Fakultät für Mathematik, Informatik und Naturwissenschaften
Lehr- und Forschungsgebiet Informatik VIII
Computer Vision
Prof. Dr. Bastian Leibe

Seminar Report

Combining 3D Shape, Color, and Motion for Robust Anytime Tracking

Frederik Zwilling
Matriculation Number: 304314

June 2015

Advisor: Aljoša Ošep

Abstract

Object tracking is a critically important task in many robotic applications, especially for autonomous cars. Although there are already various approaches, low accuracy and poor robustness because of challenges, such as occlusion, viewpoint changes, and various object classes, still are a problem. The proposed method uses a probabilistic approach to combine cues from the 3D shape, color, and motion of the tracked objects. A so called annealed dynamic histogram is used to globally search the state space for the most likely position and velocity of the tracked object. This histogram is faster than ordinary ones because it is initialized with a coarse resolution and dynamically improve the resolution in the important areas. To counter the additionally error of the coarse resolution, the measurement model takes the resolution into account and anneals the model as the resolution is improved. Evaluation of the method shows that is outperforms all common baseline methods which are the Kalman filter and ICP with various improvements. The method achieves with similar runtime an at least 25% lower RMS tracking error. Furthermore, the tracker is robust against occlusion, viewpoint and lighting changes, and variation in the object classes.

Contents

(a) Cars on a highway (b) A cyclist (c) A pedestrian

Figure 1: Various objects that have to be tracked in challenging situations [HLTS15]

1 Introduction

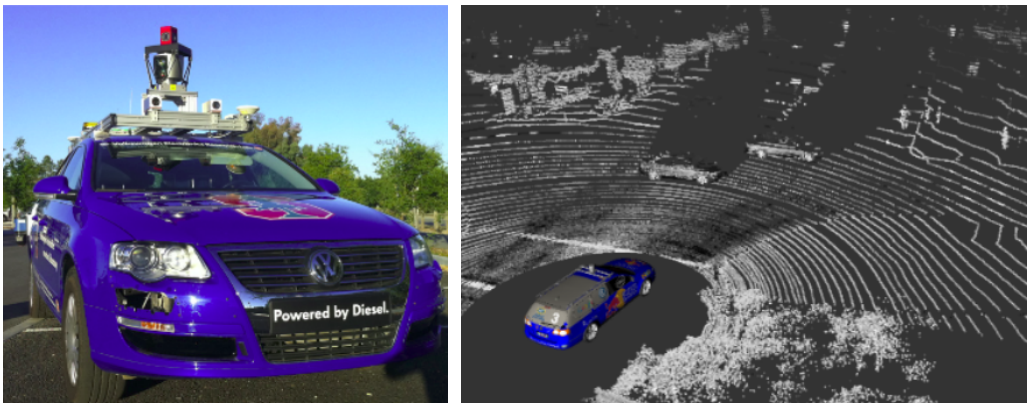
Robust and precise position and velocity estimation is an important part of object tracking. For example, it is an essential part in the collision avoidance of an autonomous car. This report presents a method by Held, Levinson, Thrun, and Savarese to solve this part of the tracking problem by combining clues of the 3D shape, color and motion of the tracked object [HLTS14]. The method uses a probabilistic measurement model derived from a Dynamic Bayesian Network. The search space, consisting of position and velocity, for each tracked object is represented in a special dynamic histogram, called *annealed dynamic histogram*. This histogram dynamically increases its resolution in the important areas and considers the local resolution in the measurement model. By expanding the measurement model with color and motion the performance of the method can be further improved. This is shown by the authors by evaluating in a static environment, where only the reference system is moving, and in a dynamic environment, where the compactness of the models build with the tracking results is used as evaluation criteria.

1.1 Motivation

Robotic applications are about to change many domains from the ground up. Especially autonomous systems could take over dangerous, exhausting and unpopular tasks and thus allow humans to do more satisfying tasks instead. Additionally, autonomous systems can be more efficient and scalable than solving the tasks by hand. Some progressive domains with autonomous robots are flying drones, which can map areas [RAK+07] or deliver packages [Ack13], autonomous cars, which take care of driving [Mar10], logistic robots, which store and grab goods in warehouses [Gui08], and domestic service robots, which can support old people and clean at home [IRdSvdZ12]. All these domains have in common that the robots have to track objects in their environment, mostly for avoiding collisions. In the case of domestic service robots, tracking is also needed for follow people. Often the reliability and precision of the tracking are limiting factors. An autonomous car, for example, can only drive fast if it is sure that it tracks all objects in the surrounding correctly and none of these objects could cause a collision. This report mainly focuses on the domain of autonomous cars. Here, the autonomous system takes care of the time consuming driving task on the one hand and could help to reduce the amount of traffic deaths (25,938 in 2013 in the EU [Eur15]) on the other hand. In this domain it is especially important to estimate the speed of various nearby objects robustly and in real time. Figure 1 shows the three main classes of objects that have to be tracked, cars, bicycles, and pedestrians, in challenging situations. The method proposed in this report solves this task and performs better than previous approaches in the car domain.

1.2 Tracking

Object tracking is the complex task to identify objects and their movement over time. It can be separated into the following parts. The first step segments the sensor data for time t, called *frame*, into detected objects. This can be done for example by separating foreground and background and find connected components



(a) (b)

Figure 2: A Velodyne LIDAR sensor mounted on a car (a) and a visualization of the point cloud generated by it (b) [TLT11].

in the foreground. In this step, it is beneficial to use the information that the most important objects to find are cars, bicycles and pedestrians [WPN12]. The second step is to associate detected object in the successive frames t and $t+1$. One method to find these matchings is to compute descriptors, such as the HOG descriptor, for the objects and find nearest neighbors in the descriptor space [TLT11]. The third step is the position and velocity estimation of the tracked object. That is the part of tracking this report is about. In the forth step, the computed trajectory of the tracked object is classified to decide, for example, if the object is dangerous to the car.

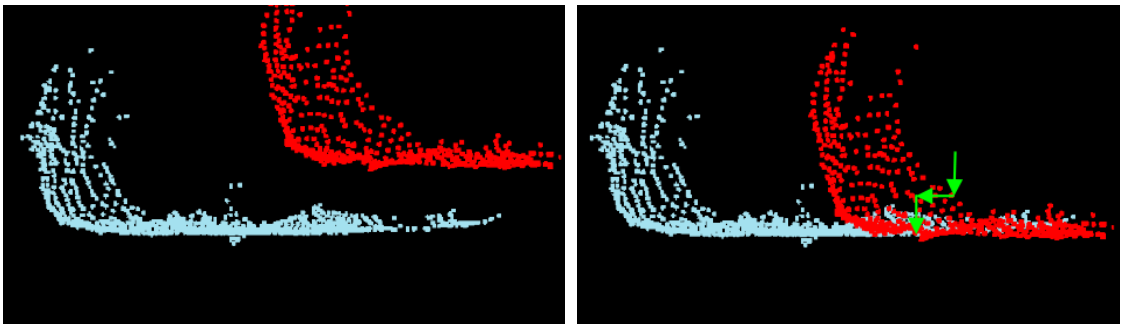
The data that is used for tracking in this report is generated by a dense laser sensor that measures the distance to surrounding objects with multiple rotating laser beams. Such a sensor and the data generated by it is shown in Figure 2. Additionally, the sensor provides a camera image similar to a panorama.

1.3 Velocity and Pose Estimation

To compute the pose and velocity estimation of an object, we first introduce a measurement model that is derived from a Dynamic Bayesian Network (*DBN*). This DBN models for each frame the dependencies between position, velocity, object surface and the observed measurement as well as the dependencies between two frames. Though, tracking is a hard problem because in cases with occlusion and major changes in the viewpoint the visible object surface can differ significantly and introduce an error. Especially in these cases, it is intuitive to also consider color and motion in the measurement model because the additional information is influenced less by occlusion and viewpoint changes.

To find the state, which is composed of position and velocity of an object, we use a histogram that maps the each histogram-chunk in the state space to the probability that this chunk causes the measurement according to our model. This allows a global search in the state space with multiple hypothesis. A standard histogram with adequate resolution would have to many chunks and thus would be to slow for real time computation. Therefore, we use a dynamic histogram that starts with a low resolution and an approximated posterior distribution. Then it dynamically increases the resolution in areas with high probability to cause the measurement. This allows to achieve a result after running the method for any time. The downside of the dynamic histogram is that the initial low resolution introduces an additional error. This can be balanced by including the resolution in the measurement model. We call this method *annealed dynamic histogram* because as the resolution increases, the distribution is annealed and approaches the true posterior.

The evaluation in Section 4 shows that the method outperforms other tracking methods by about 10%.



(a) Bad initial alignment (b) Stuck in a local maximum after 3 iterations (green arrows)

Figure 3: Possible case for bad ICP alignment results. The two point clouds belong a detected car in successive frames [HLTS15].

2 Related Work

The tracking problem has been studied for many years. The most common approaches to estimate the position and velocity of tracked objects, namely Kalman filters and Iterative Closest Point, are described in subsection 2.1. Often these approaches are efficient but sacrifice a lot of the available data by using simple representations or search only for a local maximum. Because our case of tracking objects in 3D point clouds from a laser sensor is only a special case of tracking, subsection 2.2 gives an overview of the tracking approaches with other sensors, mainly single cameras and stereo cameras. Subsection 2.3 presents grid-based approaches that are similar to annealed dynamic histograms.

2.1 Position and velocity estimation alternatives

Some simple approaches to estimate position and velocity of a tracked object first represent the object by its centroid [KMPS12, LAB+11] or bounding box [AA12, LHT+08, SFD02] and then use a Kalman filter [RN13]. This is a computationally efficient approach but discards a lot of available information by using such a simple representation. Therefore, this approach is especially fragile in cases of occlusion and viewpoint changes. In the occlusion case, the bounding box can contain only a part of the object and the centroid would be shifted. In the viewpoint-change case, the centroid and bounding box can significantly differ (e.g. when seeing a cyclist from the back and the side). Another disadvantage of the Kalman filter is that it can only represent a single hypothesis and thus performs poorly when multiple hypotheses are reasonable.

Other approaches include domain specific knowledge to use better fitting object representations. For example, the typical shape of a car with its corners or wheels can be use to achieve a more precise position of a car in a frame [WH12, DRU08, PT09]. This leads to better detection and association results, but is limited by the amount of trained object-classes. Although cars, pedestrians and cyclists are the most common classes, it is also important to detect uncommon obstacles in traffic (e.g. tractors, segways, and wheelchairs). The work presented in this report is based on [TLT11]. It allows recognition and classification of arbitrary objects and only requires labeled dataset to train the object-classes. Another approach that can track known objects as well as fully unknown objects is [ML12]. This approach differs from the previous ones by performing the tracking before the detection step what enables it to track unknown objects.

A widely used alternative to Kalman filters is the Iterative Closest Point (ICP) algorithm [FHB12, MS13, ML12]. The algorithm merges two point clouds by finding a maximum correspondence alignment. Thus the full 3D point cloud is used. It uses a hill climbing approach and therefore depends on a good initialization and can get stuck in a local maximum. This is the major drawback of ICP and has a large impact on the tracking precision as shown by [HLT13, Ols09]. Figure 3 visualizes how ICP can return a wrong alignment after an unlucky initialization. The approach presented in this report performs a global search and therefore has no problem with local optima.

2.2 Alternative Sensors

Laser range sensors, such as the one shown in Figure 2, are not the only kind of sensors used for tracking tasks. Although laser range sensors provide more precise data compared to other sensors used for tracking, they are inadequate for many applications because of their high price. A closely related alternative to laser range sensors is the usage of stereo cameras. They provide the same kind of data (point clouds and images) and are cheap. However, they have more noise then a laser sensor and can therefore cause larger errors in the tracing results. An example for the use of stereo data for tracking is [ML12]. It uses ICP to track pedestrians and unknown objects. Object tracking with a single camera has been studied for a long time and is used in many applications, e.g. in video calling [KH95]. However, single camera tracking is unsuitable in the autonomous car scenario because of missing depth information.

2.3 Grid-Based Methods

Ordinary grid-based search approaches in computer vision are used to globally search in a state space and allow multiple hypothesis in contrast to Kalman filters and ICP. However, the computational effort increases with the resolution of the grid. Therefore, ordinary histograms are often inadequate for domain which require real time computation. Dynamic histograms tackle this problem by starting with a low resolution and increasing the resolution in important areas. However, this creates the problem that the low resolution introduces an additional error.
There already are some grid-based approaches which tackle this problem. Most of them are used in Simultaneous Localization and Mapping (*SLAM*). [ECM04] uses a grid-based Monte-Carlo method that recursively expands grid cells with the most hypotheses and refines the grid resolution in these cells. This only differs from annealed dynamic histograms in the usage of a Monte-Carlo distribution instead of probabilities to cause the current measurement. The approach presented in [MSM02] self-localizes a robot with a dynamically refined grid. Here, each measurement point votes for the cells that could have caused the measurement. The cells with the most votes are refined in the next iteration. Similarly to the paper presented in this report, the approach enables anytime computation.
The main differences between previous methods and the method presented in this report are the additional consideration of color, what is not directly possible in Kalman filter or ICP approaches, and the consideration of the probabilities for points being occluded in previous frames. This leads to higher tracking robustness. Furthermore, annealed dynamic histograms allow multiple hypotheses and globally searching the state space. In contrast to other grid-based approaches, it also allows real-time computation without introducing a high error resulting from low resolution by annealing the measurement model as the resolution increases.

3 Method

The method presented in this report consists of three major parts. Subsection 3.1 introduces the probabilistic model based on a Dynamic Bayesian Network. The model is necessary to determine how likely it is that a specific position and velocity of a detected object causes the observed measurement. In Subsection 3.2 the extension of the measurement model by color and motion is shown [HLTS14]. Subsection 3.3 describes how annealed dynamic histograms are used to search the state space for the most likely position and velocity according to the measurement model. For that especially the refinement steps, that increase the resolution, and the annealing of the measurement model depending on the resolution are important.

3.1 Probabilistic Model

To derive the probability for a position and velocity to cause the observed measurement seen in the last frames, we use a Bayesian Network because it allows us to easily combine 3D shape, color, and motion cues [HLTS14]. After encoding the influences of the states, consisting of position and velocity, the surface of the object to track and the observed measurement, we can derive the wanted probability of the measurement given the state by applying well known probability rules in Bayesian Networks [RN13]. Furthermore, we use a Dynamic Bayesian Network to relate variables over successive time steps.

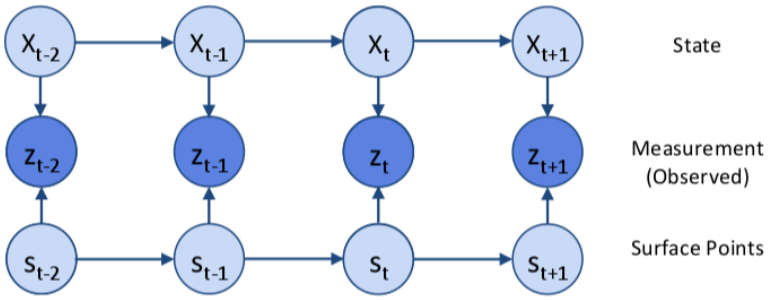


Figure 4: Dynamic Bayesian Network to derive the measurement model for tracking an object [HLTS14].

The Dynamic Bayesian Network we use to derive the measurement model is shown in Figure 4. For the frames $t-2$ to the current frame $t+1$, it includes the variables x for the state, z for the observed measurement and s for the visible surface of the object. In the following, we describe these variables in detail.
As state variable x_t, we use the composition of position and velocity $x_t = (x_{t,p}, \dot{x}_{t,p})$. $x_{t,p}$ is the linear position of the object centroid in frame t relative to the centroid of the object in frame $t-1$. In other words, the centroid of the object in frame $t-1$ is the origin of the coordinate system for $x_{t,p}$. $\dot{x}_{t,p}$ is the velocity of the object. The relative rotation and the rotational velocity of the object is not included in the state. This speeds up the method because a lower dimensional state space is considered. The authors claim that omitting the rotation is no problem because the rotational velocity of the objects present in traffic is small relative to the frame rate of the sensor, which is $10Hz$. Similarly, also the vertical movement of the objects is small because the objects we are interested in move along the ground. Therefore, we can use a 2D position and velocity instead of a 3D position and velocity. To use the methods in other domains that require considering the vertical velocity, the state space can be expanded, what results in a slow down of the method.
To include the 3D shape of the object in the model we use the latent surface variable s_t. It represents the visible surface of the object and is a set of n points $\{s_{t,1}, ..., s_{t,n}\} = s_t$ that are sampled from the visible surface. Although the true 3D shape of the object is unknown and only indirectly observable through the measurement, it is important for the model. This is similar to SLAM methods which model the environment map that corresponds to the surface in our terminology.The measurement then depends on the modeled map and the localization, or the object surface and state in our case.
When looking at the Dynamic Bayesian Network in Figure 4, it is easy to see that considering the surface is important because, when it is omitted, two successive measurements would be independent given the state variable. This would be wrong because the shape of the object is relatively consistent and therefore also the measurement points describe a similar surface. In contrast to SLAM, it is not our goal to build the shape of the observed object. We simply integrate over shapes to get the surface. The prior of the surface points $p(s_{t,i})$ is a uniform distribution over the maximum size of an object. The prior of the surface is the product over its points $p(s_t) = \Pi_i p(s_{t,i})$. The observed measurement z_t is again a set of measurement points $z_t = \{z_{t,1}, ..., z_{t,m}\}$. These points depend on the latent surface s_t and the position $x_{t,p}$. However, the measurement points do not lie directly on the surface as illustrated in Figure 5. This is caused by the sensor noise, which depends on the sensor resolution and is modeled as Gaussian noise Σ_e. The sensor resolution can roughly be modeled as a linear function of the distance between

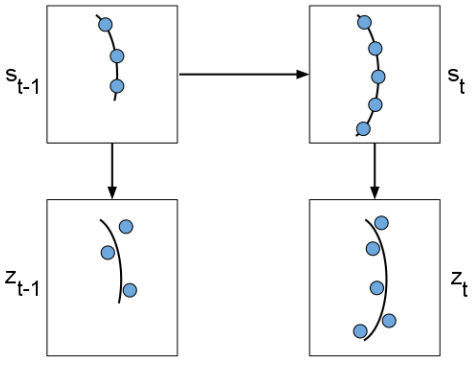


Figure 5: Measurement points z_t sampled with sensor noise Σ_e from surface points s_t. The visible surface, from which the surface points are sampled, varies from frame to frame due to occlusion and viewpoint changes [HLTS14].

object and sensor. The generation of measurement points z_t depending on the surface points s_t, the sensor noise Σ_e and the position $x_{t,p}$ can be written as

$$z_{t,i} \sim \mathcal{N}(s_{t,i}, \Sigma_e) + x_{t,p} \ . \tag{1}$$

Because the origin of the coordinate system is placed at the centroid of the object in the previous frame, the measurements of the previous frame are not shifted

$$z_{t-1,i} \sim \mathcal{N}(s_{t-1,i}, \Sigma_e) \ . \tag{2}$$

Therefore, z_{t-1} is conditionally independent from x_{t-1} what is not derivable by just using the Bayesian Network in Figure 4.
For the derivation of the measurement model, we will also need the term $p(s_t|s_{t-1})$ which stands for the probability of sampling surface points s_t from the currently visible surface given s_{t-1} from the previous frame. The sampled points might differ because of occlusions, viewpoint changes, random sampling from the surface and deformations of the object (e.g. body movement of a walking pedestrian). Every point of s_t could have been generated from a visible or occluded part of the surface in frame $t-1$. When we consider the prior probability $p(V)$ for sampling from a previously visible surface, we can use the joint distribution over these two cases:

$$\begin{aligned} p(s_{t,i}|s_{t-1}) =& p(V) * p(s_{t,i}|s_{t-1}, V) + \\ & p(\neg V) * p(s_{t,i}|s_{t-1}, \neg V) \end{aligned} \tag{3}$$

$p(s_{t,i}|s_{t-1}, \neg V)$ stands for the probability that $s_{t,i}$ is generated from a part of the surface that was not visible in the previous frame. Under the assumption that every non visible point is occluded, we can rewrite the term with constants k_1 and k_2:

$$p(s_{t,i}|s_{t-1}, \neg V) = k_1(k_2 - p(s_{t,i}|s_{t-1}, V)) \tag{4}$$

This allows simplifying Equation 3 to

$$p(s_{t,i}|s_{t-1}) = \eta_1(p(s_{t,i}|s_{t-1}, V) + k) \tag{5}$$

with the normalization constant $\eta_1 = p(V) - p(\neg V)k_1$ and the smoothing factor $k = p(\neg V)k_1k_2/\eta_1$. These constants can later be found by training.
$p(s_{t,i}|s_{t-1}, V)$ can be modeled by a Gaussian distribution

$$s_{t,i} \sim \mathcal{N}(s_{t-1,j}, \Sigma_r) \tag{6}$$

where $s_{t-1,j}$ is the closest corresponding surface point from the previous frame and Σ_r is the variance mainly resulting from the sensor resolution and the object deformation. Combining these Gaussians in Equation 5 for all surface points in s_t yields

$$p(s_t|s_{t-1}) = \eta(\mathcal{N}(s_t; s_{t-1,i}, \Sigma_r) + k). \tag{7}$$

For the measurement model used in the annealed dynamic histogram, we need to derive $p(z_t|x_t, z_1, ..., z_{t-1})$, the probability for the current measurement given the histogram cell with state x_t and the previous measurement history z_1 to z_{t-1}. Though considering all previous measurements of the object would be computationally costly. Although the measurements in different time frames are not independent given the state, as discussed earlier, we can focus only on the previous one as approximation, because the previous one has the highest impact of all measurements in the history.

$$p(z_t|x_t, z_1, ..., z_{t-1}) \approx p(z_t|x_t, z_{t-1}) \tag{8}$$

This approximation can be rewritten by using the joint distribution over all surface points in the current and previous frame:

$$p(z_t|x_t, z_{t-1}) = \iint p(z_t, s_t, s_{t-1}|x_t, z_{t-1}) \mathrm{d}s_t \mathrm{d}s_{t-1} \tag{9}$$

Because of the chain rule of probabilities and the conditional independences that could be derived from the Dynamic Bayesian Network in Figure 4, we can refine the term in the integral:

$$\begin{aligned} p(z_t, s_t, s_{t-1}|x_t, z_{t-1}) &= p(z_t|s_t, x_t)p(s_t|s_{t-1})p(s_{t-1}|x_t, z_{t-1}) \\ &\overset{(2)}{=} p(z_t|s_t, x_t)p(s_t|s_{t-1})p(s_{t-1}|z_{t-1}) \\ &= p(z_t|s_t, x_t)p(s_t|s_{t-1})\eta_2 p(z_{t-1}|s_{t-1})p(s_{t-1}) \end{aligned} \tag{10}$$

η_2 is again a normalization constant and can be used to absorb the constant term $p(s_{t-1})$. This term is constant because it is the product of a constant number of uniformly distributed probabilities of the surface points. This allows simplifying the term in the integral further:

$$p(z_t, s_t, s_{t-1}|x_t, z_{t-1}) = \eta \underbrace{p(z_t|s_t, x_t)}_{(1)} \underbrace{p(s_t|s_{t-1})}_{(7)} \underbrace{p(z_{t-1}|s_{t-1})}_{(2)} \tag{11}$$

For this term, we have modeled all probabilities as Gaussians as indicated under Equation 11. Therefore, the term only consists of a product of Gaussians and Gaussians plus a constant. If we plug in Equation 11 back into our integrals in Equation 9, we can apply a convolution of Gaussians to solve the inner integral [TBF05]. Because the convolution of two Gaussians results in a Gaussian, we can apply a convolution of Gaussians again to evaluate the outer integral and yield a Gaussian for the measurement model:

$$p(z_t|x_t, z_{t-1}) = \eta(\mathcal{N}(z_t; z_{t-1} + x_{t,p}, \Sigma_r + 2\Sigma_e) + k) \tag{12}$$

For the computation of the measurement model, we introduce the variable $\bar{z}_{t-1} = z_{t-1} + x_{t,p}$ so that the measurement in the previous frame is shifted according to x_t. Then for each $z_{t,i} \in z_t$, we have to find the closest corresponding point $\text{ccp}(z_{t,i}) = \bar{z}_{t-1,j} \in \bar{z}_{t-1}$. The measurement probability for the current measurement given the inspected state and the previous measurement can be computed by

$$p(z_t|x_t, z_{t-1}) = \eta \left(\prod_{z_{t,i} \in z_t} \exp\left(-\frac{1}{2}(z_{t,i} - \text{ccp}(z_{t,i}))^T \Sigma^{-1} (z_{t,i} - \text{ccp}(z_{t,i})) \right) + k \right) \tag{13}$$

with normalization constant η, smoothing factor k, and covariance matrix $\Sigma = 2\Sigma_e + \Sigma_r$. Again these constants and the function for Σ_r, which depends on the distance between sensor and tracked object, can be found by optimizing for a set of training data.
The computation with Equation 13 works well if the point clouds z_t and z_{t-1} have a similar size. Because of occlusion the size of the point clouds can vary. This is a problem because Equation 13 is not symmetric with respect to z_t and z_{t-1}. For example when a part of the detected object becomes unoccluded in comparison to the previous frame, z_t is larger than z_{t-1} and therefore each point in z_t that has no nearby correspondence in z_{t-1} adds a large penalty. This would result in an incorrect alignment because the tracker would try to minimize the number of badly aligned points and therefore focuses on aligning the densest parts of the point clouds. This problem can be solved by exchanging z_t and z_{t-1} so that always the smaller point cloud is aligned in the larger one. In the case of such an exchange, the implementation also has to consider the switch of the coordinate system origin.

3.2 Adding Color and Motion

So far, the measurement model considers the 3D shape of the object to compute how likely it is that a state causes the measurement. To increase the robustness and precision of the tracker, we also want to consider clues from motion and color [HLTS14]. This especially improves the performance in comparison to ICP which only uses the object shape and has no motion model.

3.2.1 Motion Model

For the motion model, we use a constant velocity model of a Kalman filter [TBF05]. In each iteration, the Kalman filter performs a measurement step and a motion step. In the measurement step, the algorithm is fed with a Gaussian that represents the probabilities for the states to cause the current measurement. In the

motion step, the expected motion of the object in the time till the last iteration is applied to the position of the object. Both steps are computed by combining two Gaussians. As a result, the algorithm gives a Gaussian for the position and motion of the object.
We just need to give the algorithm a Gaussian for the measurement step. For this, we can use the results of the measurement model that has to be represented as a Gaussian. This can be done by computing the mean μ_t and variance Σ_t over all states weighted by their probability $p(x_t|z_1,...,z_t)$.

$$\mu_t = \sum_i p(x_{t,i}|z_1,...,z_t)x_{t,i}$$
$$\Sigma_t = \sum_i p(x_{t,i}|z_1,...,z_t)(x_{t,i}-\mu_t)(x_{t,i}-\mu_t)^T$$

where $x_{t,i}$ is the state of sample i.

3.2.2 Color Model

The model we have developed up to this point considers the 3D shape and motion of tracked objects. As presented in the evaluation in Section 4, this model can be used to implement the tracker and yields reliable results. Although we also want to add color information to the model, this is important because color information is not always available (e.g. when the autonomous car is driving in the dark). To incorporate into the model, that the color of the surface points should match between two successive frames, we first need to learn the probability distribution of color matchings. This can be done by analyzing a dataset with already known alignments. For an autonomous car, such a dataset can be easily obtained by driving in a static environment (e.g. along a row of parked cars). The alignment of the tracked objects can then be computed with the speed of the car and the position of the tracked object relative to the car. For the analysis of color matchings, then the nearest points of two subsequent measurements can be used. It is reasonable to use only the correspondence pairs with a distance under a certain threshold (5cm was used by the authors of the paper). Although it would be intuitive to use all available color channels, we use only the blue color channel. This simplifies the problem, because we do not need to learn the covariance between the color channels. The blue color channel was chosen by experiments with a validation set and already increases the performance of the method. The resulting distribution for color differences follows a Laplacian distribution. To include the color distribution into our measurement model, we extend the calculation of the term $p(s_{t,i}|st-1,V)$ what is intuitive because the color is a property of the surface and, to know the color, the point has to be visible in the previous frame:

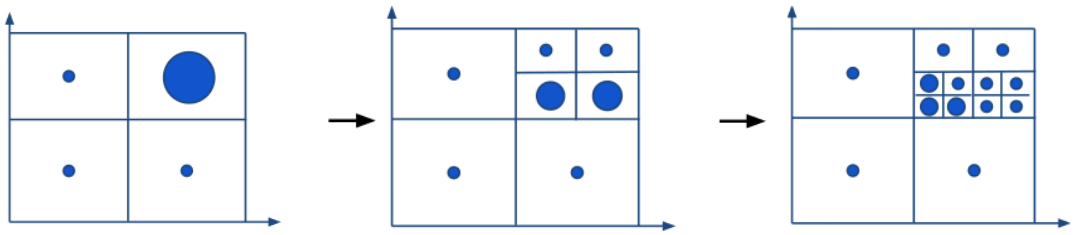
$$p(s_{t,i}|s_{t-1},V) = p_s(s_{t,i}|s_{t-1},V)p_c(s_{t,i}|s_{t-1},V) \tag{14}$$

where $p_s(s_{t,i}|s_{t-1},V)$ is the probability for the spatial matching as before and $p_c(s_{t,i}|s_{t-1},V)$ is the probability for the color match. For the term $p_c(s_{t,i}|s_{t-1},V)$, it does not suffice just to use the learned color model. This is due to changes in lighting, lens flare, and other cases that can change the observed colors significantly. Therefore, we need to introduce a probability $p(\neg C)$ for cases where the learned color model can not be applied. Now we can refine the probability for a color match:

$$\begin{aligned} p_c(s_{t,i}|s_{t-1},V) =& p(C)p_c(s_{t,i}|s_{t-1},V,C)+ \\ & p(\neg C)p_c(s_{t,i}|s_{t-1},V,\neg C) \end{aligned}$$

$p_c(s_{t,i}|s_{t-1},V,C)$ is the color distribution learned from the training set. $p_c(s_{t,i}|s_{t-1},V,\neg C)$ is the probability for a point having any color that does not match the color in the previous frame. Because we use 256 color intensities and we assume for simplicity that the colors are uniformly distributed, we can use $p_c(s_{t,i}|s_{t-1},V,\neg C) = \frac{1}{255}$.
The probability $p(C)$ that the color of two corresponding points in successive frames should match could be trained again. Though, there is a problem with the resolution of the sample in the annealed dynamic histogram. For coarse resolutions, the likelihood of a color matching is smaller than for good resolutions. Therefore, we can model $p(C)$ as function of the sampling resolution:

$$p(C) = p_c \exp\left(\frac{-r^2}{2\sigma_c^2}\right) \tag{15}$$



(a) Initialization with a coarsely divided state space (b) Cell subdivision and computation of the sample alignment probability for new cells (c) Further subdivision of multiple cells in one step

Figure 6: Dynamic refinement of an annealed dynamic histogram. The grid cells divide the state space. The size of the circles indicates the probability for the state in the cell given the measurements [HLTS14].

with sampling resolution r (as interval size, not as number of intervals), parameter p_c for the probability with ideal resolution $r = 0$, and parameter σ_c which defines how fast $p(C)$ decreases with improving resolution. Now we can revise the measurement probability by using the color probability

$$\begin{aligned} p_c(z_{t,i}|x_t, z_{t-1}, V) =& p(C)p_c(z_{t,i}|\text{ccp}(z_{t,i}), V, C) + \\ & p(\neg C)p_c(z_{t,i}|\text{ccp}(z_{t,i}), V, \neg C) \end{aligned} \tag{16}$$

what results in

$$p(z_t|x_t, z_{t-1}) = \eta \left(\prod_{z_{t,i} \in z_t} p_s(z_{t,i}|x_t, z_t - 1) p_c(z_{t,i}|x_t, z_t - 1, V) + k_3(k_4 - p_s(z_{t,i}|x_t, z_t - 1)) \right). \tag{17}$$

with $k_3 = k/(k+1)$, where k is from Equation 13, and the smoothing parameter k_4 that has to be found by training.

3.3 Annealed Dynamic Histograms

With the measurement model we have derived up to now, the probability for the current measurement given an assumed state can be computed. To search the state space for the state that is most likely given the measurement, we use an *annealed dynamic histogram* [HLTS14]. We use a histogram because it allows to globally search the state space, in contrast to ICP, and to have multiple hypothesis. The histogram divides the state space into grid cells. For each of these grid cells, we compute the probability $p(x_t|z_1, ..., z_t)$. In an ordinary histogram, it would be necessary to densely sample the state space to get a precise result. This would be computationally costly and prevents the use in a system with real-time requirements. Therefore, we use a dynamic histogram where only the areas with a high probability are densely sampled. Subsection 3.3.1 describes how the annealed dynamic histogram is initialized with a coarse resolution and is then dynamically refined to increase the density in the important areas. Though, the coarse resolution and the dynamically change of it have the drawback that in the areas with coarse resolution, the measurements are less likely to fit than in the densely sampled areas. How this problem is solved by considering the resolution of a grid cell in the measurement model is presented in Subsection 3.3.2. How the tracking estimate can be returned as result is shown in Subsection 3.3.3.

3.3.1 Dynamical Refinement

To build the annealed dynamic histogram, we start with a coarsely divided state space and compute the sample alignment probability $p(x_t|z_1, ..., z_t)$ for each cell. Figure 6(a) shows an example initialization. Then a cell is chosen to be divided into k sub-cells as in Figure 6(b) and 6(c). For the newly created cells, the sample alignment probability is computed again. To compute the probabilities for the new sub-cells, we first define $|c|$ as the volume of cell c and R as the region of cell we chose to subdivide into sub-cells c_i.

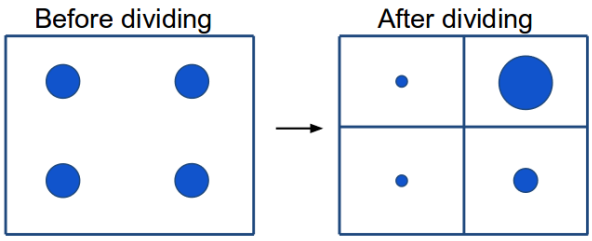


Figure 7: Before dividing a parent-cell, we assume that its probability is equally distributed among the k sub-cells. After dividing, each sub-cell has its own probability. [HLTS14].

This region does not have to be contiguous. The new probability $p(c_i)$ of sub-cell c_i can be computed as follows:

$$\begin{aligned} p(c_i) &= p(c_i \cap R) \\ &= p(c_i|R)p(R) \\ &= \frac{p(x_i|z_1,...,z_t)|c_i|}{\sum_{c_j \in R} p(x_j|z_1,...,z_t)|c_j|} p(R) \\ &= \eta p(x_i|z_1,...,z_t)p(R) \end{aligned} \tag{18}$$

Thus, the intuitive property $\sum_{c_i \in R} p(c_i) = p(R)$ holds. η is again used for normalization and depends only on the the sub-cells in region R. Therefore, the probabilities of all cells outside of R are not changed.
Loosely speaking, the criteria that chooses which cells to divide in the next step, tries to maximize the information gain. Preferred subdivisions result in a more definite distribution of probabilities in the sub-cells. In other words, the probability of the cell to divide is condensed in a few sub-cells instead of uniformly distributed. To measure this information gain, we inspect the distribution in the possible sub-cells before and after dividing. Let distribution B be the currently estimated distribution in the sub-cells before dividing. Because we only know the probability of the parent-cell, we assume that this probability is equally distributed among the sub-cells as illustrated in the left of Figure 7. After dividing the parent-cell, each of the k sub-cell has its own probability and the only constraint is that the sum of the sub-cell probabilities has to match the parent-cells probability $\sum_{j=1}^{k} p_j = P_i$. The new distribution of probabilities among the sub-cells is called A and is a better approximation of the true posterior than B.
The information gain can be computed with the Kullback–Leibler divergence *(KL-divergence)* [Kul97]. The KL-divergence of B from A measures the information loss when A is approximated with B. In our case, we can compute the KL-divergence with

$$D_{KL}(A||B) = \sum_{j=1}^{k} p_j \ln\left(\frac{p_j}{P_i/k}\right) \tag{19}$$

where A and B are the distributions after and before dividing as described before. In the mostly preferred case of a subdivision, $p_{j'} = P_i$ for one j' and $p_j = 0$ for all $j \neq j'$. This is also the case in which the KL-divergence has its maximum value

$$D_{KL}(A||B) = P_i \ln(k). \tag{20}$$

The minimum value of the KL-divergence $D_{KL}(A||B) = 0$ is reached when the probability P_i of the parent-cell is uniformly divided among the sub-cells. This is truly the minimum value because the KL-divergence is always non-negative [Kul97] and in this case there is no profit in dividing this parent-cell. With Equation 20, we can derive that the maximum KL-divergence per second is $P_i \frac{\ln(k)}{kt}$ where t is the time to compute a probability for a sub-cell with the measurement model. Because k is constant and t should roughly be the same over the whole state space, we have to divide the cells with the highest probability first to convert as fast as possible to the true posterior. If we want to divide all cells that have a higher probability than

a threshold p_{min}, we can also divide all these cells in each iteration, as it is done in Figure 6. This also yields a simple termination criteria. To maximize the the convergence speed to the true posterior, k should be chosen as small as possible. For example if the histogram has d dimensions, $k = 3^d$ could be used.

3.3.2 Annealing

With the division of cells into k sub-cells and the criteria to choose which cells to divide, we have a dynamic histogram that works in principle. Because of the coarse resolution of the histogram, especially at the initialization, we can not expect to find a good alignment between the two point clouds in the current and previous frame. The coarse resolution thus introduces an additional error-source in the method. A solution for this problem is to include the error resulting from the resolution as variance into the measurement model. We increase the variance of the measurement model Gaussian by a value Σ_g. Because the introduced error is expected to be smaller for a better resolution, Σ_g is expressed as a function proportional to the resolution in the currently inspected cell. The updated variance of the measurement model then is $\Sigma = 2\Sigma_e + \Sigma_r + \Sigma_g$. As the resolution improves, the term for Σ_g decreases towards 0. Therefore, the measurement model is annealed as the resolution is improved.

3.3.3 Resulting Tracking Estimate

After terminating because all cell-probabilities are below the threshold p_{min} or the computation was stopped due to real-time restrictions, the result of the position and velocity estimation can be returned. There are multiple possibilities what result can be returned by the histogram. The simplest approach would be to return the mean and variance of the distribution. An alternative would be to return the mode of the distribution. For sophisticated planners which would be used in an autonomous car, also the complete probability distribution as a density tree could be returned [TFBD01]. Using the mean of the distribution has the advantage that this would minimize the *Root-Mean-Square error (RMS error)* as shown in [HLTS14]. The method only determines the position and velocity of the tracked object. If an application also requires the rotation, this can easily be computed by using the computed frame-to-frame alignment and then performing coordinate descent while searching only over the rotation.

4 Evaluation

This section evaluates the tracker using the position and velocity estimation with annealed dynamic histograms and a model which combines cues from 3D shape, color, and motion. It presents the experiments and results of [HLTS14]. The main criteria of the evaluation are the robustness, the precision, and the runtime of the proposed method. The performance in these criteria is compared with other widely used tracking methods for position and velocity estimation, such as ICP and Kalman filters.
Although there are some existing evaluation methods for tracking, they mostly are not applicable or not suitable for the autonomous driving application. Many standard evaluation techniques are not applicable because the method we want to evaluate only focuses on a sub-part of tracking, namely position and velocity estimation. From the tracking evaluation techniques for tracking multiple objects, the metric called *multiple object tracking accuracy (MOTA)* can not be used because it incorporates penalties for wrong assignments called id switches. *Multiple object detection accuracy (MODA)* considers the amount of incorrect detections (e.g. false positives). *Multiple object detection precision (MODP)* measures the position detection in single frames (e.g. to find the centroid). These three metrics are not applicable here, because the presented method is not responsible to solve these problems. The metric called *multiple object tracking precision (MOTP)* would be suitable for the method because it only evaluates the precision of the position estimation given a correct matching [KR08, SN07]. However, the authors of the presented paper focus on measuring the *Root-Mean-Square error (RMS error)*

$$e_{RMS} = \sqrt{\mathbb{E}((\hat{v}_t - v_t)^2)} \tag{21}$$

where $\mathbb{E}$ is the mean and $\hat{v}_t - v_t$ is the velocity tracking error. We focus here on the velocity instead of the whole state with position and velocity because the position $x_{t,p}$ in the state is relative to the objects

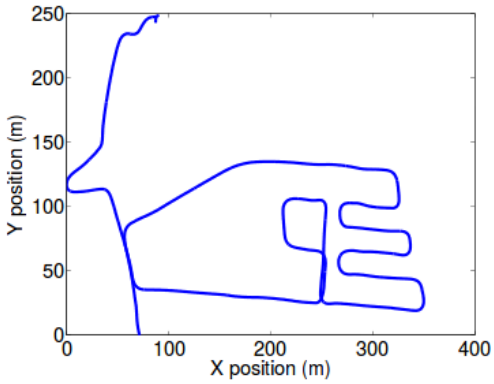
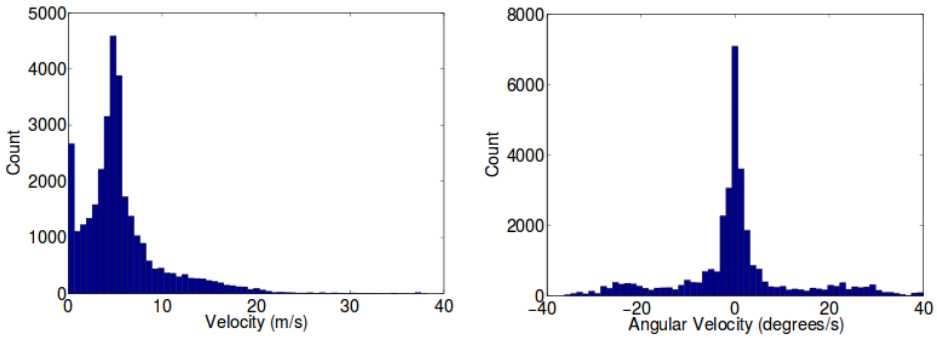


Figure 8: Path taken by the tracking car on the parking space [HLTS14].

position in the previous frame and mainly used for estimating the velocity with the motion model. The absolute position of the tracked object can easily be computed as the centroid of the point cloud and thus has not to be evaluated here. In this evaluation, the RMS error is averaged over all tracked objects. It is a widely used metric and has the property to give large errors a higher influence than many small ones have. A problem with the RMS error is that it significantly depends on the distance to the tracked object. Intuitively, the tracking error for distant objects is higher than for close objects because the point clouds have less measurement points. Because the RMS error is averaged over all tracked objects, distant objects are have a higher influence on the error than near ones. This could be changed by using the RMS error in relation to the distance [HLTS14].
To properly evaluate the performance of the tracker, a large dataset with many different objects to track is needed, because in practice an autonomous car has to track many different objects with different shapes, color distributions, and motion patterns. Therefore, the acquisition of evaluation data has to be scalable. This excludes methods as done by [MLvHW11] where the tracking and tracked cars were equipped with additional sensors to measure the positions and velocities. The authors of the paper presented in this report propose two approaches to automatically generate test data and evaluate the tracking with it. In both approaches, the car was driven around and the sensor data with object detection and object assignment was recorded. Therefore, a large amount and variety of tracked object in the surrounding of the car is automatically recorded and the data is taken from a environment that is close or equal to the application environment. In the first approach which is presented in Subsection 4.1, the car drives in a parking space where the tracked objects are not moving. By assuming that the car is standing still, the tracked objects seem to be moving and their velocity can be tracked. The results can be compared to the velocity calculated with the movement of the car and the distance to the tracked object. The second approach is described in Subsection 4.2 and utilizes a recorded dataset from the car driving in real traffic. Using the relative positions of the point clouds calculated by the tracker, the point clouds can be merged to build a model of the tracked object. Then a metric is introduced to measure the quality of the build model.
Before the tracking can be evaluated, the various parameters introduced in the method have to be determined. This was done by using a training dataset that is different from the test sets used later. The training dataset was generated by recording while the car drives past parked cars. The approach is similar to the first evaluation approach. The most important parameters determined in [HLTS14] are the initial coarse resolution of 1m for the annealed dynamic histogram, the variance $\Sigma_g = gI$ to anneal the measurement model, where g is the cell resolution and I is the identity matrix, and $p_c = 0.05$ for the color model. This results in a low probability $p(C)$ for a color match because of many underexposed frames and frames with lens flare.

4.1 Relative Reference Frame

In the first approach, the test data is recorded by a car driving on a parking space. During the recording of 6.6 minutes, about 560 cars were tracked. This large dataset allows a good quantitative evaluation. The objects were tracked in a local reference frame. This means that the tracker assumes that the recording



(a) Distribution of linear velocities

(b) Distribution of rotational velocities

Figure 9: Distribution of the ground-truth velocities of the tracked objects in the local reference frame [HLTS14].

car stands still and the tracked objects move around it. The path of the car through the parking space is shown in Figure 8. Even in this simplified scenario, many real world challenges for the tracking are present in the dataset. Viewpoint changes, changes in occlusion and color changes due to reflections from different viewpoints are similar to driving in traffic. During the recording, the linear and rotational velocity of the tracking car is logged. With this information, the ground-truth velocity of the tracked objects can be computed. When the car is driving just forward, this is easy. When the car is turning, the relative position of the tracked objects is used additionally. Because of many objects being inside and outside the turning circle of the car, the distribution of velocities of the tracked objects is wider than the velocity distribution of the recording car. These velocity distributions are plotted in Figure 9(a) and 9(b).

To evaluate only the position and velocity estimation part of the tracking pipeline, some tracked objects over a number of successive frames, called tracks, are filtered out. About 7% of the tracks were filtered out due to undersegmentation issues. Undersegmentation occurs when multiple nearby objects are not distinguished in the segmentation step and therefore wrongly detected as one object. These tracks are filtered out because of the ambiguity in the ground-truth velocity. Tracks with oversegmentation, where one object is split into multiple pieces, is not filtered out.

We now compare the performance of the proposed tracking method with several widely used baseline methods. One the one hand, we compare the RMS error and on the other hand the mean runtime. Figure 10 shows the results of the comparison. Plotted are RMS errors at different mean runtimes for the method presented in this report as well as for other baseline methods.

The first baseline method we use for comparison is a Kalman filter that uses only the centroid of point cloud in a frame in its measurement model. This method is widely used because of the easy implementation and the fast computation. As shown in Figure 10 the Kalman filter is extremely fast with a computational time below 0.1 milliseconds. This was expected because the Kalman filter only uses the centroid instead of aligning two point clouds. The RMS error is 0.78 $\mathrm{m/s}$ and shows that the Kalman filter is not very accurate. For the Kalman filter, there is only a single mean runtime because the Kalman filter only has a single computation step and no dynamic improvement steps.

Furthermore, we analyze the performance of ICP in different variants. These variants differ in the initialization of ICP and whether they are used as a measurement model in a Kalman filter. The basic point-to-point ICP algorithm is described in [RC11] and uses the centroids of the point clouds as alignment-initialization. The algorithm ran with up to 100 incremental alignment iterations. The RMS error with different mean runtimes is shown in Figure 10 and is about 1 $\mathrm{m/s}$. This is 23% worse than the RMS error of a simple Kalman filter and therefore no good tracking performance. This bad performance in comparison to the Kalman filter is caused by the missing motion model which is important for robust tracking. Without a motion model, only the last two frames can be used to estimate the velocity of the tracked object. Therefore, a single incorrect alignment as, for example, show in Figure 3 significantly impacts the tracking error. A Kalman filter with its motion model can refine the result of many more previous alignments into one Gaussian distribution and can compensate for wrong alignments because the certainty of a measurement

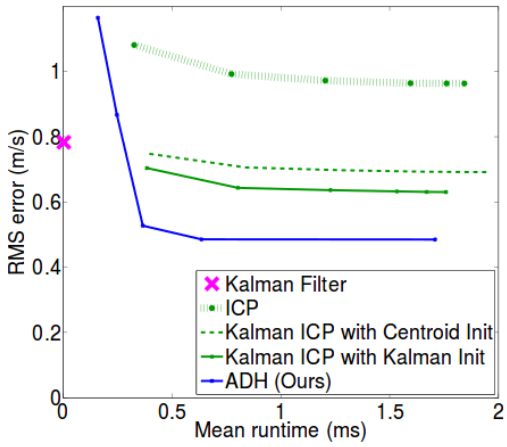


Figure 10: Comparison of the proposed method with annealed dynamic histograms with other baseline methods. Plotted is the RMS error for different runtimes [HLTS14].

can be considered as variance when feeding the measurement into the Kalman filter.

By combining the Kalman filter and ICP into one method, the drawbacks of both methods can be removed. ICP can be used as measurement model to align the point clouds in successive frames what removes the information loss in the basic Kalman filter that only uses the centroid. The result can then be used in the motion model of another Kalman filter what increases the robustness against failures of ICP. As initialization either the alignment of the centroids is used or the result of the Kalman filter that uses the centroids as described before.

The performances of the ICP variants with a Kalman filter as motion model is shown in Figure 10. The variant with centroid initialization performs 11.7% better than the simple Kalman filter and thus much better than the basic ICP algorithm without a motion model. If the initial alignment is computed by a first Kalman filter, we achieve the best performance of all baseline variants with an RMS error of 0.63 $\mathrm{m/s}$. The method presented in this report achieves with 0.43 $\mathrm{m/s}$ a lower RMS error than all baseline methods. It is more robust than ICP because it does not depend on a good initialization what is due to the global search of the annealed dynamic histogram. At the same time, the method has no drawback in the computational time. At each runtime, the result of the tracker outperforms the accuracy of ICP with the same runtime. This is due to the dynamic resolution improvement which also allows getting a result much earlier before ICP has finished its first iteration. However, this early results have a much higher RMS error. The only baseline method that has a better accuracy at a specific runtime is the pure Kalman filter which is outperformed at a larger runtime. As described before, the RMS error depends on the distance to the tracked object. For near objects with a distance smaller than 5m, the RMS error is about 0.15 $\mathrm{m/s}$. For objects with a distance greater than 70m, the method yields an RMS error of 1.8 $\mathrm{m/s}$. When comparing the method with and without usage of the color in the measurement model, the RMS error is decreased by 10.4% although the test dataset has problems with lens flare and bad lightening. The bad lightening mainly was caused by heavy shadows because the sensor data was recorded around sundown. In Figure 10 only the version with color usage is plotted. It can be expected that the advantage from using color in the measurement model is even larger under better conditions. Also the improvement in the accuracy from aligning the smaller point cloud to the larger one instead of aligning the current point cloud in the previous frame was evaluated. The version that aligns the smaller set to the larger one achieves a 10% lower RMS error than the other version. This emphasizes the importance of using this improvement. The version of the method that uses all these improvements performs with an RMS error of 0.43 $\mathrm{m/s}$ better than all other combination of used improvements.

Because the annealed dynamic histogram is a new approach, it is also evaluated here. Figure 11 shows the runtime and accuracy of the annealed dynamic histogram compared to an ordinary histogram with a densely sampled state space. The densely sampled histogram is about 138 times slower than the annealed

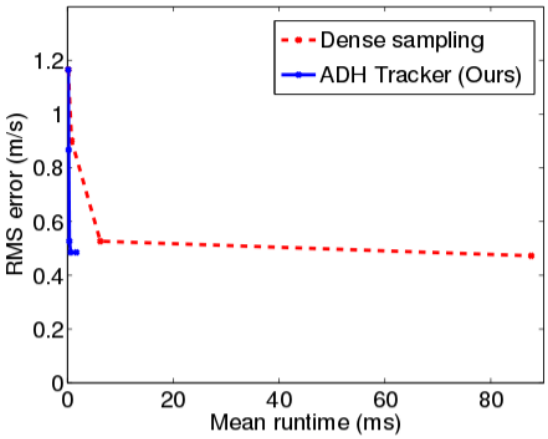


Figure 11: Comparison of runtime and accuracy for the annealed dynamic histogram and a densely sampled histogram [HLTS14].

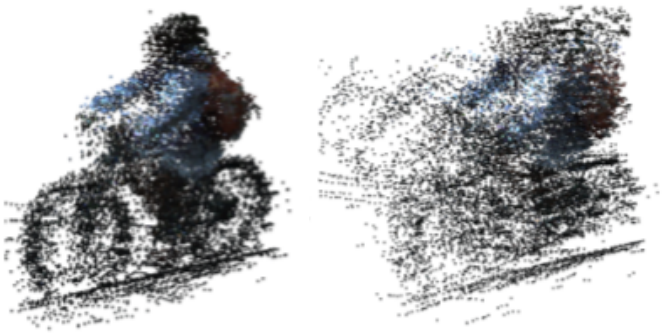
dynamic histogram for the approximately same accuracy. Furthermore, also the different strategies for cell expansion are evaluated. On the one hand, only the cell with the highest probability can be expanded in each iteration and on the other hand all cells that exceed the probability threshold p_{min} can be expanded in one step. Expanding only the cell with the highest probability leads to an RMS error that is 46.3% higher than using the other strategy. Thus, it is clear that all cells should be expanded if they exceed the threshold. The authors of [HLTS14] provide the source code and test datasets for the method. We ran the program with the test data to check if the results are reproducible. The RMS errors match the results in Figure 10. The runtimes were a bit larger because the program ran on a slower computer. The relation of runtimes between the different approaches matches the given relations. However, the runtime of the proposed method with using color is about 18 times larger than the runtime without using color. In the provided program, it is mentioned that this is as expected. In the evaluation of [HLTS14], this is not clearly stated or explained. Only the improvement in the RMS error is explained. However, this is not a problem because only the slower version with color consideration is compared in Figure 10.

4.2 Model Crispness

The previous approach to generate test data by tracking static objects in a local reference frame already provided many insights into the performance of the tracker. However, in this approach only detected cars were considered. To evaluate how the tracker performs on the variety of objects present in real traffic, another approach is necessary. With the tracking alignment which is computed frame for frame, the point clouds can be joint to build a model of the tracked object. Figure 12 shows how single point clouds of the objects and the final result that contains all point clouds look like. These models are build from the recorded sensor data of normal trips of the car through real traffic. Therefore, it allows measuring the performance in the real world. Two test datasets were used that were recorded around noon and with about a half year in between. Thus, in addition to the approach from the previous section, different seasons and times of day were evaluated. The two test sets have sensor data of a total amount of 135 pedestrians, 79 bicycles, and 63 moving cars. Therefore, the data contains a variety of different objects.
To measure the accuracy of the tracker, a metric called crispness score is introduced [SHN11]. This metric measures how accurate the model was build by analyzing how noisy the model is. Good tracking yields better frame to frame alignments and thus more clear models. Figure 13 shows the difference of a model build with the tracker presented in this report and the less precise ICP tracker improved with the motion model of a Kalman filter. The crispness score can be computed by

$$\frac{1}{T^2}\sum_{i=1}^{T}\sum_{j=1}^{T}\frac{1}{n_i}\sum_{k=1}^{n_i}G(x_k-\hat{x}_k,2\Sigma) \tag{22}$$

Figure 12: Models of tracked objects build by joining point clouds according to the computed frame to frame alignment. At the top single frames of the objects are shown [HLTS14].



(a) Tracker presented in this report (b) Kalman ICP tracker

Figure 13: Comparison of the model build with differently precise trackers [HLTS14].

Tracking Method	Object Class		
	People	Bikes	Moving Cars
Kalman Filter	0.38	0.31	0.27
Kalman ICP	0.18	0.18	0.29
ADH (Ours)	**0.42**	**0.38**	**0.33**

Table 1: Comparison of the crispness score for different trackers divided by the tracked object class [HLTS14].

where T is the number of frames the object was observed, n_i is the amount of points in frame i and $\hat{x}_k$ is the point in frame j which is nearest to the point x_k in frame i, and G is a multi-variate Gaussian for which Σ encodes the penalty for matches of different distances [SHN11, HLTS14]. The crispness score is in the interval $[0, 1]$ and a higher score means that the build model is more accurate.
Table 1 shows the crispness scores for the annealed dynamic histogram tracker, a simple Kalman filter, and ICP with a Kalman filter as motion model. The scores are divided by the tracked object classes to show how the object class influences the accuracy. For the data in the table, only frames with at least 200 points are used. The simple Kalman filter performs reasonable well through all object classes. The Kalman ICP method performs well on cars but significantly worse on pedestrians and bicycles. This is caused by the more flexible 3D shape of pedestrians and bicycles. A car has a static shape which makes the alignment easy for ICP. Pedestrians move their legs and arms and therefore have a changing shape over different frames. Also bicyclists are hard to align with ICP because their shape has many not well defined faces that cause ICP to get stuck in a local optimum. The tracking method with annealed dynamic histograms performs best in all three object classes. Intuitively, it could be expected that the crispness score for cars would have to be better than for pedestrians and bicycles because of the static shape. However, this is not the case. The authors of the paper do not mention why. A possible explanation would be that the tracked cars have a higher velocity relative to the tracking car and thus the estimation is less accurate. The evaluation with the crispness score shows that the proposed tracking method performs better than all baseline methods across all object classes and with real-world data.

5 Conclusion

In this report, the tracking method to estimate the position and velocity presented in [HLTS14] was described. Robust tracking is necessary in many robotic applications, especially for autonomous cars. The proposed method uses clues from the the 3D shape, color, and motion of the tracked object. A measurement model derived from a Dynamic Bayesian Network is used to compute the probability for a position and velocity of the tracked object given the current and previous measurements consisting of a point cloud and color information for these points. To find the most likely position and velocity, the state space is globally searched with an annealed dynamic histogram. The annealed dynamic histogram starts with a coarse sampling resolution which is dynamically improved in the important areas. Because the coarse resolution introduces an error, the measurement model considers the resolution and thus is annealed as the resolution improves. The computed alignment is then fed into a Kalman filter that models the motion of the object.
The evaluation showed that the proposed tracking method outperforms all common baseline methods, namely a simple Kalman filter and different variants of ICP. The Kalman filter is faster than the proposed method but less accurate. With the same runtime, ICP performs significantly worse than the proposed method, even with the improved variants which add a motion model and different initializations. This is due to its search for a local optimum. The proposed method performs well on real-world data, different object classes, and even problematic color information. Evaluation data was created by tracking static objects with a moving car in a local reference and by building object models and measuring their crispness score.
The authors of [HLTS14] did not evaluate the performance of the tracker in other domains and with other

sensors. Especially, the performance of the tracker with point clouds created by stereo cameras would be interesting because stereo cameras are cheaper and more widespread than the laser sensors used here. The challenge with stereo data is the larger noise. The method could be configured to this change by customizing the introduced parameters, for example the sensor variance. It can be argued that the proposed method probably also performs better on stereo data than ICP because the main problem of ICP, getting stuck in local optima, is also a problem in stereo data. Maybe the difference between the proposed method and ICP is even larger because of the larger noise.
The implementation of the proposed method is Open Source and can be found on the project page of the presented work:

`http://stanford.edu/~davheld/anytime_tracking.html`

The implementation is easy to setup, configure, and include into other C++ projects.
The tracking accuracy provided by annealed dynamic histograms and the combination of 3D shape, color, and motion is an important step towards robust tracking and thus safe and autonomous cars.

References

[AA12] Asma Azim and Olivier Aycard. Detection, classification and tracking of moving objects in a 3D environment. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pages 802–807. IEEE, 2012.

[Ack13] Evan Ackerman. Amazon Promises Package Delivery By Drone: Is It for Real? `http://spectrum.ieee.org/automaton/robotics/aerial-robots/when-drone-delivery-makes-sense`, 2013.

[DRU08] Michael Darms, Paul Rybski, and Chris Urmson. Classification and Tracking of Dynamic Objects with Multiple Sensors for Autonomous Driving in Urban Environments. In *Intelligent Vehicles Symposium, 2008 IEEE*, pages 1197–1202. IEEE, 2008.

[ECM04] Vladimir Estivill-Castro and Blair McKenzie. Hierarchical Monte-Carlo Localization Balances Precision and Speed. In *Australasian Conference on Robotics and Automation*, 2004.

[Eur15] European Commission / Directorate General Energy and Transport. Road safety evolution in EU. `http://ec.europa.eu/transport/road_safety/pdf/observatory/historical_evol.pdf`, 2015.

[FHB12] Adam Feldman, Maria Hybinette, and Tucker Balch. The Multi-Iterative Closest Point Tracker: An Online Algorithm for Tracking Multiple Interacting Targets. *Journal of Field Robotics*, 29(2):258–276, 2012.

[Gui08] Erico Guizzo. Three Engineers, Hundreds of Robots, One Warehouse. *Spectrum, IEEE*, 45(7):26–34, 2008.

[HLT13] David Held, Jesse Levinson, and Sebastian Thrun. Precision Tracking with Sparse 3D and Dense Color 2D Data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1138–1145. IEEE, 2013.

[HLTS14] David Held, Jesse Levinson, Sebastian Thrun, and Silvio Savarese. Combining 3D Shape, Color, and Motion for Robust Anytime Tracking. In *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July 2014.

[HLTS15] David Held, Jesse Levinson, Sebastian Thrun, and Silvio Savarese. Anytime Tracking. `http://stanford.edu/~davheld/DavidHeld_files/RSS2014_Poster.pdf`, 2015.

[IRdSvdZ12] Luca Iocchi, Javier Ruiz-del Solar, and Tijn van der Zant. Domestic Service Robots in the Real World. *Journal of Intelligent & Robotic Systems*, 66(1):183–186, 2012.

[KH95] James J Kuch and Thomas S Huang. Vision Based Hand Modeling and Tracking for Virtual Teleconferencing and Telecollaboration. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 666–671. IEEE, 1995.

[KMPS12] Ralf Kaestner, Jérôme Maye, Yves Pilat, and Roland Siegwart. Generative object detection and tracking in 3D range data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3075–3081. IEEE, 2012.

[KR08] Bernardin Keni and Stiefelhagen Rainer. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *EURASIP Journal on Image and Video Processing*, 2008, 2008.

[Kul97] Solomon Kullback. *Information theory and statistics*. Courier Corporation, 1997.

[LAB+11] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pages 163–168. IEEE, 2011.

[LHT+08] John Leonard, Jonathan How, Seth Teller, Mitch Berger, Stefan Campbell, Gaston Fiore, Luke Fletcher, Emilio Frazzoli, Albert Huang, Sertac Karaman, et al. A perception-driven autonomous urban vehicle. *Journal of Field Robotics*, 25(10):727–774, 2008.

[Mar10] John Markoff. Google Cars Drive Themselves, in Traffic. *New York Times*, 9, 2010.

[ML12] Dennis Mitzel and Bastian Leibe. *Taking Mobile Multi-Object Tracking to the Next Level: People, Unknown Objects, and Carried Items*. Springer, 2012.

[MLvHW11] Michael Manz, Thorsten Luettel, Felix von Hundelshausen, and H-J Wuensche. Monocular Model-Based 3D Vehicle Tracking for Autonomous Vehicles in Unstructured Environment. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2465–2471. IEEE, 2011.

[MS13] Frank Moosmann and Christoph Stiller. Joint Self-Localization and Tracking of Generic Objects in 3D Range Data. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1146–1152. IEEE, 2013.

[MSM02] R Marcello, Domenico G Sorrenti, and Fabio M Marchese. A robot localization method based on Evidence Accumulation and Multi-Resolution. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 415–420. IEEE, 2002.

[Ols09] Edwin B Olson. Real-Time Correlative Scan Matching. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 4387–4393. IEEE, 2009.

[PT09] Anna Petrovskaya and Sebastian Thrun. Model based vehicle detection and tracking for autonomous urban driving. *Autonomous Robots*, 26(2-3):123–139, 2009.

[RAK+07] Sivakumar Rathinam, Pedro Almeida, ZuWhan Kim, Steven Jackson, Andrew Tinka, William Grossman, and Raja Sengupta. Autonomous Searching and Tracking of a River using an UAV. In *American Control Conference, 2007. ACC'07*, pages 359–364. IEEE, 2007.

[RC11] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1–4. IEEE, 2011.

[RN13] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Always learning. Pearson, 2013.

[SFD02] Daniel Streller, K Furstenberg, and Klaus Dietmayer. Vehicle and object models for robust tracking in traffic scenes using laser range images. In *Intelligent Transportation Systems, 2002. Proceedings. The IEEE 5th International Conference on*, pages 118–123. IEEE, 2002.

[SHN11] Mark Sheehan, Alastair Harrison, and Paul Newman. Self-calibration for a 3D laser. *The International Journal of Robotics Research*, page 0278364911429475, 2011.

[SN07] Xuefeng Song and Ram Nevatia. Robust vehicle blob tracking with split/merge handling. In *Multimodal Technologies for Perception of Humans*, pages 216–222. Springer, 2007.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT press, 2005.

[TFBD01] Sebastian Thrun, Dieter Fox, Wolfram Burgard, and Frank Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial intelligence*, 128(1):99–141, 2001.

[TLT11] Alex Teichman, Jesse Levinson, and Sebastian Thrun. Towards 3D Object Recognition via Classification of Arbitrary Object Tracks. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 4034–4041. IEEE, 2011.

[WH12] Nicolai Wojke and M Haselich. Moving Vehicle Detection and Tracking in Unstructured Environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 3082–3087. IEEE, 2012.

[WPN12] Dominic Zeng Wang, Ingmar Posner, and Paul Newman. What Could Move? Finding Cars, Pedestrians and Bicyclists in 3D Laser Data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 4038–4044. IEEE, 2012.