
<td>7</td>
<td>bit-insertion</td>
</tr>
<tr>
<td>00101111</td>
<td>63</td>
<td>bit-flip</td>
</tr>
<tr>
<td>00110111</td>
<td>63</td>
<td>bit-insertion</td>
</tr>
<tr>
<td>00101010</td>
<td>85</td>
<td>bit-insertion</td>
</tr>
<tr>
<td>10100101</td>
<td>85</td>
<td>cyclic shift + bit-miss</td>
</tr>
<tr>
<td>01100010</td>
<td>unmapped</td>
<td>one error away from a variant of id:7</td>
</tr>
</tbody>
</table>

Table 2.2: $C_5$ code-book and $D_5$ lookup table, built using the robust method. The final table is a union over the error variants. For example 11110 is a bit-flip error of 11100. Similarly 01101 bit-shift error.

(a) Comparison of code-book sizes for dif-
(b) Table of lock-on times in seconds for ferent code-word lengths for the initial and code-book size versus sample frame-rate. robust code-book generation methods.
2.4.1 System overview

Hardware The system consists of two main components. The first component, called the **perception unit**, requires an optical sensor (for example a camera) and a processing unit. The second component, denoted as the **marker unit**, contains a set of **beacons**. A beacon is a device equipped with a micro-controller connected to some light source (for example a LED). Each beacon has a unique code based on the coded-light protocol presented in Section 2.3. The micro-controller transmits this unique binary code by alternating between the two states of the light source.

Calibration and mapping The following is a pre-processing step that needs to be carried out only once, before activating the system. The optical sensor needs to be calibrated in advance to obtain its internal parameters (e.g. the intrinsic parameters in the case of a simple camera). This can be done using one of the many tools available online, e.g., OpenCV [Bra00]. For more details on internal optical sensor calibration see [Zha00].

Now, the beacons need to be deployed as follows: in the internal approach, the beacons should be spread around the desired working environment. In the external approach, the beacons should be placed on the object(s) to be tracked; see Fig. 2.2.

![Figure 2.2: Beacon deployment.](image)

Figure 2.2: **Beacon deployment.** (Left) Internal tracking: beacons are deployed around an office environment. (Right) External tracking: beacons are deployed on a toy car.

After the deployment, the relative position of each beacon in 3D space needs to be extracted. This can be done either by using a structure from motion (SfM) pipeline [ÖVBS17] (see Section 2.4.4 for a suggested implementation), by using a depth camera, or by manually measuring the pairwise distances between every pair of beacons and then applying the well known multi-dimensional scaling.
method [BG05] to embed these beacons in 3D space. This relative position of the beacons, along with the internal calibration data, are then stored in the processing unit, where all the calculations take part.

**Beacon detection and decoding** Once the prepossessing step is complete, the processing unit can start reading a live stream of frames from the optical sensor, and apply the main procedures: detection and decoding. In the detection procedure, the goal is to detect the 2D location (region) of each of the visible beacons in the current received frame. Since a light source is to be detected, the detection can be done by searching for the brightest regions in the frame; see suggested implementation in Section 2.4.4.

After detecting a set of regions, we associate each of them with their corresponding regions from the previous frame. Once a beacon was detected for a sufficient number of consecutive frames, the perception unit can identify the unique identity of this beacon by decoding its sequence of states in those frames, and retrieve its 3D location from the previously computed list of beacon 3D locations.

**Positional tracking** If more than 4 non-planar beacons are successfully identified (decoded) in the same frame, a pose estimation procedure can be invoked to compute the desired 6DoF. If less than four beacons are detected or those detected are co-planar, then the perception unit may still be able to determine partial degrees of freedom. Alternatively, the 3D location of the beacons can be utilized to refine and enhance different localization systems running in parallel, such as a SLAM system.

### 2.4.2 Bit value extraction

**Intensity.** For near-visible illumination the binary bit representation is done by altering the intensity of the light source between high and low. The challenge in this approach is determining what constitutes as high and low intensities. We empirically found that the following method produces the most reliable results: (i) averaging the minimal and maximal intensity in $n$ consecutive bits (ii) assigning the high or low value if the intensity in a single frame is above or below the average line respectively. To justify this we note that every code has at least one high pulse and one low pulse. In addition the flashing rates are fast enough that even in fast movements the intensity average remains constant for $n$ bits. In order to correctly and consistently identify brightness levels at different distances and
in different lighting environments we ensure that the optical sensor has a fixed contrast and exposure set for all images captured in a tracking session.

**Hue.** To determine the bit value of each flash from a beacon when using red-blue flashes we use the hue (from the HSV color space) as determined by the optical sensor for each detected flash location. Although we focus primarily on binary codes using two hues, we could in fact use multiple hues and non-binary codes. All the methods described in this work for binary code generation can be easily converted to \( n \)-ary codes with the advantage that they would be shorter. Non-binary codes may actually have an advantage if we use a visible light multi-hue beacon setup because the transition between colors could be adapted to be more visually pleasing and therefore less intrusive in the environment.

### 2.4.3 Design considerations

**Beacons distribution** To effectively utilize the number of available beacon IDs, some IDs can be reused in different rooms which are visually separated from each other. For simplicity, we filter out cases where two beacons with the same ID are identified in the same frame. While setting up the beacons in the environment it is also possible to choose to take advantage of the fact that two beacons which are close to each other can be assigned codes with a large Levenshtein distance between them and two beacons which are far from each other and are unlikely to be observed at the same time may have similarly assigned codes. This process can be used as an extra layer of robustness to improve the identifiability of beacon IDs. The tracking unit can automatically receive the mapping information of new beacons in new environments via Bluetooth from room specific transmitters as they move from one area to another, for example in a museum environment with multiple rooms or on a farm with multiple fields. Each tracking unit can also retain an internal map of the different tracking environments obtained during mapping. This enables the tracking unit to detect transition between rooms by detecting large increase in the tracking error during the localization processing.

**Frame rates** The time to first decoding of the beacon ID is called the “lock-on time”. Our code are cyclic so that the lock on can occur without the need to wait for "comma" code. After the lock-on time, we can identify the beacon after every new bit with no additional time penalty. The rate at which the beacon emits its code bits is designed to match the rate at which the optical sensor samples the scene. So, the trade-off is clear: we need high frame-rates for both fast lock-on time, as well as longer code-words while we prefer lower frame-rates to limit the
processing requirements of the tracking unit. This trade-off can be seen in Table 2.1b.

2.4.4 System implementation

In this section, we discuss in detail our implementation of a positional tracking system based on the overview in Section 2.4.1. Our system was built out of low-cost off-the-shelf components with easy reproduction and accessibility in mind; see illustration in our video [vid]. All the code was written in Python using OpenCV [Bra00].

**Coded-light protocol** We used the intensity based method with infra-red (IR) illumination. The codebook for the coded-light protocol we used was generated using Procedure 2.2 with \( n = 15 \).

**Hardware** We initially chose to work with a dedicated sensor which has build-in infra-red object tracking capabilities (PixArt multiple object tracking sensor). The sensor can provide tracking data for up to 16 infra-red objects at high speeds using a fast communication protocol and without the need to use additional computing power. Despite the sensor’s benefits we have encountered numerous difficulties while calibrating and debugging our system. This was mainly caused due to the sensor being a black box device without the option to understand how the object detection and tracking works. In addition the field of view and focal points were limited. The sensor and a dedicated calibration rig which we have build can be seen in figure 2.3.

We decided to work with a readily available camera. The final configuration of the perception unit consists of a PSEye3 cheap (< 10$) webcam, with an IR pass filter attached to the lens. The camera is connected to a Raspberry Pi [Ras]. The microcomputer can be replaced by a standard laptop. Each beacon in the marker unit was built using a standard IR LED (950nm wavelength) connected to a tiny microcontroller called “A-star 328PB Micro” with a small LiPo battery, and was assigned a unique code from our chosen codebook. The microcontroller was programmed to cyclically flash the LED according to this code, where “1” and “0” in the code correspond to high and low intensity values of the LED, respectively. To distribute the light evenly in all directions a diffuser cover is placed on each LED; see Fig. 2.4. The beacons do not have a shared clock signal.
(a) The PixArt sensor connected to an Arduino. Can send fast data of up to 16 IR objects using fast serial communication.

(b) Our calibration jig for the sensor. Containing 16 IR LEDs. The jig is used similarly to the chessboard calibration tool.

Figure 2.3

Figure 2.4: System components. (Left) our homemade CoBe (coded beacons) consisting of a microcontroller, a LiPo battery, and an IR LED. (Right) An IoT board and a cheap webcam; see Section 2.4.4.
Calibration and mapping. To retrieve the camera’s intrinsic parameters, we applied the standard camera calibration pipeline [Zha00] using functions from the OpenCV library [Bra00].

We implemented our own simple SfM pipeline which also uses our detection and decoding algorithm for feature extraction and matching between frames. This pipeline applies the following steps: (i) Capturing: capture $m$ frames $F_1, \ldots, F_m$, such that every beacon appears in at least 2 different frames, and every pair of consecutive frames share at least 5 common beacons. Each beacon was identified using our decoding scheme detailed in the next paragraph. (ii) Relative frame alignment: compute the essential matrix $E_i$ between every pair $F_i$ and $F_{i+1}$ of consecutive frames as described in [Nis04](which is why 5 common markers are required between the frames), and decompose it into a rotation matrix $R_i$ and translation vector $t_i$ as in [TH84]. (iii) Triangulation: triangulate the common points between $F_i$ and $F_{i+1}$ to obtain a point cloud $P_i$. (iv) Point-cloud alignment: align the $m$ point clouds together by transforming them to a common coordinates system, to obtain a combined point cloud $P$ that contains the 3D locations of all the beacons. The optimal translation between two points clouds is simply the vector connecting their means, and the optimal rotation [Wah65] can be simply computed via SVD [Kab76]. (v) Refinement via bundle adjustment: apply a non-linear least squares optimization function which takes as an initialization the point cloud $P$ and the rough estimation $R_i, t_i$ of the camera pose in every frame $F_i$, and refines $P$ and the poses as to minimize the reprojection error [TMHF99, Mor78].

Beacon detection and decoding Since our camera has an IR pass filter and the beacons use IR LEDs, the beacons appear as bold bright blobs on an (almost) black background, which makes them easy to detect. Therefore, in each frame $F_i$ we detect the beacons using a simple contour detection function in OpenCV; see Fig. 2.5. We also store the area of each contour, which will be used to determine the bit value of the beacon (high or low). We associate the contours in $F_i$ with a contours from the previous frame $F_{i-1}$ using an optimal matching scheme [Mun57], which associates the contours so as to minimize their sum of distances in the 2D frames. We also allow partial matching, that is, associating only a subset of the contours. This is done by increasingly adding “dummy contours” and checking the decrease in the final matching cost.

Once every beacon’s contour was successfully associated for $n = 15$ consecutive frames, we estimate its bit value in each of these frames as follows. We take the
beacon’s two contours with the maximal and minimal areas, average the areas, and define the bit value of the beacon as high or low in some frame if its contour area in that frame is larger or smaller than this average, respectively. To justify this identification, we note that each code has at least one high and one low bit value. The decoded 15 bit values correspond to the beacon’s desired unique cyclic code.

**Positional tracking** If 4 or more beacon codes have been decoded for some frame $F_i$, we plug their 2D locations (the center of each contour in $F_i$) and their corresponding 3D locations (computed in the pre-processing step) into the function `SolvePnP` from OpenCV. We thus obtain the 6DoF of the camera relative to the observed object(s) / environment.

![Figure 2.5: beacon detection and decoding: (a) image received from the sensor, (b) beacons are detected and tracked (c) beacons are decoded after n consecutive frames.](image)

2.5 Real-World Experiments

In this section we put the system detailed in Section 2.4.4 to the test.

We conduct different types of experiments which demonstrate the: (i) Accuracy of our system as compared to other common methods, (ii) ability of our system to track and localize very fast moving objects and fast lock on and re-localize when occlusions occur, and (iii) applicability of our system to a wide range of applications and challenging environments, both as a standalone system or as a component fused into existing systems. The experiments are presented in [vid].

2.5.1 Accuracy

In this experiment, we tested the tracking accuracy of our system when used for the internal tracking approach. We expect to reach at least the same accuracy as...
that of the external approach due to the static position of the camera.

**Rails test** We mounted our perception unit (camera) on a precise 2-axis moving rail system shown in Fig. 2.6, which provided the exact ground truth position of the camera, up to the sub-millimeter accuracy. We programmed the two axis rail system to move the camera in a rectangular shape of size 1.2m×1.2m, while the exact position of the camera when capturing every frame was known. The experiment is presented in our video (at 00:10).

**Servo test** To test the accuracy of our system when the camera rotates, we mounted the perception unit on a rotating servo with angular control; see Fig 2.6. The servo was programmed to rotate 12 times clockwise, each by 5 degrees, and then rotate back 12 times anti-clockwise, by 5 degrees each. The experiment is presented in our video (at 00:41).

![Figure 2.6: Systems used as aground truth in our accuracy testing experiments. (Left) Two axes rail system. (Right) A servo.](image)

In both tests we compared our system to one of the widely used techniques for pose estimation, which places a chessboard with known geometry in the observed scene. The chessboard pattern is a basic representative for the family of fiducial marker-based approaches. We used the same camera (PSEye3) for both methods. A large A3 chessboard pattern was placed in front of the moving camera, such that it is clearly visible during the entire test. Both methods aim to detect 2D features and find their corresponding 3D features in the predefined point cloud. The PnP method (from OpenCV) was then used in both cases to recover the 6DoF [LMNF09].

Fig. 2.7–2.9 present the accuracy results of both tests. As shown there, our system outperformed the widely used chessboard method, for both position and orientation accuracy. Our system yielded an error up to ×5 times smaller, while
(a) **Position accuracy.** (Left:) 2D top view of the recovered camera position. (Right:) the recovered Z-axis values (camera height).

(b) **Orientation accuracy** for each frame for each of the 3 angles.

Figure 2.7: **Rail test accuracy.**
Figure 2.8: **Rails test cumulative position error** compared to the ground truth, from the beginning of the test till each frame. Using our markers, the total sum of errors throughout the entire experiment was roughly $0.6 \times 10^4$, as compared to $3.5 \times 10^4$ when using the chessboard pattern.

also being more stable and consistent. Furthermore, our beacons deployment was easier to scale up in order to cover larger areas, and easily detected from far away compared to the chessboard; see our video [vid].

### 2.5.2 Handling fast moving objects and occlusions

To test our system’s performance as an external positional tracking system, we mounted our beacons on a fast moving toy race car, as illustrated in Fig. 2.2, and placed our perception unit in a fixed location in the room from which it can observe the race car. The goal was to compute the 6DoF of the car in two different scenarios: (i) when the car is moving fast and/or performing complex maneuvers, and (ii) when the car passes behind some object and is occluded by it for some amount of time, before it is observed again. The test setup can be seen in figure 2.10.

The two tests are presented in the our video [vid] (at 01:09 and 01:54 respectively). As shown, our system was able to accurately localize the fast moving car also when performing fast maneuvers. It is also clear that our system requires a very small amount of time to re-track and re-localize the car after it has been occluded, as also predicted by Table 2.1b.
Figure 2.9: Servo test orientation accuracy and stability.

Figure 2.10: External tracking of a fast toy race car
2.5.3 Challenging environments

Many existing SLAM systems have very poor performance in environments which lack texture or are very repetitive, for example, a room with flat uniformly colored walls, long symmetric hallways, and staircases. To that end, we tested both the ORBSLAM2 system [MAMT15], which is a computationally-heavy and sophisticated system, and our system’s performance in a typical two story staircase. Unfortunately, the ORBSLAM2 system did not even manage to capture initial features in such an environment due to symmetry and uniform color. Our system’s performance, using the same optical sensor, is demonstrated in our video [vid] (at 02:26). The test environment can be seen in figure 2.11 As presented, using 16 of our coded beacons, we were able to accurately find the location and navigate in a stable manner within a two story staircase. This test is a proof of concept that our low cost and computationally-light system can be integrated into an existing SLAM pipeline, such as ORBSLAM2, to help overcome challenging scenarios.

![image](a) Our camera  ![image](b) RGB camera

Figure 2.11: A staircase with a lack of texture where SLAM algorithms fail. Our beacons are clearly visible and can help SLAM algorithms

2.5.4 Potential Use-Cases

Other potential use-cases and applications for our system include:

(i) **Head tracking and pose estimation** of a human or a humanoid. This application is presented in our video [vid] (at 02:52).

(ii) **Tracking and localization of multiple objects in the external approach.** Different beacons may be placed on more than one moving object, to obtain the 6DoF of multiple moving objects, e.g., a car and a drone...
simultaneously. However, this requires applying the initial mapping step for each object on its own.

(iii) **Outdoor navigation** when either the lighting conditions are challenging e.g., at night or in a dynamic environment such as a forest or a plantation, where the plants are very similar and move rapidly. In this scenario, most vision (RGB) based methods will likely to be inaccurate and have a lot of outliers. However, our system is expected to work properly in the absence of light due to its use of static previously deployed IR-based beacons which are easily detected in such conditions.

(iv) **Warehouse automation.** Using our coded beacons, manufacturing robots or inventory counting drones can autonomously localize and navigate accurately throughout the warehouse, even at dark.

(v) **Under water navigation** is a tough task [PSSL13] mainly due to the lack of GPS signal, very poor lighting conditions, and highly noisy images which make the detection of image features an almost impossible task. This motivates the use of previously deployed markers such as our beacons, which simplifies the detection step in the noisy images, and are not affected by the lack of light. This can be useful for autonomous underwater research vessels.
Chapter 3

Conclusion and future work

In this work we presented a beacon light coding protocol which is robust to noise and does not need direct signal communication. This protocol enables fast and accurate detection and identification of beacons using an optical sensor with minimal computational burden. We have explained in detail the development of our code-book and code words which are provably error resistant and have an efficient decoding process. We then provided a detailed description of the building blocks for a positional tracking system which is based on our protocol. We continue by presenting an implementation of such a tracking system using low cost and accessible hardware, which can either operate as a stand-alone system or be integrated with a SLAM pipeline to enhance its performance. The performance of this system was evaluated and compared to other common methods and to a ground truth. Our system has low installation cost, low energy footprint, is easily deployable, can be easily adapted to different environments, does not need direct signal communication and can be used by multiple users without a centralized processing system. We believe this approach has much potential for increasing robustness in tracking for future industrial and possibly consumer level applications.

3.1 Future Work

Integration with other technologies. Similarly to other optical tracking systems, integration with other technologies could greatly improve the performance of the system. Integration with inertial sensors is a common example. Such integration can allow the optical system to work at lower frame rates, resulting in a simpler system, yet provide drift free information to the inertial sensor. The overall system would be lighter, faster and more robust.
Proof of concept application. We implemented our system as a proof of concept for testing and lab experiments. The next step will be to implement the system in one of the practical scenarios described in 2.5.4. Underwater tasks for example could greatly gain from such implementation as there is currently no robust enough solution for autonomous systems in real world applications.
Bibliography


HTC-VIVE. tracking system. "https://www.vive.com".


NDI. Measurement systems. "https://www.ndigital.com/".


Diederick C Niehorster, Li Li, and Markus Lappe. The accuracy and precision of position and orientation tracking in the htc vive virtual reality system for scientific research. *i-Perception*, 8(3):2041669517708205.

Oculus. Vr headsets and equipment. "https://www.oculus.com".

OptiTrack. Motion capture systems. "https://www.optitrack.com".


Vicon. Motion systems. "https://www.vicon.com/".

Demonstration video of our system and experiments: https://drive.google.com/file/d/1yD-4W4hqaIUhafevhDh9sYz2IfJK64dn/view?usp=sharing.


הRTL של החיבור אשר נמצאות על הגה עומון ואבصبיבת חתולה בבוש השימוע
(Ӧקבר אטינאול או אוריינט). גלופית החיבור הבינוני בין המערつな דרור בלעしたい
על כל החיבור והدور ובמיוחד המימוש. דרך מצויה(Random) וב униjadi מתופר.
דריך לחות בור יגון לבצע את המימנו. חומך את השיקולים שיש לבעש.
ברפסית המימונו במרחב חתולה באומן השימוע.

尼斯יונם ביעול האמיתיות

לאור זה החיבור ביצוע ניסיוןắn받י עטיפת השימוע על המערנן באופנים,
לבינימן במאפשרות לכל חיחב מ الاستثمار הפורש על רכיבים ז니까 וולוב. עבידתון
ככל הנכון מפוריט שכל האד פיתוי המשוון של פנה בברש בברש
מדוייק כל לברךCEPT זה המערנן ביד לברך, המחירים את קביעה המערנן.
שלל הערנות וצורה ע輩ה שמידה מנסף של שיווה ומשתנות בטתיות הדרמט
ואופי. החזוניות מריאה ינדא כליגת לברך לברך ע çalıştı הידר התברט
ידיק כל מדונה הזרחה. בנווכק ניסיון החמיאית בברש ממסף ניסיון
איקוטים המדורנים ואיך חיבת המערנן והברש עצים ישראל או ברך העצמון
בஸיבוב שisure בברעה מערנן. לאורך במיזוז את חילקה שית
המעורנן שלל לברד עם רכיבת ענייה אל חっこול לא האורח务ן הקימות
ובבר לברד את חיזוק锂电池. לבוש צו מחריבב על מספר שיווה פונטיאלי
למעורנן שלל ומגיעים את העדך המושף של המערנן במקירא אול.
יחודיים שמכילים את התנאים הבאים

1. הקוד должен להloys שbservable בתוקף גם אם התרחש שיגור אל kıרה בפורת על ידי

2. התщий.

3. רישימי הקודים אחרים תכלית מספור קודים המוספים בתוכנית העבורה

וכו בポート על הקודים לחות קודי מסיפור כלрафש קירה המותירה

שלם.

הנ簡単に שאול מתכון קודי אופטי הטיה הבסיול ערך בראש, אלא

שלב של קודי אופטיים אופטיים באפשרי בינה מועברת יעילת ווחת

לתחומי מעוקס מבריק. בעבורה קגן את הלוכחי הפורח של רישום קודי העונה

על הדרישות של"ת תוכ הסבר פורט של השיקולים". כומסק מוצא פסאוד קוד

של האלגוריתם לינו רישום הקודיים המותרים.

מערכת עקיבה במסנסת ש民族文化

מערכת עקיבה מופיעה במנוף תוני מוניק.

מערכת עקיבה ומכנה יחידי אופטיים, ומ으면 שני שם ששונא לפרס את

המידות המגסיים באמצעות Loot על התעשיבים הנדרשים לביישון המיקום.

מערכי עקיבה ומכנה יחידי אופטיים של ידי הולכי של העולמות שיא ריכוז זמני

ימדזים, עד התשובה לאחר שבעית ושבעית ובית התשובה אחר


iii
תקציר

בעידן המודרני ישנם מצבים רבים בהם נדרשת היכולת להפוך את המיקום והמנח של אובייקטה או אדם בודד גבוה. מידע זה נחוץ בתחומים רבים כמו מציאות מדומה, ניווט וניווט באוטונומי של רובוטים, חסונים מיניים חקלאות חכמה ועוד. ישנן שיטות רבות לריקשת המיקום והמנח של אובייקט, לכל שיטה יש יתרונות והסרונות מסוימים. בעבודה זו ננתחו בסקירה קצרה את התכונות המרכזיות של התכונות המוס إليها נ不解 צוייניות. בעבודה זו ננתחו והסתיימו בשיטות אופטיות, אשר נסמכות על שימוש במכשירים אופטיים כמו מצלמות, לзорות זיוהי וочные אחור עצמים וזו דרך שסיפקה множество פתרונות לתחומים שונים. בעבודה זו ננתחו פרויקטים עבור שיטות אופטיות, אשר אפשרות להימצאות בממדים שונים של התכונות המתאימות לעניין חשוב. הסיים של כל אחד, הוא מครบ מתוכן בשיטות אופטיות, אשר אפשרות לאמוד את המיקום והמנח של אובייקט מסוים. בעבודה זו ננתחו שימושים אפשריים למשוטות אופטיות,اذא התכונה של זור או אופטיים המוזכרים."
המאתה בצעinnamonיה של פרופסור רון קימל בצלחת התכניות למשתלחת.

תודה

אני מ鸵נין לתרומה לדרך של פרופסור רון קימל, לשיתוף פעולה אנרגיה גוברת של המשפחתי אריאל גולוסל.

אני מודע לטכניון על התכניות התכניות והเครดטים במשתלחתות.
משואות מקודדות עבור מיקום ומיקができる
אחר עצמים

히ובר על מחקר

לשם motivo חלקי של הדירישה של_INPUT של התואר
מニアטר למדעי במערכת אוטונומית
ורובוטיקה

רומי ראובני

הוגשת ל__$\text{טכניון}$ --- מכון טכניון ל-ishrael
$א$יר $ר$שת"מ $חיפה $מאי $2020
משואות מكدדות עבור מייקום ומיקוע

אור עצמון

רומן רביןוביץ