METRIC SCALE CALCULATION FOR VISUAL MAPPING ALGORITHMS

Alexander Hanel[1,*], Adrian Mitschke[1,2,3], Richard Boerner[1], Dominik Van Opdenbosch[3], Ludwig Hoegner[1], David Brodie[2], Uwe Stilla[1]

[1]Photogrammetry and Remote Sensing, Technical University of Munich, Munich, Germany
(alexander.hanel, adrian.mitschke, richard.boerner, ludwig.hoegner, stilla)@tum.de
[2]ESG Elektroniksystem- und Logistik-GmbH, Fuerstenfeldbruck, Germany - david.brodie@esg.de
[3]Chair of Media Technology, Technical University of Munich, Munich, Germany - dominik.van-opdenbosch@tum.de

Commission II, WG II/4

KEY WORDS: Visual SLAM, Point Cloud, Pose Estimation, Scale Calculation, Object Detection

ABSTRACT:

Visual SLAM algorithms allow localizing the camera by mapping its environment by a point cloud based on visual cues. To obtain the camera locations in a metric coordinate system, the metric scale of the point cloud has to be known. This contribution describes a method to calculate the metric scale for a point cloud of an indoor environment, like a parking garage, by fusing multiple individual scale values. The individual scale values are calculated from structures and objects with a-priori known metric extension, which can be identified in the unscaled point cloud. Extensions of building structures, like the driving lane or the room height, are derived from density peaks in the point distribution. The extension of objects, like traffic signs with a known metric size, are derived using projections of their detections in images onto the point cloud. The method is tested with synthetic image sequences of a drive with a front-looking mono camera through a virtual 3D model of a parking garage. It has been shown, that each individual scale value improves either the robustness of the fused scale value or reduces its error. The error of the fused scale is comparable to other recent works.

1. METHODS FOR CAMERA LOCALIZATION AND MAPPING

A vehicle can be guided along a route on public roads with knowledge about the map of the route and a continuous comparison of the planned and the actual position of the vehicle along the route (Grewal et al., 2007). Therefore, the car has to be equipped with an electronic system obtaining the actual position of the car. An electronic system available in a high number of cars offered on the market bases on a GPS receiver. In this case, the metric position of the car in the map can be computed from GPS measurements.

Guiding a vehicle in more complex environments, like in dense urban areas or in indoor environments, has to rely on another type of electronic system, as the GPS signal may not be available or provide only a low position accuracy (Cui and Ge, 2003). An electronic system becoming available in more and more cars offered on the market in recent years is based on a camera (Statista, 2013), which perceives the environment around the car. In this case, the position of the car has to be localized by matching the environment perception obtained by the camera with the map. A common way to represent the environment perceived by a camera are point clouds obtained by matching visual cues across multiple images (Ros et al., 2012).

Several groups of methods are available for environment mapping and localization of a mono camera. Multi-view stereo methods (Hartley and Zisserman, 2003) map the environment for example by a point cloud generated from a high number of images. As these methods rely on known interior and exterior orientations of the camera (Furukawa et al., 2015), are they designed to create the point cloud of the environment from the images and not to estimate the unknown orientations of additional images. Visual odometry methods (e.g. (Nister et al., 2004)) are designed to estimate the trajectory of a moving camera by matching point features between image pairs in an image sequence. The focus is put on a short processing time capable to handle continuously new images, with negative influence on the consistency of the trajectory and the map (Scaramuzza and Fraundorfer, 2011). Similarly, structure from motion methods obtain a map and the camera trajectory from an image series (e.g. (Fitzgibbon and Zisserman, 1998)), but include an optimization step for the map and the trajectory, often by a bundle adjustment. Neither the interior orientation, nor exterior orientations of the camera have to be known a priori. Visual simultaneous localization and mapping (V-SLAM) methods provide also a map and the camera trajectory (e.g. (Lemaire et al., 2007)) without the need of a priori knowledge. Instead of structure from motion, V-SLAM is capable to work in real-time, an important property for use in vehicles. Compared to the also real-time capable visual odometry, V-SLAM methods can achieve a higher precision and consistency of both map and trajectory by using additional constraints like loop closures (Scaramuzza and Fraundorfer, 2011).

Point clouds of the environment can be grouped depending on the available a priori information as follows (Heyden and Astrom, 1996) (Bebis et al., 2010): if the interior orientation of the camera is not known, a point cloud is projective. If the interior orientation is known, the point cloud can be called Euclidean with a scale ambiguity. If both the interior orientation and metric scale information are known, the point cloud can be called Euclidean with metric scale or Euclidean with absolute scale. In the last case, localization using the camera returns the metric position of the car in the map.

The metric scale can either be provided by the visual sensor perceiving the environment or by additional knowledge. In a calibrated multi-camera system, the metric scale is available by the known baseline between the cameras (e.g. (Nister et al., 2006)).

*Corresponding author

(Clipp et al., 2008) present an approach to obtain the metric scale for a multi-camera system with non-overlapping fields-of-view. For a mono camera, (Song et al., 2016) present an approach to obtain the scale if the height of the camera above the ground plane is known. In a vehicle driving on an uneven road, there might be rotations around the pitch axis, causing height changes of the camera relative to the ground. (Scaramuzza et al., 2009) present an approach to obtain the metric scale for a wheeled vehicle moving in a plane, if the distance of the camera from the rear axle of the vehicle is known. Their approach is independent from rotations around the pitch axis, but still requires known information about the camera mounting in the vehicle.

In this contribution, a method to calculate the metric scale for an Euclidean point cloud using a priori information about the environment recorded with a mono camera is proposed; the a priori information is taken from typical elements in the environment; installing special equipment, like a calibration pattern is not necessary. A priori information about the camera mounting is not necessary, as well. Typical elements, like dominant building structures shown in the reconstructed point cloud or objects shown in images of the environment are used to derive Euclidean distances in the point cloud. The a priori information provides the metric size of these distances, allowing to compute the metric scale and to obtain the Euclidean point cloud with metric scale.

The remainder of this paper is organized as follows: The method to calculate the scale is presented in section 2. Experiments with point clouds created from synthetic image sequences are described in section 3. The results of the scale calculation are shown and discussed in section 4. The paper ends with a conclusion in section 5.

2. METRIC SCALE CALCULATION FOR POINT CLOUDS

In order to calculate the metric scale of an Euclidean point cloud captured with a calibrated monocular camera, a priori knowledge about the metric size of dominant structures in the environment or the objects in it can be used. The key is to find distances d_{cloud} in the Euclidean point cloud corresponding to the dominant strctures or objects and to compare them against the corresponding metric distances d_{real}, which are known a priori. The scale factor s is then computed straight-forward by (equation 1)

$$s = \frac{d_{real}}{d_{cloud}} \tag{1}$$

where s = calculated scale
d_{real} = metric distance
d_{cloud} = Euclidean distance in the point cloud

The metric scale is obtained in a two-step process, which is being described in the following. In order to handle arising scale drift during environment mapping, the process should be applied repeatedly during environment mapping and is therefore based on local parts of the point cloud, not the complete one. Hereby, scale change addresses the gradual change of the scale of an interatively created point cloud over time. A local part is a subset of all 3D points of the point cloud, which is calculated from the images between two selected keyframes, e.g. determined by the ORB-SLAM2 (Mur-Artal and Tardós, 2017) algorithm the authors are using to create the point cloud. In the first step, the scale is calculated by two independent approaches. The first approach evaluates peaks in the density distribution of the 3D points in the point cloud along a specified axis, the second approach evaluates unique points on reference objects seen in images of the environment. Individual scale values from multiple appropriate distances in both approaches are averaged. In the second step, the independently calculated scale values from the density-based and the object-based approach are fused to a combined scale value.

2.1 Gravity Alignment

Before density-based analysis can be performed along a horizontal or vertical axis of a given point cloud, an important preprocessing step is to align the point cloud according to gravity, i.e. so that walls in the point cloud become parallel to the z-axis. Gravity alignment can become important, as all camera pose parameters estimated by SLAM can drift over time, leading potentially to a roll of the estimated poses relative to the point cloud for example, as previous experiments of the authors have shown. The method used for gravity alignment in this contribution was first described by (Al-Nuaimi et al., 2017). It is assumed that the point cloud is extended most along its horizontal plane and least extended along its normal axis, what is likely to be fulfilled in parking garages, which require a lot of space for parking lots in a horizontal plane, but not a large room height; besides, in parking garages, it can be assumed that the camera is moving in a horizontal plane. By computing the point position covariance matrix, which indicates the point scatter along the two horizontal and the vertical axis, the belonging eigenvalues $\lambda_{1...3}$ and eigenvectors $v_{1...3}$ are obtained. The normal vector of the horizontal plane of the point cloud then is represented by the eigenvector v_{min} belonging to the smallest eigenvalue λ_{min}. By finding the rotation that aligns this eigenvector to the z-axis of the gravity-aligned coordinate system, the transformation required for gravity alignment of the point cloud can be obtained and applied to the 3D points.

2.2 Density-Based Scale Calculation

For buildings like a parking garage, their building plan can be used as a priori knowledge about metric sizes in the captured environment: a top-down view plan provides a priori information about the building layout, while a side-view plan represents its height profile. Having the knowledge from the plan, Euclidean distances d_{cloud} for scale calculation can be obtained from point density distributions in the point cloud (Hilsenbeck et al., 2012a). As buildings are sometimes not built as planned, the building plan might contain errors. Therefore, scale values for new local parts of the point cloud are compared to previous scale values to recognize outliers. For a parking garage, previous experiments of the authors have shown that peaks in the point density distribution in the point cloud of the tested parking garages - including empty ones - correspond to:

- **Driving lane width:** The driving lane is considered to be an empty space with side objects like pillars, walls or cars at its left and right side. The point cloud therefore is expected to have point density peaks at the left and right edge between the driving lane and the side objects.
- **Floor-to-ceiling distance:** The room of a parking garage is considered to be the space between the floor plane and the ceiling plane. The point cloud therefore is expected to have point density peaks at the floor and the ceiling.

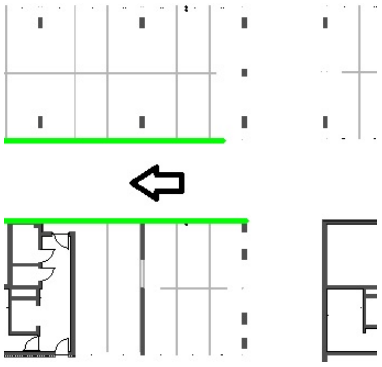

Figure 1. Obtaining the metric driving lane width: Closest parallel line pair (green) to the camera (black arrow) is found in a top-down view building plan of a parking garage by Hough transform.

In order to extract the corresponding metric distances from an image of the top-down view or the side-view building plan, a global line detection using a Hough transform (Duda and Hart, 1972) is applied. The image coordinate system hereby represents the metric size of the building. Depending on the current camera position and orientation in the plan, the two closest lines, which are parallel to each other and to the orientation of the camera, are selected (see figure 1 for the top-down view plan). The distance between the selected lines is calculated and used as metric distance d_{real} for scale calculation. The camera position and orientation have to be known, e.g. from V-SLAM using the metric-scaled point cloud; the coordinate systems of point cloud and plan are aligned based on the known position and orientation of the camera at the entry lane of the parking garage.

Knowing the metric distances d_{real} of the driving lane width and the room height, the corresponding Euclidean distances d_{cloud} in the point cloud have to be obtained. As already mentioned, both distances in the point cloud can be obtained from point density peaks, assuming that the desired peaks are the most characteristic ones in the density distribution. Previous experiments have led to this statement. Depending on the environment, point density peaks corersponding to other building structures, like the complete room breadth, can be used. After gravity alignment as described in section 2.1, the lane width can be obtained from the difference of the peak positions in the point distribution along the horizontal axis perpendicular to the driving lane. The floor-to-ceiling distance can be obtained from the difference of the peak positions along the z-axis.

To find the left and right lane edge in the given point cloud, an approach based on the work of (Hilsenbeck et al., 2012b) is used (figure 2). An anchor point that is expected to be located between the lane edges, represented by point density peaks, has to be calculated first; only its horizontal position is needed, it is calculated by the mean position of all 3D points in the current local part of the point cloud. By restricting the points to the local part, effects from scale drift, occurring especially during mapping of a larger environment with a mono camera (Ros et al., 2012), can be avoided. Starting from this anchor point, an initial horizontal lane orientation vector $\underline{l}_1$ is provided by the first principal component vector v_1, which has been obtained by previous gravity alignment, corresponding to the largest eigenvalue of the covariance matrix of 3D points of the point cloud. This initialization is assumed to be valid, as long as the point cloud has its largest extension in the direction of the driving lane. In an angle interval

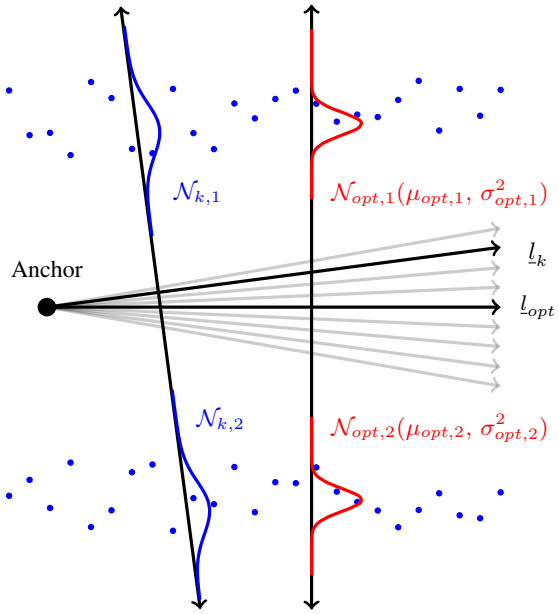

Figure 2. Lane width is obtained by fitting a bimodal Gaussian model (blue, red) to multiple horizontal axes (black lines with arrows on both ends) perpendicular to different lane orientation vectors $\underline{l}_k$. Final lane width is calculated by the difference of the mean values $\mu_{opt,2}$ and $\mu_{opt,1}$ of the bimodal Gaussian model $\mathcal{N}_{opt,1/2}$ with minimal average variance.

$[-\alpha; +\alpha]$ around the initial lane orientation vector, further horizontal lane orientation vectors are sampled equidistantly. The horizontal distances of the local 3D points from each lane orientation vector are calculated. A bimodal Gaussian model, as shown in figure 2, is fitted to the distance distribution belonging to each lane orientation vector $\underline{l}_k$; as seen in direction of the driving lane, one Gaussian bell $\mathcal{N}_{k,1}(\mu_{k,1},\ \sigma^2_{k,1})$ is on the left side of the anchor point, the other $\mathcal{N}_{k,2}$ is on the right side. The lane orientation vector $\underline{l}_{opt}$ with the minimal average variance of $\mathcal{N}_{opt,1}$ and $\mathcal{N}_{opt,2}$ is then considered as best fit to the edges. The difference of $\mu_{opt,1}$ and $\mu_{opt,2}$ then denotes the Euclidean driving lane width $d_{lane,cloud}$ in the point cloud.

The idea to determine the floor-to-ceiling distance in the given point cloud is similar. Instead of using the lane orientation, the density of the z-values of the 3D points in the gravity-aligned point cloud is evaluated directly in a histogram as proposed by (Al-Nuaimi et al., 2017). These authors assume that the 3D points belonging to the floor and ceiling cause density maxima along the z-axis. But as floor and ceiling are often poorly textured surfaces, it has to be expected that the corresponding density peaks are remarkably smaller than the global peak; own experiments with different image sequences of parking garages have proven this expectation, with the majority of the 3D points being on the surface of parked cars. Therefore, the peak detection via a bimodal Gaussian model is not seen as suitable here. Instead, the histogram of the z-values of the 3D points is created with a number of histogram bins appropriate to make local maxima visible. The outmost points based on their z-values are considered as outliers and removed in advance. The first and last local peak in the histogram are assumed to represent the floor and the ceiling, respectively. These local peaks are detected by iterating from the first and last bin to the central bin and checking if the number of points contributing to a bin is higher than the number contributing to its adjacent bins. The distance between the last and first

local peak then denotes the Euclidean floor-to-ceiling distance $d_{height,cloud}$.

2.3 Object-Based Scale Calculation

The second approach (pipeline see figure 3) calculates the scale s based on the known metric size of traffic signs shown in an image of the environment. The approach is described in the following for circular traffic signs, but can be transferred to other objects in general, too.

The diameter of a circular traffic sign can be used as known metric size d_{real}. The metric size can be obtained for example from governmental regulations (e.g. (Department of Transport - Ireland, 2010)). The size of the diameter in the point cloud coordinate system d_{cloud} is obtained by the following steps: A traffic sign shown in an image of the environment is detected and the ellipsoidal shape of its contour is extracted by fitting a geometric primitive to the image part containing the detection. The image coordinates of the ellipse vertices of the major axis are calculated, giving the diameter in the image coordinate system (Elder, 2017). The image rays of the ellipse vertices are intersected with a plane fitted to the 3D points in the point cloud belonging to the traffic sign and thereof the 3D coordinates of the ellipse vertices are obtained in the point cloud coordinate system. The distance between the 3D coordinates gives finally the Euclidean diameter d_{cloud} in the point cloud.

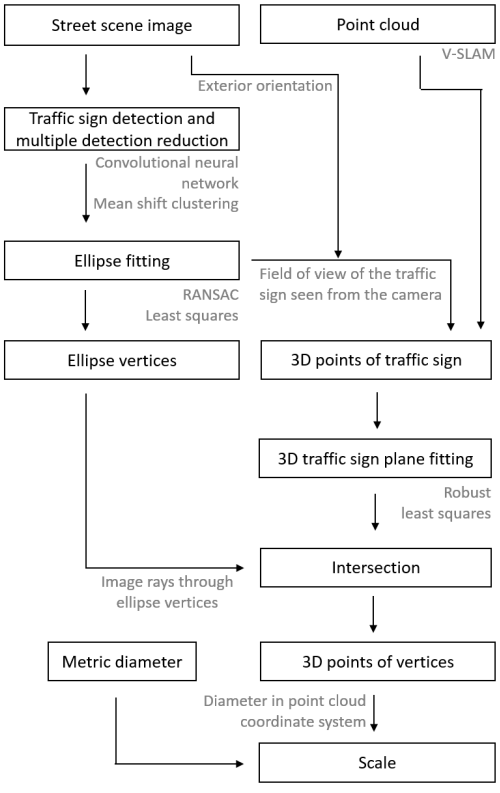

Figure 3. Pipeline to calculate the scale with the object-based approach.

The detector uses a sliding window approach to detect one or more traffic signs in an image pyramid. Step by step, regions of interest with different sizes and positions in the image are evaluated by a deep learning traffic sign detector described by (Hanel and Stilla, 2018). Positive detections of a sliding window approach are typically multiple regions with slightly different positions and sizes for each traffic sign shown in an image. To avoid errors in the later shape fitting step, mean shift clustering (Fukunaga and Hostetler, 1975) is used to reduce the multiple regions to a single region of interest for each traffic sign. Thereby without the need to determine the number of traffic signs manually, all positive detected regions are grouped into several clusters depending on their distance to a cluster center. Each cluster center defines a single region, the size of the region is defined by the mean size of all regions contributing to this cluster.

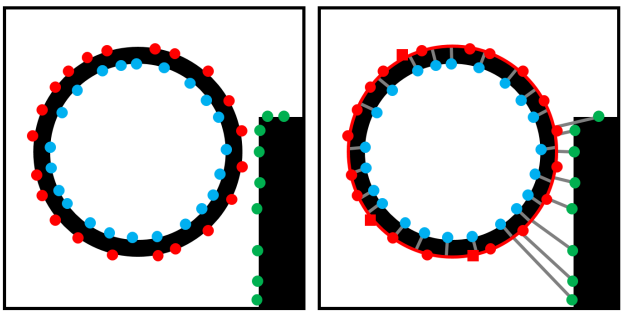

Figure 4. Intermediate steps for traffic sign shape extraction. Left: Binary image (idealized visualization) with contour points (red, green, blue balls) along the edges of different image parts (traffic sign shape as black circle, background object as black rectangle). Right: Sample contour points (red squares) are used to fit an ellipse (red circle). RANSAC determines iteratively the consensus set of contour points along the edge of the traffic sign shape using the distance (gray lines) between contour points and fitted ellipses.

The shape of the traffic sign is extracted by fitting an ellipse to a region of interest, considering that a real-world circle is projected to an ellipse in an image in general. The extraction obtains the ellipse parameters position, size and orientation in the image. A binary image is created out of the region of interest using an absolute global threshold (exploiting known color of traffic signs) to separate the shape of the sign contour from other objects in the image. Contour points are extracted along the edges of different image parts (figure 4) with an unique intensity in the binary image (algorithm from (Suzuki and Abe, 1985)). The RANSAC algorithm (Fischler and Bolles, 1981) selects iteratively the consensus set of contour points belonging to the edge of the traffic sign. Therefore, ellipse parameters are calculated with sampled contour points in each iteration and the set of all contour points within a maximal distance to the ellipse is determined. The largest set is used as consensus set. The number of iterations is chosen to have at least one set of sample points without outliers with a probability of 99 %. The final ellipse parameters are estimated in a least-squares-adjustment using the consensus set.

To calculate the scale, the 3D coordinates of the ellipse vertices have to be obtained in the point cloud coordinate system to establish a link between the point cloud and the known metric size. As the orientation of the major axis of an ellipse in an image is perpendicular to the line of sight of the camera (Elder, 2017), the image coordinates of the vertices in two images do not correspond to the same object coordinates. Therefore, the 3D coordinates of the vertices can’t be obtained by triangulation from two or more images. (Soheilian and Brédif, 2014) provide a method to recover the 3D circle parameters from multi-view using multiple optimizations, which is considered as computationally more expensive than our approach, which is described in the following paragraph.

Instead, the 3D coordinates are obtained by intersecting their im-

age rays with a plane fitted to the 3D points belonging to the traffic sign. Therefore, a cone starting in the projection center of the image and going through the corners of the region of interest is defined. The cone is transformed from the camera coordinate system to the point cloud coordinate system. By intersecting the cone with the point cloud, a set of 3D points is selected. For the following, it is assumed that in an urban environment, 3D points belong to planes. Further, that the traffic sign belongs to the closest plane as seen from the camera. To group the 3D points into planes with different distances from the camera, a histogram is used. It represents the horizontal distances of the selected 3D points from the projection center. The distance is supposed to be a valid separator for the planes: first, traffic signs and so their planes are approximately perpendicular to the optical axis of a front looking vehicle camera. Second, the field of view of the cone is typically small, as the traffic sign covers only a part of the image, leading to similar distances even for planes diagonal to the camera. A ”peakiness test” is applied to get characteristic local maxima in the histogram. The 3D points with a distance within a threshold around the first maximum are considered to be belonging to the plane of the traffic sign. The 3D position and orientation of the plane is fitted by a robust least squares adjustment using the 3D points within the threshold. The 3D coordinates of the ellipse vertices are finally calculated by intersecting their image rays given in the point cloud coordinate system with this plane and thereof their distance d_{cloud} is obtained.

2.4 Fusion to a Combined Metric Scale

The approaches described in sections 2.2 and 2.3 provide three independent scale values. There might be situations, where the scale value of an individual approach lacks accuracy, for example when the environment captured in the images changes quickly, like in curve drives. To increase the robustness of the scale calculation, a combined metric scale s_{fused} is obtained by a weighted mean of the independent scale values valid for the same local part of a point cloud. Additionally, a multiplicative factor f_{method} is added to the weight for each approach to penalize strong deviations of the scale calculated for the current local part of the point cloud from the scale calculated for former parts.

$$f_{method} = 1 - \frac{|s_{fused,t-1} - s_{method}|}{s_{fused,t-1}} \quad (2)$$

where f_{method} = multiplicative factor for w_{method}
$s_{fused,t-1}$ = former computed scale value
s_{method} = current computed scale value

The weight of the scale of each individual approach is obtained as follows:

- **Driving lane width:** The weight w_{lane} for the scale s_{lane} is calculated from the average of the variances $\sigma^2_{opt,i}$ of the best-fit bimodal Gaussian model $\mathcal{N}_{opt,1/2}$ (equation 3). Furthermore, the multiplicative factor f_{lane} (see equation 2) is applied in order to penalize strong deviations from previous scale computations.

$$w_{lane} = f_{lane} \cdot \tanh\left(\frac{2}{\sigma^2_{opt,1} + \sigma^2_{opt,2}}\right) \quad (3)$$

where w_{lane} = weight for s_{lane}
$\sigma^2_{opt,1/2}$ = variances of bimodal Gaussian model
f_{lane} = multiplicative factor (see equation 2)

- **Floor-to-ceiling distance:** The weight w_{height} for the scale s_{height} is calculated from the average of the local variances $\sigma_{local,1/2}$ of the peaks for floor and ceiling (equation 4). Again, the factor f_{height}, computed as shown in equation 2, is applied to penalize strong deviations from the last computation.

$$w_{height} = f_{height} \cdot \tanh\left(\frac{2}{\sigma^2_{local,1} + \sigma^2_{local,2}}\right) \quad (4)$$

where w_{height} = weight for s_{height}
$\sigma^2_{local,1/2}$ = local variances of outmost histogram peaks
f_{height} = multiplicative factor (see equation 2)

The value of the three neighboring histogram bins are used for each local peak to calculate the variance; the lower the local variances are, the more robust the local peaks are assumed to be.

- **Object-based scaling:** The weight w_{object} for the scale s_{object} is calculated from the variance σ^2_{image} of the set of scale values obtained from the traffic signs detected in one image (equation 5):

$$w_{object} = f_{object} \cdot \tanh\left(\frac{1}{\sigma^2_{image}}\right) \quad (5)$$

where w_{object} = weight for s_{object}
σ^2_{image} = variance of the scales from one image
f_{object} = multiplicative factor (see equation 2)

The combined metric scale is calculated as follows (equation 6):

$$s_{fused} = \frac{w_{lane} \cdot s_{lane} + w_{height} \cdot s_{height} + w_{obj} \cdot s_{obj}}{w_{lane} + w_{height} + w_{obj}} \quad (6)$$

where s_{fused} = fused scale
$w_{lane}, w_{height}, w_{obj}$ = scale weights
$s_{lane}, s_{height}, s_{obj}$ = individual scale values

3. DATASETS AND EXPERIMENTS

As test scenario, a synthetic 3D model of a parking garage is used. The parking garage consists of a driving lane and parking lots perpendicular to the lane. The model is created by means of 3D computer graphics with circular traffic signs along the driving lane. Template for the virtual parking garage model is the building plan of a real existing parking garage. The density of traffic signs in the virtual model corresponds to the density in the afore-mentioned real existing garage.

Camera	Pinhole camera model
Geometric resolution	640 x 480 px
Radiometric resolution	RGBA, 8 bit
Lens	35 mm, no distortions
Settings	24 fps

Table 1. Technical data of the synthetic camera used to render several image series along different camera drives through the 3D model of a parking garage.

A sequence of RGB images (example in figure 5) is rendered for each drive with a front-looking camera (camera specifications see table 1) through the model: drive 1 along a straight part of the driving lane; drive 2 along a part of the driving lane with two curves and a final parking maneuver heading into a parking lot. Contrary to a real-world dataset, the ground truth size of the scenario and its objects as well as the ground truth trajectories of the camera drives are available for a synthetic dataset. Thereby comparison of the scale values calculated with the proposed method to the actual scales values becomes possible.

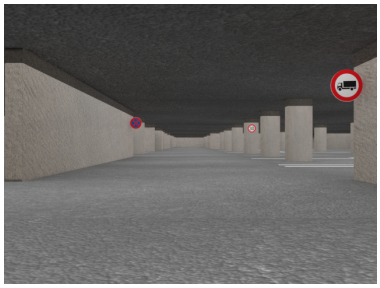

Figure 5. Rendered RGB image during a camera drive through the modeled parking garage. The known metric lane width, the room height and diameter of the traffic signs are used to calculate the metric scale.

As application to calculate the metric scale, the ORB-SLAM2 algorithm (Mur-Artal and Tardós, 2017) is applied to each image sequence k, returning each time an estimate for the point cloud of the environment and the camera trajectory. The ground truth metric scale is obtained by comparing and aligning the estimated ORB-SLAM2 trajectory with the ground truth trajectory of the scenario and by scaling it to minimize the translational error. The ground truth metric scale is assumed to be constant for the whole point cloud, as the ORB-SLAM2 algorithm has not shown to be prone to scale drift in the tested sequences. Using the proposed method, the metric scale is calculated with the local part of the estimated point cloud provided for every 10th keyframe during the ORB-SLAM2 processing; an image of the synthetic sequence is considered to be a keyframe, if the camera moves into a part of the environment with a low overlap to already mapped parts. The number ’10’ is chosen as trade-off between a desired high amount of 3d points in the point cloud and a desired low computational effort. The described steps to estimate the point cloud and to calculate the metric scale are repeated for each image sequence several times for statistical reasons; indeterministic in the algorithm is the initialization of the camera pose at the beginning and the RANSAC used for traffic sign shape fitting.

The individual approaches and the fusion approach are evaluated and discussed with regard to the following aspects:

- **Scale precision:** The scale precision is analyzed by a relative scale error $e_{scale,k,i}$ comparing the scale value $s_{k,i}$ calculated for each repetition i to its respective ground truth value (equation 7):

$$e_{scale,k,i} = \frac{|s_{k,i} - s_{gt,k}|}{s_{gt,k}} \tag{7}$$

where $e_{scale,i}$ = relative scale error
$s_{k,i}$ = calculated scale for repetition k, i
$s_{gt,k}$ = ground truth scale for sequence k

- **Trajectory precision:** The trajectory is analyzed using the root-mean-square error defined in (Sturm et al., 2012) (equations 8, 9). This error measure allows analyses in absolute metric values as well as in relative values with respect to the trajectory length (Geiger et al., 2012):

$$E_{traj} = \left(\frac{1}{n} \sum_{t=1}^{n} ||\underline{x}_{est,t} - \underline{x}_{gt,t}||^2 \right)^{1/2} \tag{8}$$

$$e_{traj} = \frac{E_{traj}}{l} \tag{9}$$

where E_{traj} = absolute metric trajectory error
e_{traj} = relative trajectory error
n = number of estimated positions
$\underline{x}_{est,t}$ = estimated camera position at time t
$\underline{x}_{gt,t}$ = ground truth camera position at time t
l = trajectory path length

4. DISCUSSION OF THE RESULTS FOR METRIC SCALE CALCULATION

4.1 Traffic Sign Detection and Shape Fitting

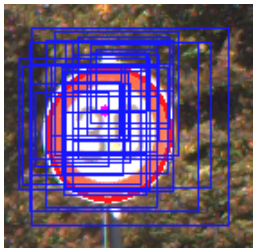

Figure 6. Multiple detections (blue rectangles) are typical for the traffic sign detector used for this paper.

The traffic sign detector used for object-based scale calculation yields typically multiple detections for a traffic sign in an image (real-world image example see figure 6). The detector shows a high classification certainty on both classes (see confusion matrix in table 2). Detections with a low classification certainty are removed, they might be false positives, which could lead to a wrong scale value. The average RMS for fitting an ellipse to a detected region of interest is 1 px. Regions above a RMS threshold are removed as well.

4.2 Scale Error of the Individual Approaches

The difference (see table 3) between the mean and median of the relative scale error e_{scale} of scale values s_{lane}, which are calculated for different local parts of the point cloud from the lane

		True class	
		TS	No TS
Predicted class	TS	96 %	4 %
	No TS	4 %	96 %

Table 2. Confusion matrix for the convolutional neural network for traffic sign (TS) detection.

width, is remarkable. The low median error, compared to using other structures or objects in the environment, indicates that using the lane width can provide the most precise scale values, while the higher mean error indicates that the scale values calculated for some local parts might be outliers. Analysis of the image sequence at the time points of outliers shows that they occur especially when the camera is moved around curves along the drive way or when simulating a parking maneuver at one of the parking lots. Table 3 further shows a larger mean and median error for scales calculated from the room height, which can be interpreted that this building structure is less suitable for scale calculation than the driving lane width. This observation is surprising, as the virtual parking garages don’t contain cars, on which 3D points could be lying, which then could blur the density peaks in the point cloud in the vertical direction.

The mean and median relative error for scale values from the object-based approach are higher than for the afore-mentioned density-based approaches exploiting building structures, as table 3 shows as well. Analysis of the image sequences shows that the calculcated scale values have largest variations when the camera is moved around a curve with only a short viewing range until the next wall in front and a low number of traffic signs in these images therefore. The object-based approach can therefore rather be seen as complementary extension, which is independent from dynamic objects in the parking garage, compared to being a standard approach for scale calculation. It can become important for scale calculation, when the driving lane borders or the floor and ceiling are occluded in the images by dynamic objects, especially cars, in the garage.

Method	Mean (%)	Median (%)
Lane Width Det.	24.54	8.93
Ceiling Height Det.	24.72	19.27
Object Det.	32.79	30.24
Fused Lane/Height	14.69	9.83
Fused all	14.45	11.94

Table 3. Mean and median of the relative scale error e_{scale} for both synthetic sequences

4.3 Scale Error of the Combined Approach

Fusion of the three individual approaches using equation 6 results in both reduced mean error and median error (table 3) of the combined scale value compared to the individual scale values. The errors of the combined scale value are in the same range as described by other authors; for example, the method of (Kaminsky et al., 2009) achieves a mean error of 25%, while the more recent method of (Ni et al., 2013) achieves 12%. In addition, the second cited method relies on the assumption that the complete point cloud is already known. Our approach instead uses only on local parts of the point cloud and can therefore be used ”online” during V-SLAM mapping. Table 3 further shows, that the median error for the fusion of all three approaches is worse than for a fusion of the two density-based approaches only.

4.4 Trajectory Error of Density-based Approaches

A small scale error in the camera trajectory is crucial for the density-based approaches, because they rely on extracting the lane width or ceiling height at the correct position in the metric building plan, i.e. the camera position has to be given with a metric unit. Therefore, the camera trajectory obtained by the V-SLAM algorithm with an Euclidean unit is scaled to obtain a metric unit. Using synthetic data has the benefit that the ground truth trajectory of the camera is known: The absolute trajectory error E_{traj} for drive 1 averaged over i repetitions is 2.72 m with a standard deviation of 0.57 m; the ground truth length of the trajectory is 63 m and the relative trajectory error e_{traj} therefore is 4.3 %. Compared to the work of (Gräter et al., 2015), our approach achieves a similar trajectory error. Considering the desired use case of a parking garage, the trajectory error is sufficient to localize the car on a driving lane. To perform a parking maneuver, the trajectory error has to be seen as too high, as a common parking lot has a width of only around 2.5 m.

5. CONCLUSION

In this paper, a method to calculate the metric scale for a point cloud of an indoor environment, like a parking garage, generated by a visual mapping algorithm, has been proposed. The length of dominant environment structures in the point cloud, like the width of the driving lane or the room height of the parking garage are derived from peaks in the point densities in the point cloud. Alternatively, the size of traffic signs in the point cloud is derived by projecting traffic signs detected in images onto the point cloud. A priori known knowledge about the lane width, room height or the traffic sign size provides the metric lengths for the lengths derived from the point cloud, which are then set in ratio to calculate individual scale values. The combined scale is obtained by a weighted fusion of the individual scale values. It has been shown for synthetic image sequences of camera drives through a 3D model of a parking garage that the calculated metric scale is either more robust or more precise for each of the three individual approaches. It has further been shown, that the combined scale can improve precision and robustness compared to the individual approaches.

An obvious future research step is to apply the proposed method to an acquired image series to test the quality of the scale calculation under real conditions. To further extend the scale calculation for automotive applications, the metric distance between the two wheel centers on one side of a known car model can be used as additional scale information. Statistical tests could be used to evaluate hypotheses for different car models in the case of an uncertain model recognition from images. As maps created with monocular mapping algorithms show especially errors in the moving direction of the camera, the 3D position of traffic signs could be used to compensate these errors longitudinal to the driving lane. To make the method better applicable to other indoor environments than parking garages, the use of common standardized objects like doors or windows could be investigated.

REFERENCES

Al-Nuaimi, A., Hilsenbeck, S., Garcea, A. and Steinbach, E., 2017. 6dof decoupled roto-translation alignment of large-scale indoor point clouds. *Computer Vision and Image Understanding* 157, pp. 72–89.

Bebis, G., Boyle, R., Parvin, B., Koracin, D., Chung, R., Hussain, M., Tan, K., Crawfis, R., Thalmann, D. et al., 2010. *Advances in Visual Computing: 6th International Symposium, ISVC 2010, Las Vegas, NV, USA, November 29-December 1, 2010, Proceedings*. Advances in Visual Computing: 6th International Symposium : Proceedings, Springer.

Clipp, B., Frahm, J.-M., Pollefeys, M., Kim, J.-H. and Hartley, R., 2008. Robust 6dof motion estimation for non-overlapping multi-camera systems. In: *IEEE Workshop on Applications of Computer Vision (WACV 2008)*, IEEE Workshop on Applications of Computer Vision, IEEE, pp. 1 – 8. IEEE Workshop on Applications of Computer Vision (WACV 2008); .

Cui, Y. and Ge, S. S., 2003. Autonomous vehicle positioning with gps in urban canyon environments. *IEEE Transactions on Robotics and Automation* 19(1), pp. 15–25.

Department of Transport - Ireland, 2010. Traffic Signs Manual.

Duda, R. O. and Hart, P. E., 1972. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM* 15(1), pp. 11–15.

Elder, J., 2017. Determining rotations between disc axis and line of sight. Website. http://web.ncf.ca/aa456/scale/ellipse.html. Accessed 2017-10-29.

Fischler, M. A. and Bolles, R. C., 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24(6), pp. 381–395.

Fitzgibbon, A. W. and Zisserman, A., 1998. Automatic camera recovery for closed or open image sequences. In: *European Conference on Computer Vision*, Springer-Verlag, pp. 311–326.

Fukunaga, K. and Hostetler, L., 1975. The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Transactions on Information Theory* 21(1), pp. 32–40.

Furukawa, Y., Hernández, C. et al., 2015. Multi-view stereo: A tutorial. *Foundations and Trends® in Computer Graphics and Vision* 9(1-2), pp. 1–148.

Geiger, A., Lenz, P. and Urtasun, R., 2012. Are we ready for autonomous driving? the kitti vision benchmark suite. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gräter, J., Schwarze, T. and Lauer, M., 2015. Robust scale estimation for monocular visual odometry using structure from motion and vanishing points. In: *2015 IEEE Intelligent Vehicles Symposium (IV)*, IEEE.

Grewal, M., Weill, L. and Andrews, A., 2007. *Global Positioning Systems, Inertial Navigation, and Integration*. Wiley.

Hanel, A. and Stilla, U., 2018. Iterative calibration of a vehicle camera using traffic signs detected by a convolutional neural network. In: *Proceedings of the 4th International Conference on Vehicle Technology and Intelligent Transport Systems*.

Hartley, R. and Zisserman, A., 2003. *Multiple View Geometry in Computer Vision*. Cambridge books online, Cambridge University Press.

Heyden, A. and Astrom, K., 1996. Euclidean reconstruction from constant intrinsic parameters. In: *Proceedings of 13th International Conference on Pattern Recognition*, Vol. 1, pp. 339–343 vol.1.

Hilsenbeck, S., Mller, A., Huitl, R., Schroth, G., Kranz, M. and Steinbach, E., 2012a. Scale-preserving long-term visual odometry for indoor navigation. In: *2012 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pp. 1–10.

Hilsenbeck, S., Möller, A., Huitl, R., Schroth, G., Kranz, M. and Steinbach, E., 2012b. Scale-preserving long-term visual odometry for indoor navigation. *International Conference on Indoor Positioning and Indoor Navigation*.

Kaminsky, R. S., Snavely, N., Seitz, S. M. and Szeliski, R., 2009. Alignment of 3d point clouds to overhead images. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pp. 63–70.

Lemaire, T., Berger, C., Jung, I.-K. and Lacroix, S., 2007. Vision-based slam: Stereo and monocular approaches. *International Journal of Computer Vision* 74(3), pp. 343.

Mur-Artal, R. and Tardós, J. D., 2017. ORB-SLAM2: an open-source SLAM system for monocular, stereo and RGB-D cameras. *IEEE Transactions on Robotics* 33(5), pp. 1255–1262.

Ni, K., Armstrong-Crews, N. and Sawyer, S., 2013. Geo-registering 3d point clouds to 2d maps with scan matching and the hough transform. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 1864–1868.

Nister, D., Naroditsky, O. and Bergen, J., 2004. Visual odometry. In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, Vol. 1, pp. I–652–I–659 Vol.1.

Nister, D., Naroditsky, O. and Bergen, J., 2006. Visual odometry for ground vehicle applications. *Journal of Field Robotics* 23, pp. 2006.

Ros, G., Sappa, A., Ponsa, D. and Lopez, A. M., 2012. Visual slam for driverless cars: A brief survey. In: *Intelligent Vehicles Symposium (IV) Workshops*, Vol. 2.

Scaramuzza, D. and Fraundorfer, F., 2011. Visual odometry [tutorial]. *IEEE Robotics Automation Magazine* 18(4), pp. 80–92.

Scaramuzza, D., Fraundorfer, F., Pollefeys, M. and Siegwart, R., 2009. Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints. In: *2009 IEEE 12th International Conference on Computer Vision*, pp. 1413–1419.

Soheilian, B. and Brédif, M., 2014. Multi-view 3D circular target reconstruction with uncertainty analysis. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* pp. 143–148.

Song, S., Chandraker, M. and Guest, C. C., 2016. High accuracy monocular sfm and scale correction for autonomous driving. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38(4), pp. 730–743.

Statista, 2013. Worldwide shipments of cameras for cars in 2012, 2013 and 2020 (in million units). https://www.statista.com/statistics/262015/worldwide-shipments-of-cameras-for-cars/, accessed 2018-01-08.

Sturm, J., Engelhard, N., Endres, F., Burgard, W. and Cremers, D., 2012. A benchmark for the evaluation of rgb-d slam systems. In: *Proc. of the International Conference on Intelligent Robot Systems (IROS)*.

Suzuki, S. and Abe, K., 1985. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing* 30(1), pp. 32–46.