Image and Vision Computing 68 (2017) 53–63

Contents lists available at ScienceDirect

Image and Vision Computing

journal homepage: www.elsevier.com/locate/imavis

Regression of feature scale tracklets for decimeter visual localization☆

David Wong[a,*], Daisuke Deguchi[b], Yasutomo Kawanishi[a], Ichiro Ide[a], Hiroshi Murase[a]

[a] *Graduate School of Informatics, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan*
[b] *Information Strategy Office, Nagoya University, Furo-cho, Chikusa-ku, Nagoya 464-8601, Japan*

ARTICLE INFO

Article history:
Received 5 April 2016
Received in revised form 22 June 2017
Accepted 21 July 2017
Available online 5 August 2017

Keywords:
Ego-localization
Monocular vision
Feature scale

ABSTRACT

Localization along a route is an everyday necessity for in-vehicle navigation systems, and a vital task for automated driving technologies. Visual ego-localization promises reliable accuracy even in challenging urban environments where Global Positioning Systems (GPSs) can fail. Using cameras for localization against a pre-constructed database requires either the creation of a dense three-dimensional feature point map and pose estimation of a query camera relative to this map, or image matching along a database route to determine the capture position of the query camera based on the most similar database image. While the latter method is potentially less computationally intensive and can provide a more compact database, localization accuracy is limited by the discrete positioning information at database frame capture locations. In this paper we propose an image matching method that makes use of image features which are pre-matched during database construction, allowing linear regression coefficients for the relationship between capture position and feature scale to be calculated. The capture position of matched query features can then be estimated to sub-database spacing resolution. By incorporating the visual localization system into a Bayes estimator, we demonstrate an average monocular vision localization accuracy of 0.33 m in tests on actual vehicle image streams.

1. Introduction

Current in-car navigation systems typically rely on a Global Positioning System (GPS) for ego-localization. While sufficient for assisting the driver with directions and general positioning in intelligent vehicle systems, the localization accuracy provided by consumer GPS setups is typically inadequate for automated driving tasks. In addition, in some common automotive environments, GPS fails completely — for example, when the signal is blocked inside parking buildings or tunnels. GPS is also unsuitable where multi-level roads exist, as its altitude resolution is not sufficient to determine which level is being traversed. While these localization failures may not be of vital importance to simple navigation systems, as we move toward intelligent vehicles that use precise localization information to determine suitable vehicle control behavior, accurate and fast localization in all environments is critical. Vehicle localization systems using expensive Inertial Measurement Units (IMUs), laser scanners, and Real Time Kinematc (RTK) GPS have been extensively researched. However, the hardware simplicity and cost effectiveness of localization using only visual cameras make it a very attractive solution.

Computer vision for automotive ego-localization involves determining the current position of the vehicle relative to a known map. The constrained nature of roadways allows localization to be performed by matching the current image from a vehicle-mounted camera to a stream of pre-captured database images. Visual localization techniques can be broadly separated into two types: pose-based methods, which attempt to calculate the exact pose of the camera view from a dense three dimensional map of points [1,2]; and path-following methods, which determine how far along a given database sequence the current capture position is, based on image similarity [3–7]. Pose-based methods can be very accurate, but depend heavily on the correct matching of many feature points and calculation of image geometries. Accurately calibrated cameras and significant processing power are usually required for real-time operation. Path-following methods, however, provide a more general localization result which is often suitable for automotive applications because of the constrained motion allowed by roads. However, the localization accuracy of path-following methods is limited by the spacing between database images. A broad spacing makes for a smaller database size and faster localization but reduces

☆ This paper has been recommended for acceptance by Branislav Kisacanin.
* Corresponding author.
E-mail addresses: davidw@murase.is.i.nagoya-u.ac.jp (D. Wong), ddeguchi@nagoya-u.jp (D. Deguchi), kawanishi@i.nagoya-u.ac.jp (Y. Kawanishi), ide@i.nagoya-u.ac.jp (I. Ide), murase@i.nagoya-u.ac.jp (H. Murase).

http://dx.doi.org/10.1016/j.imavis.2017.07.004
0262-8856/© 2017 Elsevier B.V. All rights reserved.

the spatial resolution of the database. Therefore a technique to accurately localize a query image captured from a location between two database images is significant for path-following visual localization methods.

In this paper, in order to overcome the localization accuracy limitation of path-following localization systems from database image spacing, we propose the use of scale interpolation of matched local features between frames. Our system takes a scene captured from a camera and returns a location estimate by comparing feature points from the input image with those within the database. Where most path-following localization systems determine the current vehicle localization based on visual similarity between the input image and database images, the proposed method makes use of the continuous scale property of matched image feature points to provide location estimates at a higher resolution than possible when using only discrete database image locations. Visual feature points extracted from corner detectors such as the Scale Invariant Feature Transform (SIFT) [8] have a size, or scale property. The scale of these features change with their distance from the camera, and we make use of this change together with the constrained motion of a vehicle traveling along a known roadway to provide more accurate localization from a query image. For localization within large maps, the database size is an important consideration. By storing only the strongest feature points in the database, and pre-matching database features, our method limits the database size while allowing efficient feature point matching in the localization step. We introduced the use of interpolation between frames using feature scale in our previous work [9]. In this paper we expand on this concept and provide the following contributions:

1. A database structure for image features, which are organized into groups of corresponding features from consecutive database frames which we call "feature scale tracklets". A novel inclusion is the linear regression coefficients for each "feature scale tracklet" which describe the relationship between feature scale and image capture position.
2. A localization method where each feature from a query image frame is matched to a corresponding database "feature scale tracklet", and its scale is used to create a current position estimate. We propose the combination of individual feature estimates into a Probability Distribution Function (PDF).
3. A Bayes estimator for the system. With a linear motion model we show that a Kalman filter is appropriate for position estimation using the proposed visual localization method.

We evaluate our method with a real-world dataset and analyze the database size and performance.

This paper is organized as follows: In Section 2 we give a brief overview of related research. We describe our novel contributions and proposed methodology in Section 3. The implementation of the proposed system is detailed in Section 4. Experimental results are presented in Section 5 followed by a discussion in Section 6, and the paper is concluded in Section 7.

2. Related work

There are many ego-localization methods that use computer vision to achieve position estimates. Here we present an overview of some of the approaches that are related to the proposed method.

2.1. Local feature-based pose estimation methods

Many visual localization methods rely on local feature extraction and description. Features are extracted from images and used to build a visual database of the environment, typically using some kind of invariant feature extraction method such as SIFT [8]. More modern extraction and description methods include Speeded Up Robust Features (SURF) [10], which offer some performance advantages by using integral images, and binary descriptors such as Binary Robust Independent Elementary Features (BRIEF) [11], Oriented FAST and Rotated BRIEF (ORB) [12], Binary Robust Invariant Scalable Keypoints (BRISK) [13] and Fast Retina Keypoints (FREAK) [14]. Binary descriptors are typically used with a Features from Accelerated Segment Test (FAST) [15] detection method, incorporating machine learning into the corner detection process for determining good feature locations. All feature-based methods depend on matching of features in the database to features from a query image. Typically, the database features are processed to create semi-dense scene representations with 3D positions for each feature point [1,16]. Localization then takes place by calculating the geometry constraints between a candidate set of database features and their matches within the query image, in order to estimate the camera pose. Pose estimates are localized within a world map using a pose graph [17,18], or a Bayes estimator [19,20]. These methods have much in common with the Simultaneous Localization And Mapping (SLAM) [21] techniques used in robotics, with the key difference being that the map construction stage is able to be performed off-line in the vehicle localization case.

Local feature based methods suffer from several issues when applied to the vehicle localization problem. The number of matched features necessary for exact camera pose estimation is large, with a recursive inlier selection process such as Random Sample Consensus (RANSAC) [22] required for estimation of image-to-image geometry. When too few consistent inlier feature matches are found, localization fails. The calculation of the camera pose is a computationally intensive process, and when the distance between the query and database images is small, short baseline degeneracies [23] can occur. In addition, the number of database features required leads to very large databases which can be difficult to handle for real-world localization, where the database for an entire region or city may have to be stored or downloaded to the system. Pose estimation from extracted features also requires accurately calibrated cameras, and maintaining calibration for a camera in service on a consumer vehicle is a potentially difficult task.

2.2. Path-following methods

For automotive localization, an exact camera pose estimate is not necessarily required. Vehicles traveling along a road generally follow prescribed lanes, so accurate positioning longitudinally along the lane is sufficient for many automated driving tasks. There are many implementations of path-following methods using vision, which generally provide more scalable performance at the cost of lower precision when compared to pose estimation methods. Path-following methods typically compute whole image discrepancies to determine the most likely position of a query image along a stream of database images. A whole-image discrepancy measure can be formulated by using the Euclidean distance of dimension reduced images [4], whole image descriptors constructed in a similar way to SURF descriptors [5], or a low bit-rate image sequence instead of single images [3]. Dynamic Time Warping (DTW) [24] can then be used for image matching [3,4], or alternatively a Bayes estimator can be employed [5,6] to generate position estimates.

Techniques that rely on whole image matching can be affected by situations where images captured at the same location may vary significantly, for example where dynamic traffic environments create occlusions. There are also methods that use features for image matching without direct pose estimation, offering advantages in these situations — usually visible features can still be matched even where a scene may be partially blocked by a passing truck, for example. Image matching techniques that use feature points include monitoring the epipole position of features matched between query and database images [7]. When two images were captured from similar locations, the epipole position of matched features moves toward the outside of the image, and this can be used as a similarity measure within a DTW method. However, this system does still require calculation of the essential matrix.

3. Contributed concepts

The method proposed in this paper aims to overcome some of the issues of visual localization in the automotive environment. While local feature points are used to offer robustness in real-world traffic, we employ a path-following method which does not involve camera pose estimation. Therefore camera calibration is not important for localization, and localization can be performed even when few feature matches are found. We use the scale of matched features to interpolate the capture position of a query image along a series of database features, with the addition of a Bayes filter to improve localization results and error analysis. The system overview of our method is shown in Fig. 1. The main novel contributions of this paper can be explained as follows:

1. We combine corresponding features from consecutive database frames into "feature scale tracklets" and determine the parameters of the linear relationship between feature scale and capture position using least squares regression. Query image features are matched to database feature scale tracklets, and their scales are used to calculate a capture position estimate per tracklet, which are then combined into a PDF. See Section 3.1.
2. The feature scale-based localization system is incorporated into a Bayes estimator which allows filtering of the localization results with a general motion model (see Section 3.2). The use of a Bayes estimator also allows the inclusion of control inputs and other measurement information such as odometry if available. In Section 3.3 we describe how the Bayes estimator is implemented using a Kalman filter.

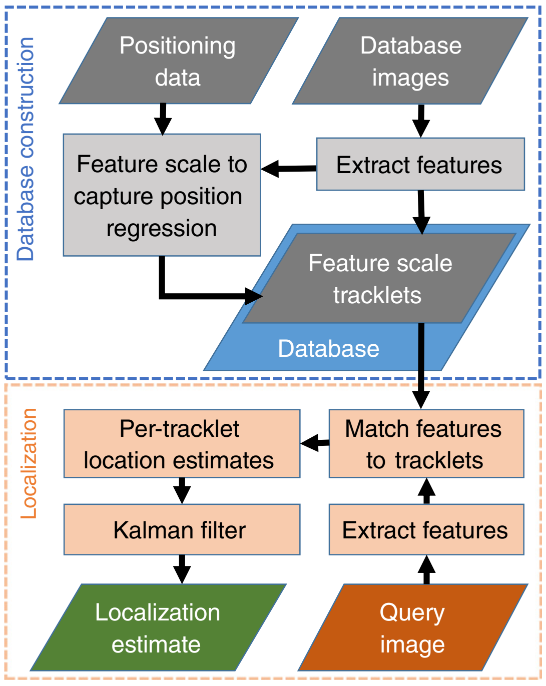


Fig. 1. System flow of the proposed method.

3.1. Probability distribution function from tracklet estimates

In our previous work [25] we proposed the "feature scale tracklet" concept. In this paper, we introduce a method for using scale and capture position regression within tracklets to create a PDF of the overall visual localization estimate for an input query frame. The key concept of "feature scale tracklets" is that corresponding features between images have a continuous scale, and in the case of a vehicle moving linearly along a road, the feature scales increase with each consecutive image. As the scale property is continuous, we can model the relationship between scale and capture position with linear least squares. This can be used to interpolate the capture position of a query feature even between database capture locations.

3.1.1. Feature scale tracklet

Feature scale tracklets are created in the database construction phase. SIFT feature points are extracted from each database frame and are matched in consecutive frames. The resulting database is a web of interconnected features, arranged into "feature scale tracklets", $T \in \mathcal{T}$ where $\mathcal{T}$ is the full set of database tracklets. A feature tracklet T contains a list of M pre-matched database features from sequential frames,

$$T = \{g_1, g_2, \dots, g_m, \dots, g_M\},$$
$$g_m = \{\lambda_m, a_{\lambda_m}, \mathbf{u}_m, s_m, \mathbf{p}_m, l_m\}, \tag{1}$$

with each $m = 1, 2, \dots, M$ as the index along the tracklet. Here λ_m is the database image index which refers to the database image containing the feature g_m, and a_{λ_m} is the feature index within the corresponding λ_m. The descriptor of feature g_m is denoted as $\mathbf{u}_m$, s_m is the feature scale, and $\mathbf{p}_m$ is the vector containing the feature pixel positions. The longitudinal distance along the current database segment is denoted l_m, and defines the capture location of the feature g_m. While the capture locations are actually two-dimensional co-ordinates, they are converted into one-dimensional values to represent the length in meters along the current database segment. Where the road splits and new segments form, the segments are annotated appropriately. This notation is suitable because lateral motion of the vehicle is constrained and of less interest than the longitudinal position along the road. As we discuss in Section 3.1.2, it also simplifies the modeling of position with feature scale.

An example of three feature scale tracklets are displayed over four database frames in Fig. 2, with each tracklet in a different color. In this diagram, the features from a query frame are shown matched to database tracklets using the average descriptor of the tracklet $\mathbf{d}(T)$ (see Section 4.1.1). The tracklet features are shown as extracted from their parent database images λ_m. The interpolation of position from feature scale displayed in this diagram is the core of the visual localization process, which is described in Section 3.1.3. A set of regression coefficients Θ are provided by each tracklet that is matched to a query feature, and these are used to perform the interpolation based on the query feature's scale. The individual position estimates are then combined and used within the Bayes estimator, implemented in this case with a Kalman filter (see Section 3.2).

3.1.2. Tracklet parametrization

Instead of performing a multivariate linear regression to build a per-tracklet model of feature scale changes with capture position [9], in this paper we perform a simple bivariate least squares regression made possible by resolving the capture positions into a single longitudinal distance along the current database segment. The regression process allows the inclusion of scale information from all corresponding feature points in the database, giving a more robust

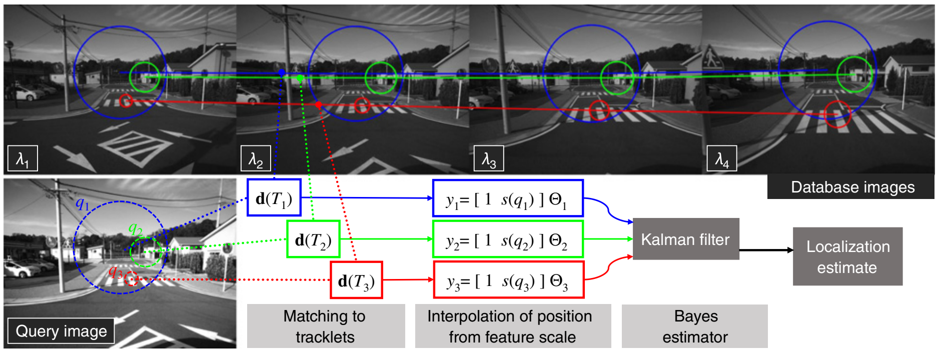


Fig. 2. Example query image and a string of database images displaying three example tracklets, each highlighted in a different color. The query image features, matched to the tracklets through the average descriptors, are shown matched to the tracklet line based on the interpolated position along the tracklet, calculated from the query feature scale and tracklet regression coefficients.

position estimate per tracklet and allowing filtering based on the coefficient of determination.

If $\mathbf{l}$ is the vector of capture coordinates $l_1, l_2, \ldots, l_M$ of M consecutively matched database features within the tracklet, and S is the $M \times 2$ design matrix containing the corresponding feature scale row vectors $\mathbf{s}_m = [1 s_m]$, then the coefficients for the linear regression Θ can be calculated by ordinary least squares as

$$\Theta = (S^{\mathrm{T}}S)^{-1}S^{\mathrm{T}}\mathbf{l}. \tag{2}$$

The coefficients calculated for each tracklet form a parametrization which allows a direct capture position estimate when supplied with a corresponding query feature's scale.

3.1.3. Visual localization

When N features from a query image are matched to a subset of database feature scale tracklets, each matched query feature q_i and corresponding T_i (where $i = 1, 2, \ldots, N$) create an estimate of the query capture position, y_i, using the database coefficients determined for each tracklet in Eq. (2) and the query feature scale $s(q_i)$ as

$$y_i = [1 \quad s(q_i)]\,\Theta_i. \tag{3}$$

The results of the individual estimates can be combined to form a single position estimate by averaging their values as

$$y = \frac{1}{N}\sum_{i=1}^{N}[1 \quad s(q_i)]\,\Theta_i. \tag{4}$$

The variance of the overall position estimate can be calculated as

$$\sigma_y^2 = \frac{1}{N}\sum_{i=1}^{N}(([1 \quad s(q_i)]\,\Theta_i) - y)^2. \tag{5}$$

One of the contributions of the proposed method is the recognition that the multiple estimates form a Gaussian distribution which is suitable for inclusion in a Bayes estimator as a measurement model. When combined, the query position estimates form the following PDF:

$$P(y \mid x) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x-y)^2}{2\sigma_y^2}\right), \tag{6}$$

where x is the capture position of the query image, y is the mean of the individual observations y_i, and σ_y^2 is the variance of the distribution.

3.2. Bayes estimator

This paper proposes the use of a Bayes estimator for tracklet-based visual localization updates, based on the contribution of Section 3.1.3. A Bayes estimator maximizes the *a posteriori* likelihood of the state of a dynamic system. If a variety of information is available about the system state, the Bayes estimator can combine this information, together with the confidence of each state estimate, to provide the most probable current state. For a moving vehicle, the current position can be estimated by predicting the next state from previous states using a motion model, and then updating the predicted state using sensor measurements. The Bayes estimator combines these two estimates by recursively predicting the PDFs that describe the prediction estimate (state transition model) and the measurement estimate (measurement model). The per-tracklet position estimate provides a variance estimate and has a Gaussian distribution (see Section 6) which makes a Bayes estimator a good choice for including other measurements, control inputs, or a motion model, in order to improve the overall localization performance. In this section we formulate a standard Bayes estimator for our system.

The location of the vehicle along the current database segment at the current time step k (with associated system time t_k) is denoted as x_k and represents the state of the vehicle. Assuming a Markov model, the probability distribution function of the vehicle position x_k is conditioned on the previous state x_{k-1} and the measurement update of Eq. (4) provided by the visual localization system y_k. The state transition model is denoted $P(x_k \mid x_{k-1})$ and the measurement model $P(y_k \mid x_k)$. First, the transition model is used to predict the location of the vehicle as

$$P(x_k \mid y_{k-1}) = \int P(x_k \mid x_{k-1})P(x_{k-1} \mid y_{k-1})\mathrm{d}x_{k-1}, \tag{7}$$

followed by the update state incorporating new measurements from the visual localization system within the measurement model as

$$P(x_k \mid y_k) = \alpha P(y_k \mid x_k)P(x_k \mid y_{k-1}), \tag{8}$$

where α is a scalar to ensure that the result integrates to one. In absence of any control input or odometry information, the transition model is a simple constant velocity model described in Section 4.2.1. The measurement model is a direct application of the query position estimate PDF in Eq. (6). In this particular application, we assume that our measurement and state transition models are linear, which allows us to use a Kalman filter.

3.3. Kalman filtering

We are dealing with a unimodal system with a Gaussian state transition model and measurement model, so we can use a Kalman filter to avoid the difficult calculations of finding the maximum *a posteriori* estimate from Eqs. (7) and (8). The Kalman prediction step for this system is formulated as

$$\hat{x}_k = x_{k-1} + \Delta x + w_k,$$
$$\hat{\sigma}_k^2 = \sigma_{k-1}^2 + \sigma_{w_k}^2, \tag{9}$$

where w_k is the transition model noise, and Δx is the predicted position change determined by the chosen motion model. We can now use the measurement model to perform the update as

$$x_k = \hat{x}_k + K_k(y_k - \hat{x}_k),$$
$$\sigma_k^2 = (1 - K_k)\hat{\sigma}_k^2, \tag{10}$$

where K_k is the Kalman gain, calculated at each step as

$$K_k = \frac{\hat{\sigma}_k^2}{\hat{\sigma}_k^2 + \sigma_y^2}. \tag{11}$$

More implementation details of the proposed methods are presented below in Section 4. Fig. 2 summarizes how the contributed processes perform localization using example tracklets.

4. Localization system implementation

Our method consists of two main stages — the database capture and construction, and the localization of an input query image. The database capture involves taking images from a vehicle-mounted camera together with accurate localization information for every capture position. The database is then constructed by extracting and pre-matching local feature points in consecutive database frames. Localization is performed by extracting features from a query image and determining the position of the capture location based on scale differences between these features and their corresponding database feature matches. The database construction stage is described in Section 4.1 and the localization stage in Section 4.2.

4.1. Database construction

The method proposed in this paper requires all localization routes to be previously traversed and captured by a special data capture vehicle. The data capture itself requires images from a single forward facing camera, and accurate localization information for the vehicle at each capture location. For the experiments presented in Section 5, we used a Mobile Mapping System (MMS) to capture database image streams. This system combines information from an IMU, odometry sensors and a highly accurate GPS to provide accurate capture locations for each image.

The series of database images are processed to create the visual database, which actually only consists of labeled feature descriptor information and discards the original images themselves. SIFT feature points are extracted from each frame and are matched in consecutive frames. This is an important step, as it affords the following:

1. Features can be pruned from the database. Only features that appear consistently in multiple consecutive database frames, and have distinct descriptors, are kept for the final database. For feature matching we use the Fast Library for Approximate Nearest Neighbors (FLANN) matcher [26] for computational efficiency and apply a Nearest Neighbor Distance Ratio (NNDR) test to filter strong matches. Consistent matches are found by tracking feature positions and projecting a feature search area into the next database image that is consistent with the previous feature motions for each current feature.
2. Consecutively matched features can be grouped together and can share a single descriptor. Consecutive features matched between database frames have very similar descriptors, as a result of the matching process. In order to make the descriptor more general for future matching to query image features, and also to reduce the database size, the collection of matched features in the database have their descriptors averaged to create a single feature descriptor for the group of features. This process is described in more detail in Section 4.1.1.

4.1.1. Feature scale tracklet construction

Database images have SIFT features extracted and tracklets are constructed by matching features from consecutive frames and grouping them as described in Section 3.1. For each tracklet, the parameters that describe the relationship between capture position and feature scale are then calculated as described in Section 3.1.2. Additionally, the residual sum of squares, SS_{res} of the set of feature positions with respect to scale can be calculated as

$$SS_{res} = \sum_{m=1}^{M} (l_m - \mathbf{s}_m\Theta)^2. \tag{12}$$

Similarly the total sum of squares SS_{tot} can be calculated as

$$SS_{tot} = \sum_{m=1}^{M} \left(l_m - \bar{l}\right)^2, \tag{13}$$

where $\bar{l}$ is the mean of the capture locations $l_1, l_2, \ldots, l_M$. A measure of variability in the scale to position relationship for each tracklet is the ratio of the residual sum of squares to the total sum of squares, known as the coefficient of determination, R^2. This can be can be calculated as

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}}. \tag{14}$$

This ratio determines the closeness of the regression fit and can be used to prune tracklets that display highly non-linear scale change with position. The R^2 value is between zero and one, with one representing a perfect linear fit. Tracklets with an R^2 over a chosen threshold can be selected for inclusion in the database. We found in our experiments that a value of 0.8 is an effective threshold for filtering tracklets with mismatches.

We also propose the averaging of feature descriptors within each tracklet to reduce the database size. Once the features have been collected into "feature scale tracklets", they make up a string of matched features of increasing scale and varying image position. However, they all have similar descriptors which led to them being matched to each other. We can now assign a single descriptor $\mathbf{d}(T)$ as

$$\mathbf{d}(T) = \frac{1}{M} \sum_{m=1}^{M} \mathbf{u}_m, \tag{15}$$

which provides the average descriptor for each tracklet T. When matching query image features to the features in the database, candidate database tracklets are filtered by their capture location range $(l_1 : l_M)$ and their pixel locations, and then a query match can be found from normal feature matches to the tracklet descriptors. This is discussed further in Section 4.2.2.

4.2. Query image localization

Before a query image is localized relative to the database tracklets, the system must retrieve the relevant sections of the database. The database may be retrieved as a single file with indexed tracklets, or for very large databases, split into many files of a more manageable size for reading or streaming into the localization system. The current vehicle position estimate is used to select candidate tracklets from the database based on the capture positions which they cover. Where no current position estimate exists, initialization is performed (see Section 4.2.4). The visual localization process, as described in Section 3.1.3, is then performed within the Bayes estimator (described in Section 3.2). This requires two parts; the state transition model for predicting the next state of the vehicle, and the measurement model for updating the prediction based on the visual localization measurements from the tracklet estimates.

4.2.1. State transition model implementation

Assuming that no vehicle odometry information is available, we employ a constant velocity state transition model. The new vehicle position PDF is approximated to be a Gaussian with mean μ and variance σ_w. We calculate the previous distance covered between query image frames, and adjust this distance for timing differences in the next time step as

$$\Delta x = \left(\frac{x_{k-1} - x_{k-2}}{t_{k-1} - t_{k-2}} \right) (t_k - t_{k-1}). \tag{16}$$

The calculated Δx is added to the last predicted vehicle location such that $\mu = x_{k-1} + \Delta x$. This provides the following Gaussian probability distribution function

$$P(x_k \mid x_{k-1}) = \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp\left(-\frac{(x_{k-1} - \mu)^2}{2\sigma_w^2} \right), \tag{17}$$

which can be used in the transition model of Eq. (7), and in the case of a Kalman filter, Eq. (9).

4.2.2. Measurement model implementation

The key component of this localization method is the visual localization estimation. The candidate database tracklets are selected based on the previous position x_{k-1}. A tracklet is selected for matching if the span of capture positions of the tracklet $l_1, l_2, \ldots, l_M$ intersect with the query region between x_{k-1} and $x_{k-1} + \rho\Delta x$, where ρ is a factor to determine the range of the database tracklets to include for potential matching. Increasing the value of ρ increases the number of tracklets included in the matching process, which can give better position estimates at the expense of slower matching time. We found in experimental testing that a ρ value of 2.0 usually encompassed all tracklets which would produce inlier position estimates.

The resulting set of tracklets are then matched to the query image features by comparing the tracklet average descriptors to the query image feature descriptors, using the same FLANN matching technique [26] as used in the database construction stage. The set of matches are pruned by removing matches where the query feature scale $s(q)$ is not within the range of the tracklet scales, $s_1, s_2, \ldots, s_M$. The final set of matches are then used to complete the measurement model described by Eqs. (8) and (10) in Section 3.2.

4.2.3. Kalman filter implementation

In practice, the visual position estimate update is much more accurate than the estimate from the motion model. Even when the visual position estimate has a high error (which usually occurs when the number of query features matched to tracklets becomes very low), the predicted variance may not be particularly high. Therefore we weight the Kalman gain (Eq. (11)) such that the influence of the motion model has little affect unless $y_k - \hat{x}_k$ exceeds a threshold, at which point the measurement variance σ_y^2 is weighted to increase the filtering power of the state transition prediction. The output of the Kalman filter, as presented in Eq. (10), is a position estimate x_k at each query image capture, with a measure of its confidence σ_k^2.

4.2.4. Initialization

When the vehicle is first localized, there is no information available about the vehicle velocity for use in the Kalman filter. To commence localization, coarse positioning information from GPS is used to determine the general region, and tracklets covering close-by locations are included for potential matches to the original query frame features. The first few frames are localized using visual estimates only. The approximate localization from GPS could be replaced with a vision-only approach by sequence matching over dimension reduced images [3]. This would remove the dependence on GPS, but would potentially require a few minutes to capture a sequence suitable for general position recognition. The increase in database size would be small as only a few bits are required for sequence template matching.

5. Experiments

To evaluate the localization performance of the proposed method, a dataset captured at a driving school was used. The location allowed various lane changes, traversal of intersections, and maneuvers to be safely performed away from busy traffic. Fig. 3 shows the vehicle paths. Examples of database images and their corresponding query images are presented in Fig. 4.

A sophisticated data capture system was employed to provide accurate ground-truth data in this dataset. This process is presented in Section 5.1, and the methods used for evaluation are introduced in Section 5.2. The database construction and localization results are reported in Section 5.3. In Section 5.4 we evaluate the performance of the Kalman filter by comparing with a particle filter on both straight and curved road areas.

5.1. Vehicle configuration

The database and query image streams were captured using a Mitsubishi Electric MMS-X320R MMS. This system incorporates three 5 megapixel cameras, three laser scanners, GPS, and odometry hardware. The three cameras are roof mounted, and only one forward facing camera was used in the experiments. None of the data collected from the laser scanners were used in this research. Capture position estimation for database construction and ground-truth is by a fusion of wheel encoder odometry, a high-accuracy GPS unit and an IMU. This process is performed with proprietary tools and gives a claimed root mean square localization error of 6 cm. The system provided an estimated average error of below 1 cm in the experiments that were conducted, aided by good GPS satellite coverage. The MMS produces timestamped images with corresponding capture position coordinates, with an image capture rate of up to 10 Hz. The actual image rate depends on synchronization with other hardware, but was at approximately 2 m intervals for most of our dataset. The camera uses a wide angle lens (horizontal: 80°, vertical: 64°), and although the native capture resolution of the MMS camera is 2400 × 2000 pixels, we found that localization performance was maintained after downscaling to 600 × 500 pixels which improved computation speed. The localization information supplied by the MMS for each image was used for tracklet construction in the database streams, and provided

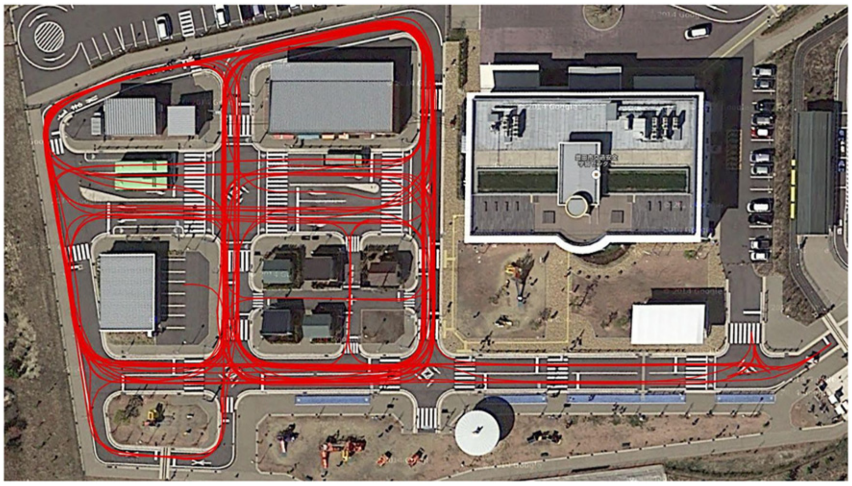


Fig. 3. The dataset location, showing the driving paths used.
Source: Satellite imagery: Google, ZENRIN.

a ground-truth for the query stream images to allow performance evaluation.

5.2. Evaluated methods

To evaluate the performance of the proposed system, we compared five localization methods. The experiments used a standard desktop computer with a 3.5 GHz Intel i7 processor. No GPU, multi-threading, or optimization was used to increase performance, and the OpenCV C++ library [27] was used to implement feature detectors and descriptor extractors. The tested methods are summarized as follows:

1. **Comparative method 1: WISURF.** This method uses the Whole-Image SURF (WISURF) descriptor [5] to perform image matching. By comparing WISURF descriptors, this technique selects the database image that minimizes the descriptor distance. We implemented this method using DTW to compare image streams. Localization is performed by applying the localization information from the matched database image to the query image directly.
2. **Comparative method 2: Scale only.** This method is based on our previous work [28] and uses the average scale change between matched features of individual images as a cost measure within a DTW framework. Like the proposed method, it uses matched feature scale comparisons; however, it does not use the pre-matching and tracklet construction techniques presented in this paper.
3. **Comparative method 3: Tracklet-based image matching.** This method uses the pre-matched feature scale tracklets presented in Section 3.1.1 and finds the closest database image match per feature, but it does not incorporate any form of interpolation or regression of feature scale between database image positions. Localization is performed by using the localization information from the predicted closest match directly. This method is based on our previous work [25].
4. **Proposed method 1: Proposed method without Kalman filter.** This method uses all techniques introduced in this

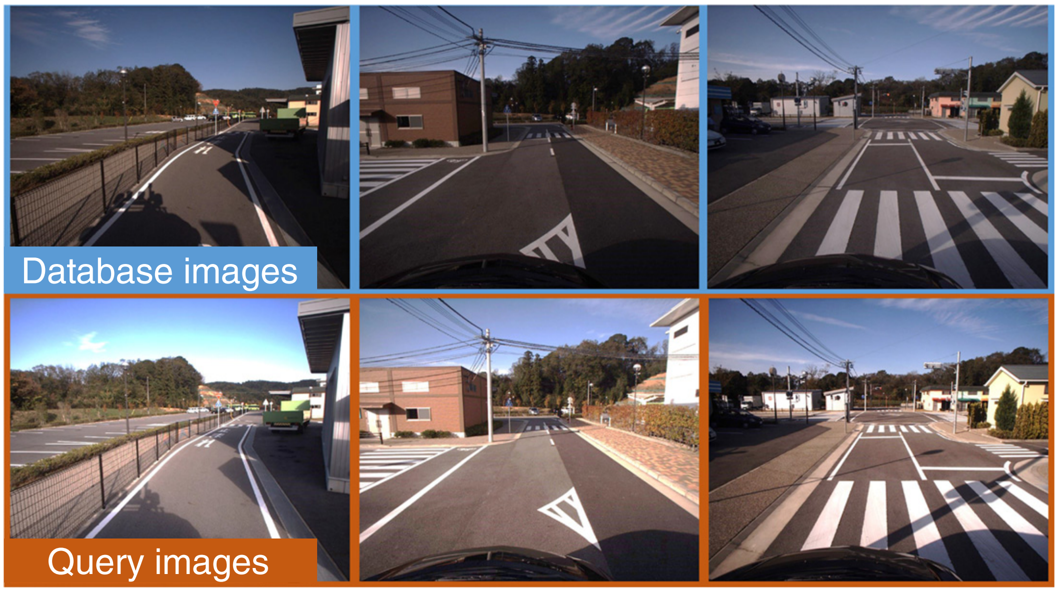


Fig. 4. Examples of query images from the dataset and their corresponding database images.

paper up to the Bayes estimator for position estimation with a Kalman filter. The feature tracklet regression models described in Section 3.1 and the descriptor averaging proposed in Section 4.1.1 are employed to create per feature interpolated position estimates. Localization is performed without a state transition model and proceeds with the average of the visual location estimates only.

5. **Proposed method 2: Proposed method with Kalman filter.** This method uses all techniques introduced in this paper. It expands on proposed method 1 to incorporate a Bayes estimator in the form of a Kalman filter as described in Sections 3.2 and 3.3.

Note that in the experimental results that follow, feature descriptor averaging was only applied to the proposed methods. It can also be applied to comparative method 3, but we used standard per-feature descriptors in comparative method 3 as per our previous work [25]. We tested all feature tracklet methods with and without descriptor averaging and found that despite consistently halving the database size, localization accuracy was unaffected.

5.3. Localization results

The localization results of the proposed methods and comparative methods were evaluated from the same database and query image streams. A summary of the average localization errors of the tested methods is presented in Table 1. The localization error rates of the different methods are shown in Fig. 5. The proposed method 2 achieved an average localization error of 0.33 m, representing a 77% improvement over the feature scale image matching method alone. Compared to proposed method 1, a 21% improvement was observed from the application of the Bayes filter for localization. The filter also managed to reduce the localization variance and maximum error, as the motion model provides smoothing of the localization results.

5.4. Estimator evaluation

The Kalman filter used in the proposed method is optimal for linear problems with Gaussian noise, but a vehicle may not strictly adhere to these properties. Especially where cornering occurs, the linear system assumption may no longer be appropriate. Therefore, the suitability of the Kalman estimator was evaluated by implementing a particle filter for comparison. A particle filter is also a Bayes estimator, but since it uses Monte Carlo random sampling for measurement and transition, limited assumptions are made about the underlying system model. This makes it an effective estimator for highly non-linear systems, but it is much more computationally expensive than a Kalman filter and may also suffer from divergence due to the random sampling nature of state propagation. We implemented a simple particle filter with Sampling Importance Resampling (SIR) [29]. We used the same motion model as described in Section 4.2.1, and experimented with various numbers of particles.

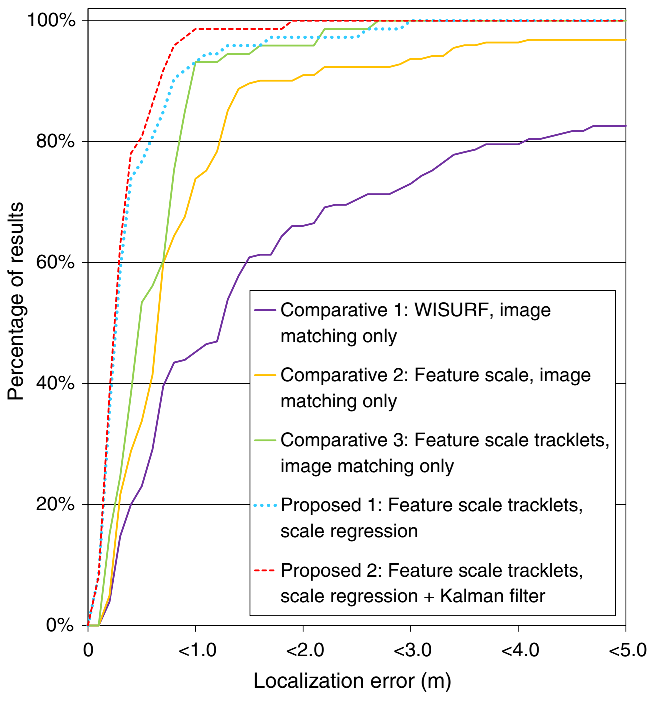


Fig. 5. Localization errors of the evaluated methods.

We found a few thousand particles were sufficient for consistent and reproducible results, and used 4000 for the results presented here.

The comparison of the particle filter estimator and proposed Kalman estimator is presented in Table 2 and Fig. 6. To determine if the particle filter has advantages over the Kalman filter in areas of the dataset where the vehicle maneuvers turns, the evaluation results are split into areas where a corner was traversed and straight sections of road. The results show that there was no significant difference in localization accuracy when using a particle filter, confirming that a Kalman filter is a suitable choice of estimator for this system.

6. Discussion

Here we present a discussion about the performance of the evaluated methods.

Apart from the comparative methods 1 and 2, all of the evaluated methods used the feature scale tracklets, so the image matching performance was the same for comparative method 2 and the two proposed methods. While the use of feature scale tracklets provided much better image matching results than comparative method 2 which uses only single matched feature pairs for scale comparisons, the differences between the three feature scale tracklet-based methods are more subtle. Using regression within the

Table 1
Localization results of proposed and comparative methods.

Method	Avg. error (m)	Standard deviation (m)	Max. error (m)	Database size (kB/m)
Comparative method 1				
WISURF descriptors, image matching only	3.71	1.68	4.61	**1.42**
Comparative method 2				
Feature scale, image matching only	1.00	1.32	11.04	191.23
Comparative method 3				
Feature scale tracklets, image matching only	0.61	0.46	2.66	80.96
Proposed method 1				
Feature scale tracklets, scale regression	0.42	0.49	2.92	40.19
Proposed method 2				
Feature scale tracklets, scale regression + Kalman filter	**0.33**	**0.27**	**1.82**	40.19

Bold values indicate the minimum value across the evaluated methods.

Table 2
Results of evaluation with different estimators.

Road type	Estimator	Avg. error (m)	Standard deviation (m)	Max. error (m)
Curved	No filter	0.39	0.44	2.58
	Particle filter	0.38	0.36	**1.80**
	Kalman filter	**0.35**	**0.31**	1.82
Straight	No filter	0.44	0.54	2.92
	Particle filter	0.32	**0.23**	**0.88**
	Kalman filter	**0.31**	**0.23**	0.96

Bold values indicate the minimum value across the evaluated methods.

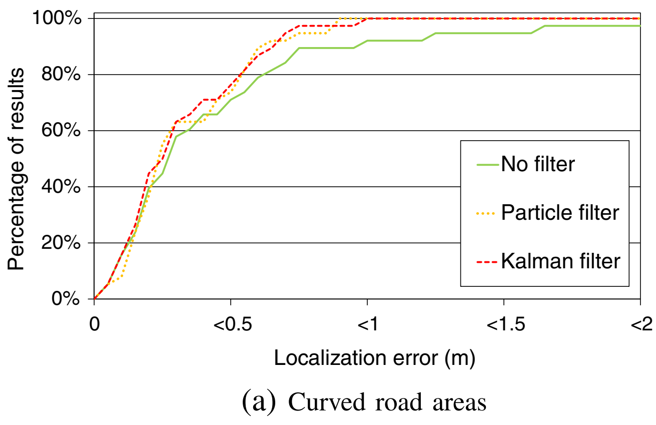
database (proposed method 1) made a significant 31% improvement in localization error over the best image matching only method (comparative method 3). This illustrates the effectiveness of the use of feature scale to interpolate position within a group of pre-matched features from consecutive database frames. The large number of estimates per query frame makes the visual localization system ideally suited to inclusion in a Bayes estimator, as variance can be easily calculated over the samples. A Kalman filter is a natural choice for including a simple motion model in the localization estimate, and also allows simple inclusion of control inputs and other sensor localization estimates if available. The simple constant speed motion model we applied smoothed the localization results enough to enable a 21% improvement in localization accuracy. It also reduced the variance of the localization error by 70%, as shown in Table 1 (where standard deviations are displayed instead of variance for unit consistency. The corresponding drop in standard deviation is 45%).

The visual localization method proposed in this paper relies on two assumptions. The linear regression performed on the tracklets constructed as described in Section 3.1 assumes that feature scale varies linearly with capture position. The strength of the linear fit can be evaluated using the coefficient of determination, R^2 (Eq. (14)). In our experiments, we found that most tracklets containing only correct matches have an R^2 very close to one, suggesting that the linear fit is a good model for feature scale change with forward motion. The regression lines of the example tracklets from Fig. 2 are shown in Fig. 7. The coefficients of each tracklet are displayed in the linear fit equation for each line, together with the R^2 value. The coefficient of determination is an important filter measure in the proposed method. By only using tracklets containing an R^2 above a certain threshold, we can effectively filter out features which do not show stable linear scale increase with capture position, and this also results in the removal of tracklets containing mis-matched features. Unlike pose estimation methods, the proposed method can generate a localization estimate even when only a few tracklets are matched within the query image features; in fact, even one matched tracklet will produce an estimate. The coefficient of determination is therefore useful to select a smaller group of feature matches which provide quality location estimates.

The second assumption is that the distribution of the per-tracklet visual estimates is a zero-mean Gaussian. In order to confirm that the noise distribution v_k of y_k is a zero-mean normal distribution such that $v_k \sim \mathcal{N}\left(0, \sigma_{y_k}^2\right)$, we plotted the location estimate error frequencies over many localization steps. The resulting distribution, shown in Fig. 8, proves that the noise distribution is indeed closely Gaussian.

The Gaussian nature of the visual position estimate is the reason that a Bayes estimator was chosen for localization. Some of the related works [3,4,7] use a DTW framework for localization estimation instead. These methods perform sequence matching between the database and query image streams, minimizing a cost function to determine the most likely location of the vehicle. The proposed method is related in that it creates a position estimate for each database feature by minimizing the feature scale difference cost, relative to a regression line constructed for each database tracklet. However, by pooling all of the individual feature estimates, a measurement update ideally suited for a Bayes filter is created. This allows easy inclusion of a motion model for smoothing and improving the localization results. DTW is a suitable estimator when interpolation between image frames is not used, but for localization precision that exceeds the database image spacing some form or regression is required.

If the system is assumed to be linear, a Kalman filter is an optimal estimator. A non-linear estimator, such as an Unscented Kalman (UKF) [30] or particle filter [29] can also be applied to this method but adds complexity, with potentially limited benefit. The particle



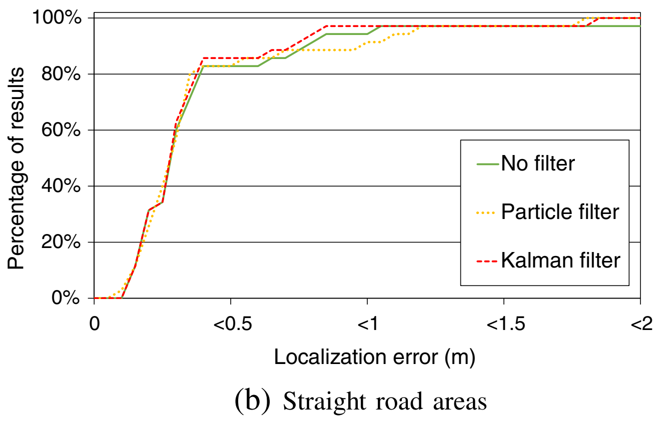


Fig. 6. Localization error rates of the proposed Kalman estimator, compared with a particle filter and no filter on both (a) straight, and (b) curved sections of road.

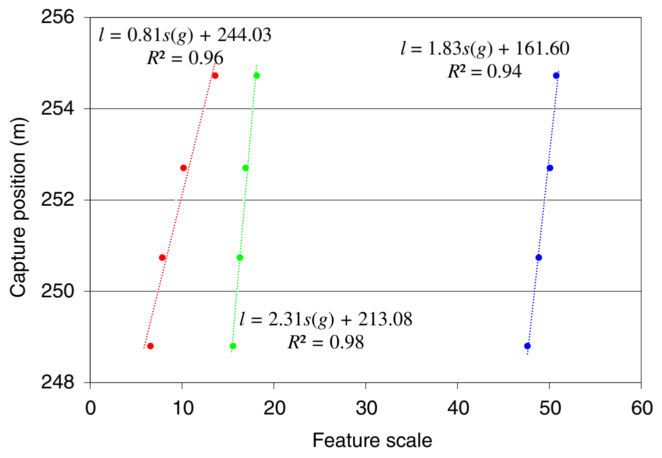


Fig. 7. The linear regression fit of the example tracklets in Fig. 2. The resulting regression coefficients and R^2 values are shown next to the regression line of each tracklet.

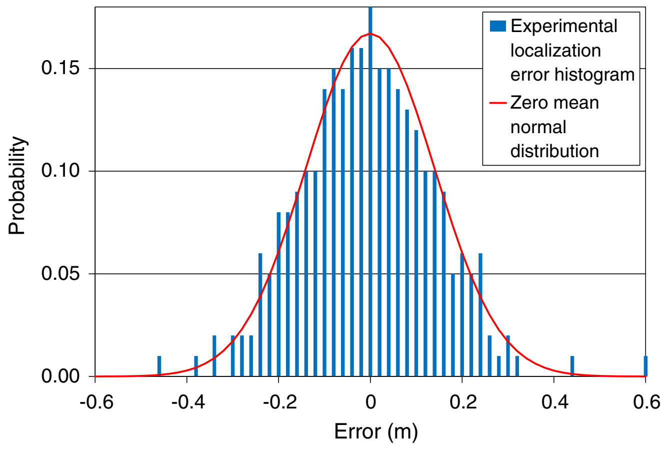


Fig. 8. The frequency histogram of the individual tracklet localization error v_k compared with the standard, zero-mean normal distribution with variance σ_z^2.

filter we tested provided no significant improvement in localization accuracy, even in curved sections of road, showing that the linear system assumption is appropriate for this application.

Another important consideration for automotive localization system is the database size. Many visual systems that use features have an impractically large visual database even after preprocessing. Although our feature tracklet database is of respectable compactness when compared to denser feature point methods [1,2], the ability to share a single descriptor over consecutively matched features makes a significant difference to the database size. The descriptor component of the database is reduced by at least 60%, which equates to an overall halving of the database size. The resulting database fits approximately 25 m into 1 MB of storage. While it would be possible to compress this database into a much smaller file size — in this implementation, the descriptor values are stored as raw text — even in its current format, downloading the database in real time at 100 km/h would use less bandwidth than streaming a standard 720p movie. This should be obtainable by modern mobile data networks.

In the experiments presented in this paper, the camera type and mounting system used for both query and database images was the same. While it would be desirable to use a variety of configurations to test robustness, this was not possible with the hardware available. However, because this method uses feature scale only and does not rely on accurate pixel position like pose estimation methods do, it should be robust to alternative mounting positions and uncalibrated cameras. Knowledge of the camera focal length should be sufficient to allow localization using a different camera from the one used in the database construction.

7. Conclusion

We proposed a method for visual ego-localization using local feature points within a pre-constructed database. Arranging matching features from consecutive database frames into tracklets allows the creation of individual tracklet position estimators by fitting feature scale to capture position with linear regression. The per-feature localization estimates are approximately normally distributed so give a convenient structure for inclusion in a Bayes estimator. By adding a constant velocity motion model, a 21% reduction in average localization error to 0.33 m and a 70% reduction in variance were observed. Grouping features into tracklets also allows averaging of descriptors within each tracklet, halving the size of the database.

Future work includes the dynamic update of the database during localization, so that localized query images can contribute to the ongoing maintenance of the database. One drawback of using feature points is their limited illumination invariance, so we would also like to test the use of databases which vary depending on the lighting conditions.

Acknowledgments

Parts of this research were supported by MEXT (17H00745), Grant-in-Aid for Scientific Research.

References

[1] H. Lategahn, C. Stiller, Vision-only localization, IEEE Trans. Intell. Transp. Syst. 15 (3) (June 2014) 1246–1257.
[2] X. Qu, B. Soheilian, N. Paparoditis, Vehicle localization using mono-camera and geo-referenced traffic signs, Proc. 2015 IEEE Intelligent Vehicles Symposium (IV2011), Seoul, Korea, June 2015. pp. 605–610.
[3] M. Milford, Visual route recognition with a handful of bits, Proc. 2012 Robotics: Science and Systems Conf., Sydney, Australia, July 2012. pp. 297–304.
[4] H. Uchiyama, D. Deguchi, T. Takahashi, I. Ide, H. Murase, Ego-localization using streetscape image sequences from in-vehicle cameras, Proc. 2009 IEEE Intelligent Vehicles Symposium (IV2009), Xi'an, China, June 2009. pp. 185–190.
[5] H. Badino, D.F. Huber, T. Kanade, Real-time topometric localization, Proc. 2012 IEEE Int. Conf. on Robotics and Automation (ICRA2012), St. Paul, MN, USA, May 2012. pp. 1635–1642.
[6] D. Xu, H. Badino, D.F. Huber, Topometric localization on a road network, Proc. IEEE/RSJ 2014 Int. Conf. on Intelligent Robots and Systems (IROS2014), Chicago, IL, USA, Sept. 2014. pp. 3448–3455.
[7] H. Kyutoku, T. Takahashi, Y. Mekada, I. Ide, H. Murase, On-road obstacle detection by comparing present and past in-vehicle camera images, Proc. 12th IAPR Conf. on Machine Vision Applications (MVA2011), Nara, Japan, June 2011. pp. 357–360.
[8] D. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (Nov. 2004) 91–110.
[9] D. Wong, D. Deguchi, I. Ide, H. Murase, Position interpolation using feature point scale for decimeter visual localization, Proc. 2015 IEEE Int. Conf. on Computer Vision (ICCV2015) Workshops, Santiago, Chile, Dec. 2015. pp. 1–8.
[10] H. Bay, A. Ess, T. Tuytelaars, L. Van Gool, Speeded-Up Robust Features (SURF), Comput. Vis. Image Underst. 110 (3) (June 2008) 346–359.
[11] M. Calonder, V. Lepetit, C. Strecha, P. Fua, BRIEF: Binary Robust Independent Elementary Features, Proc. 11th European Conf. on Computer Vision (ECCV2010), Part IV, Crete, Greece, Lecture Notes on Computer Science, vol. 6314, Sept. 2010. pp. 778–792.
[12] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, ORB: an efficient alternative to SIFT or SURF, Proc. 2011 IEEE Int. Conf. on Computer Vision (ICCV2011), Barcelona, Spain, Nov. 2011. pp. 2564–2571.
[13] S. Leutenegger, M. Chli, R.Y. Siegwart, BRISK: Binary Robust Invariant Scalable Keypoints, Proc. 2011 IEEE Int. Conf. on Computer Vision (ICCV2011), Barcelona, Spain, Nov. 2011. pp. 2548–2555.
[14] A. Alahi, R. Ortiz, P. Vandergheynst, FREAK: Fast Retina Keypoint, Proc. 2012 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR2012), Providence, RI, USA, June 2012. pp. 510–517.
[15] E. Rosten, T. Drummond, Machine learning for high-speed corner detection, Proc. 9th European Conf. on Computer Vision (ECCV2006), Part I, Graz, Austria, Lecture Notes on Computer Science, vol. 3951, May 2006. pp. 440–443.
[16] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, P. Sayd, Monocular vision based SLAM for mobile robots, Proc. 18th IAPR Int. Conf. on Pattern Recognition (ICPR2006), Hong Kong, China, vol. 3, Aug. 2006. pp. 1027–1031.
[17] T. Botterill, S. Mills, R.D. Green, Bag-of-words-driven, single-camera simultaneous localization and mapping, J. Field Rob. 28 (2) (March 2011) 204–226.
[18] G. Grisetti, R. Kümmerle, C. Stachniss, W. Burgard, A tutorial on graph-based SLAM, IEEE Intell. Transp. Syst. Mag. 2 (4) (Winter 2010) 31–43.
[19] M. Cummins, P. Newman, Appearance-only SLAM at large scale with FAB-MAP 2.0, Int. J. Rob. Res. 30 (9) (Aug. 2011) 1–24.
[20] G. Sibley, C. Mei, I. Reid, P. Newman, Vast-scale outdoor navigation using adaptive relative bundle adjustment, Int. J. Rob. Res. 34 (14) (July 2010) 1688–1710.
[21] H.F. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: part I, IEEE Robot. Autom. Mag. 13 (2) (June 2006) 99–110.
[22] M.A. Fischler, R.C. Bolles, Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography, Comm. ACM 24 (6) (June 1981) 381–395.
[23] G. López-Nicolás, C. Sagüés, J.J. Guerrero, Parking with the essential matrix without short baseline degeneracies, Proc. 2009 IEEE Int. Conf. on Robotics and Automation (ICRA2009), Kobe, Japan, May 2009. pp. 1098–1103.

[24] M. Muller, Dynamic time warping, Information Retrieval for Music and Motion, Springer, Berlin Heidelberg, Sept. 2007, pp. 69–84.
[25] D. Wong, D. Deguchi, I. Ide, H. Murase, Single camera vehicle localization using feature scale tracklets, IEICE Trans. Fundam. Electron. Commun. Comput. Sci. E100-A (2) (February 2017) 702–713.
[26] M. Muja, D.G. Lowe, Fast approximate nearest neighbors with automatic algorithm configuration, Proc. 4th Int. Conf. on Computer Vision Theory and Applications (VISAPP2009), Lisboa, Portugal, vol. 1, Feb. 2009. pp. 331–340.
[27] G. Bradski, The OpenCV Library, Dr. Dobb's J. Softw. Tools 25 (11) (2000) 120–126. http://opencv.org/.
[28] D. Wong, D. Deguchi, I. Ide, H. Murase, Vision-based vehicle localization using a visual street map with embedded SURF scale, Computer Vision — ECCV 2014 Workshops Proc., Part I, Zurich, Switzerland, Lecture Notes on Computer Science, vol. 8925, Sept. 2014. pp. 167–179.
[29] L. Marchetti, G. Grisetti, L. Iocchi, A comparative analysis of particle filter based localization methods, RoboCup 2006: Robot Soccer World Cup X, Bremen, Germany, Lecture Notes on Computer Science, vol. 4434, Springer, Berlin Heidelberg, June 2006, pp. 442–449.
[30] E.A. Wan, R.V.D. Merwe, The unscented Kalman filter for nonlinear estimation, Proc. IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (AS-SPCC2000), Oct. 2000. pp. 153–158.