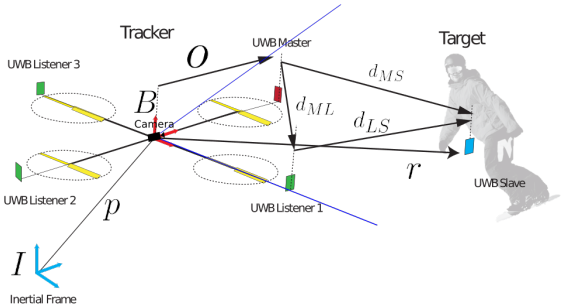
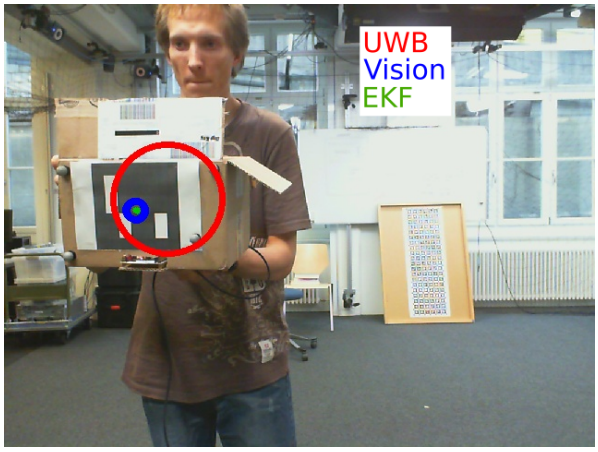
Robust object tracking in 3D by fusing ultra-wideband and vision





Andreas Ziegler

Semester Project
June 2016

Supervisors:
Benjamin Hepp
Prof. Dr. Otmar Hilliges
Prof. Dr. Luc Van Gool

ETH zürich Advanced Interactive Technologies BIWI Computer Vision

Abstract

Motivation Object tracking is an important part for many applications especially for robotic systems interacting with humans.

Problem statement Ultra-wideband (UWB) systems as well as vision based object trackers are widely known and used. Both of the systems have their advantages and disadvantages. UWB systems can provide the location of an object in 3D with an accuracy of approximate $10cm$ whereas vision based object trackers can only provide the location of an object in 2D pixel coordinates but with a more precise accuracy than UWB systems.

Approach So why not combine these two sources of information? Exactly this concept should be developed and evaluated in this semester project. The 3D position measured by the UWB system should be fused with the 2D pixel coordinates of a visual object tracker with an Extended Kalman Filter (EKF). A re-detection mechanism for the visual tracker should be implemented in addition to increase the usability as well as the stability of the system.

Result The proposed system shows a significantly better accuracy compared with the 3D positions measured by the UWB system. This proof of concept enables to apply this system to a wide range of applications and also allows further extensions.

Acronyms

UWB Ultra-wideband

EKF Extended Kalman Filter

KCF Kernelized Correlation Filters

ROS The Robot Operating System

Contents

List of Figures

List of Tables

List of Algorithms and Listings

1

Introduction

Object tracking is an important building block for many interactive systems, especially for robotic systems interacting with humans. State-of-the-art robust approaches detect and recognize a small number of pre-defined object types like humans, birds or cars which were learned beforehand during the training of the detector. As for many applications tracking of arbitrary objects is desirable, i.e. a bottle, a hand, an animal, a face etc., these object trackers are not enough flexible. Online visual tracking on the other hand deals with the challenging task of tracking an object based on an initial bounding box in an image. This faces the fundamental problem of very limited labeled data and as a consequence any such tracking approach has to balance plasticity and drift, in particular when an object should be re-detected after loss of tracking. In this semester project a new approach is proposed. A fusion of Ultra-wideband (UWB) and visual measurements to track an object in 3D by fusing both modalities in a principled manner.

This semester project focuses on visual tracking with correlation filters. This is typically susceptible to drift and has low accuracy in the radial direction. The aim is to compensate for this with an additional existing sensor modality based on multilateration with UWB signals. A single tracker consists of multiple UWB units that track a single UWB unit on the target, providing a 3D position and covariance of the target. Because of the arrangement of the UWB units the tangential accuracy of the UWB position is relatively low. The visual tracker will provide a 2D measurement. Together both observations should be fused in a principled manner using an Extended Kalman Filter (EKF) that will combine the strength of both approaches.

2

Related Work

2.1 Ultra-wideband (UWB)

In this semester project, an UWB system as described in [Tobias Naegeli 2016] was used to get the location and the velocity of the object, relative to the measurement setup. This UWB system is manually calibrated with the help of a motion capture system (VICON) and has an accuracy of $\approx 10cm$ but also provides a distance in the z-direction in contrast to the used visual tracker described next, which only gives x- and y-coordinates in pixels. The adaption of the UWB system used in this semester project is described in chapter 3.

2.2 Vision based object tracking

Because the currently best performing vision based object trackers are mostly built with Kernelized Correlation Filters (KCF), an implementation of the KCF tracker [Henriques et al. 2015] was used as vision based object tracker[Haag 2015]. It implements the KCF tracker, with FHOG features proposed in [Felzenszwalb et al. 2010] and a few adaption, more precisely speaking, the default scale adaption proposed in [Danelljan et al. 2014a], a more robust filter update scheme from [Danelljan et al. 2014b] and a target loss functionality presented in [Bolme et al. 2010].

KCF exploit the fact that translated and scaled patches, as normally used to train discriminative classifiers, are riddled with redundancies and therefore can be represented as a circulant matrix. As is standard with correlation filters, the input patches are weighted by a cosine window to smoothly remove discontinuities at the image boundaries, caused by the cyclic shifts. Circulant matrices can then be diagonalized with the Discrete Fourier Transform, which reduces storage as well as computation by several orders of magnitude. As demonstrated in

[Henriques et al. 2015] kernel regression has the same complexity as its linear counterpart with this approach.

Thanks to this properties of the circulant matrices, the KCF tracker can be implemented with only a few lines of code. The bulk of the functionality of the KCF tracker is implemented in three functions: "train" which implements Equation 2.1, "detect" which implements Equation 2.2, and "kernel_correlation" which implements Equation 2.3.

$$\hat{\alpha} = \frac{\hat{y}}{\hat{\vec{k}}^{\vec{x}\vec{x}} + \lambda} \tag{2.1}$$

where $\vec{k}^{\vec{x}\vec{x}}$ is the first row of the kernel matrix $\mathbf{K} = C(\vec{k}^{\vec{x}\vec{x}})$, and $\hat{}$ denotes the DFT of a vector.

$$\hat{\vec{f}}(z) = \hat{\vec{k}}^{xz} \odot \hat{\vec{\alpha}} \tag{2.2}$$

where $\vec{k}^{\vec{x}\vec{z}}$ is the kernel correlation of $\vec{x}$ and $\vec{z}$.

$$\vec{k}^{\vec{x}\vec{x}'} = exp\Big(-\frac{1}{\sigma^2}(\|\vec{x}\|^2) + \|\vec{x}'\|^2 - 2\mathcal{F}^{-1}(\sum_c \hat{\vec{x}}_c \odot \hat{\vec{x}}'_c)\Big) \tag{2.3}$$

For more detailed information and also the derivation of the presented equations, please see [Henriques et al. 2015].

As this method is much faster than other algorithms and can be implemented in only a few lines of code, it makes it really suitable to use on low-power devices as it is common in the robotics area.

2.3 Extended Kalman Filter (EKF)

The Extended Kalman Filter (EKF) [Chu 2009] in this semester project is used to fuse the 3D position of the tracked object from the UWB system and the 2D pixel coordinates from the vision based object tracker. The EKF model and the update steps are described in detail in chapter 4.

3

Setup

The first part of this chapter describes how the different elements of the setup are brought together. This includes the following steps:

- How the camera was calibrated
- How the camera and the UWB system were mounted
- The matching proceeder which matches the two coordination systems together

This part also explains with the help of which frameworks the mentioned steps were done.

In the second part of this chapter the ROS setup (ROS nodes and messages), which was used, is described.

3.1 Camera calibration

It is well known that cameras, as the one used in this semester project, suffer from distortion (radial as well as tangential) [Szeliski 2010], [Bradski]. The required constants to remove this distortions as well as the intrinsic camera matrix, are determined by the camera calibration.

In this semester project, the camera calibration framework of the OpenCV library [Bradski] was used, which is easy to handle and well known to work properly.

3.2 UWB and camera mounting

The UWB system and the camera are mounted in a way, that the center of their relative coordinate systems have a fixed, not too large displacement. This setup is shown in Figure 3.1. In this setup the origin of the camera coordinate system is a bit above of the one from the UWB system (indicated by the blue arrow in Figure 3.1), which is positioned in the middle of the monitor screen.

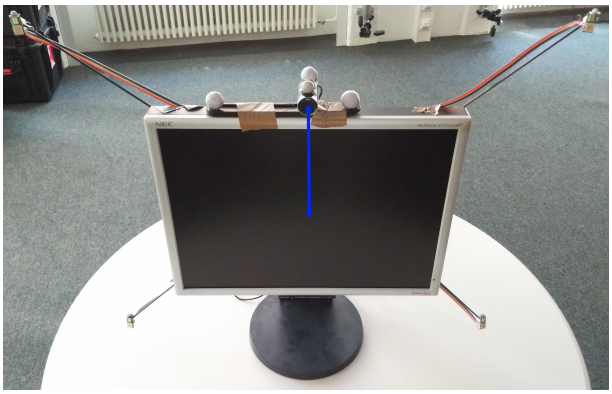


Figure 3.1: *UWB and camera setup.*

3.3 Matching frames

The coordinate system of the UWB setup and the one of the camera are not the same, as sketched in Figure 3.2. As the goal of this semester project is to fuse the locations provided by the UWB system and by the visual tracker, the locations must be transformed from one to the other. To achieve this, the location of the object in 3D must be available from both systems (UWB and vision).

For the vision system the ArUco [Garrido-Jurado et al. 2014] library and an ArUco marker were used to get the 3D coordinates of the object, as explained in subsection 3.3.1.

The scale, the translation as well as the rotation between the two coordinate systems were then determined with the matching proceeder, explained in subsection 3.3.3, with the help of the Kabsch algorithm [Kabsch 1976].

3.3.1 ArUco

ArUco [Garrido-Jurado et al. 2014] presents itself as a minimal library for augmented reality applications, based on OpenCV. ArUco is a marker system specialized for camera pose estimation in different applications such as augmented reality, robot localization, etc. ArUco contains an algorithm for the generation of markers as well as marker boards and an algorithm for the automatic detection of markers. A third contribution of ArUco is a solution to the occlusion problem in augmented reality applications which is not of interest for this semester project.

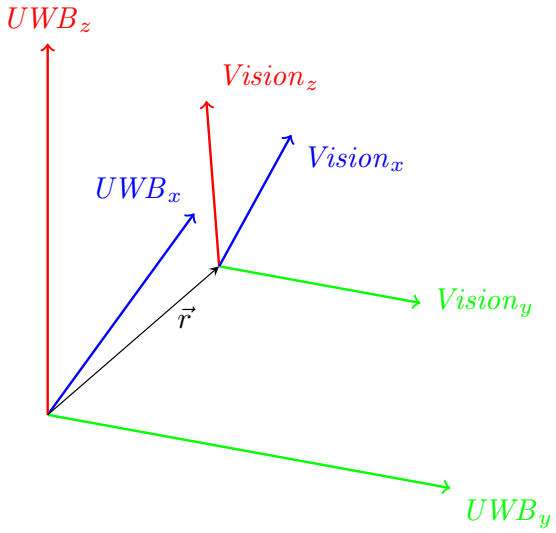


Figure 3.2: *The two coordinate systems relative to each other.*

To get 3D vision coordinates, an ArUco marker was mounted on the object, as shown in Figure 3.3, right in front of an UWB target and an application was written which uses the ArUco library to read out the 3D coordinates of detected ArUco markers and saves them for later usage by the matching proceeder.

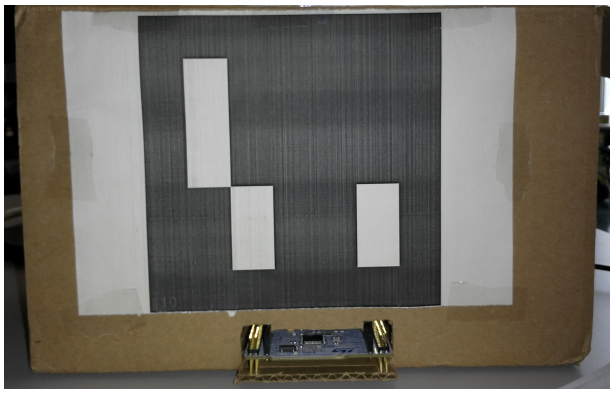


Figure 3.3: *ArUco marker and UWB target.*

3.3.2 Kabsch

The Kabsch algorithm [Kabsch 1976] calculates the optimal rotation matrix and translation vector that minimize the root mean squared deviation between two sets of corresponding points. In this semester project the Kabsch algorithm determines the optimal rotation matrix and translation vector between the coordination systems of the UWB system an the one of the camera.

3.3.3 Matching proceeder

For the matching proceeder a Matlab script was written, shown in Listing 3.1, which performs the following steps:

- Calculates the mean of the UWB and the vision (ArUco) coordinates
- Centralizes the coordinates of both systems
- Calculates the scale from the centralized coordinates
- Scales the vision (ArUco) coordinates
- Executes the Kabsch algorithm to calculate the rotation matrix $\mathbf{U}$, the translation $\vec{r}$ and the least root mean squared error $lrms$

To transform 3D points from the UWB coordinate system to the vision coordinate system the translation $\vec{r}$ is applied to the 3D coordinate and then the rotation matrix $\mathbf{U}$ and the scale are applied as shown in Equation 3.1

$$\begin{bmatrix} x_{Vision} \\ y_{Vision} \\ z_{Vision} \end{bmatrix} = \frac{1}{scale} \cdot \mathbf{U} \left(\begin{bmatrix} x_{UWB} \\ y_{UWB} \\ z_{UWB} \end{bmatrix} - \vec{r} \right) \tag{3.1}$$

To transform the covariance matrix $\mathbf{C}$ from the UWB coordinate system to the vision coordinate system, a new rotation matrix $\mathbf{U}' \in \mathbb{R}^{6\times6}$ has to be applied to the covariance matrix in the UWB coordinate system like

$$\mathbf{C}' = \frac{1}{scale^2} \mathbf{U}' \mathbf{C} \mathbf{U}'^T \tag{3.2}$$

where

$$\mathbf{U}' = \begin{bmatrix} \mathbf{U} & \mathbf{0} \\ \mathbf{0} & \mathbf{U} \end{bmatrix} \tag{3.3}$$

The determined rotation matrix $\mathbf{U}$ and the translation vector $\vec{r}$ applied on a set of measured points by the UWB systems together with the set of measured points by ArUco results in a data set as shown in Figure 3.4

Listing 3.1: *Matching proceeder*

```
% Calculate mean
mean_uwb = mean(uwb(1:3,:), 2);
mean_aruco = mean(aruco(1:3,:), 2);

% normalize data
aruco_centred(1,:) = (aruco(1,:) - ...
                    mean_aruco(1)*ones(1,length(aruco(1,:))));
aruco_centred(2,:) = (aruco(2,:) - ...
```

```
                    mean_aruco(2)*ones(1,length(aruco(1,:))));
aruco_centred(3,:) = (aruco(3,:) - ...
                    mean_aruco(3)*ones(1,length(aruco(1,:))));

uwb_centred(1,:) = uwb(1,:) - ...
                    mean_uwb(1)*ones(1,length(uwb(1,:)));
uwb_centred(2,:) = uwb(2,:) - ...
                    mean_uwb(2)*ones(1,length(uwb(2,:)));
uwb_centred(3,:) = uwb(3,:) - ...
                    mean_uwb(3)*ones(1,length(uwb(3,:)));

% calculate scale
scale = norm(uwb_centred)/norm(aruco_centred);

% Scale aruco
aruco = scale .* aruco;

% Perform Kabsch
[U, r, lrms] = Kabsch(aruco, uwb);
```

3.4 The Robot Operating System (ROS) setup

In this semester project three different ROS setups were used to perform the tasks of recording the video from the camera as well as recording the measurements from the UWB system, and collecting the data required for the matching proceeder and for the object tracking task.

The first setup to record the video and the measurement of the UWB system is described in subsection 3.4.1.

The setup to gain the required data to perform the matching proceeder, explained in subsection 3.3.3, is described in subsection 3.4.2.

The last ROS setup that was used in this semester project, is the setup introduced in subsection 3.4.3, which is used for the main task of object tracking.

3.4.1 Recording setup

In the recording setup, shown in Figure 3.5, a rosbag node records the messages from the two nodes publish_image and uwb. The messages "/camera/video" from the publish_image node is the image stream from the camera. The node uwb sends the messages "/uwb/tracker" which consists of the position, the velocity of the target as well as their covariances.

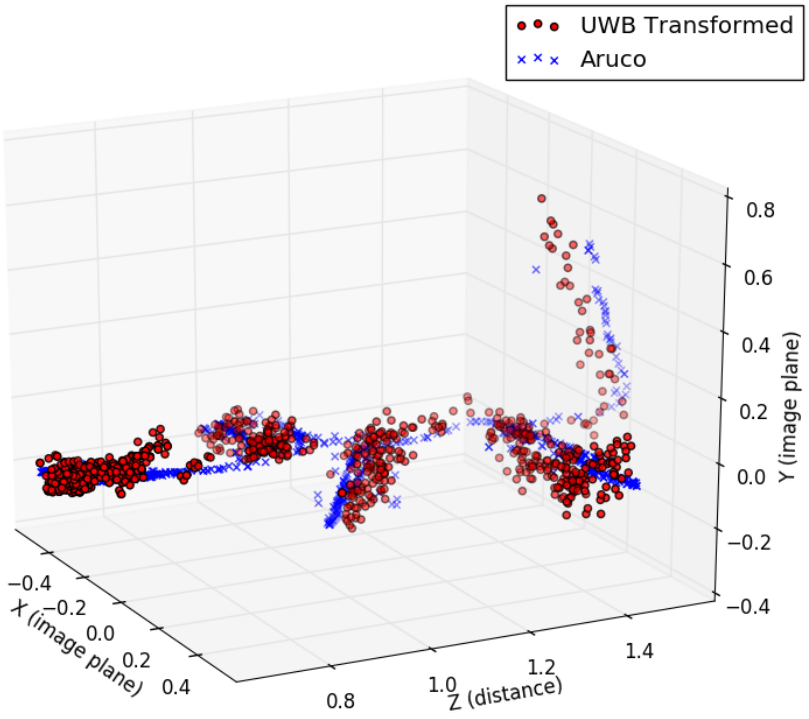


Figure 3.4: *Matching of the position set of ArUco and UWB.*

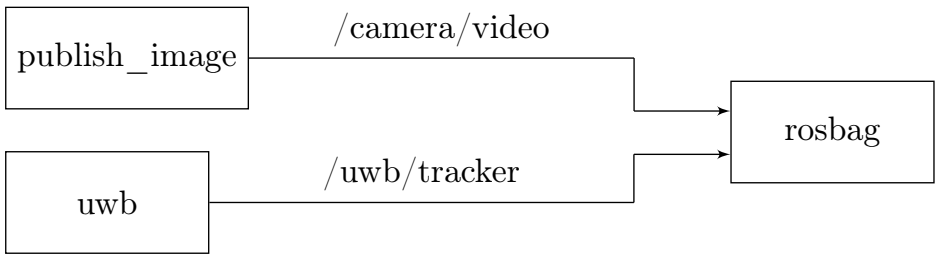


Figure 3.5: *Block diagram of the ROS nodes and messages for the recording setup.*

3.4.2 Kabsch setup

To save the required data to perform the matching proceeder described in subsection 3.3.3, the setup, shown in Figure 3.6, was used. In this ROS setup the node uwb_aruco receives the messages "/camera/video" and "/uwb/tracker". It saves the positions measured by the UWB system directly and performs the ArUco marker detection to get the positions detected by ArUco and also saves them.

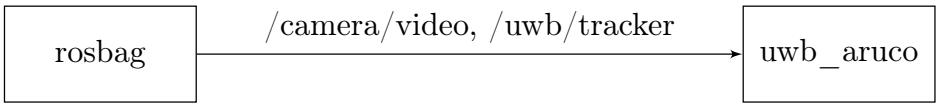


Figure 3.6: *Block diagram of the ROS nodes and messages for the Kabsch setup.*

3.4.3 Tracking setup

This ROS setup, shown in Figure 3.7, is used for the main task of tracking an object. In this setup, the node vision_tracker performs the object tracking on the images it receives from the node rosbag in the messages "/camera/video". It also receives the fused positions in the messages "/fusing/coordinates" from the EKF for the case, that the vision tracker loses the object and has to re-detect it. The node vision_tracker then publishes the position of the tracked object in the messages "/vision_tracker/vision_coordinates". The node fusing receives the positions measured by the UWB system in the messages "/uwb/tracker" as well as the positions of the object detected by the node vision_tracker in the messages "/vision_tracker/vision_coordinates". With an EKF, described in chapter 4, the node fusing estimates the position of the tracked object.

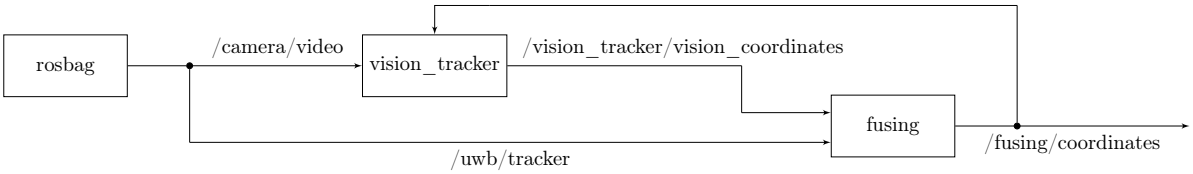


Figure 3.7: *Block diagram of the ROS nodes and messages for the tracking setup.*

4

Fusing of the two measurement sources

The position of the object is estimated with an Extended Kalman Filter (EKF), using a model of the system and the measurements from the UWB system and also from the visual tracker. A block diagram of this pipeline is shown in Figure 4.1.

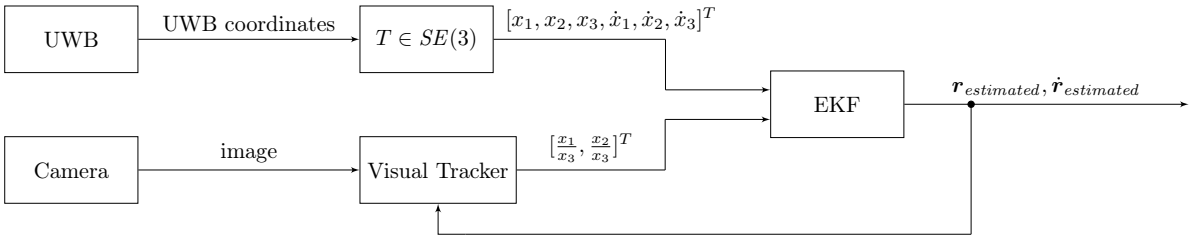


Figure 4.1: *Block diagram of the EKF pipeline.*

4.1 Extended Kalman Filter (EKF)

4.1.1 System Model

The state $\vec{x} = [\vec{r}, \dot{\vec{r}}]^T$ of our model consists of a position $\vec{r} \in \mathbb{R}^3$ and a velocity $\dot{\vec{r}} \in \mathbb{R}^3$. The discrete process model with timestep ΔT is given by

$$\vec{x}(k) = \vec{q}_{k-1}(\vec{x}(k-1), \vec{v}(k-1)) \tag{4.1}$$

with

$$q_{k-1}(\vec{x}(k-1), \vec{v}(k-1)) = \mathbf{B}\vec{x}(k-1) + \vec{v}(k-1) \tag{4.2}$$

where

$$\mathbf{B} = \begin{pmatrix} \mathbf{I}_3 & \Delta\mathbf{T} \\ \mathbf{0} & \mathbf{I}_3 \end{pmatrix} \tag{4.3}$$

and

$$\vec{v}(k-1) \sim \mathcal{N}(\vec{0}, \mathbf{Q}) \tag{4.4}$$

where

$$\mathbf{Q} = \begin{pmatrix} \mathbf{I}_3 & \mathbf{0} \\ \mathbf{0} & \mathbf{q}_v \end{pmatrix} \tag{4.5}$$

and $\mathbf{q}_v \in \mathbb{R}^{3\times3}$ is the velocity covariance of the process noise.

There are two possible measurements. The first measurement is a direct measurement of the state $\vec{x}$ from ultra-wideband (UWB) multilateration:

$$\vec{z}_1(k) = \vec{h}_{k,1}(\vec{x}(k), \vec{w}_1(k)) \tag{4.6}$$

with

$$\vec{h}_{k,1}(x(k), w_1(k)) = \mathbf{H}_1\vec{x}(k) + \vec{w}_1(k) \tag{4.7}$$

where

$$\mathbf{H}_1 = \mathbf{I}_6$$

and

$$\vec{\omega}_1(k) \sim \mathcal{N}(\vec{0}, \mathbf{R}_1)$$

where $\mathbf{R}_1 \in \mathbb{R}^{6\times6}$ is the covariance of the UWB measurement. The second measurement is a projection of the position $\vec{r}$ as seen by a camera:

$$\vec{z}_2(k) = \vec{h}_{k,2}(\vec{x}(k), \vec{w}_2(k)) \tag{4.8}$$

with

$$\vec{h}_{k,2}(\vec{x}(k), \vec{w}_2(k)) = \mathbf{H}_2(\vec{x}(k)) + \vec{w}_2(k) \tag{4.9}$$

where

$$\mathbf{H}_2(\vec{x}) = [\frac{x_1}{x_3}, \frac{x_2}{x_3}]^T \in \mathbb{R}^2 \tag{4.10}$$

and

$$\vec{\omega}_2(k) \sim \mathcal{N}(\vec{0}, \mathbf{R}_2) \tag{4.11}$$

where $\mathbf{R}_2 \in \mathbb{R}^{2\times2}$ is the covariance of the camera measurement.

4.1.2 The Extended Kalman Filter steps

There are two steps which have to be performed in an iterative fashion, when running the EKF.

Step 1: Prior update/Prediction step

In the first step, a prediction for the mean of the states $\hat{\vec{x}}_p(k)$ and the co-variance matrix $\mathbf{P}_p(k)$ is calculated from the linearized system model.

$$\hat{\vec{x}}_p(k) = q_{k-1}(\hat{\vec{x}}_m(k-1), 0) = \mathbf{B}\hat{\vec{x}}_m(k-1) \tag{4.12}$$
$$\mathbf{P}_p(k) = \mathbf{A}(k-1)\mathbf{P}_m(k-1)\mathbf{A}^T(k-1) + \mathbf{L}(k-1)\mathbf{Q}\mathbf{L}^T(k-1) \tag{4.13}$$

where

$$\mathbf{A}(k-1) = \frac{\partial q_{k-1}(\hat{\vec{x}}_m(k-1), 0)}{\partial \vec{x}} \tag{4.14}$$
$$= \frac{\partial}{\partial \vec{x}}\big(\mathbf{B}\vec{x}(k-1) + \vec{v}(k-1)\big) \tag{4.15}$$
$$= \mathbf{B}(k-1) \tag{4.16}$$
$$\mathbf{L}(k-1) = \frac{\partial q_{k-1}(\hat{\vec{x}}_m(k-1), 0)}{\partial \vec{v}} \tag{4.17}$$
$$= \frac{\partial}{\partial \vec{v}}\big(\mathbf{B}\vec{x}(k-1) + \vec{v}(k-1)\big) \tag{4.18}$$
$$= \mathbf{I}_6 \tag{4.19}$$

and with the initial values $\hat{\vec{x}}_m(0) = x_0$ and $\mathbf{P}_m(0) = \mathbf{0}$. Therefore equation 4.13 becomes

$$\mathbf{P}_p(k) = \mathbf{B}(k-1)\mathbf{P}_m(k-1)\mathbf{B}^T(k-1) + \mathbf{Q}$$

Step 2: A posteriori update/Measurement update step

In the second step, the information gained from the measurements is used to perform an a posteriori update, resulting in a updated mean of the states $\hat{\vec{x}}_m(k)$ and an updated co-variance matrix $\mathbf{P}_m(k)$.

$$\mathbf{K}(k) = \mathbf{P}_p(k)\mathbf{H}^T(k)\Big(\mathbf{H}(k)\mathbf{P}_p(k)\mathbf{H}^T(k) + \mathbf{M}(k)\mathbf{R}(k)\mathbf{M}^T(k)\Big)^{-1} \tag{4.20}$$
$$\hat{\vec{x}}_m(k) = \hat{\vec{x}}_p(k) + \mathbf{K}(k)\Big(\vec{z}(k) - \begin{bmatrix} \vec{h}_{k,1}(\hat{\vec{x}}_p(k), 0) \\ \vec{h}_{k,2}(\hat{\vec{x}}_p(k), 0)) \end{bmatrix}\Big) \tag{4.21}$$
$$\mathbf{P}_m(k) = \big(\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)\big)\mathbf{P}_p(k) \tag{4.22}$$

where

$$\mathbf{H}(k) = \left[\frac{\partial h_k(\hat{\vec{x}}_p(k),0)}{\partial \vec{x}}\right] \tag{4.23}$$

$$= \begin{bmatrix} \frac{\partial}{\partial \vec{x}}\left(\mathbf{H}_1\hat{\vec{x}}(k) + \vec{\omega}_1(k)\right) \\ \frac{\partial}{\partial \vec{x}}\left(\mathbf{H}_2(\hat{\vec{x}}(k)) + \vec{\omega}_2(k)\right) \end{bmatrix} \tag{4.24}$$

$$= \begin{bmatrix} \mathbf{I}_6 & & & & & \\ \frac{1}{\hat{x}_3} & 0 & -\frac{\hat{x}_1}{\hat{x}_3^2} & 0 & 0 & 0 \\ 0 & \frac{1}{\hat{x}_3} & -\frac{\hat{x}_2}{\hat{x}_3^2} & 0 & 0 & 0 \end{bmatrix} \tag{4.25}$$

$$\mathbf{M}(k) = \left[\frac{\partial h_k(\hat{\vec{x}}_p(k),0)}{\partial \vec{w}}\right] \tag{4.26}$$

$$= \begin{bmatrix} \frac{\partial}{\partial \vec{\omega}}\left(\mathbf{H}_1\hat{\vec{x}}(k) + \vec{\omega}_1(k)\right) \\ \frac{\partial}{\partial \vec{\omega}}\left(\mathbf{H}_2(\hat{\vec{x}}(k)) + \vec{\omega}_2(k)\right) \end{bmatrix} \tag{4.27}$$

$$= \mathbf{I}_8 \tag{4.28}$$

Therefore equation 4.20 becomes

$$\mathbf{K}(k) = \mathbf{P}_p(k)\mathbf{H}^T(k)\Big(\mathbf{H}(k)\mathbf{P}_p(k)\mathbf{H}^T(k) + \begin{bmatrix} \mathbf{R}_1 & \mathbf{0} \\ \mathbf{0} & \mathbf{R}_2 \end{bmatrix}\Big)^{-1}$$

4.1.3 Implementation

The implementation of the EKF was done in a Python script within the ROS environment. The ROS nodes used in this semester project and their function are listed in Table 4.1.

For more details as well as for the used parameters, please see [Ziegler 2016].

Node name	File/Script name	Function
publish_image	publish_image_node	Publishes images, recorded with a camera, as ROS messages.
uwb	uwb_tracker_node	Publishes the information provided by the UWB system as ROS messages.
uwb_aruco	uwb_aruco_node.py	Receives ROS messages from the uwb and publish_image node, detects ArUco markers and saves the locations provided by both systems into a hdf5 file.
vicon_aruco	vicon_aruco_node.py	Receives ROS messages from the VICON system and publish_image node, detects ArUco markers and saves the locations provided by both systems into a hdf5 file.
vision_tracker	vision_tracker_node	The KCF tracker publishes the 2D pixel coordinates as ROS messages as well as boolean ROS messages which indicate, if the object is lost or not.
fusing	fusing_node.py	The EKF receives ROS messages from the vision_tracker and uwb node and publishes the fused positions as ROS messages.
visualization	visualization_node	Displays the positions provided by the UWB, the vision tracker and by the EKF on top of the picture.
validation	validation_node.py	Records the required information to perform the evaluation.

Table 4.1: *Table with the implemented ROS nodes and their function.*

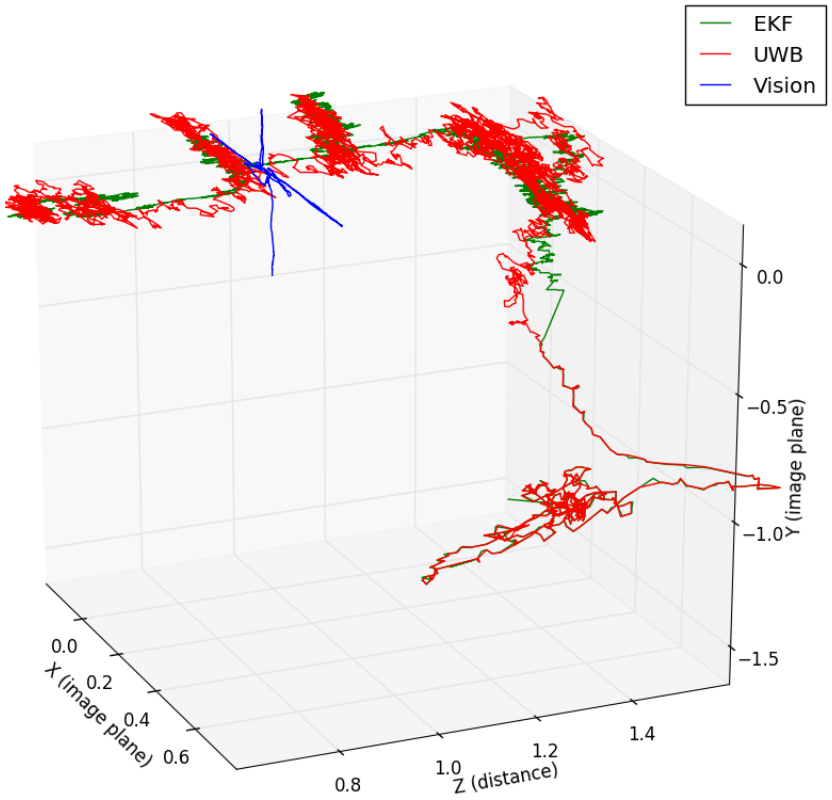


Figure 4.2: *3D plot of the points measured by the UWB (red), the points detected by the visual tracker (blue) and the fused positions (green).*

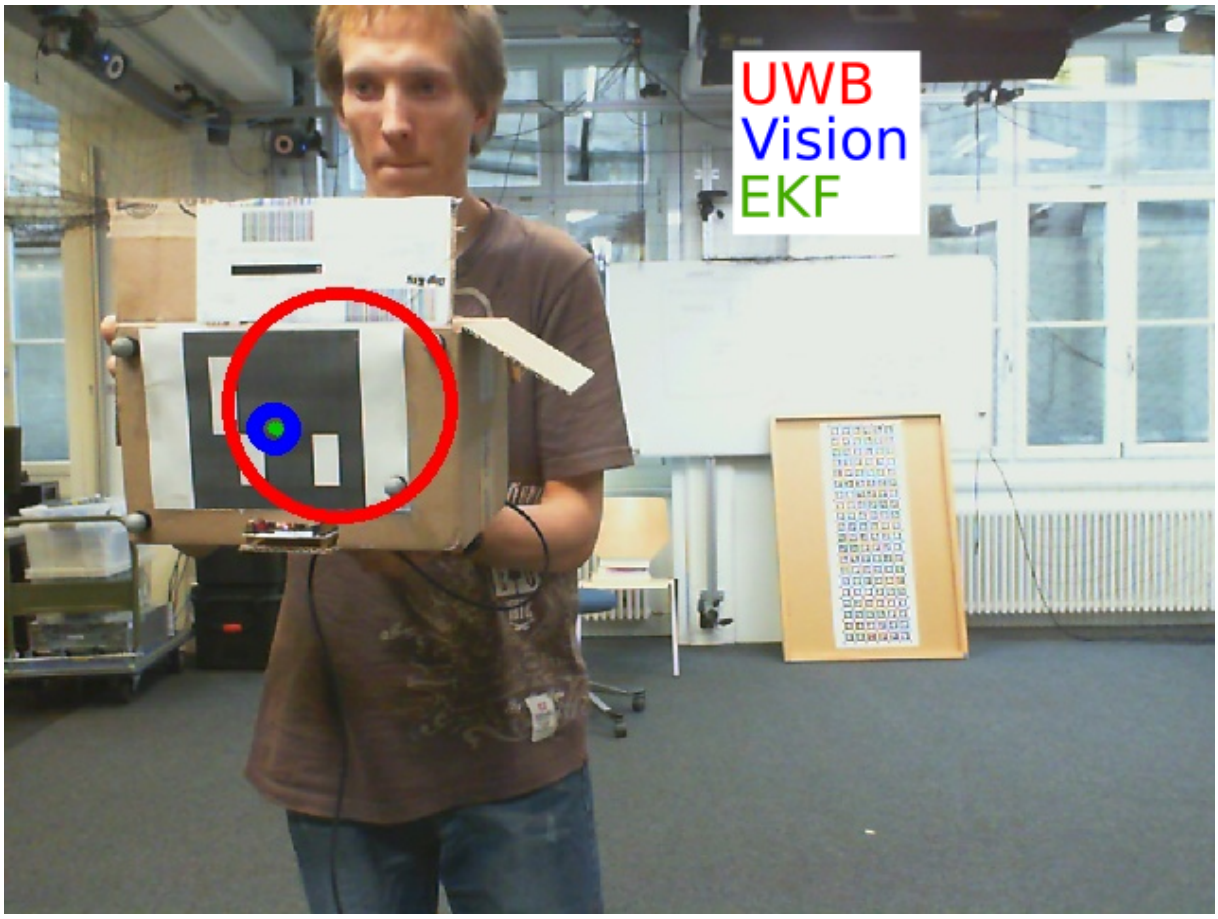


Figure 4.3: *Recorded picture with the point measured by the UWB (red), the point detected by the visual tracker (blue) and the fused position (green). The radii of the UWB and EKF circle indicate the covariances of the measured/estimated position.*

5

Re-detection of the object in the visual tracker

5.1 Motivation

The implementation of the Kernelized Correlation Filters (KCF) tracker in [Haag 2015] does not provide any sophisticated re-detection. To be able to re-detect the object, it has to pass the location, where the tracker has lost it. Otherwise the tracker is not able to re-detect the object as it does not search at other locations than the one of the object's last appearance.

As with our system the Extended Kalman Filter (EKF) gets the location of the object measured by the Ultra-wideband (UWB) system also if the vision based tracker can't detect the object, this information can be used to re-detect the object in the image.

This information feedback from the EKF to the Visual Tracker is shown in Figure 4.1.

5.2 Method

The re-detection mechanism works as follows. If the object can't be detected by the KCF tracker, the re-detection mode is activated and the detection parameters (PSR and "response threshold") are adapted. The KCF tracker now takes the 2D location provided by the EKF and tries to detect the object, with the adapted detection parameters, at these new locations. The KCF tracker needs to detect the object in five consecutive frames to deactivate the re-detection mode and to reset the detection parameters to the standard values. This ensures, that the visual tracker re-detects the object and not something else. The whole mechanism is presented in the

pseudo code 5.1.

Algorithm 5.1 Re-detection mechanism

```
1:  while true do
2:      if target not detected then
3:          Adapt PSR and response threshold
4:          Set re-detection = true
5:      else
6:          if Object was detected in 5 consecutive frames then
7:              Reset PSR and response threshold
8:              Set re-detection flag = false
9:      if re-detection flag = true then
10:         Take 2D position from EKF
```

6

Experiments and Results

6.1 Experiments

To measure the accuracy of the developed system, a ROS setup similar as described in subsection 3.4.3 was used. The Extended Kalman Filter (EKF) works in the camera coordinate system with approximately $80Hz$. To eases the whole handling, the experiments were recorded with rosbag and then afterwards off-line evaluated with a separate Python script.

To have a ground truth to compare with, the 3D coordinates of the tracked object measured by a motion capture system (VICON) were additionally recorded.

For the comparison between the accuracy of the UWB system and the one of the EKF, both systems were compared with the mentioned ground truth by the root mean squared error

$$\textit{rmse} = \sqrt{\frac{1}{N-1}\sum_{i}^{N}\left((x_{m,i}-x_{V,i})^2+(y_{m,i}-y_{V,i})^2+(z_{m,i}-z_{V,i})^2\right)}$$

and by the root mean squared error of only the x and y axis

$$\textit{rmse}_{xy} = \sqrt{\frac{1}{N-1}\sum_{i}^{N}\left((x_{m,i}-x_{V,i})^2+(y_{m,i}-y_{V,i})^2\right)}$$

where the subfix m stands for measurement and represents coordinates which either come from the UWB system or from the EKF. The subfix V on the other hand stands for VICON which is the ground truth in this experiments. N is, as common for *rmse*, the number of points in the data set.

6.2 Results

In several recorded experiments, the *rmse* and the $rmse_{xy}$ of the EKF were significantly lower compared with the *rmse* and the $rmse_{xy}$ of the UWB system. The results of some experiments are listed in Table 6.1. A 3D plot of the measured coordinates by the VICON system (ground truth), the UWB system and the coordinates fused by the EKF from the data set of experiment number 1 is shown in Figure 6.1. Figure 6.2 shows the dataset of experiment number 5, where the visual tracker has to re-detect the object.

Experiment number	*rmse* of UWB	*rmse* of EKF	$rmse_{xy}$ of UWB	$rmse_{xy}$ of EKF
1	0.0667	0.0350	0.0621	0.0270
2	0.0771	0.0364	0.0734	0.0260
3	0.1304	0.0379	0.1275	0.0312
4	0.1169	0.0344	0.1126	0.0265
5*	0.1273	0.1195	0.1159	0.1055

Table 6.1: *Table with listed rmse and $rmse_{xy}$ of the UWB system and the EKF for the different experiments.*

* In experiment number 5 the object goes several times out of camera view and therefore the visual tracker has to re-detect the object. During the periods in which the visual tracker has lost the object, the resulting positions of the EKF are not longer better than the positions measured by the UWB system.

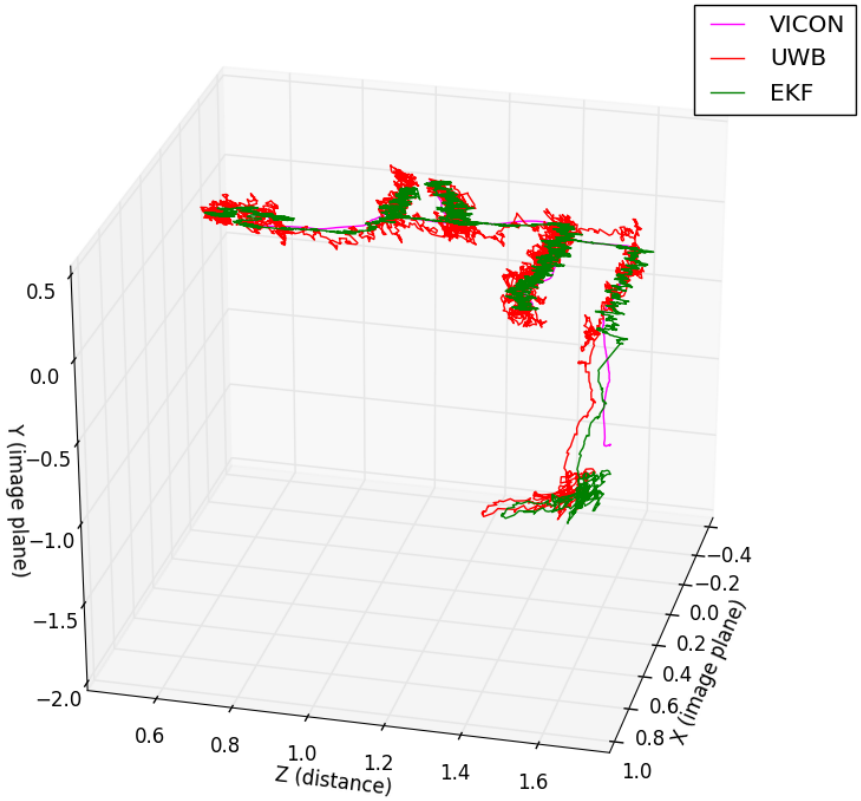


Figure 6.1: *3D plot of the coordinate points measured by the UWB system (red), the points measured by the VICON system (magenta) and the fused positions (green) of the experiment number 1.*

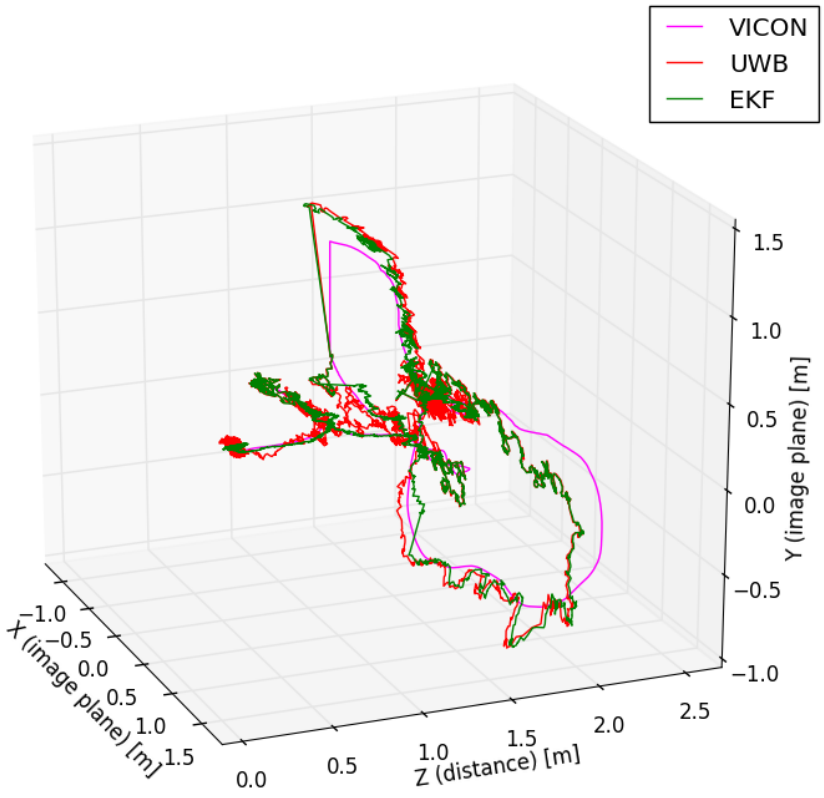


Figure 6.2: *3D plot of the coordinate points measured by the UWB system (red), the points measured by the VICON system (magenta) and the fused positions (green) of the experiment number 5 in which the object goes several times out of the camera view and the visual tracker has to re-detect the object.*

7

Conclusion and Outlook

7.1 Conclusion

In this semester project I have explored the proposed approach of fusing UWB and vision for a robust object tracking in 3D.

The results show that the proposed method of fusing less accurate 3D coordinate measurements from the UWB system with more precise 2D pixel coordinate measurements from the vision based tracker with an Extended Kalman Filter (EKF) has a significantly improved accuracy compared to the coordinates measured solely by the UWB system. The idea of combining these two sources has therefore proven to be beneficial. The additionally implemented re-detection mechanism for the visual tracker also performs well under normal conditions.

This semester project was meant to be a proof of concept. The proposed method could be applied in many applications in different fields as robotics, human-computer-interaction, entertainment, rescue, etc., to mention only a few.

As part of this semester project I have also implemented the approach in a modular fashion within the ROS environment and made it public accessible on GitHub[Ziegler 2016].

7.2 Limitations

The current EKF implementation does not consist of an outlier-detection. Since the standard EKF is not robust to outliers, noisy input signals may result in an unstable state estimation.

7.3 Outlook

The UWB system used in this semester project runs with a frequency of approximately $80Hz$ as well as the implemented EKF. If an UWB system with a higher frequency would be used, the currently in Python implemented EKF won't be able to process all the measurements provided by the UWB system and by the vision based tracker. To encounter this problem, a faster implementation in C++ would be conceivable.

The proposed fusing method contains a re-detection mechanism for the cases, when the object goes out of the camera view or moves to fast. The currently implemented re-detection mechanism is however dependent of the object and the environment and some detection parameters need to be adjusted to achieve good re-detection performance. A more sophisticated re-detection system could improve the stability as well as the usability of the proposed method.

So far the used UWB system has to be calibrated manually, for example with a motion capture system (VICON). With an ArUco marker and the ArUco library [Garrido-Jurado et al. 2014] an automated calibration proceeder for the UWB system could be developed which would simplify the setup of the whole system.

Up to now, for the vision based tracker the desired object has to be marked manually by a user which restricts the usability of the system in many real-world applications. Automatic visual target detecting with the help of the information provided by the UWB system would widen the application area of the proposed system.

In this semester project only one object at a time can be tracked. In many applications tracking of multiple targets is of interest. A multi target tracking system consisting of multiple objects equipped with distinguishable UWB targets and a vision based multi target tracker would allow to track multiple object simultaneously.

Bibliography

BOLME, D. S., BEVERIDGE, J. R., DRAPER, B. A., AND LUI, Y. M. 2010. Visual object tracking using adaptive correlation filters. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

BRADSKI, G. *Dr. Dobb's Journal of Software Tools*.

2009. *Extended Kalman Filter and System Identification*. Springer Berlin Heidelberg, Berlin, Heidelberg, 108–130.

DANELLJAN, M., HÄGER, G., KHAN, F. S., AND FELSBERG, M. 2014. Accurate scale estimation for robust visual tracking. In *Proceedings of the British Machine Vision Conference BMVC*.

DANELLJAN, M., KHAN, F. S., FELSBERG, M., AND WEIJER, J. V. D. 2014. Adaptive color attributes for real-time visual tracking. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

FELZENSZWALB, P. F., GIRSHICK, R. B., MCALLESTER, D., AND RAMANAN, D. 2010. Object detection with discriminatively trained part based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 9, 1627–1645.

GARRIDO-JURADO, S., NOZ SALINAS, R. M., MADRID-CUEVAS, F., AND MARÍN-JIMÉNEZ, M. 2014. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition 47*, 6, 2280 – 2292.

HAAG, K., 2015. cf_tracking. https://github.com/klahaag/cf_tracking.

HENRIQUES, J. F., CASEIRO, R., MARTINS, P., AND BATISTA, J. 2015. High-speed tracking with kernelized correlation filters. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*.

KABSCH, W. 1976. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A 32*, 5 (Sep), 922–923.

SZELISKI, R. 2010. *Computer Vision: Algorithms and Applications*, 1st ed. Springer-Verlag New York, Inc., New York, NY, USA.

TOBIAS NAEGELI, BENJAMIN HEPP, O. H. 2016. Omni-directional person tracking on a flying robot using occlusion-robust ultra-wideband signals. *IROS 2016*.

ZIEGLER, A., 2016. Robus object tracking in 3D by fusing ultra-wideband and vision. `https://github.com/AndreasAZiegler/ROTI3DBFUWBAV`.