Dynamic Augmented Kalman Filtering for Human Motion Tracking under Occlusion Using Multiple 3D Sensors

Atle Aalerud[1] and Geir Hovland[1]

[1]Mechatronics Group, Faculty of Engineering and Science, Department of Engineering Science, University of Agder, N-4898 Grimstad, Norway. atle.aalerud@uia.no

Abstract **– In this paper real-time human motion tracking using multiple 3D sensors has been demonstrated in a relatively large industrial robot work cell. The proposed solution extends state-of-the-art by augmenting the constant velocity model and Kalman filter with low-pass filtered velocity states. The presented method is able to handle occlusions by dynamically inclusion in the Kalman filter of only those 3D sensors which provide valid human position data. Human motion tracking was achieved at a frame rate of 20 Hz, with a typical delay of 50 ms to 100 ms and an estimation accuracy of typically 0.10 m to 0.15 m.**

E.1 Introduction

With recent advances in 2D and 3D sensor technologies, algorithms and processing power, real-time human motion tracking has become a possibility. Tracking of human motion has many potential application areas. Some examples are safety systems for human-robot collaboration, e-health applications such as fall detection, and for collision detection and avoidance in intelligent autonomous systems.

YOLOv3 [1] is a state-of-the-art, real-time object detection system. In this paper, multiple sensor nodes together with YOLOv3-tiny are used to detect humans in the sensor images and extract the 3D position from depth images. The human position measurements are fed into a constant velocity model and Kalman filtering is used to track human motion in real-time. One novelty in the presented paper is the augmentation of the constant velocity model with filtered states to reduce the noise in the estimated human motion. For collision avoidance planning applications in particular, it is desirable to use human velocity estimates which are not distorted by high-frequency noise.

In [2], Linder et al. presented a benchmark on multi-modal people tracking from mobile platforms in very crowded and dynamic environments. Their framework integrated four existing tracking approaches and was based on the Robot Operating System (ROS) running under Ubuntu and both 2D lasers and 3D RGB-D type sensors were included. One of the methods used was the RGB-D upper-body detector from [3]. More examples of state-of-the-art human motion detectors can be found in the reference list of [2]. The framework used in our paper is similar to [2] by using ROS and running on Ubuntu. However, the upper-body detector used in this paper is the one developed in [4] and the experimental results presented in this paper includes only the motion of one human for

ease of benchmarking, but the method would handle multiple humans.

The paper is organized as follows: Paper E.2 describes the methodology used, Paper E.3 describes the framework and experimental setup, Paper E.4 outlines the experiments while the results are presented in Paper F.5. Discussion and conclusions are given in Papers E.6 and E.7.

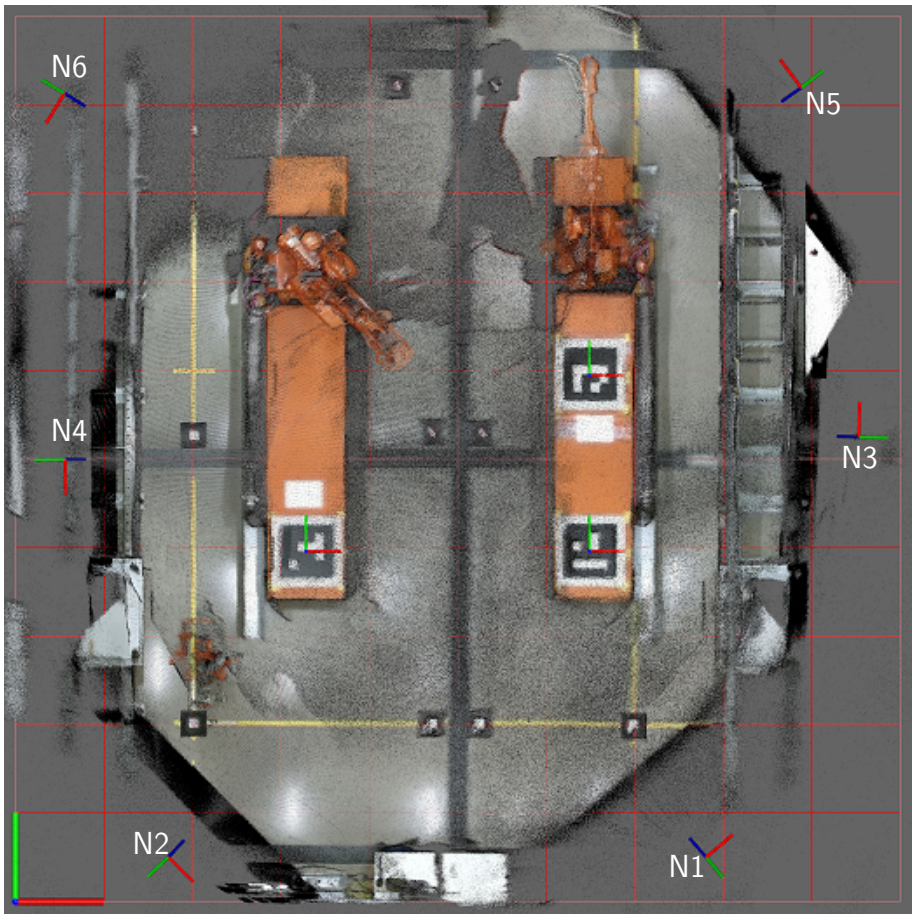

Figure E.1: Orthogonal view of a combined point cloud depicting the Industrial Robotics Lab at the University of Agder. Sensor node poses are annotated N1-N6. The sensors are located 4.3 m above the floor and are angled approx. 60° from horizontal. The origin of the world frame is located at the bottom-left, where the red and green lines represent the X and Y-directions, respectively.

E.2 Methodology

For completeness this section contains the background material for human motion detection using Kalman filtering. The section contains several novelties: A) dynamic sized Kalman filter depending on the number of sensors which are not occluded, B) human motion velocity filtering using augmented variables, C) a new test criterion for benchmarking (SNRMSE).

E.2.1 Kalman Filter

A Kalman filter was designed to refine the position estimates from the human detector and to estimate velocity states that can be used to predict a future human position.

The discrete Kalman filter is well known in the literature. Nevertheless, the governing equations are included here for completeness and the reader's convenience. The filter is typically defined by a predicting and an updating phase where the predicting phase consists of the equations:

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{F}\hat{\boldsymbol{x}}_{k-1} \tag{E.1}$$

$$\boldsymbol{P}_k^- = \boldsymbol{F}\boldsymbol{P}_{k-1}\boldsymbol{F}^T + \boldsymbol{Q}\ . \tag{E.2}$$

Here, $\hat{\boldsymbol{x}}_k^-$ is the a priori state estimate calculated from the transition matrix, $\boldsymbol{F}$, and the previous posteriori state estimate, $\hat{\boldsymbol{x}}_{k-1}$. $\boldsymbol{P}_k^-$ is the a priori covariance matrix and $\boldsymbol{Q}$ is the process noise matrix.

In the second phase, the predicted estimates are updated by the equations:

$$\hat{\boldsymbol{x}}_k = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K}_k\boldsymbol{y} \quad | \quad \boldsymbol{y} = (\boldsymbol{z}_k - \boldsymbol{H}_k\hat{\boldsymbol{x}}_k^-) \tag{E.3}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^-\boldsymbol{H}_k^T{\boldsymbol{S}_k}^{-1} \quad | \quad \boldsymbol{S}_k = \boldsymbol{H}_k\boldsymbol{P}_k^-\boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{E.4}$$

$$\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_k^- \quad . \tag{E.5}$$

The updated state estimate, $\hat{\boldsymbol{x}}_k$, is calculated from $\hat{\boldsymbol{x}}_k^-$, the Kalman gain, $\boldsymbol{K}_k$, and the residual, $\boldsymbol{y}$. This residual is the difference between the measured states, $\boldsymbol{z}_k$, and the estimated states found by multiplying the measurement matrix, $\boldsymbol{H}_k$, with $\hat{\boldsymbol{x}}_k^-$. The Kalman gain, $\boldsymbol{K}_k$, balances the influence caused by prediction uncertainty and measurement uncertainty. Here, the innovation covariance matrix, $\boldsymbol{S}_k$ describes the system uncertainty and weighs the uncertainty for each of the sensor measurements using the added sensor noise matrix, $\boldsymbol{R}_k$. Lastly, the covariance matrix, $\boldsymbol{P}_k$, is updated.

Designing a model of the system, in this case human motion, is not a trivial task. For example a constant velocity Kalman filter will not be able to track rapid changes in heading, but it will catch up once signals enter steady state. A constant acceleration filter, on the other hand, may react quickly, but it can also misinterpret noise as acceleration during steady state. None of these are exact models of human motion, but a constant velocity model was found most suitable for this application where stability was the main concern. The transition matrix, $\boldsymbol{F}$, modelled using a constant velocity model in two dimensions is

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{E.6}$$

where Δt is the discrete time step. The corresponding position and velocity states are

$$\boldsymbol{x} = \begin{bmatrix} x & \dot{x} & y & \dot{y} \end{bmatrix}^T \tag{E.7}$$

where x and y show the human position in x and y direction. Since the velocity model is a rough simplification of human motion, it is necessary to artificially boost the magnitude of

the process noise, $\boldsymbol{Q}$, to account for the mis-matched model. In this paper, the piecewise constant white acceleration model, as described in [5, ch.6.3] and [6], was applied. The state equation for this model is

$$\boldsymbol{x}_k = \boldsymbol{F}\boldsymbol{x}_{k-1} + \Gamma w_{k-1} \quad . \tag{E.8}$$

The piecewise constant white acceleration noise, is defined by $\boldsymbol{\Gamma} w_{k-1}$. Here, $\boldsymbol{\Gamma}$ is the noise gain and w_{k-1} is the piecewise constant noise. This artificial acceleration noise has a duration of Δt which, in turn, will give a velocity increment of $w_{k-1}\Delta t$. The effect on position is $w_{k-1}\Delta t^2/2$. Inserting for all states yields the noise gain matrix

$$\boldsymbol{\Gamma} = \begin{bmatrix} \frac{1}{2}\Delta t^2 & \Delta t & \frac{1}{2}\Delta t^2 & \Delta t \end{bmatrix}^T . \tag{E.9}$$

Assuming no correlation between x and y, the covariance of the process noise is

$$\begin{aligned} \boldsymbol{Q} &= E\left[\boldsymbol{\Gamma} w_{k-1} w_{k-1} \boldsymbol{\Gamma}^T\right] = \boldsymbol{\Gamma}\boldsymbol{\Gamma}^T \sigma_v^2 \\ &= \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & 0 & 0 \\ 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} \\ 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 \end{bmatrix} \sigma_v^2. \end{aligned} \tag{E.10}$$

As the sensor setup was comprised of multiple sensor nodes that performed human detection in a non-synchronous manner, the filter had to be able to scale from zero to multiple separate pose readings per iteration. Hence, $\boldsymbol{H}_k$, $\boldsymbol{R}_k$ and $\boldsymbol{K}_k$ were scaled according to the number of received measurements in $\boldsymbol{z}_k$. Only positions were detected, so the measurement matrix for a single sensor was

$$\boldsymbol{H}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{E.11}$$

whilst matrices for multiple sensors were concatenated in the combined measurement matrix

$$\boldsymbol{H}_k = \begin{bmatrix} \boldsymbol{H}_1 \\ \vdots \\ \boldsymbol{H}_{n_k} \end{bmatrix} , \tag{E.12}$$

where the number of sensors was

$$n_k = \frac{m}{2} \;\middle|\; \begin{matrix} \boldsymbol{z}_k \in \mathbb{R}^{m\times 1} \\ \boldsymbol{H}_{n_k} = \boldsymbol{H}_1 \in \mathbb{R}^{2\times 4} \end{matrix} \quad , \tag{E.13}$$

where m was number of rows in the measurement vector $\boldsymbol{z}_k$.

We assume that there was no correlation in the noise between sensors, so the covariances were zero. The diagonal values of $\boldsymbol{R}_k$ are typically based on the sensors variances. In our application, the sensors may have a low variance, but there was a significant bias. The detections were based on measuring position of the human upper body using RGB-D cameras and applying an offset to estimate the body center. However, this estimate had a bias depending on the human orientation.

Further, the human detection algorithm used as input in this paper returned some false positives that will appear as outliers, but since there were several sensors available, pre-filtering and clustering was possible. From the position cluster used as input to the system it was possible to implement a dynamic variance.

As described in [7], the average difference between measurements can be written

$$\boldsymbol{\mu}_k = \frac{\sum_{j=1}^{n_k} \left(\sum_{i=j+1}^{n_k} \left(|\boldsymbol{z}_{k,j} - \boldsymbol{z}_{k,i}| \right) \right)}{n_k(n_k - 1)} \ , \tag{E.14}$$

where $\boldsymbol{z_k}$ is the vector of x- and y-measurements for all sensors and n_k is the number of sensors for sample k. Setting the sensors standard deviation to

$$\boldsymbol{\sigma}_k = \frac{\boldsymbol{\mu}_k}{\lambda} \ , \tag{E.15}$$

where λ is a scaling factor set to 1.5, yields the dynamic variance $\boldsymbol{\sigma}^2_{k,d}$. For tuning purposes, sensor noise is implemented as a combination of static and dynamic variance

$$\boldsymbol{\sigma}^2_k = \rho_1 + \rho_2 \, \sigma^2_{k,d} \ , \tag{E.16}$$

where ρ_1 and ρ_2 are constant tuning parameters. Using the combined variance yields the single sensor noise matrix

$$\boldsymbol{R}_{1_k} = \begin{bmatrix} \rho_1 + \rho_2 \, \sigma^2_{k,d,x} & 0 \\ 0 & \rho_1 + \rho_2 \, \sigma^2_{k,d,y} \end{bmatrix} \tag{E.17}$$

and the complete sensor noise matrix:

$$\boldsymbol{R}_k = \begin{bmatrix} \boldsymbol{R}_{1_k} & \dots & \boldsymbol{0} \\ \vdots & \ddots & \vdots \\ \boldsymbol{0} & \dots & \boldsymbol{R}_{n_k} \end{bmatrix} \ . \tag{E.18}$$

E.2.2 Augmented Kalman Filter (AKF)

Designing and tuning a Kalman Filter is inevitable a trade-off between state smoothness and accuracy. In this case, the application requires a smooth velocity prediction for estimating a future human path. This would require tuning the KF to have high confidence in the model (low $\boldsymbol{Q}$) and low confidence in measurements (high $\boldsymbol{R}_k$). High tracking accuracy, on the other hand, requires higher confidence in the measurements and low confidence in the model. To alleviate this trade-off one can tune the filter for required tracking and then apply an averaging filter for the velocity states.

As explained in [8], an exponentially weighted moving average (EWMA) filter in discrete form can be written on the form

$$x_{f,k} = (1 - a)x_{f,k-1} + ax_k \ , \tag{E.19}$$

where the $x_{f,k}$ is the filtered state and a is the filter parameter given by

$$a = \frac{\Delta t}{T_a + \Delta t} \ , \tag{E.20}$$

which, in turn, contains the filter time constant T_a and the time-step Δt. The velocity model from Equations (E.6) and (E.7) yields the position

$$x_k = x_{k-1} + \Delta t \dot{x}_{k-1} \ , \tag{E.21}$$

Inserting Equation (E.21) in Equation (E.19) and rearranging terms yields the filtered position

$$x_{f,k} = a x_{k-1} + a\Delta t \dot{x}_{k-1} + (1-a) x_{f,k-1} \ . \tag{E.22}$$

The filtered velocity is calculated in a similar manner where the time derivative of Equation (E.19) is

$$\dot{x}_{f,k} = (1-b)\dot{x}_{f,k-1} + b\dot{x}_k \tag{E.23}$$

and the filter parameter is

$$b = \frac{\Delta t}{T_b + \Delta t} \tag{E.24}$$

such that the filter time constant T_b does not need to be the same as T_a. The constant velocity model, Equation (E.6), states that

$$\dot{x}_k = \dot{x}_{k-1} \ . \tag{E.25}$$

Hence, by inserting Equation (E.25) in Equation (E.23) and rearranging terms, the filtered velocity is given as

$$\dot{x}_{f,k} = b\dot{x}_{k-1} + (1-b)\dot{x}_{f,k-1} \ . \tag{E.26}$$

The filtered states are introduced as state variables such that the new vector for two dimensions is

$$\boldsymbol{x} = \begin{bmatrix} x & \dot{x} & x_f & \dot{x}_f & y & \dot{y} & y_f & \dot{y}_f \end{bmatrix}^T . \tag{E.27}$$

The augmented transition matrix, including Equation (E.22) and Equation (E.26) for two dimensions, is

$$\boldsymbol{F} = \begin{bmatrix} 1 & \Delta t & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ a & a\Delta t & 1\text{–}a & 0 & 0 & 0 & 0 & 0 \\ 0 & b & 0 & 1\text{–}b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & \Delta t & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & a & a\Delta t & 1\text{–}a & 0 \\ 0 & 0 & 0 & 0 & 0 & b & 0 & 1\text{–}b \end{bmatrix} \tag{E.28}$$

and the measurement matrix for a single sensor becomes

$$\boldsymbol{H}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} . \tag{E.29}$$

The process noise matrix, $\boldsymbol{Q}$, must also be augmented for the filter to work. However, as seen in Equation (E.2) and Equation (E.4), this is indirectly multiplied with $\boldsymbol{H}_k^T$ causing only column one and five to be used. The new states have only covariances in these columns whilst the variances are found in column three, four, seven and eight. As

the covariances are not considered to add value in this case and other columns are not used, no additional process noise is calculated. Thus, the new zero-padded process noise matrix is

$$\boldsymbol{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{\Delta t^3}{2} & \Delta t^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\Delta t^4}{4} & \frac{\Delta t^3}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{\Delta t^3}{2} & \Delta t^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \sigma_v^2. \tag{E.30}$$

The augmented versions of $\boldsymbol{x}$, $\boldsymbol{F}$, $\boldsymbol{H}_1$ and $\boldsymbol{Q}$ are used by the governing equations of the discrete Kalman filter, Equations (E.1) to (E.5). This enables the augmented filter to estimate an EWMA filtered position and velocity as additional states without affecting the existing states for position and velocity.

E.2.3 Tuning and Validation

The Kalman filter was tuned by iteratively adjusting σ_v^2, ρ_1 and ρ_2. The filter configurations were applied to a recorded dataset (ROS-bag) for identical input and the performance was evaluated using the method of Bar-Shalom et al. [5, ch. 5.4] as described below.

No innovation magnitude bond test or whiteness test (autocorrelation) was performed due to variable innovation, $\boldsymbol{y}$. Nevertheless, to test for unbiasedness and conistency, the χ^2-tests *normalized estimation error squared* (NEES) and *normalized innovation squared* (NIS) were performed. To further evaluate bias, a *normalized mean estimation error* (NMEE) was conducted.

NEES is defined as

$$\epsilon_k = \tilde{\boldsymbol{x}}_k{}^T \boldsymbol{P}_k^{-1} \tilde{\boldsymbol{x}}_k \ , \tag{E.31}$$

where

$$\tilde{\boldsymbol{x}}_k = \boldsymbol{x}_k - \hat{\boldsymbol{x}}_k \ . \tag{E.32}$$

Under the linear Gaussian assumption, H_0, the expected value of ϵ_k is

$$E[\epsilon_k] = n_x \ , \tag{E.33}$$

where n_x is the dimension of $\boldsymbol{x}$. However, in our test we have not included the augmented states and will use $n_x = 4$. For multiple runs, i.e. Monte Carlo simulations, the N-run average NEES (ANEES) is

$$\bar{\epsilon}_k = \frac{1}{N} \sum_{i=1}^{N} \epsilon_{k,i} \ , \tag{E.34}$$

and the new χ^2 probability density function will have Nn_x degrees of freedom. By utilizing the ergodicity of the state errors, we can use the time-average rather than an ensamble average. Thus, the time-average NEES for K time iterations is

$$\bar{\epsilon} = \frac{1}{K} \sum_{k=1}^{K} \epsilon_k \tag{E.35}$$

and $K\bar{\epsilon}$ will, under H_0, have a χ^2 density with Kn_x degrees of freedom. The NEES acceptance criterion is

$$\bar{\epsilon} \in [r_1, r_2] \ , \tag{E.36}$$

where the interval is defined

$$P\bar{\epsilon} \in [r_1, r_2] | H_0 = 1 - \alpha \ . \tag{E.37}$$

We use $\alpha = 0.05$, i.e., the two-sided 95% confidence region. For a single run, $K = 1$ which yields the acceptance interval

$$\begin{aligned} [r_1, r_2]/K &= [\chi^2_{4\cdot 1}(0.025), \chi^2_{4\cdot 1}(0.975)]/1 \\ &= [0.484, 11.143] \end{aligned} \tag{E.38}$$

whilst using $K = 840$ time iterations yields the interval

$$\begin{aligned} [r_1, r_2]/K &= [\chi^2_{4\cdot 840}(0.025), \chi^2_{4\cdot 840}(0.975)]/840 \\ &= [3.811, 4.194] \end{aligned} \ . \tag{E.39}$$

Note that increasing the number of averaged samples naturally reduces the interval needed for 95% certainty.

Similar to the NEES test, the NMEE shows if the filter errors are consistent with the covariance matrix, but here the results are shown individually for each state. Based on the time-average described above, the error is defined

$$\bar{\mu}_j = \frac{1}{K} \sum_{k=1}^{K} \frac{\tilde{\boldsymbol{x}}_{j,k}}{\sqrt{\boldsymbol{P}_{jj,k}}} \ , \tag{E.40}$$

where $\boldsymbol{P}_{j,j,k}$ is the j^{th} diagonal covariance matrix element for the k^{th} sample. The above, $\bar{\mu}_j$, is assumed distributed $\mathcal{N}(0, 1/K)$ with a standard deviation of $1/\sqrt{K}$. Similar to NEES, the acceptance criterion is

$$\bar{\mu}_j \in [-r, r] \ , \tag{E.41}$$

where the interval is defined

$$P\bar{\mu}_j \in [-r_1, r_1] | H_0 = 1 - \alpha \tag{E.42}$$

$$r = r_1/\sqrt{K} \ . \tag{E.43}$$

For $\alpha = 0.05$ we get the two-sided 95% region such that the acceptance criterion for a single sample, $= 1$, can be found by inverse lookup of the normal cumulative distribution

$$[-r, r] = [-r_1, r_1] = [-1.96, 1.96] \ . \tag{E.44}$$

For $K = 840$ samples, the acceptance region is

$$[-r, r] = [-r_1/\sqrt{840}, r_1/\sqrt{840}] = [-0.068, 0.068] \ . \tag{E.45}$$

Note that there is no actual mean for a single sample (Equation (E.40), so in this case we will call it *normalized estimation error* (NEE), but it refers to the same method.

The NIS test is the same as the NEES, but is produced for the innovation vector and corresponding innovation covariance. The test should show that the innovation is unbiased and white under the hypothesis that the filter is consistent. The error is defined as

$$\epsilon_{y,k} = \boldsymbol{y}_k^T \boldsymbol{S}_k^{-1} \boldsymbol{y}_k \tag{E.46}$$

and time-average NIS (ANIS) estimated mean is

$$\bar{\epsilon}_y = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{y}_k^T \boldsymbol{S}_k^{-1} \boldsymbol{y}_k \ . \tag{E.47}$$

Similar to NEES, NIS is expected to have a χ^2 distribution and acceptance criteria as described in Equations (E.36) and (E.37), but here, the expected value of $\epsilon_{y,k}$ is

$$E[\epsilon_{y,k}] = n_z \ , \tag{E.48}$$

where n_z is the dimension of $\boldsymbol{z_k}$. In this paper, the dimension of $\boldsymbol{z_k}$ is not constant, (see Equations (E.3) and (E.12),) but the test is still attempted with $\boldsymbol{z_k} = 2$. Using the same two-sided 95% confidence region as before yields

$$\begin{aligned} [r_1, r_2]/K &= [\chi^2_{2\cdot 1}(0.025), \chi^2_{2\cdot 1}(0.975)]/1 \\ &= [0.051, 7.378] \end{aligned} \tag{E.49}$$

for a single iteration whilst using $K = 840$ time iterations yields the interval

$$\begin{aligned} [r_1, r_2]/K &= [\chi^2_{2\cdot 840}(0.025), \chi^2_{2\cdot 840}(0.975)]/840 \\ &= [1.867, 2.138] \end{aligned} \ . \tag{E.50}$$

In addition to the mentioned statistical tests, root mean square error (RMSE) values are evaluated for position, velocity, speed and course. RMSE equations are omitted for brevity as they assumed to be well known by the reader. However, it is noted that the RMSE of position is the RMS of the Euclidean distance error. To evaluate the combination of RMSE values, we define the metric *sum of normalized RMSE* (SNRMSE) as

$$\text{SNRMSE}_c = \sum_{i=1}^{n} \left(\frac{e_{c,i} - \min\limits_c(e_{c,i})}{\max\limits_c(e_{c,i}) - \min\limits_c(e_{c,i})} \right) , \tag{E.51}$$

where c are the different configurations and n is the index of RMSE values for the configuration.

E.3 Framework

The Industrial Robotics Lab (IRL) at the University of Agder depicted in Figure E.1 is the industrial environment to be used for human tracking. This lab functions as a large robotic cell where both industrial robots and humans need to operate. Thus, it is crucial that the positions of the human workers are known so that safety systems can be developed for the

robots. The specific setup of the sensor network and the lab have been described in [9], but key information is included here for completeness.

The mapped environment covered an area of 10 m × 10 m where sensors were placed according to Figure E.1 at 4.3 m height. This sensor network comprised of six nodes in waterproofed cabinets, where each cabinet was equipped with a Kinect V2 RGB-D camera based on an active infrared (IR) sensor using the time-of-flight principle. The automatic extrinsic calibration of the sensor nodes was described in [10].

Further, each node contained an NVIDIA Jetson TX2 Development board. These boards processed the sensor data before sending requested data via Gigabit Ethernet to a personal computer (PC). The PC used in this paper was equipped with a 3.6 GHz Intel Core i7-7820x central processing unit (CPU), 32 GiB system memory and an NVIDIA GeForce GTX 1080 Ti graphics processing unit (GPU). The communication between the sensor nodes and the central PC is handled by the open-source middleware Robot Operating System (ROS) [11] running on Ubuntu 16.04.

The multiple ROS nodes that performed human detections was adaped from [4] where YOLOv3-tiny, [1], was trained to detect the upper body of humans. Removing limbs from the detection reduces the problem of occlusion. Further, the detection nodes extracted the 3D position based on the depth image of the Kinect V2.

A ROS node on the central PC received the positions from all connected sensor nodes and processed them using the augmented Kalman filter as described further in Paper E.2.

E.4 Experiments

To test the AKF, a (ROS-bag) dataset was recorded. Here, position detections and images from all sensor nodes were recorded while a person walked in a predefined pattern. This route was used as ground truth so errors could be calculated. To increase the accuracy of the person with regards to the planned route, markers were placed at the floor at 0.5 m intervals using known coordinates as reference. These markers were used as walking steps. Further, to control the walking speed, a metronome was used to keep the pace. By setting the metronome at 120 BPM, a walking speed of 1 m/s was ensured. This method may cause some small position deviations, but using tracking equipment was not an option in this specific case due to occlusion.

Each sensor node ran detections at 20 Hz. These detections were processed on the central node that executed the AKF at 20 Hz. Thus, there may have been a maximum time delay of 50 ms between the detection time stamp and filter processing. When evaluating a walking speed of 1 m/s we see that this can introduce a maximum error of 50 mm. Detections were currently not extrapolated to compensate for this as it was considered sufficiently small.

The planned route was a rounded rectangle, with 6 m length and 1 m width, placed in the middle of the lab between the two robot tracks shown in Figure E.1. As the sensors had a limited range and field of view it was expected that the individual node's detections would depend on the actual position of the person. Occlusion would also reduce the amount of detections. A large gantry crane covered large portions of N1 and N2's field

of view as the boom crossed the lab at $x = 2.5\,\text{m}$ during the experiment. Further, the industrial robots on both sides of the planned path caused occlusion depending on the position of the person.

The extracted data from the dataset covers the time $t = [10\,\text{s}, 52\,\text{s}]$ which yields three clockwise rounds of the path starting at coordinate $(4.5\,\text{m}, 2.0\,\text{m})$. The sequence consists of 840 samples at a sampling interval of 0.05 s.

E.5 Results

In Table E.1 the 4 best test results of 21 tested tuning configurations are shown. The top of the table describes filter parameters where the main Kalman parameters are σ_v^2, ρ_1 and ρ_2, but also two different selections of T_a and T_b are also shown. Below the separator, the test results are listed according to the described tests from Paper E.2.3. Where applicable, corresponding equations are listed. The number of samples is $K = 840$ for all tests. The best $\text{RMSE}_{\text{course}}$ is found in configuration D, where the value equals 18.6°. Note that all plots in this section are generated using data from configuration D.

Table E.1: Filter configurations and test results.

Configuration		**A**	**B**	**C**	**D**
$\boldsymbol{T_a}$, $\boldsymbol{T_b}$	[s]	0.1 , 0.5	0.1 , 0.5	0.1 , 0.25	0.1 , 0.25
$\boldsymbol{\sigma_v^2}$		10	10	10	10
$\boldsymbol{\rho_1}$, $\boldsymbol{\rho_2}$		0.05 , 0	0.01 , 0.5	0.05 , 0	0.01 , 0.5
ANEES Equation (E.35)		4.208	8.085	7.084	4.931
NMEE_x Equation (E.40)		−0.073	−0.027	−0.057	0.007
$\text{NMEE}_{\dot{x}}$ Equation (E.40)		0.022	0.006	0.024	0.014
NMEE_y Equation (E.40)		−0.256	−0.341	−0.271	−0.330
$\text{NMEE}_{\dot{y}}$ Equation (E.40)		0.015	−0.038	−0.010	−0.007
ANIS Equation (E.47)		1.139	0.961	1.373	0.635
$\text{RMSE}_{\text{position}}$	[m]	0.120	0.133	0.157	0.110
$\text{RMSE}_{\text{velocity}}$	[m/s]	0.376	0.416	0.380	0.311
$\text{RMSE}_{\text{speed}}$	[m/s]	0.210	0.261	0.196	0.172
$\text{RMSE}_{\text{course}}$	[rad]	0.379	0.372	0.406	0.325
SNRMSE Equation (E.51)		1.915	3.074	2.929	0

The estimated positions and states calculated by the AKF are shown in Figures E.2 and E.3. Figure E.2 visualizes x- and y- position estimates and corresponding detections for a single round of the path. This figure also illustrates the field of view and occlusion of the different sensors. For example, one can see that detections near the sensor position (as shown in Figure E.1) are more accurate than detections far away. In Figure E.3, the estimated positions, velocities, speed and course are shown. Figures E.3(**a**) and E.3(**b**) include the detections from the sensors, but they are omitted for clarity in the legend.

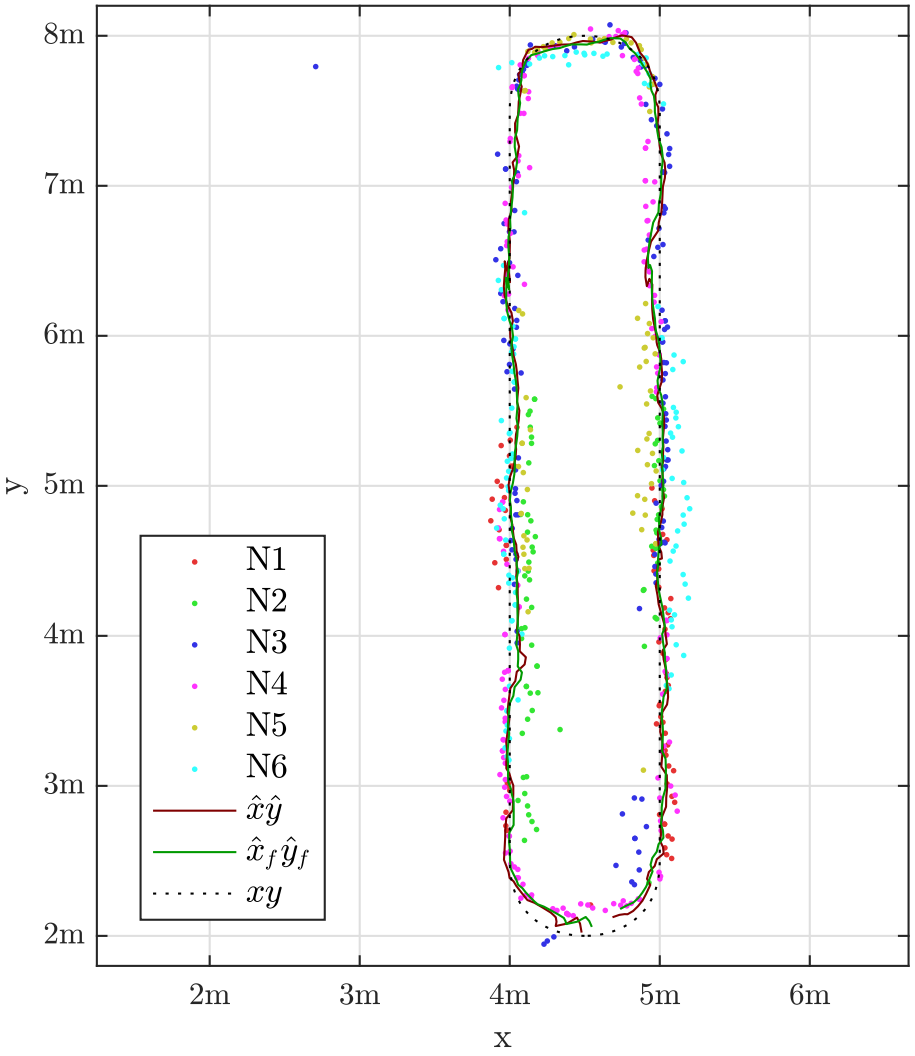

Figure E.2: Positions from the middle round, $t = [10\,\text{s}, 24\,\text{s}]$, where the walking pace was 1 m/s in clockwise direction. Detections from sensors N1-N6 are illustrated using dots in separate colors. Red and green line show the estimated position and EWMA-filtered position, respectively. The dotted line is the reference.

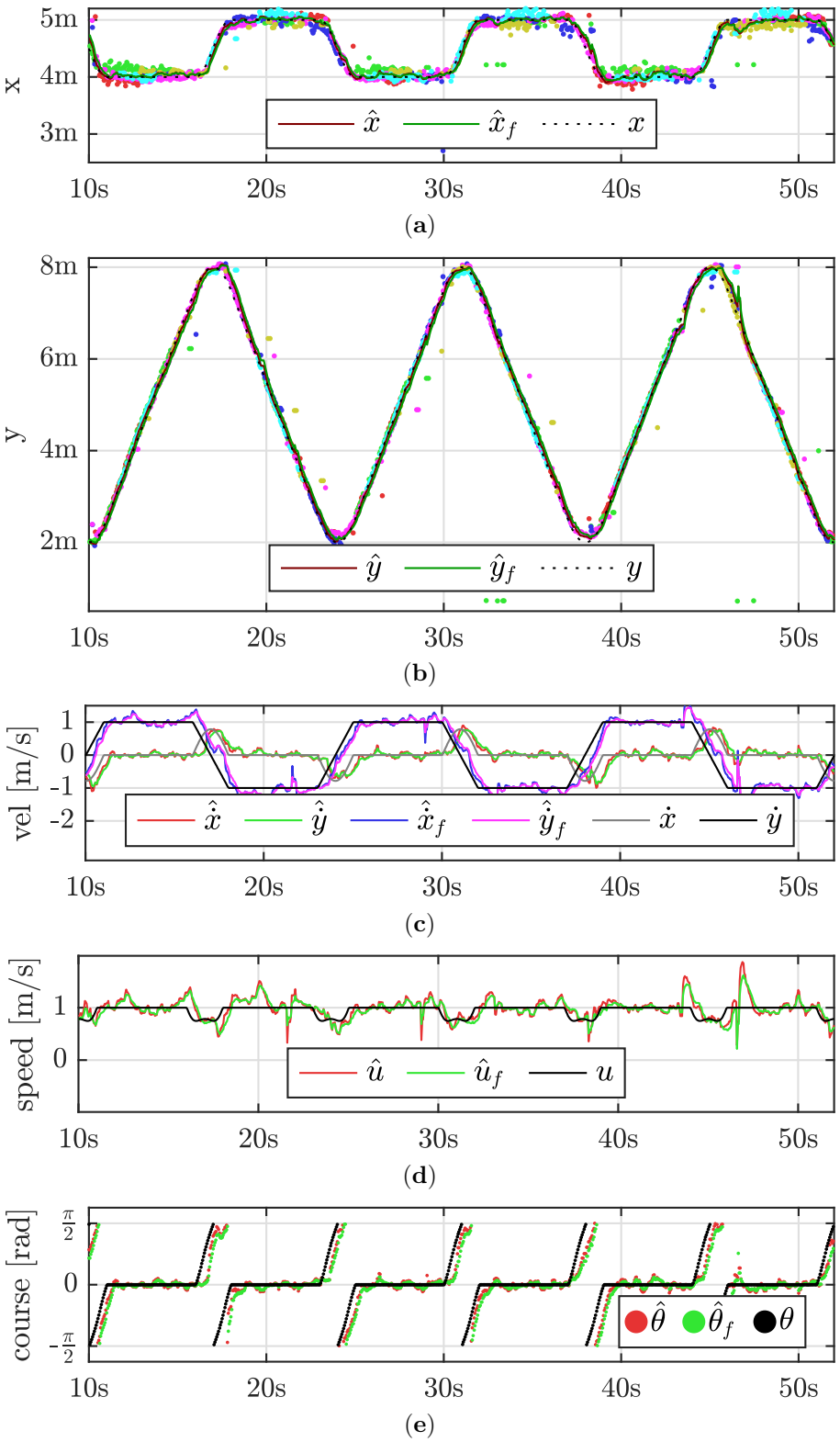

Figure E.3: Timeplots of outputs from the AKF compared to the reference shown in black. Position estimates for x and y are shown in (**a**) and (**b**), respectively. Here, detections are included according to legend in Figure E.2, but not shown in the respective legend. Velocity is shown in (**c**), speed in (**d**) and course in (**e**) where direction is positive clockwise from positive (and negative) y-axis.

The statistical single run tests are visualized in Figure E.4 along with their two-sided 95% confidence intervals for a single sample and 840 samples average. The boundaries of NEES in Figure E.4(**a**) were shown in Equations (E.38) and (E.39), the boundaries of NEE in Figure E.4(**b**) were shown in Equations (E.44) and (E.45), and the bondaries of ANIS in Figure E.4(**c**) were shown in Equations (E.49) and (E.50).

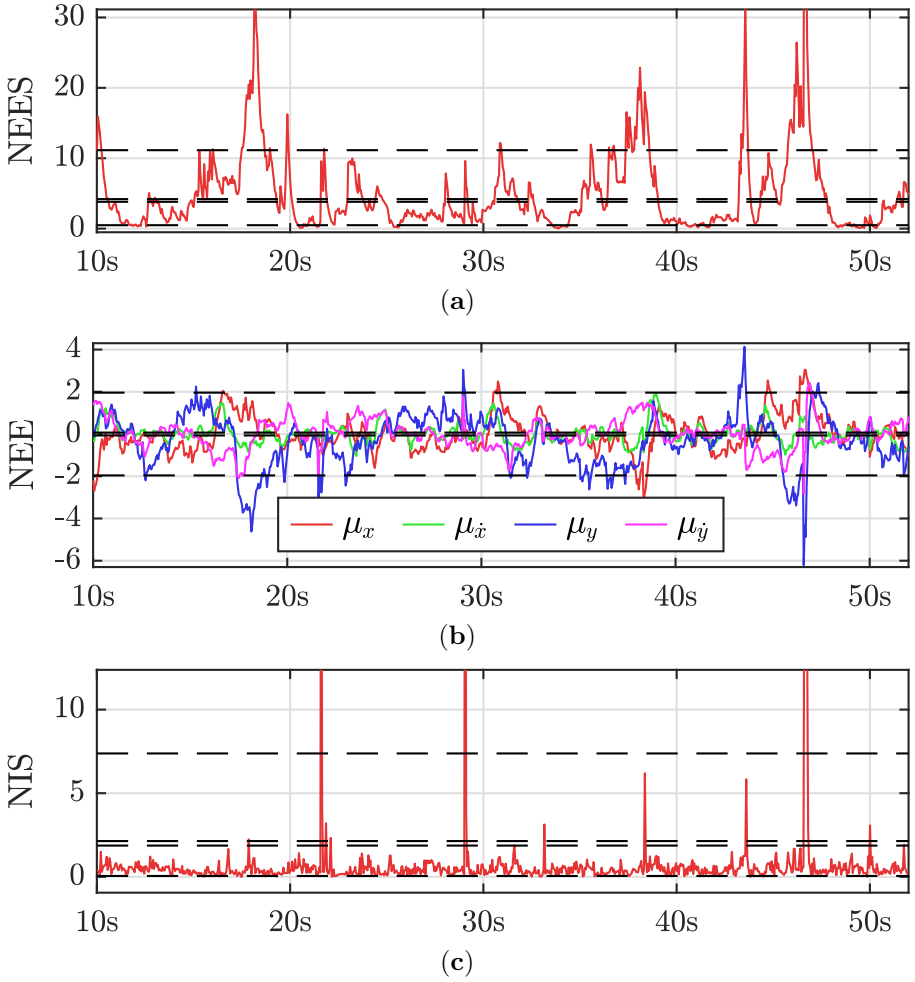

Figure E.4: Statistical tests of the AKF with the two-sided 95% confidence region for a single sample and 840 samples average. (**a**) show the single run NEES, (**b**) illustrate the single NEE for the evaluated states and (**c**) show single run NIS.

All evaluated absolute errors are shown in Figure E.5. Note that the EWMA filtered states generally have larger errors in transient conditions. Because all errors are calculated using a non-delayed reference, an error is a natural consequence of the filter induced delay. Thus, EWMA filtered states are not used for benchmarking or filter selection. The x and y position errors are shown in Figure E.5(**a**) whilst the Euclidean position error is shown in Figure E.5(**b**). Velocity error in x and y direction is found in Figure E.5(**c**) and the speed error is displayed in Figure E.5(**d**). Lastly, the course error is illustrated in Figure E.5(**e**).

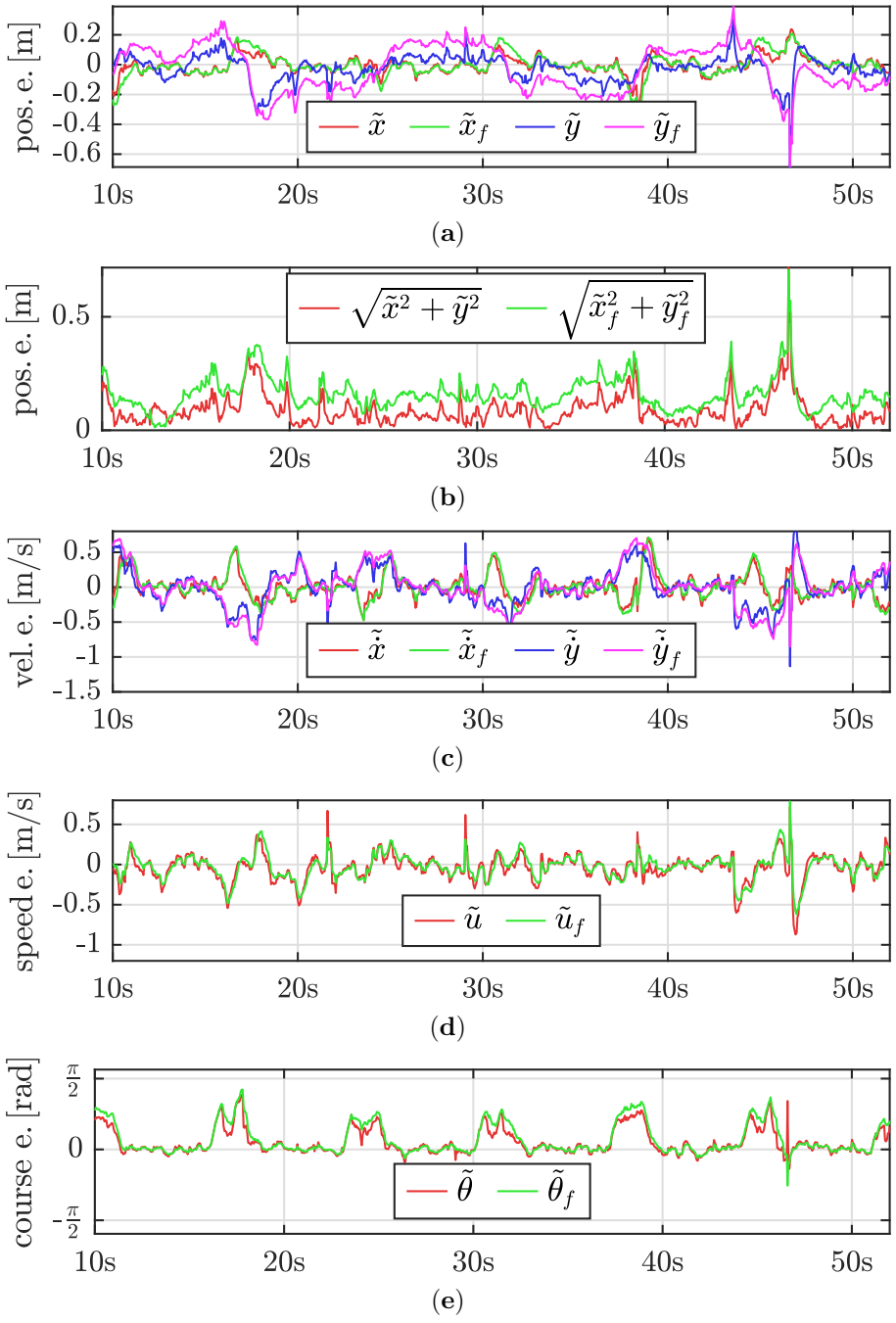

Figure E.5: Error between estimated values and reference path. (**a**), position error in x- and y-direction. (**b**), Euclidean distance error. (**c**), velocity error in x- and y-direction. (**d**), speed error. (**e**), course error.

E.6 Discussion

As the considered system is modelled with reduced order, we do not expect to meet all test criteria. Nevertheless, it is necessary to strive to make the filter as close to being consistent as feasible, but also try to acchieve as small errors as possible. In this application, it is of particular interest to get a stable and reliable course estimate.

In Table E.1 we found that none of the ANNES and ANIS test results are within the acceptance criteria defined in Equation (E.39) and Equation (E.50), respectively. This indicates that the filter is not consistent as we have a mismatch. This is likely caused by sensor bias that has not been evaluated individually for the sensors. It is also possible that the ground truth data is inaccurate as this was generated based on planned path and not recorded. However, this would not affect the NIS test. By investigating the NMEE results of the individual (non-augmented) states, one can see that the values are mainly accepted as zero according to Equation (E.45). The exception is NMEE_y where all values are too large. This can also be seen in Figure E.4(**b**) where μ_y is clearly outside the single sample bondary for more than 5% of the samples. This is one of the main contributors in the large errors seen in the NEES value in Figure E.4(**a**). Thus, it is possible that we have a sensor bias in y-direction. This bias is expected to be different for sensor N1-N6 depending on the mounting position. Further investigation of the sensor bias is left as possible future work.

The choice of tuning configuration depends heavily on the RMSE values. All of the listed position errors are acceptable for safely locating humans in the lab as the safety zones will need to be significantly larger than the worst position RMSE of 0.13 m. However, to predict a future position for safety applications, the course is important. Hence, the choice of configuration D is obvious as this has the lowest RMSE values. Here, the course, in particular, is cruicial. The SNRMSE metric, $\text{RMSE}_{\text{course}}$, is helpful in rating which configuration is better. Therefore, it can be a good cost function for optimization.

By investigating Figures E.3(**e**) and E.5(**e**) we see that the moving average filtering has reduced the errors for stable conditions at the expense of an added error during course change. Even if one would like the system to be both fast and stable it is not often possible. For human tracking and prediction in an industrial setting, a stable course during constant velocity is more valuable than precision of rapid course change. However, by utilizing the proposed method of the AKF, different states can be used for different applications. For example visualization and course estimation can use the averaged states while position can be determined by the regular state.

Some advantages found in using an AKF of dynamic size is that the filter will adapt to the number of received measurements whilst a constant filter is required to be scaled to the maximum number of sensors. If a large area is to be mapped by numerous sensors, it is likely that the dynamic filter can save some computation cost assuming all sensors will not have data to transmit at the same time. On the other hand, unless some sort of configuration lookup table is used, all sensors should have the same noise and bias properties. The work in this paper illustrates that this is not necessarily the case even if identical sensors are used.

The implementation of the proposed AKF should be further investigated with regards

to the statistical properties and tests that can be applied when the filter has variable innovation, and thus, variable matrix sizes. The statistical properties can be analyzed using simulations, but further analysis on real data should be based on high accuracy data such as laser tracker- or motion capture measurements. Further, a sensitivity analysis should be done as the filter is of reduced order compared to actual human motion and a model mismatch can be expected. A real-time online monitoring system of the filter performance would also be useful for industrial deployment.

E.7 Conclusion

In this paper an augmented Kalman filter with dynamic size to fuse a variable number of position measurements from RGB-D sensors has been implemented. The ability to handle a dynamic number of sensors is a significant advantage in terms of scalability. The method proposed in this paper would be able to cover a large volume with a larger number of sensors where the majority of the sensors do not see the human or are occluded by obstacles. In this paper sensor occlusion has been demonstrated by including an overhead gantry system in the workcell.

Compared to other forms of the Kalman filter, such as UKF and EKF, the computational cost of the augmented filter is low which again is an advantage for scalability.

The main novelty of the presented solution is the inclusion of filtered states in the Kalman filter for human velocity tracking. For 3D visualization and collision avoidance planning applications in particular it is desirable to use human velocity estimates which are not distorted by high-frequency noise.

Human motion tracking was achieved at a frame rate of 20 Hz, with a typical delay of 50 ms to 100 ms and an estimation accuracy of typically 0.10 m to 0.15 m. The lowest accuracies were achieved when the human changes direction, since a constant velocity model has been used in the Kalman filter.

Acknowledgment

The research presented in this paper has received funding from the Norwegian Research Council, SFI Offshore Mechatronics, project number 237896.

References

[1] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," Apr. 2018. arXiv: 1804.02767

[2] T. Linder, S. Breuers, B. Leibe, and K. O. Arras, "On multi-modal people tracking from mobile platforms in very crowded and dynamic environments," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2016. doi: 10.1109/ICRA.2016.7487766. ISBN 978-1-4673-8026-3 pp. 5512–5519.

[3] O. H. Jafari, D. Mitzel, and B. Leibe, "Real-time RGB-D based people detection and tracking for mobile robots and head-worn cameras," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, May 2014. doi: 10.1109/ICRA.2014.6907688. ISBN 978-1-4799-3685-4 pp. 5636–5643.

[4] E. Borøy and S. Jarlsby, "Human Detection and Position Estimation Using RGB-D Cameras in a Large Industrial Robotic Cell," Master's thesis, University of Agder, 2019.

[5] Y. Bar-Shalom, X.-R. Li, and T. Kirubarajan, *Estimation with Applications to Tracking and Navigation*. New York, USA: John Wiley & Sons, Inc., 2001. ISBN 047141655X

[6] R. R. Labbe, *Kalman and Bayesian Filters in Python*, 2018.

[7] S. Ripegutu, "Multi-Camera Person Tracking Using Positional Features, Kalman filters, and Unsupervised Classification," Master's thesis, University of Agder, 2019.

[8] F. Haugen, *Advanced DYNAMICS and CONTROL*. TechTeach, 2012. ISBN 9788291748177

[9] A. Aalerud, J. Dybedal, E. Ujkani, and G. Hovland, "Industrial Environment Mapping Using Distributed Static 3D Sensor Nodes," in *2018 14th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA)*. Oulu: IEEE, Jul. 2018. doi: 10.1109/MESA.2018.8449203. ISBN 978-1-5386-4643-4 pp. 1–6.

[10] A. Aalerud, J. Dybedal, and G. Hovland, "Automatic Calibration of an Industrial RGB-D Camera Network Using Retroreflective Fiducial Markers," *Sensors*, vol. 19, no. 7, p. 1561, Mar. 2019. doi: 10.3390/s19071561

[11] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA workshop on open source software*, vol. 3, no. 3.5, Kobe, May 2009, p. 5.