Hand Tracking using Kalman Filter for Safe Human-Robot Interaction

HASRET GÜMGÜMCÜ[1] & DANIEL LAUMER[1],
JAN DIRK WEGNER[1], PAUL BEARDSLEY[2] & KONRAD SCHINDLER[1]

Abstract: The goal of this bachelor thesis was to develop an algorithm to detect human activity with the use of an Intel RealSense camera and determine if the activity intrudes into a specified safety zone. The Intel RealSense is a depth camera, which means that we know the distance from the camera to the objects in the scene for every pixel. The Intel RealSense camera can also deliver tracked coordinates of hand joints which can be used as a signal of human activity. In order to be able to keep the safety zone small and at the same time reduce false positives, we use a Kalman Filter with which we are able to solve two problems at once: Predict future hand movements and at the same time filter the output to realize a smooth visualization. We were able to implement an application which is able to foresee quick hand movements and signal those in direction of the camera as potentially hazardous. The filter smooths the raw coordinates as desired and gives a nice visualization of the hand. To ensure an overall security we added another module which uses the 3D coordinates of the depth information to also check other objects than the hands against the security zone which however does not use prediction. Our developed method can be a starting point for further research purposes and a potential ground base to implement a profound safety zone for a real robot.

1 Motivation and Problem

Nowadays there is a huge change in the use and importance of vision based applications especially with automation in many big markets. Many autonomous processes need to have the ability to 'see', to conceive and understand the surroundings in order to react properly. However, there are still unsolved problems and questions in this area. Especially when we look at the research in robotics and suchlike topics, which depend on reliable vision based methods for the automation process to get along with the environment, a question arises: How can we ensure the safety of such human-robot interactions? As long as the person reacts as imagined by the programmer everything is fine, but a human is not a robot and can act unexpectedly. In this thesis we drill down on this safety problem and develop a security algorithm which protects interacting people of being hurt from a movement of the robot and at the same time the robot of getting damaged.
The initial setup for our thesis is the following: There is a robot on top of a table which is supposed to be able to interact with humans. This interaction could be for example playing board

[1] ETH Zürich, Institut für Geodäsie und Photogrammetrie, Stefano-Franscini-Platz 5, CH-8093 Zürich, Schweiz, E-Mail: hasret.g46@gmail.com, daniel.laumer@gmail.com, [jan.wegner, konrad.schindler]@geod.baug.ethz.ch
[2] Disney Research Zurich, Stampfenbachstrasse 48, CH-8006 Zürich, Schweiz, E-Mail: pab@disneyresearch.com

games. The robot has an Intel RealSense camera mounted on its top which is a so called depth camera that can deliver 3D coordinates for each pixel of the image.
The goal of this thesis is now the following:
Design an algorithm, which uses the functionalities of the Intel RealSense and filtering methods to establish a safety zone around the robot and issues a warning when something enters that zone.
Since it is a tabletop interaction, the main focus is on the hands. The Intel RealSense has a hand tracking option included, so it is possible to get the raw coordinates of 22 joints at any time a hand is in the field of view. An option would be to use these coordinates and check, if they are behind the safety zone. This approach would require a big safety zone in order to be safe, because if a hand is moving with high velocity directed to the robot, there should be enough time to shut down. Due to the fact that many hand movements will neither be directed to the robot nor reach it, it produces many false negatives. In this thesis we want to go a step further. The algorithm should distinguish if the hand is likely to enter the zone or not. To do that, a prediction of the movement is required. If the hand is still or just moving alongside the security zone, the algorithm should not issue any warning. However, if the hand movement is directed to the camera and is likely to be too close to the robot, it should shut down. As a result, the safety zone can be made much smaller and still offer the same level of security and at the same time, it allows much closer and smoother interaction with the robot itself without being interrupted with false negatives.

Fig. 1: The two possible solutions to solve the safety problem

2 Methodology

2.1 Safety Realization

To realize our application we define a safety zone. Since the field of view of the camera is limited to one direction a simple plane defined by 3 points $\boldsymbol{p}_1, \boldsymbol{p}_2, \boldsymbol{p}_3$ is sufficient. To determine if any hand joint $\boldsymbol{p}$ is entering the safety zone we can use the following mathematical approach:

$$Safety\ Status = \begin{cases} 0\ for\ (\boldsymbol{p} - \boldsymbol{p}_1) \cdot \boldsymbol{n} > 0 \\ 1\ for\ (\boldsymbol{p} - \boldsymbol{p}_1) \cdot \boldsymbol{n} < \boldsymbol{0} \end{cases}$$

with 0 meaning outside and 1 meaning inside of the safety zone and $\boldsymbol{n}$ being the normal vector of the plane. To make sure that $\boldsymbol{n}$ and $\boldsymbol{p}_1$ look in the same direction we use a vector to a point far away from the origin (perpendicular to the camera sensor) and calculate its dot product with $\boldsymbol{n}$ and $\boldsymbol{p}_1$. Both results must have the same algebraic sign.
For this entire process we use the joint coordinates $\boldsymbol{p}$ of the predicted hand to detect dangerous interactions beforehand.
Although focus of our project is on safe human-robot interaction, we need to provide more than a safety status just for the predicted hand. For an overall safe environment, we add a depth map feature where any objects in the field of view range of the camera can be measured. Each pixel of the depth map goes through the same calculation as the predicted joint coordinates to find out if there is anything inside the safety zone. The only difference is that the security check is made with the real time raw data of the depth map and there is no prediction. Its function is just to have an overall secure environment.

2.2 Visualization

For the real time visualization of the application we can show the hand with the raw, predicted and filtered coordinates at the same time. As default we use the filtered hand because the visualization is smooth since that the jittery party of the raw hand is filtered. The hands are visualized by using a spherical representation for each joint with the joint coordinates as the center. A bone which connects two joints is represented by a cylinder where its size and the orientation is calculated with the coordinates of the two joints which it should connect.

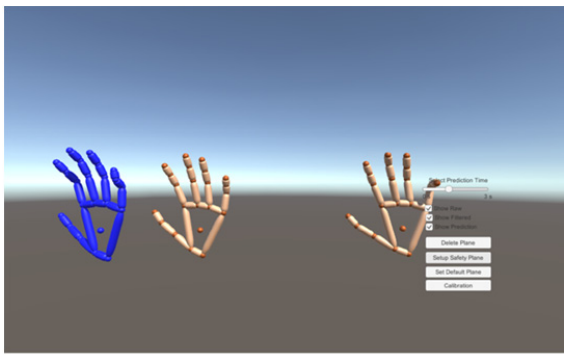

Fig. 2: Visualization of the hands, from left to right: Predicted, Filtered, Raw

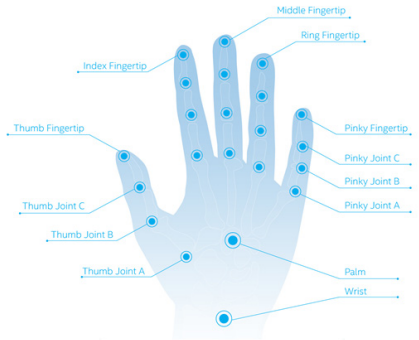

Fig. 3: All detectable joints with the Intel RealSense camera INTEL (2016)

For the depth map, there is a rectangular box in the upper left corner which shows every pixel red where its measured 3D coordinates are inside the safety zone and white otherwise (see Fig. 4).
The entire visualization of the application is made to get real-time information when implementing the software for any projects. The entire visualization is run in Unity.

2.3 Kalman Filter

To achieve the prediction of the hand we use the Kalman Filter, which was first introduced by KALMAN (1960). Kalman Filtering consists of two main parts, prediction and correction, where the state vector $\boldsymbol{x}$ describes position and velocity and P contains its variances and covariances.
The first step is the **prediction**. There we make a kind of educated guess of how the state of the system is going to be in the near future. For that we need the actual state and the prediction model. The prediction model is an equation which describes the nature of how the state of the system changes over time. It is a recursive function which is in the form:

$$\boldsymbol{x}_n = f(\boldsymbol{x}_{n-1})$$

Usually, the prediction model is a normal equation of motion by Newton. As proposed by KOHLER (1997), when tracking hands, the movement between two time steps is often assumed as linear and the acceleration is omitted and modeled as noise, so the model has the following form:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{n,apriori} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}_{n-1} + \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}_{n-1} \cdot \Delta t$$

Linear models like this can be directly transformed into matrix notation:

$$\boldsymbol{x}_{n,apriori} = A \cdot \boldsymbol{x}_{n-1}$$

For the predicted P matrix we perform a propagation of variance. Since the matrix $\boldsymbol{A}$ is already calculated, this is an easy task:

$$P_{n,apriori} = A \cdot P_{n-1} \cdot A^T$$

The second step is the **correction**. The prediction gets corrected by the actual measurements which are stored in the vector $\boldsymbol{z}_n$ and their errors and noise in the matrix R. The H matrix is called conversion matrix and handles the conversion of different units or scales. To perform the correction, we use a weighted mean of the measured and the predicted state:

$$\boldsymbol{x}_{n,apost} = \boldsymbol{x}_{n,apriori} + K \cdot \left(\boldsymbol{z}_n - H \cdot \boldsymbol{x}_{n,apriori}\right)$$

$$P_{n,apost} = P_{n,apriori} - K \cdot H \cdot P_{n,apriori}$$

The weight matrix K is the Kalman Gain Matrix. It is actually a kind of inverse of the ratio between the two covariance matrices of the measured and the predicted state and is calculated as follows:

$$K = \frac{P_n \cdot H^T}{H \cdot P_n \cdot H^T + R}$$

Both steps, prediction and correction, are repeated over the course of the filtering operation. In practice, all matrices could change along the process but most of them are assumed constant as proposed in (WELCH & BISHOP 2006). Only the value of $\mathbf{z}_n$ change as well as the outputs $\boldsymbol{x}_n$ and P_n.

3 Results

We implemented the developed method as a running software in C# that demonstrates the system's capabilities. Our main interface is shown in Fig. 4:

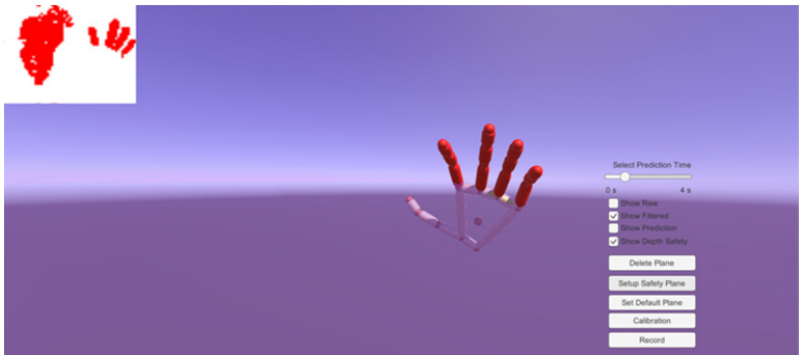

Fig. 4: Visualization of the algorithm

We represent the safety zone as a blue transparent plane and the visualized hand changes the color when its inside in order to verify it the algorithm is working. With one click you can also show the predicted hand and its intrusion into the zone. In the upper left corner, the safety status of the depth map is displayed as a small image stream.
To assess how good the filtering of the coordinates works, we chose a qualitative approach of recording some seconds and plotting the distance from the camera to one joint against the time. This way we can easily see how much noise the raw coordinates contain and what the Kalman Filter makes of this data. We can observe that the Kalman filtered outputs (red line in Fig. 5) are much smoother than the original raw measurements.

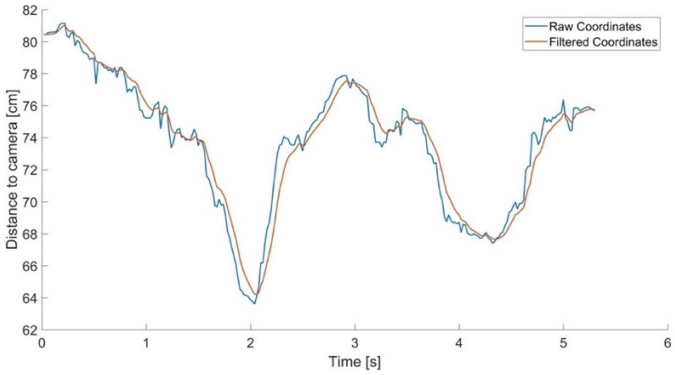

Fig. 5: Distance from the camera to the index fingertip over time, raw and filtered coordinates

We estimated measurement noise (values in R) with the empirical standard deviation of a still hand. We took the mean over all joints, which resulted in a standard deviation of 0.1 cm for the position and 0.7 cm/s for the velocity. In Fig. 6 we can see how variation of these values affects the filtering process. If the measurement errors are small, the filter stays close to the observations and there is virtually no filtering happening. With increasing uncertainty of the measurement, more and more filtering takes place.

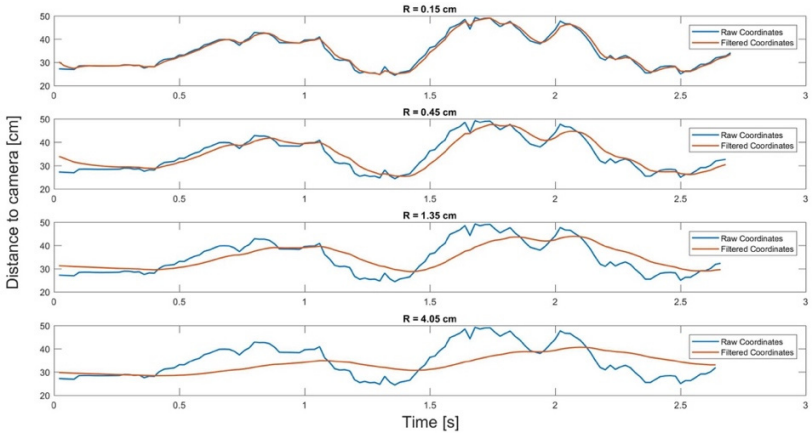

Fig. 6: Influence of the R value on the filtering process (Given values are the mean over the matrix)

4 Conclusions and Outlook

In our bachelor thesis we developed, implemented, and tested a hand-tracking tool for depth sensors. Our main methodological contribution is the adaption of a Kalman-filtering approach to human-robot interaction. Experiments with our method demonstrate that the method manages to guarantee a safety zone around the robot in order to avoid injury of humans and unnecessary damage to the robot.

In the future, we want to add additional parts like human heads, kids toys etc. to the safety zone predictor. Additionally, it is worth investigating if adding more depth sensors to the set up helps making predictions more robust. Another interesting topic would be experimenting with partial interference by daylight, which may cause the (infrared pattern-based) depth camera to fail.

5 References

INTEL, 2016: Intel RealSens SDK 2016 R2 Documentation. Intel RealSense SDK 2016 R2 Documentation.

KALMAN, R.E., 1960: A New Approach to Linear Filtering and Prediction Problems. Transactions of the ASME-Journal of Basic Engineering, **82**, Series D, 35-45.

KOHLER, M., 1997: Using the Kalman Filter to track Human Interactive Motion – Modelling and Initialization of the Kalman Filter for Translational Motion. University Dortmund, Informatics VII.

WELCH, G. & BISHOP, G., 2006: An Introduction to the Kalman Filter. University of North Carolina at Chapel Hill.