International Journal of Recent Engineering Science (IJRES),
ISSN: 2349-7157, Volume 4 Issue 6 November to December 2017

World Scale Estimation Monocular Visual Odometry with Ground Feature

Xiang Liu, Zuolei Sun*,Weijie Chen
(Machine Perception and Interaction Group (MPIG),
College of Information Engineering, Shanghai Maritime University, Shanghai, 201306, China)

Abstract

Scaleambiguity is the major challenge in the application of monocular visual odometry (MVO). A sound approach to MVO world scale estimation is proposed in this paper. Firstly, the Speed Up Robust Feature (SURF) is employed to extract features and KLT (Kanade-Lucas-Tomasi) optical flow functions as the visual feature matcher between the consecutive images. Then RANSAC is used to refine the feature matching to reduce the mismatches introduced by noise, the procedure can improve the accuracy of fundamental matrix. In order to get good ground feature which is used to overcome the scale ambiguity, the region of interest (ROI) where the ground in the images is selected and adaptively adjusted when the camera view is changed. The performance of the proposed approach is demonstrated with the widely used KITTI benchmark and compared with the classical MVO algorithm.

Keywords: *Accurate scale, Image processing, Motion estimation, Visual odometry*

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM), is a key skill for autonomous mobile system. The goal of SLAM is the joint estimation of both the robot's pose and a model of its surrounding environment. And that can be summarized as three issues: Where am I? Where am I going? How can I get there?

Just as people look at the surroundings with their eyes, robot's eyes are sensors. Most of the early works use a laser rangefinder as the main sensor. More recently, visual sensors have become the dominant choice (i.e. camera). Obviously, it called visual SLAM. Visual odometry (VO) [1] is the front-end technology of visual SLAM. VO is good at localization according to images from cameras.

The main distinction we make when discussing visual odometry systems is that between stereo and monocular systems. Although monocular visual odometry (MVO) systems are hard pressed to compete with stereo systems due to the difficulties imposed by a single camera setting, a single camera is much cheaper and easier to use than stereo cameras. MVO significant improvements in the last few years.

One of monocular challenges is scale estimation: we can't know the scale of true word from only images input. It should be supplemented by other ways to determine the scale such as add another sensor like inertial measurement unit (i.e. IMU), finding something's size is fixed (i.e. road signs) or some parameters have been known (i.e. height of camera). In this paper, we will introduce an implementation of MVO, use the height of camera and ground feature to estimate the scale, and experimentally confirm the overall performance of the method.

II. RELATED WORK

Visual odometry calculates the camera's position changes by many frames continuous images from different moments. VO was presented by Matthies et al in 1980 in [2], and they established a system framework contains feature extraction, feature matching or tracking and motion estimation. And the main steps to achieve MVO nowadays are similar to this in [3]: image correction, feature extraction, feature matching or tracking, motion estimation and determine the scale finally.

A camera is a mapping between 3D word and 2D image. In [4], it introduced how a camera works and the image will have a certain distortion because of camera is always not perfect. Before we do visual odometry, calibrating the camera and getting its' intrinsic parameters first, which called calibration matrix K.

About image processing, there are various algorithms of extracting feature points such as Harris, SIFT and SURF [5], [6], [7], and each have advantages and disadvantages. Then we can match point using rich feature descriptors or scene flow. Compared to the traditional rich feature matching like Fast Library for Approximate Nearest Neighbors (FLANN) in [9], using different approaches for selecting proper scene flow are more popular in recent years. For example, NOTF in [20] comparing and discussing different 3D-flows and choose the best one to estimate the ego-motion that leads to a high precision result. Optical flow among scene flow is the most widely used especially the KLT [10] to track feature. Feature matching will be wrong sometimes, and many methods were found to solve that like Random Sample Consensus (RANSAC) to distinguish inliers and outliers, and others in [9], [11],

[12]. The newest famous ROCC [21] and its upgrade version RotROCC [22] ranked #10 and #7 separately in Kitti, and the two algorithms are just based on optical flow and RANSAC. The great improvement in ROCC is that they propose an optical flow dependent feature-adaptive scaling and distinguish inliers from outliers and then they normalized reprojection error in RotROCC to get a better result. In this paper, we just use normal optical flow based KLT and RANSAC to reject outliers.

About motion estimation, it introduction what's essential matrix E and fundamental matrix F, how they work and how to compute in [4], [13]. We use matched points to get them and usually estimate the ego-motion by decomposing them to get rotation and translation together. However, SOFT in [23], which ranked #6 in Kitti proposed in 2015, estimating rotation by the five point method and using the three point method for translation estimation separately. Only slightly less obvious is the fact that the overall scale of the scene cannot be determined when only using a single camera. A lot of ideas to make sure the scale in [14], [15], [16], [24]. Combining visual sensor and another sensor will have highest precision and robust, because it works well even though temporary lack of visual features and can play sensors respective advantages. The V-LOAM combining visual and lidar in [24] ranked #1 in Kitti, which starts with visual odometry and is improved by lidar odometry. Although it's attractive, the cost is too high to achieve for us. The idea that fixed height of camera in [14] is more suitable with our robot model, and we compute the scale using the height and ground feature with ROI in this paper.

III. FEATURE PROCESSING

In this section, we describe the feature processing part of the algorithm. The results of this part are image coordinates of a subset of points that will be used in motion estimation.

A. Feature Extraction

SURF [6] is an accelerated version of Scale Invariant Feature Transform (SIFT) [7]. The feature extracted by SURF has scale invariance and rotation invariance. The SURF uses the local maxima of Hessian matrix determinants to locate the feature points. In paper [6] the Gaussian second order differential filtering model is simplified so that we get approximate solution of Hessian matrix:

$$\mathrm{Det}(H_{approx}) = D_{xx}D_{yy} - (0.9D_{xy})^2 \quad (1)$$

D is the approximate of Gaussian second order partial conduction and point convolution, and it can be easily calculated using the integral image. Changing the size of the filtered template to obtain the scale space. And then we do non-maximum suppression (NMS). A point is compared with 26 fields around the current scale, the previous scale, and the next scale. It will be a feature point when it's the max or the min of 26 points around.

In order to make the feature points have rotation invariance, we should find the direction of feature. And in paper [6], it statistics harr wavelet feature around the feature point with the range of 60°in a circle of radius 6s (s is the scale of the feature point), and make the max value as the main direction.

When we know the main direction, get 20*20 points around the feature, and divided into 4*4 areas each has 5*5 points, just like the Fig 1 shows. For each area, we compute every points' Harr wavelet response in x and y direction, and summation them like the Fig 1. As a result, every feature descriptor has 4*4*4=64 dimensional vector.

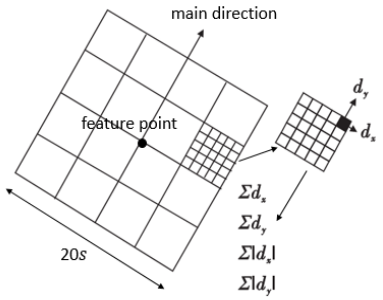

Fig 1. Feature Descriptor

B. Feature Tracking

Optical flow is be used in this paper to match points at two frames of images. And using KLT [10] algorithm to achieve it. Optical flow is the process of matching selected points from one image to another, assuming both images are part of a sequence and relatively close to one another [8]. There are three premise assumptions of KLT:

1) Brightness constant

2) Time continuous or motion displacement is small

3) Spatial consistency, adjacent points have similar movements

The input of MVO are continuous images from a moving camera. So images fully consistent with the three prerequisites, and it is very applicable. The goal of KLT is that tracking a point we know in an image and find where the point is in the next frame image. And the way is computing movement, just the optical flow, of the point between two images. According to the brightness constant assumption and the third assumption, the same point and its local neighborhood in two images has similar grayscale value. Definition of the residual function:

$$\varepsilon(\mathrm{d}) = \varepsilon(d_x, d_y) = \sum_{x=u_x-w_x}^{u_x+w_x} \sum_{y=u_y-w_y}^{u_y+w_y} (I(x,y) - J(x+d_x, y+d_y))^2 \quad (2)$$

I and J mean the grayscale value of two image. The point we know is u and w is the window size. And optical flow is d in this formula. When we get the minimum value, the grayscale is the most similar in other words, we just think we tracking this point.

KLT deal with small pixel displacement according to the second assumption. Solution for this is a pyramidal implementation like figure. The original image is Layer 0 and every up one layer the image size halved, and the same to coordinate of u. The window size w is constant for all pyramid. The structure of the image pyramid is shown in Fig 2.

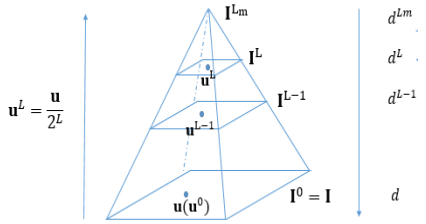

Fig 2. Imagepyramid

With the image being smaller, the pixel displacement being smaller, too. Computing optical flow dlm in the highest layer first. However, optical flow dlm-1 in next layer isn't simply double because the error will also be double. We need get a new dlm-1 on the basis of the previous layer dlm, and do this until the original image. The residual function only have one minimum value, so that when its derivative value is 0, the d is we want and we can easily tracking the point.

Feature tracking will be wrong sometimes, that will affect the follow analysis. Using RANSAC to reject mismatches [11], so that we can use right matches to compute a better result. RANSAC is a mathematical model to get rid of noise and bad data, and it makes data be beater. It is based on the following principle: In each iteration, a minimum number of random samples is taken from the correspondences to create a motion hypothesis. Then, a score for each feature is calculated that describes whether it supports the hypothesis. If the motion estimate reaches a predefined support of the features, the non-supporting features are marked as outliers. Otherwise, a new random sample is drawn and the next iteration starts.

Obviously, we will get lots of fundamental matrix F by matched points (the next section). Some will be good that from right matching. However some F will be bad because of wrong matching and noise. RANSAC can filter out good F, which means right matching, as inliers and get rid of the bad. And finally, computing new F by inliers.

IV. MOTION ESTIMATION

Motion estimation is split into two parts. Firstly, the rotation and translation are estimated without scale, and secondly, the scale is estimated by ground feature and ROI.

A. Motion without Scale

The core to calculate camera's pose is epipolar geometry [4]. From the matching points of the preceding and succeeding frames we can calculate essential matrix E or fundamental matrix F, whose parameters contains information of rotation R and translation t between two frames. And we can get R and t by matrix decomposition. The relationship between E and F is:

$$\mathrm{E} = K'^{T} F K (3)$$

It means that we can get fundamental matrix directly from the original images but hard to estimate motion accurately because of camera distortion. And essential matrix is calibrated and the estimate rotation and translation exactly. As a result, we usually get F first and then compute E with K and finally decompose it to estimate motion.

Normalization is necessary before computing. Just like RotROCC normalized reprojection error in [22], we normalized feature points here. In this way coordinate dependency can be almost completely removed. And it leads to more accurate results. The feature point is scaled to the average distance from point $x=(u, v, w)^T$ to the origin is equal to $\sqrt{2}$.

According to the epipolar geometry, we know theformula:

$$x'^{T} F x = 0 \quad (4)$$

The matched feature points (x, x') are known, and what we want is parameters of E. If parameters we unknown write as a vector f, and coordinates of x and x' write as a matrix A, as a result we can write (4) to

$$Af = 0 \ (5)$$

We can see it's a general homogeneous linear equations. Because of F is 3×3, the vector f has 9-dimensional. It can be solution by least squares when there have at least 8 groups of matched feature points [17]. After using RANSAC we transform the F with K to E, and the E can be decomposition to R and t by SVD, but four possible solutions are obtained. Linear triangulation will help us to find the right one [4].

B. Scale Estimation

MVO using one camera, unlike binocular visual odometry, so that we can't know the scale of true word from images only. It should be supplemented by other ways to determine the scale like methods are described in Chapter 2. In this paper, we suppose the ground is flat and the camera height is fixed, whose value is h*, like Fig 3. We also assume the camera is parallel to the ground and the ground feature point is $\mathbf{X}_i = (X_i, Y_i, Z_i)^T A$ after reconstruction [14].

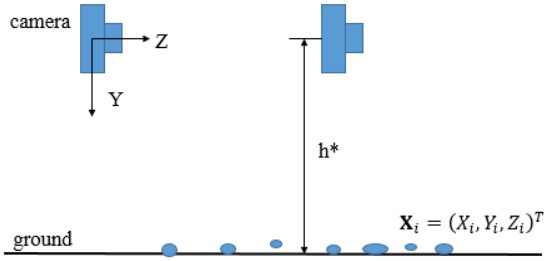

Fig 3. Camera and Ground Feature Model

The Gaussian kernel weight is used to calculate the vertical distance $\bar{h}$ from the reconstructed ground feature to the camera. The formula is shown as (6). The (7) is kernel function and the variance is given by (8).

$$\bar{h} = \frac{argmax\sum k(Y_i - h_i)}{ji} \quad (6)$$

$$\mathrm{k}(x) = \exp\left(\frac{-x^2}{2\sigma_h^2}\right) \quad (7)$$

$$\sigma_h = \frac{1}{50} med\|\bar{\mathbf{X}}_\mathbf{i}\|_1 \quad (8)$$

This method is improved on the basis of viso-mono, libviso2, and it's effective when all the features come from ground. However, Gaussian kernel function will have deviation if the features come from others, such as building and trees. So we should identify the features, and select the ground features.

As we all know, the road (or street) is always in substantially the same region of the images when the camera is parallel to the ground. And its' gray values are close to each other and different from the rest in the image generally. Selecting ROI of images, a region we need where the road must will be, as feature area to extract ground features. An example is given in Fig 4.The image comes from KITTI [18], sequence 0, image_0, and number 000000.png, whose resolution is 1241×376. The area (u, v) is be chosen as ROI, whose u ∈ [200,800] and v ∈ [300,376). Circles are ground feature as a result shown in the fig 4.

Fig 4 Ground Features

The vertical distance from ground to camera is always larger than others. Once a feature point is not ground feature is extracted, its' distance value will smaller than ground feature. We change the kernel function to classification, like (9). We can adaptive amplification of ROI when the number of feature is too small and ROI is constrained by this formula. In this way, we can always get enough and right ground feature to compute the scale.

$$\mathrm{k}(x) = \begin{cases} exp\left(\frac{-x^2}{2\sigma_h^2}\right) others \\ exp\left(\frac{-x^2}{2(0.1\sigma_h)^2}\right) x > 0 \end{cases} \quad (9)$$

Finally, we solution $\bar{h}$ by (6). Multiply the scale to the translation t, and then the scale of image to the actual can be determined.

$$\mathrm{S} = \frac{h^*}{\bar{h}} (10)$$

Obviously, the road is not flat completely. The more bumps the road is, the greater error of the scale will be. In other words, we can know the conditions of road we through according to the translation error to some extent by this method. And the method need to be further improved.

V. EXPERIMENTAL RESULTS

All the image data in this paper are from KITTI, and the code is based on openCV2.4.9, libviso2, MATLAB 2015b and Visual Studio 2015.

The data set from KITTI is captured by two high-resolution color and grayscale cameras fixed on an off-road vehicle on rural roads and motorway in a medium-sized city called Karlsruhe. It provides camera intrinsic parameters K, real motion trajectory measured by GPS/oxts and a translation and rotation error evaluation code [19].

In this paper, the first 2000 images of image_0 of sequence 00, sequence 05 and sequence 08 from KITTI were tested for mapping path. And error evaluation using the whole dataset of each sequence. The calibration matrix K of sequence 00 is

$$\mathrm{K} = \begin{pmatrix} 718.856 & 0 & 607.1928 \\ 0 & 718.858 & 185.2517 \\ 0 & 0 & 1 \end{pmatrix}$$

And the location test result is shown in Fig 5.

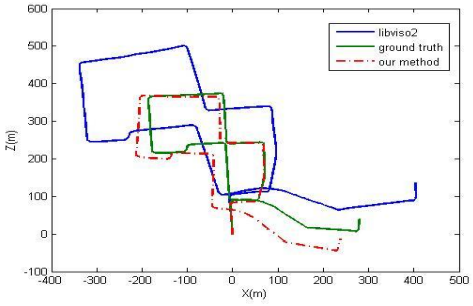

Fig 5. Location Test Result of Sequence 00

The calibration matrix K of sequence 05 and 08 is

$$\mathrm{K} = \begin{pmatrix} 707.0912 & 0 & 601.8873 \\ 0 & 707.0912 & 185.1104 \\ 0 & 0 & 1 \end{pmatrix}$$

And the location test results are shown in Fig 6.

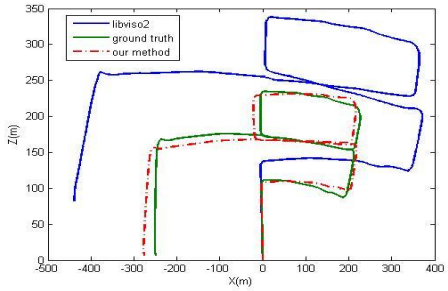

Fig 6.(A) Location Test Result of Sequence 05

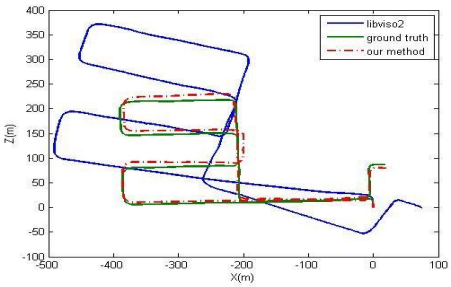

Fig 6.(B) Location Test Result of Sequence 08

Comparing with figure 5 and figure 6, we can see that our method is much better than libviso2. The results are more matching with the truth and scale estimates accurately. Especially two sequences in Fig 5, there're better when dealing with sequence 05 and 08 in Fig 6 than sequence 00 in Fig 5. The causes of this maybe the parameter of camera which leads to Fig 6 is the same one and different from the camera of sequence 00. And another posable reason is that the road of sequence 00 is much worse than other two. It is truth that when we play the images of sequence 00 one by one, the road is bumpy and the camera lens always up and down. However, we assume the road is flat and the shaking let it's hard to compute a right scale so that it leads to a bad result.

Evaluation of relative translation error and relative rotation error by Kitti evaluation code are shown in Table 1.

Table 1.Evaluation of libviso2 and our Method

	sequences	00	05	08
libviso2	Translation error	0.327345	0.352596	0.349086
	Rotation error	0.00049	0.000721	0.000404
Our method	Translation error	0.064519	0.105512	0.107859
	Rotation error	0.000316	0.000367	0.000272

The error data of evaluation results in this table are from stats.txt with each sequence without conversion after running the evaluation code from Kitti. The errors are shown of sequence 00 in the table is statistical value for 4540 images (all images of sequence), and the same to sequence 05 is 2760 images and sequence 08 is 4070 images. It's no doubt that our method's errors are smaller than libviso2's. Especially the translation error of libviso2 are more than three times than our method's error. And our rotation error is nearly half of libviso2's. It's surprising that sequence 00 is the best one according to the evaluation results in Table 1, different from the path we plot in Fig 5. Maybe we use only 2000 frames to show and the rotation error really effects path mapping. The rotation error of 08 is the smallest and we see the path in Fig 6 b matches better with ground truth than others.

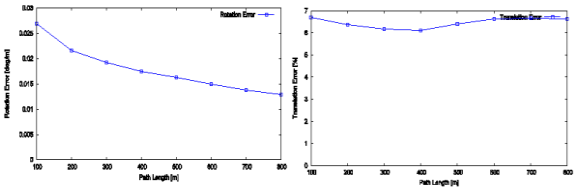

Fig7 Rotation Error and Translation Error of Sequence 00

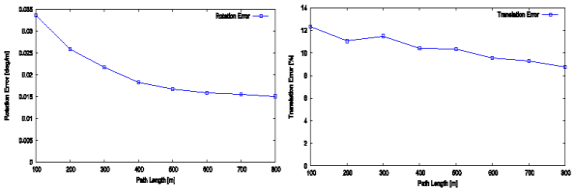

Fig8 Rotation Error and Translation Error of Sequence 05

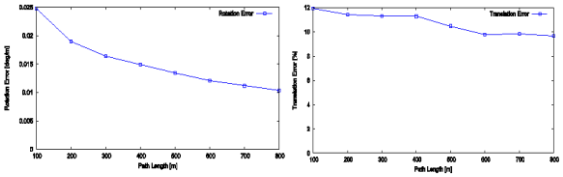

Fig9 Rotation Error and Translation Error of Sequence 08

The evaluation results of errors change with path length are shown in Fig 7, Fig 8 and Fig 9. Rotation errors are conversion to deg/m and we can see that rotation errors are always less than 0.03 deg/m and the error decreases with the path length increasing. In addition, translation errors is average of nearly 10% about 05 and 08, and translation error of 00 is smaller with an average nearly 6.5%. It's quite a good result of MVO, whose scale is really a challenge.

VI. CONCLUSION

In this paper, We have implemented a scale - based monocular visual odometry. However, there are also many to do for improvement. Our method haven't use any filtering like bundle adjustment or Kalman filter. Two defects of our method are regrettable which we will solve next. One is the code running slowly and it is not suitable of real-time. We will try to change code structure or other algorithm to improve the speed. The other is not robust enough. When we test the code, we found that it works well with a small dataset, but it sometimes almost not work leads to high error. And this method is depend on the flat ground assumption which is quite limited by road conditions.

REFERENCES

[1] Howard A. Real-time stereo visual odometry for autonomous ground vehicles, F, 2008 [C]. IEEE.3946-3952.
[2] Matthies L, Shafer SA. Error modeling in stereo navigation [J]. Robotics and Automation, IEEE Journal of, 1987, 3(3): 239-248.
[3] Scaramuzza D, Fraundorfer F. Visual odometry [tutorial] [J]. Robotics & Automation Magazine, IEEE, 2011, 18(4): 80-92.
[4] Hartley R, Zisserman A. Multiple view geometry in computer vision [M]. Cambridge university press, 2003.
[5] Harris C, Stephens M. A combined corner and edge detector; proceedings of the Alvey vision conference, F, 1988 [C]. Citeseer.50.
[6] Bay H, Tuytelaars T, Van Gool L. Surf: Speeded up robust features [M]. Computer vision–ECCV 2006. Springer. 2006: 404-417.
[7] Ng PC, Henikoff S. SIFT: Predicting amino acid changes that affect protein function [J]. Nucleic acids research, 2003, 31(13): 3812-3814.
[8] Daniel Lélis Baggio, Emami S. et al. Mastering OpenCV with Practical Computer Vision Projects [M]. Packt Publishing Ltd, 2012.
[9] Muja M, Lowe D G. Fast Approximate Nearest Neighbors with Automatic Algorithm Configuration[J]. VISAPP (1), 2009, 2: 331-340.
[10] Lucas BD, Kanade T. An iterative image registration technique with an application to stereo vision; proceedings of the IJCAI, F, 1981 [C].674-679.
[11] Scaramuzza D, Fraundorfer F, Siegwart R. Real-time monocular visual odometry for on-road vehicles with 1-point ransac; proceedings of the Robotics and Automation, 2009 ICRA'09 IEEE International Conference on, F, 2009 [C]. IEEE.4293-4299.
[12] Sami Brandt.Maximum Likelihood Robust Regression with Known and Unknown Residual Models, Statistical Methods in Video Processing Workshop,2008
[13] Richard Szeliski. Computer Vision:Algorithms and Applications.Springer-Verlag New York, Inc. , 2010 , 21 (8) :2601-2605
[14] Kitt BM, Rehder J, Chambers AD, et al. Monocular visual odometry using a planar road model to solve scale ambiguity [J]. 2011.
[15] Davison AJ, Reid ID, Molton ND, et al. MonoSLAM: Real-time single camera SLAM [J]. Pattern Analysis and Machine Intelligence, IEEE Transactions on, 2007, 29(6): 1052-1067.
[16] Klein G, Murray D. Parallel tracking and mapping for small AR workspaces, F, 2007 [C]. IEEE.225-234.
[17] Hartley RI. ªIn Defense of the Eight-Point Algorithm [J]. IEEE Trans Pattern Analysis and Machine Intelligence, 1997, 19(6): 580-593.
[18] Geiger A, Lenz P, Stiller C, et al. Vision meets robotics: The KITTI dataset [J]. The International Journal of Robotics Research, 2013, 0278-3649.
[19] Geiger A, Lenz P, Urtasun R. Are we ready for autonomous driving? the kitti vision benchmark suite; proceedings of the Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on, F, 2012 [C]. IEEE.3354-3361.
[20] J Deigmoeller, J Eggert.Stereo Visual Odometry Without Temporal Filtering. Springer International Publishing , 2016
[21] M Buczko, V Willert. How to Distinguish Inliers from Outliers in Visual Odometry for High-speed Automotive Applications.Intelligent Vehicles Symposium , 2016
[22] M Buczko, VWillert..Flow-Decoupled Normalized Reprojection Error for Visual Odometry. IEEE International Conference on Intelligent Transportation Systems, 2016
[23] I Cvišić, I Petrović. Stereo odometry based on careful feature selection and tracking. European Conference on Mobile Robots , 2015 :1-6
[24] J Zhang, S Singh.Visual-lidar Odometry and Mapping: Low-drift, Robust. IEEE International Conference on Robotics & Automation , 2015 , 2015 :2174-2181