

Depth estimation with ego-motion assisted monocular camera

Mostafa Mansour^{1,2}, Pavel Davidson¹, Oleg Stepanov², Jukka-Pekka Raunio³, Mohammad M. Aref⁴, and Robert Piché¹

¹Faculty of Information Technology and Communication Sciences, Tampere University, Finland

²Department of Information and Navigation Systems, ITMO University, St. Petersburg, Russia

³AAC Technologies, Tampere, Finland

⁴Faculty of Engineering and Natural Sciences, Tampere University, Finland

Abstract—We propose a method to estimate the distance to objects based on the complementary nature of monocular image sequences and camera kinematic parameters. The fusion of camera measurements with the kinematics parameters that are measured by an IMU and an odometer is performed using an extended Kalman filter. Results of field experiments with a wheeled robot corroborated the results of the simulation study in terms of accuracy of depth estimation. The performance of the approach in depth estimation is strongly affected by the mutual observer and feature point geometry, measurement accuracy of the observer’s motion parameters and distance covered by the observer. It was found that under favorable conditions the error in distance estimation can be as small as 1% of the distance to a feature point. This approach can be used to estimate distance to objects located hundreds of meters away from the camera.

Index Terms—computer vision, depth-from-motion, extended Kalman filter, image sequence, inertial sensing, sensor fusion

I. INTRODUCTION

The image sequence acquired by a moving monocular camera in a static environment contains detailed information about both the motion of the camera and the shape of the environment. This phenomenon, called the motion parallax effect, uses corresponding image points in multiple views [1]. Given the image of a 3D point in two or more views it can be reconstructed by triangulation. The process of the reconstruction of a 3D scene from 2D images using the parallax effect, known as structure from motion (SfM), combines point correspondence information from multiple points over multiple frames to solve for scene structure and camera motion. Closely related to SfM is also relative position and depth estimation using monocular vision.

SfM approaches that are based solely on images are sensitive to outliers in correspondence estimates and they are usually not robust enough for many practical applications. The difficulties arise from ambiguities in local photometric information due to illumination changes, occlusions, specularities, repetitive structures and objects that move independently of the scene [2]. The knowledge of camera motion can help to improve performance of the SfM algorithms in the presence of outliers. Fusion of visual information and measurements of camera motion by inertial sensors is becoming common practice in applications like augmented and virtual reality. In

these applications the fusion of visual and inertial information is used to estimate the camera’s 3D angular and translational motion [3]–[5].

Movement of the object’s projection on the image plane is uniquely related to the camera’s kinematic parameters (angular and translational velocities). The equations that relate the coordinates and velocity of a feature point’s projection with the camera’s translational and angular velocities were derived by Longuet-Higgins et al. [6]. They showed that from a monocular view of a rigid, textured surface it is possible to determine the motion of the eye relative to it from the velocity field of the changing retinal image. They also found that the depth computed based on motion parallax is specified completely by the retinal velocity. Corke showed in [7] that information about camera motion from the inertial sensors can also be used to predict feature position and significantly reduce the computational burden.

Approaches for depth estimation from a sequence of monocular images with known camera motion were proposed in [3], [4]. The authors use a pixel-based algorithm that estimates depth and depth uncertainty at each pixel and incrementally refines these estimates over time. The algorithm processes an image sequence taken with small inter-frame displacement and produces an on-line estimate of depth. The feature-based Kalman filter algorithm was restricted to known lateral camera motion. The prediction model in the Kalman filter used the current depth map and an estimate of the camera motion to predict a depth map for the next image in the sequence. In [8], [9] the authors present a batch algorithm and a recursive iterated EKF for estimating sensor motion and scene structure by considering all of the measurements from a camera, gyro, and accelerometer simultaneously. In these works, in addition to conventional projective cameras, omnidirectional cameras and wide angle lenses were used.

In [10] Corke et al. proposed loosely and tightly coupled frameworks for concurrent estimation of structure and motion based on camera and inertial sensors. In the loosely coupled approach, the separate inertial sensing and SfM blocks are running at different rates and exchanging information. In the tightly coupled approach a single high-order EKF combines the disparate raw data of vision and inertial sensors in a single, optimum filter, rather than cascading two filters, one for each

sensor. The state vector comprises the kinematic variables that describe the camera motion and the 3D coordinates of N scene points. Additionally the state vector may be augmented by unknown camera intrinsic parameters and inertial sensor bias and scale parameters. The state vector can include more than 20 states, which has implications for computational load and for tuning. The combined IMU and vision sensing strategy is robust to vision drop-outs and is able to determine relative position with minimal requirements on the vision system.

Visual-inertial SfM and depth estimation approaches can be categorized according to their applications. The instrumentation always includes an IMU and, optionally, it can include an odometer for wheeled robots, or a doppler velocity log for underwater robots. The applications comprise UAVs [11], [12], underwater robots [13]–[15], visual servoing for robotic arm manipulators [16]–[19] and wheeled robots [20], [21]. When the visual-inertial SfM approach is applied to UAVs or underwater robots it has to deal with the presence of disturbances and uncertain dynamics.

Visual-inertial SfM can be also used to acquire depth map in real time for augmented reality (AR) applications [22]. One of the primary technical challenges facing all AR systems is to accurately localize and track displays moving relative to content in the physical world. In these applications the depth is usually recovered through computer-vision techniques that are based on binocular disparity and shading visual cues. However, it is difficult to make robust real-time implementation of these SfM techniques and therefore, in some cases, laser rangefinders are used. Another problem is that binocular disparity and shading visual cues work only within short range to the features. To overcome these difficulties robust visual-inertial odometry can be used instead [23].

Some researchers have conducted field experiments to evaluate the accuracy of the depth estimation of their approaches and some other important characteristics like time of convergence to the true depth. Dani et al. [15] designed a reduced-order nonlinear observer for depth estimation that was implemented on an underwater robot. They reported a convergence time of 25 sec in field tests. De Luca et al. [20] tested their approach on a wheeled robot with point features located several meters away from the camera; the convergence time was about 30 sec. Grabe et al. [11] reported 25 sec convergence time for depth estimation with their EKF-based estimator that used optical flow. The target was the floor plane located only 1.5 m below a UAV's camera. Matthies et al. [24] used a Kalman filter with known camera poses to estimate the depth to an object using optical flow. They reported a convergence time of 10 seconds.

Motion parallax can be combined with other depth cues such as stereovision, kinetic depth effect, shading and occlusion to obtain a more stable estimate of viewing distance. Landy et al. [25] proposed to fuse motion parallax information with stereo disparity by minimizing the inconsistency between depth from disparity and depth from motion parallax.

This work is based on our previous paper [21] that provided an algorithm and simulation study for the depth-from-motion problem. The theoretical results from [21] are now confirmed by real-life experiments with a wheeled robot equipped with

a monocular camera, odometer and inertial sensors. The approach has the ability to determine relative position by tracking just one feature point. Moreover, many feature points can be tracked simultaneously using the parallel implementation of the same algorithm.

This paper is organized as follows. Section II describes a mathematical model that relates image measurements with the depth and camera kinematic parameters. The solution algorithm is presented in Section III. Section IV presents the implementation details, the results of simulation study and field test results. Finally, Section V summarizes our approach for monocular depth estimation with measured camera motion and gives recommendations for future research in this area.

II. SYSTEM DESCRIPTION AND MODELING

A. Geometry of camera and feature point

Consider a non-holonomic mobile robot equipped with an odometer, an inertial measurement unit (IMU), and a monocular camera. The robot moves in a 2D plane and its camera is detecting a fixed feature point located at $P^N = [X^N, Y^N, Z^N]^T$ in an earth-fixed navigation frame $\{N\}$. For simplicity, we assume that the robot base coordinate frame $\{r\}$ and the IMU coordinate frame are aligned with the x^r axis, the forward motion direction of the robot base, z^r axis directed upward, and y^r axis completing a right hand coordinate frame. The camera is installed such that its $\{C\}$ coordinate frame's z^c axis directed along its optical axis which is in the direction of the robot x^r axis, its y^c axis is directed downward, and its x^c axis completing a right-hand coordinate frame. The rotation matrix from the robot coordinate frame to camera coordinate frame is R_r^c .

The geometry of the feature point and the camera in the navigation frame is described in Fig. 1. From the geometry, as explained in [13], the feature point can be described as

$$P^N(t) = q^N(t) + R_c^N P^c(t), \quad (1)$$

where R_c^N is the rotation matrix from the camera coordinate frame $\{C\}$ to the navigation frame $\{N\}$, P^c is a feature point vector resolved in the camera coordinate frame $\{C\}$ and q^N is the camera frame origin resolved in the navigation frame $\{N\}$. The kinematics of this system can be obtained by differentiation of (1), yielding

$$\dot{P}^N(t) = \dot{q}^N(t) + \dot{R}_c^N P^c(t) + R_c^N \dot{P}^c(t), \quad (2)$$

$$\dot{R}_c^N = R_c^N [\Omega^c(t)]_\times, \quad (3)$$

where

$$[\Omega^c(t)]_\times \triangleq \begin{bmatrix} 0 & -\omega_z^c(t) & \omega_y^c(t) \\ \omega_z^c(t) & 0 & -\omega_x^c(t) \\ -\omega_y^c(t) & \omega_x^c(t) & 0 \end{bmatrix} \quad (4)$$

is the skew-symmetric matrix of the camera angular velocity vector $\Omega^c(t) = [\omega_x^c(t), \omega_y^c(t), \omega_z^c(t)]^T$. Taking into account the fact that $\dot{P}^N = 0$ and substituting (3) into (2) we can write

$$\dot{P}^c = -R_N^c \dot{q}^N(t) - [\Omega^c(t)]_\times P^c. \quad (5)$$

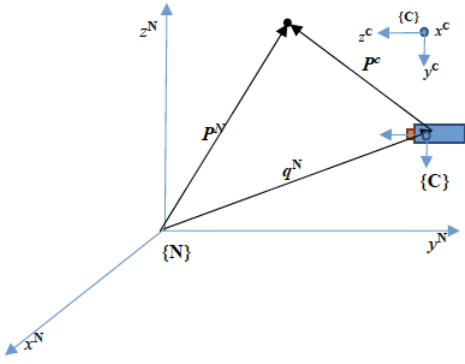


Fig. 1. The camera and the feature point geometry in the navigation frame.

The left-hand side of (5) is the feature point kinematics resolved in the camera coordinate frame, i.e. $\dot{P}^c = [\dot{X}^c, \dot{Y}^c, \dot{Z}^c]^T$, while the first term in the right-hand side represents the camera velocity vector resolved in the camera coordinate frame, i.e.

$$R_N^c \dot{q}^N(t) = [V_x^c(t), V_y^c(t), V_z^c(t)]^T. \quad (6)$$

For writing simplicity, we drop the time dependency argument. Substituting parameters with their values into (5) and rearranging the terms gives

$$\dot{X}^c = -V_x^c + \omega_z^c Y^c - \omega_y^c Z^c, \quad (7a)$$

$$\dot{Y}^c = -V_y^c - \omega_z^c X^c + \omega_x^c Z^c, \quad (7b)$$

$$\dot{Z}^c = -V_z^c + \omega_y^c X^c - \omega_x^c Y^c. \quad (7c)$$

Given that a non-holonomic 2D motion is considered, the rotation will be only around the y^c axis of the camera and the velocity vector will have only a component in the direction of the motion vector. According to the relative position of the camera in the robot coordinate frame, the motion vector will be only along the z^c axis of the camera. In other words, $\omega_z^c = \omega_x^c = 0$ and $V_x^c = V_y^c = 0$. By applying these constraints on the model in (7), we obtain the following model of the kinematics of the feature point vector resolved in the camera coordinate frame:

$$\dot{X}^c = -\omega_y^c Z^c, \quad (8a)$$

$$\dot{Y}^c = 0, \quad (8b)$$

$$\dot{Z}^c = -V_z^c + \omega_y^c X^c. \quad (8c)$$

From (8), one can notice that the feature point Y^c coordinate does not change since the motion of the robot is only in 2D.

1) *Pin-hole camera model and image formation:* In this paper we used a monocular camera that can be described by a pin-hole camera model [1], [26]. In this model a 3D feature point in the camera coordinate frame is projected to a 2D image point on the camera image plane (uov) (Fig. 2). According to the properties of the perspective projection in a pin-hole camera [1], [26], image point coordinates can be

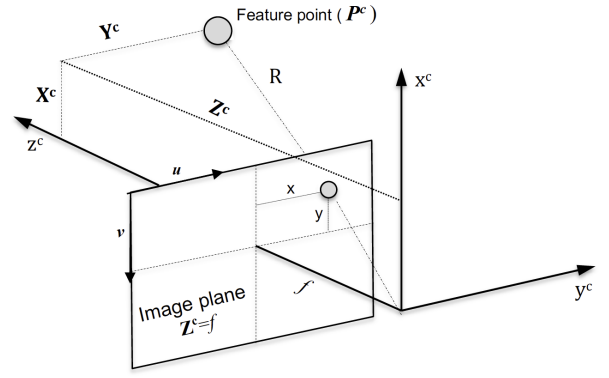


Fig. 2. Perspective projection in a pin-hole camera.

calculated as

$$x = f \frac{X^c}{Z^c} + c_x, \quad (9a)$$

$$y = f \frac{Y^c}{Z^c} + c_y, \quad (9b)$$

where (x, y) are the image point coordinates, f is the camera focal length and (c_x, c_y) are principal point coordinates, all measured in pixels. Rearranging the terms in (9a) yields

$$X^c = (x - c_x) \frac{Z^c}{f}. \quad (10)$$

Substituting (10) into (8c) we obtain

$$\dot{Z}^c = -V_z^c + \omega_y^c (x - c_x) \frac{Z^c}{f}. \quad (11)$$

The focal length and the principal point are camera intrinsic parameters. They should be estimated before using the camera, as described later (see IV-B3). The kinematics of an image point can be obtained by differentiating (9), yielding

$$\dot{x} = \frac{Z^c \dot{X}^c - X^c \dot{Z}^c}{(Z^c)^2 / f}, \quad (12a)$$

$$\dot{y} = \frac{Z^c \dot{Y}^c - Y^c \dot{Z}^c}{(Z^c)^2 / f}. \quad (12b)$$

To cope with high degree of non-linearity in the state dynamics the range coordinate Z^c in (8), (11) and (12) is represented with its reciprocal $\xi = \frac{1}{Z^c}$ [13]. Montiel et al. [27] showed that this explicit parametrization of the reciprocal depth allows a Gaussian distribution to faithfully represent uncertainty in depth for a depth range from nearby to infinity. Substituting (8) into (12) and using (11) we obtain

$$\dot{x} = - \left(f + \frac{x - c_x}{f} \right) \omega_y^c + V_z^c \xi (x - c_x), \quad (13a)$$

$$\dot{y} = V_z^c \xi (y - c_y) - \frac{\omega_y^c (x - c_x) (y - c_y)}{f}, \quad (13b)$$

$$\dot{\xi} = V_z^c \xi^2 - \frac{\omega_y^c \xi (x - c_x)}{f}. \quad (13c)$$

The system of equations (13) can be interpreted as a shaping filter whose outputs are the image point coordinates and the

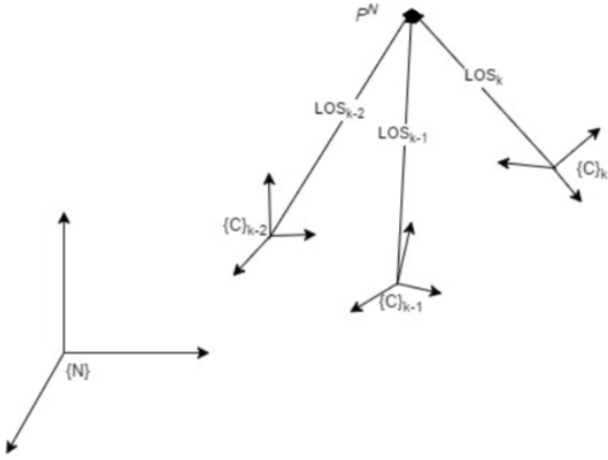


Fig. 3. Generating data for simulation studies

reciprocal of the depth while its inputs are camera angular and linear velocities.

III. PROBLEM STATEMENT AND SOLUTION ALGORITHM

This section presents an improvement to the approach described by Davidson et al. [21], [28], [29] for the depth estimation from a sequence of images while tracking a feature point. In [21], [28] the authors presented a simpler normalized version of the model in (13) where $f = 1$ and $(c_x, c_y) = (0, 0)$. In this paper, a complete camera model is considered and the intrinsic parameters of the camera, i.e. f and (c_x, c_y) are used in model (13). These parameters will be given in simulation studies and will be determined using camera calibration procedure in field test implementation (see IV-B3).

A. Problem statement

The Line-of-Sight (LOS), i.e. the direction to the feature point, can be measured by a camera. This LOS changes according to the camera motion as illustrated in Fig. 3. The changing in the LOS represents the kinematics of a feature point and can be described in (7).

The kinematic relation between the feature point and its projection on the camera image plane was described using the system of equations (13) where the camera linear and angular velocities are used. The velocities can be measured in the coordinate frame of the robot using an odometer and an IMU then transformed to the camera coordinate frame using the rotation matrix R_r^c . However, the measured velocities include error components. So, the true velocities can be represented as

$$\begin{aligned}\omega_y^c &= \tilde{\omega}_y^c - \Delta\tilde{\omega}_y^c, \\ V_z^c &= \tilde{V}_z^c - \Delta\tilde{V}_z^c.\end{aligned}\quad (14)$$

where \tilde{V}_z^c and $\tilde{\omega}_y^c$ are noisy measured camera linear and angular velocities and $\Delta\tilde{V}_z^c$ and $\Delta\tilde{\omega}_y^c$ are corresponding measurement errors. In this paper, these errors are considered to be independent white noises with autocorrelation functions

$E(\Delta\tilde{V}_{y_t}^c \Delta\tilde{V}_{y_\tau}^c) = \sigma_V^2 \delta(t-\tau)$ and $E(\Delta\tilde{\omega}_{y_t}^c \Delta\tilde{\omega}_{y_\tau}^c) = \sigma_\omega^2 \delta(t-\tau)$, respectively. Substituting (14) into (13) yields

$$\begin{aligned}\dot{x} &= -\left(f + \frac{x - c_x^2}{f}\right) (\tilde{\omega}_y^c - \Delta\tilde{\omega}_y^c) + (\tilde{V}_z^c - \Delta\tilde{V}_z^c) \xi(x - c_x) \\ \dot{y} &= (\tilde{V}_z^c - \Delta\tilde{V}_z^c) \xi(y - c_y) - \frac{(\tilde{\omega}_y^c - \Delta\tilde{\omega}_y^c)(x - c_x)(y - c_y)}{f} \\ \dot{\xi} &= (\tilde{V}_z^c - \Delta\tilde{V}_z^c) \xi^2 - \frac{(\tilde{\omega}_y^c - \Delta\tilde{\omega}_y^c) \xi(x - c_x)}{f}.\end{aligned}\quad (15)$$

In summary, the process model (15) can be formulated as

$$\dot{X} = \mathbf{S}(X, \tilde{V}_z^c, \tilde{\omega}_y^c, q), \quad (16)$$

where $X = [x, y, \xi]^T$ is the state vector and $q = [\Delta\tilde{\omega}_y^c, \Delta\tilde{V}_z^c]$ is the vector of process noise components that are modeled as zero-mean white noises with power spectral density matrix

$$Q_c = \begin{bmatrix} \sigma_\omega^2 & 0 \\ 0 & \sigma_V^2 \end{bmatrix}. \quad (17)$$

The state vector in (16) is initialized as a normally distributed multivariate random variable $X_0 \sim N(\hat{X}_0, P_0)$. The initial guess can be obtained by triangulation, for example, using two (or some few) consecutive camera measurements for coarse distance estimation. The measurements can be described as

$$\mathbf{z}_k = \begin{bmatrix} x \\ y \end{bmatrix}_k + v_k, \quad (18)$$

where v_k is the measurement error at time step k . It is modeled as a white noise with a covariance matrix R . The measurement model (18) can be rewritten as

$$\mathbf{z}_k = HX_k + v_k, \quad (19)$$

where X_k is the state vector evaluated at time step k and the measurement matrix is

$$H = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \quad (20)$$

The problem under investigation can be summarized as estimation of the depth when its reciprocal is described by the continuous-time shaping filter (16), using the sequence of camera measurements modeled by (18).

B. Solution algorithm

To solve the problem under investigation numerically, the model in (16) should be discretized. It can be written in a discrete form as

$$X_k = X_{k-1} + \int_{t_{k-1}}^{t_k} S(X_{k-1}, \tilde{V}_{z_{k-1}}^c, \tilde{\omega}_{y_{k-1}}^c) dt + G(X_{k-1})q_{k-1} \quad (21)$$

where q_{k-1} is the vector of discretized measurement errors that are considered to be zero-mean white Gaussian noise. The integral in (21) can be estimated using different numerical integration methods, e.g. Euler or Runge-Kutta methods. In

this paper the fourth order Runge-Kutta method is used. Equation (21) can be written as

$$X_k = \Psi \left(X_{k-1}, \tilde{V}_{z_{k-1}}^c, \tilde{\omega}_{y_{k-1}}^c, q_{k-1} \right). \quad (22)$$

The problem under investigation is a nonlinear problem because the process model is nonlinear. Hence, posterior distribution and the state vector can be estimated recursively using a Bayesian filter. In this paper, EKF will be considered for solving the problem under investigation.

C. Extended Kalman Filter

EKF is an approximate version of the optimal filter for nonlinear filtering problems [30]. The prediction and update steps of the (first order) EKF are [31]:

Prediction:

$$X_k^- = \Psi \left(\hat{X}_{k-1}, \tilde{V}_{z_{k-1}}^c, \tilde{\omega}_{y_{k-1}}^c, 0 \right), \quad (23)$$

$$P_k^- = A_{k-1} P_{k-1} A_{k-1}^T + G_{k-1} \frac{Q_c}{\Delta t} G_{k-1}^T. \quad (24)$$

Update:

$$K_k = P_k^- H (H P_k^- H^T + R)^{-1}, \quad (25)$$

$$\hat{X}_k = X_k^- + K_k (z_k - H X_k^-), \quad (26)$$

$$P_k = (I - K_k H) P_k^-, \quad (27)$$

where

$$A_{k-1} = \frac{\partial}{\partial X} \Psi \left(X_{k-1}, \tilde{V}_{z_{k-1}}^c, \tilde{\omega}_{y_{k-1}}^c, q_{k-1} \right)_{X=\hat{X}_{k-1}, q_{k-1}=0}, \quad (28)$$

$$G_{k-1} = \frac{\partial}{\partial q} \Psi \left(X_{k-1}, \tilde{V}_{z_{k-1}}^c, \tilde{\omega}_{y_{k-1}}^c, q_{k-1} \right)_{X=\hat{X}_{k-1}, q_{k-1}=0}. \quad (29)$$

Here, the nonlinear model (22) is used to predict the state according to (23). Then, the model is linearized to obtain matrices A_{k-1} and G_{k-1} (see Appendix for the details), which are then used to obtain the prediction error covariance matrix in (24). The measurement update (25)-(27) is the same as in a standard Kalman filter.

IV. IMPLEMENTATION AND RESULTS

This section presents the simulation studies and field test for the algorithm introduced in the previous section. First, simulation studies for the influence of different motion parameters on estimation accuracy are presented in subsection IV-A. Then, the results of the field test for the proposed algorithm are presented and discussed in IV-B.

A. Simulation study

The proposed algorithm is tested using simulated data in order to get insights into how the algorithm works. Also, it will help us to study the effect of different parameters, such as feature point coordinates and linear velocity, on depth estimation accuracy. This subsection is organized as follows. First, a block diagram for robot's motion and measurement modeling is introduced along with the mathematical equations describing the behavior of each item in the block diagram. Then, the results of the simulation studies are presented and discussed.

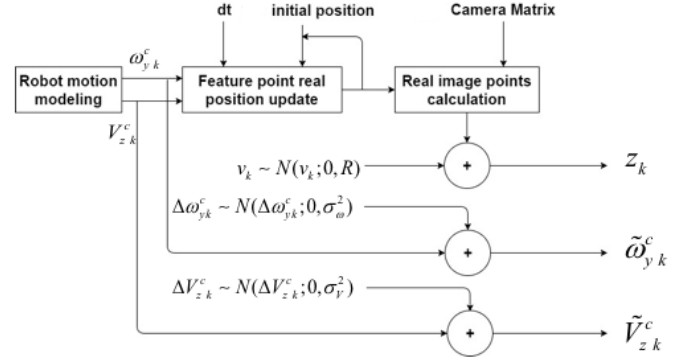


Fig. 4. Generating data for simulation studies

1) *Motion and measurement modeling:* Fig. 4 gives insights into how the data are generated for simulation. There are three main sub-blocks: a camera motion modeling block, a feature point position update block and a measurement modeling block. These blocks are described in the following lines.

Motion modeling: In this sub-block, true values for angular and linear velocities of the camera are generated. These velocities are then disturbed by white noises to model the noisy signals obtained from the IMU and the odometer.

Feature point coordinates update: Due to camera motion, the feature point coordinates, resolved in the camera coordinate frame $\{C\}$, changed at every time step. By approximate time-integration of (8), the updated coordinates of the feature point can be obtained as

$$\begin{aligned} X_k^c &= X_{k-1}^c - \tilde{\omega}_{y_{k-1}}^c Z_{k-1}^c \Delta t, \\ Y_k^c &= Y_{k-1}^c, \\ Z_k^c &= Z_{k-1}^c - \tilde{V}_{z_{k-1}}^c \Delta t + \tilde{\omega}_{y_{k-1}}^c X_{k-1}^c \Delta t. \end{aligned} \quad (30)$$

Image point calculation: The projection of the updated feature point, i.e. the updated image point, can be computed using (9). The image point is then disturbed by a white noise to model noisy camera measurements.

2) *The results of simulation study:* The modeled motion and measurement parameters are fed to EKF algorithm to estimate the state vector X recursively as explained in the previous section (see III-B and III-C). In Fig. 5 a block diagram of the EKF implementation is presented. The rest of this section is organized as follows. The influence of the feature point location, related to the focus of expansion, on depth estimation accuracy is presented and discussed. Then, the effect of the camera linear velocity on depth estimation accuracy is investigated. Finally, the scalability of the problem under investigation is considered by introducing a dimensionless scalability parameter. This parameter describes the expected estimation error as a function of the difference in the apparent position of a feature point viewed along two different lines of sight for the same observation scenario regardless of camera linear velocity. The parameters used in this simulation study are presented in Table I.

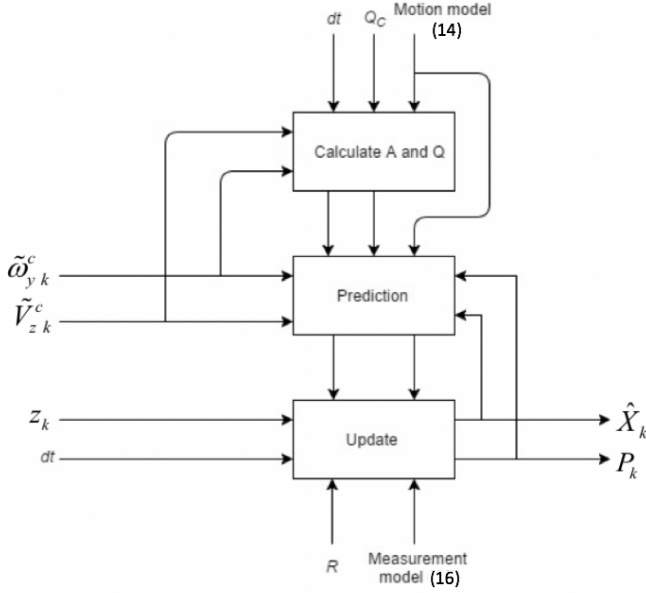


Fig. 5. EKF implementation

TABLE I

THE KALMAN FILTER PARAMETERS FOR THE SIMULATION STUDY.

time step size (dt)	0.1 s
observation time (T)	3 s
σ_ω	0.001 rad/s $1/\sqrt{H_z}$
σ_V	0.01 m/s $1/\sqrt{H_z}$
P_0	$\text{diag} [10 \text{ pixel}^2, 10 \text{ pixel}^2, 9 \text{ m}^{-2}]$

Feature point location (camera trajectory): The focus of expansion is a point at which there is no optical flow. This point is located on the camera optical axis. If a feature point is located at, or near to, the focus of expansion, it will be projected at, or near to, the camera principal point and therefore there will be little or no disparity between image points in successive frames. Therefore, it is difficult to accurately estimate the distance to a feature point that is located near the focus of expansion. In this study, a Monte-Carlo simulation of the depth estimation of four feature points with 1000 realizations of the measurement error using the parameters in Table I and a robot linear velocity equal to 0.5 m/s, i.e, $V_z^c = 0.5$ m/s. Initial error in distance estimation is set to 3 m. The feature points are located at different distances from the optical axis, i.e. from the focus of expansion.

Fig. 6 shows how the accuracy of distance estimation depends on the mutual geometry of a feature point and a camera. From the figure it can be seen that the filter converges faster for points that are located farther from the focus of expansion. This fact could be taken into account during the practical implementation of the proposed algorithm. That is, the camera trajectory, i.e. robot motion, could be planned in such a way that keeps the feature point away from the focus of expansion. By doing so, better estimation accuracy and faster convergence of the filter can be achieved. Also, it should be noted that it is a scalable problem. It means that the farther the

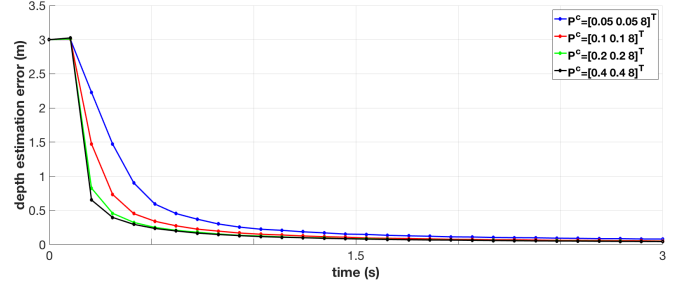


Fig. 6. Influence of the feature point distance from the focus of expansion on depth estimation accuracy. Monte-Carlo simulations (1000 realizations) for different feature points.

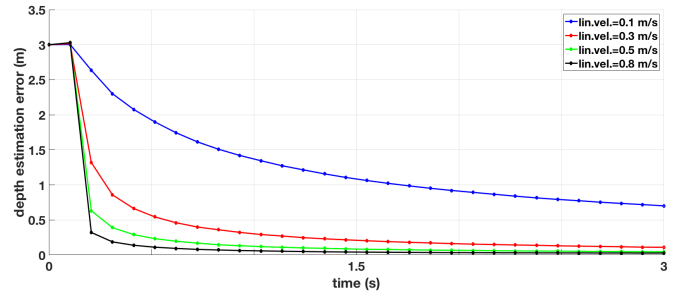


Fig. 7. Influence of the linear velocity of the camera on the depth estimation accuracy. Monte-Carlo simulations (1000 realizations) for different linear velocities.

feature point is from the camera, the farther its image point needs to be from the focus of expansion. This is because the disparity between image points in successive frames for far feature points is smaller than the disparity for the near ones.

3) *Linear velocity:* To investigate how the camera linear velocity affects the depth estimation accuracy, four different values for a constant camera linear velocity are tested: 0.1, 0.3, 0.5, and 0.8 m/s. The parameters used in this simulation study are presented in Table I. The feature point is located at $P^c = [0.4 \ 0.4 \ 8]^T$.

In Fig. 7 shows the mean depth estimation errors from 1000 Monte Carlo simulations with different measurement realizations. We can see that the greater the linear velocity, the faster the filter converges. The explanation is that when the camera moves faster the disparity between the image points in successive frames will be larger and hence the triangulation angle will be wider.

4) *Problem scalability:* The results illustrated in Fig. 7 are scalable. Depth estimation depends not only on the camera linear velocity but also on how far the feature point is. This scalability feature can be captured using the dimensionless scalability parameter $L = VT/D_0$ that was used in [32], [33], where V is the camera's speed averaged over the observation time interval, T is the length of the observation time interval and D_0 is the initial range to the feature point, i.e. the Euclidian norm of the feature point initial position vector resolved in the camera coordinate frame $\{C\}$ [21], [28].

The scalability parameter L describes the ratio between the covered distance and the initial range to the feature point. It represents the ability of the camera to create a difference in the apparent position of a feature point viewed along two different

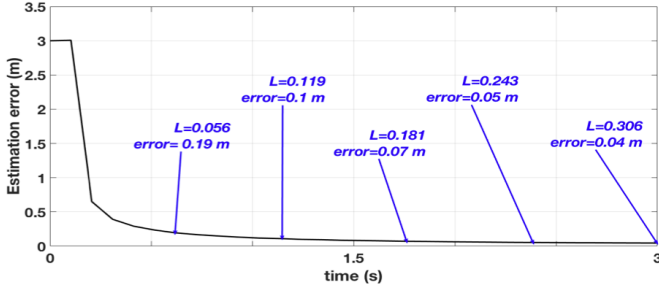


Fig. 8. Influence of the length of the observation time interval on the depth estimation accuracy.

TABLE II

THE RELATION BETWEEN THE RELATIVE ERROR IN DEPTH ESTIMATION AND THE SCALABILITY PARAMETER L USING SIMULATED DATA.

L	0.05	0.12	0.18	0.24	0.31
Relative error	3.2%	2.1%	1.7%	1.5%	1.35%

lines of sight. It should be noted that this parameter does not take into account different camera trajectories (different observation scenarios). Thus for the same L , the depth estimation accuracy will be different for different trajectories. At the same time, for the same camera trajectory, depth estimation accuracy will be the same for the same values of L regardless of the value of the camera linear velocity.

Fig. 8 shows how the error in depth estimation depends on the dimensionless scalability parameter L for a moving camera with a linear velocity 0.5 m/s detecting a feature point initially located at $P^c = [0.4 \ 0.4 \ 8]^T$. It helps to understand what potential accuracy can be achieved if the scale (dimensions) of the viewing scene are different from those that was used in the presented experiments.

The relative error in depth estimation can be defined as the percentage ratio of the error in depth estimation at any time step and the true depth at the same time step. Table II shows the relative error for the values of scalability parameter L illustrated in Fig. 8. We can notice that the relative error decreases with the increase in the value of L .

B. Field test

This subsection presents the field test results for the proposed algorithm. First, the implementation details about the used components are presented. Then, the experiment setup and the results are introduced and discussed.

1) *Implementation details:* The Robotnik TurtleBot2 robot (Fig. 9) is used. The robot's Kobuki base includes a calibrated IMU and a calibrated odometer connected to the wheels. The base is connected to a Linux (Ubuntu 14.4) portable computer and power is provided by a battery installed in the lower part of the base. Three Point Grey Grasshopper3 cameras are installed on the robot and connected to the computer. The middle camera only is used in this experiment. A Robot Operating System Indigo (ROS Indigo) is running on the computer. ROS drivers for the sensors (the IMU, the odometer,

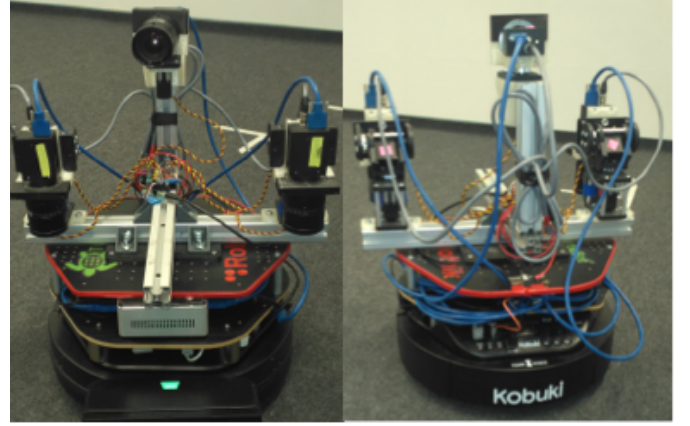


Fig. 9. Front and rear views of the TurtleBot2 robot. Only the central camera is used in the experiment.

and the camera) are installed on the robot computer. When the robot begins to function, written ROS *nodes* allow the sensors to transmit time stamped ROS *messages* that are collected remotely using a Linux user computer connected to the robot computer via a wireless Local Area Network (LAN). The collected messages are saved in one ROS *bag file* and available for offline post processing.

2) *Experiment setup:* The robot is driven manually to a feature point in a preplanned trajectory. The feature point is a chessboard fixed on a wall in the laboratory. The robot is driven by a remote user transmitting the input signals on the corresponding ROS topics over a wireless LAN. While driving the robot, the IMU and the odometer measure the angular and the linear velocity of the robot. At the same time, the camera captures images of the feature point at 5 frames/second. The ROS time stamped messages from the IMU, the odometer, and the camera are sent to the user computer and saved in a ROS bag file. Then the signals are extracted from these messages, synchronized and processed.

3) *Preprocessing and camera calibration:* The measured angular and linear velocities are transformed from the robot coordinate frame to the camera coordinate frame using the rotation matrix R_r^c . The transformed velocities are stored with their corresponding time stamps. Each frame is processed to detect the feature point.

The camera's intrinsic matrix is calibrated using different images of a chessboard and the Camera Calibration Toolbox in Matlab [34]. The Matlab Camera Calibration Toolbox's function *detectCheckerboardPoints* is used to detect the chessboard. The function is based on the algorithm introduced by Geiger et al. [35]. The algorithm automatically extracts corners to sub-pixel accuracy and combines them to (rectangular) checkerboards or chessboard-like patterns. We are interested in the image point of the chessboard's upper-left corner. The image points of the chessboard origin are stored with the corresponding time stamps of their frames.

4) *Feature points detection and tracking:* The image captured with camera describes the intensity of light reflected from the surface of objects. The intensity can rapidly change in the image for example because of the color changes or

the orientation changes of the object. A point where intensity changes in two directions is called a corner in image. There exist a large variety of corner detection algorithms and in this work the algorithm described in [36] was applied. This approach was described in our previous work [21].

Feature points can be found from each image frame separately. However, the location of feature points differs from one frame to another and therefore the features between the consecutive frames must be matched by tracking them over the image sequence. In our experiments we used the Kanade-Lucas-Tomasi (KLT) feature-tracking algorithm, an iterative method with image pyramids [37], [38]. The KLT tracker detects a number of image points the corners in the checkerboard. However, we need only one image point at the time and track it in the consecutive frames. It is worth to mention that significant disparity between the image points in two consecutive frames should be guaranteed for the algorithm to converge. We found from our experiments that the algorithm does not converge well if the disparity is less than 10 pixels.

5) *Synchronization and processing*: The velocities and the image points are sorted in ascending order according to their time stamps. Then, the pipeline of the proposed algorithm is launched by assigning appropriate initial values for the state vector \hat{X} and the covariance matrix P . The initial time value is assigned equal to the time stamp of the first signal. Then, the sorted signals are processed, in order, at every step as follows. First, the time step size is computed by taking the difference between the current time stamp and the previous one. Then, the signal is processed according to its type.

For example, if the signal is an angular velocity, the state is propagated using only the angular velocity using (30) by setting ω_y^c equal to the value of the processed signal and \tilde{V}_z^c equal to zero. The propagation is similar with the linear velocity signal. It is processed by setting \tilde{V}_z^c equal to the value of the processed signal and ω_y^c equal to zero. The measurement update is carried out only if the processed signal is a camera image point. The proposed algorithm was tested on a computer with 3.5 GHz intel core i7 processor. The required time for EKF updating step using one feature is around 0.06 ms.

6) *Experiment results*: The experiment is performed by driving the robot forward to the feature point, stopping it, then driving it backward away from the feature point. This motion is repeated to imitate the way in which a heavy fork-lift robot may be moved when carrying different objects from a place to another in a warehouse or factory. This motion can be described as a repeated low-speed motion with low dynamics and without sharp turns. The ground truth depth was obtained using the *perspective-n-points* algorithm described in [39] and known size of the checkerboard.

In the experiment, the robot is at rest for about 1.5 s, then begins a series of back and forth motions. The angular and linear velocities captured by the IMU and the odometer are shown in Fig. 10. The results of offline application of the proposed depth estimation algorithm on real data are presented in Fig. 11. We see that after the robot starts to move, the filter needed 1.5 to 2 seconds (about 10 frames) to converge. This is approximately the same amount of time the filter needed to converge in the simulation studies (see

TABLE III
THE RELATION BETWEEN THE RELATIVE ERROR IN DEPTH ESTIMATION AND THE SCALABILITY PARAMETER L USING REAL DATA.

L	0.042	0.085	0.127	0.174	0.23
Relative error	9.05%	2.14%	0.92%	0.26%	0.17%

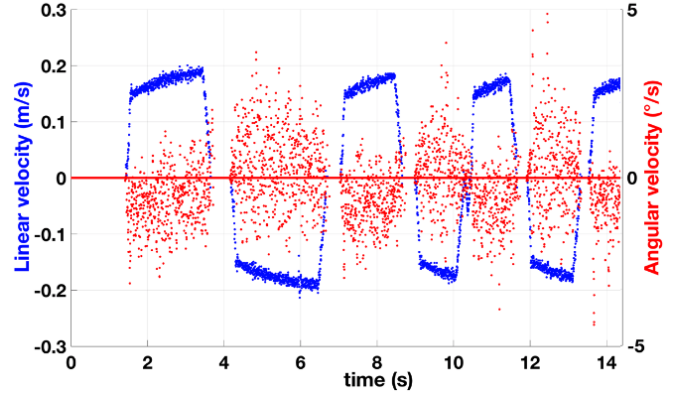


Fig. 10. Discrete received linear (blue) and angular (red) velocities of the camera.

Fig. 8). The scalability parameter L was computed for the forward motion in the first 20 seconds. From Fig. 11, we can notice that the error in depth estimation decreases with the increase in the value of L . The same result was obtained in simulation studies (see Fig. 8). The relation between the relative error and L parameter is described in table III. We can notice that the relative error decreases with the increase in the value of L . This result is expected from simulation studies (see Table II).

Fig.12 shows the actual and computed errors in the inverse depth (ξ) estimation. The actual error is computed as difference between the reference and estimated values of the inverse depth. The computed error is a square root of the diagonal element of the covariance matrix estimated by the EKF (σ_ξ). From Fig.12, we can assure filter consistency as the real estimation error is located inside the error interval predicted by EKF.

V. CONCLUSIONS AND FUTURE WORK

In this paper, a depth estimation approach using a monocular camera is proposed. The approach is based on using EKF for fusing the information extracted from a sequence of camera images, i.e. image points, with the angular and linear velocities of the camera measured by an IMU and an odometer respectively. The novel contribution of this work is a practical approach to the 6 DOF depth-from-motion problem that can be implemented on a mobile robot equipped with a standard camera, and a mathematical model to predict depth accuracy for different observation scenarios. The approach is suitable for real-time implementation and it can be used to estimate distance to objects located hundreds of meters away. Our

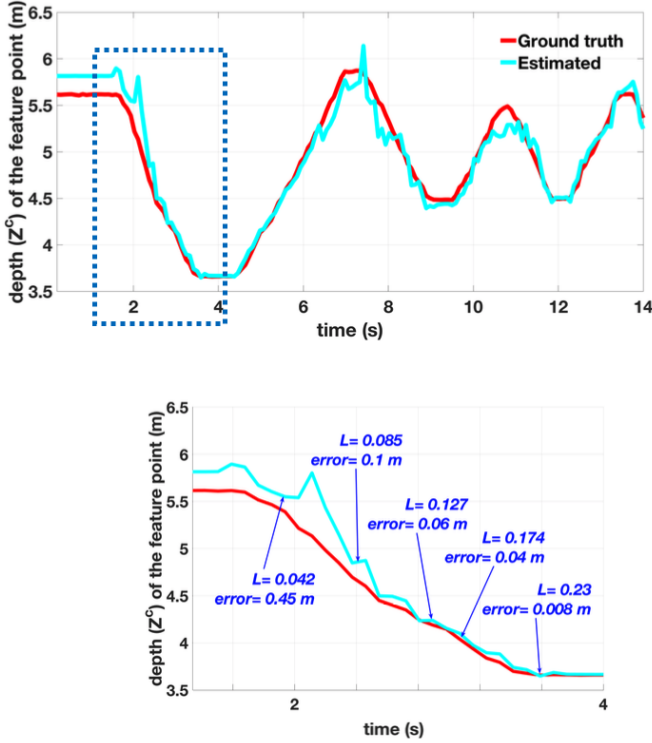


Fig. 11. Depth estimation using real data. The lower part emphasizes filter convergence during the first 4 seconds

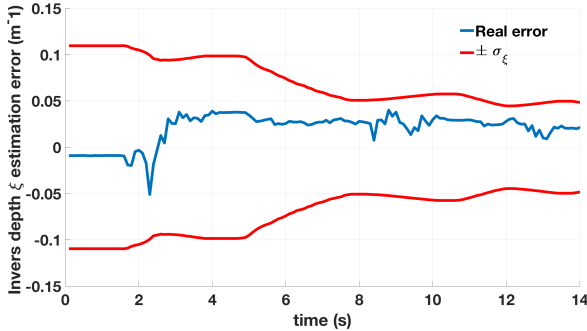


Fig. 12. Actual and computed errors in the inverse depth (ξ) estimation.

approach is based on a probabilistic model that takes into account the uncertainties in the camera's motion. The approach is robust to brief line of sight obstructions and to errors in the measurement of the camera's ego motion. The convergence time of the depth estimation to the true value is several times shorter compared to state-of-the-art solutions [12]. We also introduce a mathematical model for assessing the accuracy of the depth estimation algorithm for different observation scenarios that can include various distances between a camera and points and different camera velocities. In the following lines, some issues related to the robustness and the area of the implementation of this approach are addressed.

A. Estimation accuracy and convergence time

Accuracy of the depth estimation depends, mainly, on the mutual geometry between the camera and feature point. If the

camera motion creates significantly different viewpoints of an object, the distance to this object can be estimated accurately and the errors in distance estimation can be as low as 1% of the distance. The accuracy also depends on the measurement errors in translational and angular velocities. However, in our previous study [21] we showed that for wheeled vehicles equipped with an odometer and reasonably good MEMS IMU the influence of these errors on depth estimation accuracy is small.

The convergence time depends on the initial guess in depth estimation. In the examples that are shown in Fig.6 and Fig.7, the state vector was initialized with 3 m depth error. Reducing the initial uncertainty will help the filter to converge faster. In real-life scenarios the initial guess is usually obtained using simple triangulation. In this case the error in initial depth estimation does not exceed 50% of the distance to the object. The proposed algorithm can cope with this initial uncertainty well.

B. Robustness of the approach

Concerning the robustness of the approach, three important factors should be considered. **The first factor** is the location of the feature point related to the focus of expansion. As presented in IV-A2, the approach converges faster for the points that are far from the focus of expansion. Therefore, to get a faster convergence, the trajectory should be planned in such a way that allows the feature point to be located away from the focus of expansion.

The second factor is the type of the camera motion. Since this is a triangulation-based approach, translation motion helps the depth estimation: the faster the camera moves, the bigger the change in the LOS, the wider the triangulation angle, the better the accuracy. This was seen in the results of simulation studies (see IV-A3). At the same time, rotation does not help the approach. On the contrary, it will add an ambiguity. In rotational motion, there is no changing in the LOS, hence, there is not any triangulation angle and the filter will not be able to estimate the depth properly.

The third factor is the initial state of the filter. EKF uses a linearized version of the motion model. The initial state is the first point of linearization. Choosing the point of linearization is very important, otherwise the filter may diverge. There are, at least, three different ways to choose the initial state for the proposed approach. *The first way* is to consider the geometry of the space if the robot moves in an enclosed space. By knowing the dimensions of the space and the initial position of the robot, we can, approximately, guess the initial depth of the feature point. *The second way* is using an additional feature point with known position as an auxiliary point. By making a triangulation between the two points and the camera, the depth can be estimated and used as an initial value in the EKF.

The third way is getting two different images for the feature point from two different positions and consider the camera in the two position as a stereo pair. Using an IMU and an odometer, the rotation and the transition of the camera between the two positions can be calculated. With known relative geometry of the camera in two position, an approximated value

for the depth can be estimated. In this paper the first and the second way were used because the feature point was a chessboard with known geometry among its corners and the experiment was conducted in an indoor laboratory.

C. Area of implementation

The proposed approach can be used for any implementation that requires estimating the distance to a feature point under two conditions. The first one is that the motion should be translation-dominant motion without high maneuvering (sharp turns). The second condition is that the feature point should not be too close to the focus of expansion of the camera. A fork-lift robot in warehouses or in factories may be a good implementation site for the proposed approach. Another important possible application for the proposed approach is a depth map. By taking more than one feature point (around 500 or 1000) and estimating the depth for each of them, a depth map can be obtained. Depth maps are usually obtained using stereo cameras, but this proposed approach may open the door towards using a monocular camera to get a depth map.

D. Future work

Our future work will address a comparison between the depth estimation using stereo vision and the proposed approach. The proposed approach might be improved if other non-linear filters, such as Unscented Kalman filter or particle filter are used to solve the depth from motion problem.

Taking advantage of additional features, when available, should provide significant benefits, like improved accuracy and increased robustness. This multi-feature cooperative approach can provide additional sensor measurements without adding additional hardware. It also adds flexibility to the sensing strategy, whereby a feature that is traveling out of the field of view of the camera can be augmented by a more appropriate feature before the first feature is lost.

In this work the detected feature points are not associated with any specific object and the current goal is to obtain an image where each chosen feature point contains information about the depth or distance to the camera. In our future work we are planning to develop approaches for vanishing points identification and data association. The latter can help to identify specific objects including moving ones and build a map of the environment. The vanishing points correspond to distant landmarks and therefore their projections to the image plane are not moving unless the camera is rotating.

ACKNOWLEDGMENT

The authors thank the staff of the TUT Mobile Robotics Lab. for their help in arranging the field experiments. This work was partially supported by the Russian Foundation for Basic Research, project no. 18-08-01101?, and by the Government of the Russian Federation (grant 08-08).

REFERENCES

- [1] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.
- [2] O. Özyeşil, V. Voroninski, R. Basri, and A. Singer, "A survey of structure from motion," *Acta Numerica*, vol. 26, pp. 305–364, 2017.
- [3] L. Matthies, R. Szeliski, and T. Kanade, "Incremental estimation of dense depth maps from image sequences," in *Proc. Computer Society Conf. on Computer Vision and Pattern Recognition, CVPR'88*, pp. 366–374, IEEE, 1988.
- [4] L. Matthies, *Dynamic stereo vision*. PhD thesis, Carnegie Mellon University, 1989.
- [5] J. Hernandez, K. Tsotsos, and S. Soatto, "Observability, identifiability and sensitivity of vision-aided inertial navigation," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, pp. 2319–2325, IEEE, 2015.
- [6] H. C. Longuet-Higgins and K. Prazdny, "The interpretation of a moving retinal image," *Proc. R. Soc. Lond. B*, vol. 208, no. 1173, pp. 385–397, 1980.
- [7] P. Corke, "An inertial and visual sensing system for a small autonomous helicopter," *Journal of Field Robotics*, vol. 21, no. 2, pp. 43–51, 2004.
- [8] D. Strelow and S. Singh, "Online motion estimation from image and inertial measurements," in *Proc. Workshop on Integration of Vision and Inertial Sensors (INERVIS), Coimbra, Portugal*, June 2003.
- [9] D. Strelow and S. Singh, "Motion estimation from image and inertial measurements," *The International Journal of Robotics Research*, vol. 23, no. 12, pp. 1157–1195, 2004.
- [10] P. Corke, J. Lobo, and J. Dias, "An introduction to inertial and visual sensing," *The International Journal of Robotics Research*, vol. 26, no. 6, pp. 519–535, 2007.
- [11] V. Grabe, H. H. Bulthoff, and P. R. Giordano, "A comparison of scale estimation schemes for a quadrotor UAV based on optical flow and IMU measurements," in *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 5193–5200, IEEE, 2013.
- [12] J. Delmerico and D. Scaramuzza, "A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots," *Memory*, vol. 10, p. 20, 2018.
- [13] A. Huster and S. M. Rock, "Relative position sensing by fusing monocular vision and inertial rate sensors," in *Proc. 11th Int. Conf. Advanced Robotics (ICAR'03), Coimbra, Portugal*, vol. 3, pp. 1562–1567, IEEE, 2003.
- [14] A. Huster, *Relative position sensing by fusing monocular vision and inertial rate sensors*. PhD thesis, Stanford University, 2003.
- [15] A. Dani, N. Fischer, Z. Kan, and W. Dixon, "Globally exponentially stable observer for vision-based range estimation," *Mechatronics*, vol. 22, no. 4, pp. 381–389, 2012.
- [16] R. Spica, P. R. Giordano, and F. Chaumette, "Plane estimation by active vision from point features and image moments," in *Proc. Int. Conf. Robotics and Automation (ICRA)*, pp. 6003–6010, IEEE, 2015.
- [17] R. Spica, P. R. Giordano, and F. Chaumette, "Active structure from motion: Application to point, sphere, and cylinder," *IEEE Transactions on Robotics*, vol. 30, no. 6, pp. 1499–1513, 2014.
- [18] O. Tahri, D. Boutat, and Y. Mezouar, "Brunovsky's linear form of incremental structure from motion," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1491–1499, 2017.
- [19] A. Martinelli, "Vision and IMU data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 44–60, 2012.
- [20] A. De Luca, G. Oriolo, and P. Robuffo Giordano, "Feature depth observation for image-based visual servoing: Theory and experiments," *The International Journal of Robotics Research*, vol. 27, no. 10, pp. 1093–1116, 2008.
- [21] P. Davidson, J.-P. Raunio, and R. Piché, "Monocular vision-based range estimation supported by proprioceptive motion," *Gyroscope and Navigation*, vol. 8, no. 2, pp. 150–158, 2017.
- [22] R. T. Azuma, "A survey of augmented reality," *Presence: Teleoperators & Virtual Environments*, vol. 6, no. 4, pp. 355–385, 1997.
- [23] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, pp. 298–304, IEEE, 2015.
- [24] L. Matthies, T. Kanade, and R. Szeliski, "Kalman filter-based algorithms for estimating depth from image sequences," *International Journal of Computer Vision*, vol. 3, no. 3, pp. 209–238, 1989.
- [25] M. S. Landy, L. T. Maloney, E. B. Johnston, and M. Young, "Measurement and modeling of depth cue combination: In defense of weak fusion," *Vision research*, vol. 35, no. 3, pp. 389–412, 1995.
- [26] P. Corke, *Robotics, vision and control: fundamental algorithms in MATLAB®*, vol. 73. Springer Science & Business Media, 2011.

- [27] J. M. Montiel, J. Civera, and A. J. Davison, "Unified inverse depth parametrization for monocular SLAM," in *Proc. Robotics: Science and Systems*, 2006.
- [28] P. Davidson, J.-P. Raunio, and R. Piché, "Accurate depth estimation from a sequence of monocular images supported by proprioceptive sensors," in *Proc. 23rd Int. Conf. on Integrated Navigation Systems, St. Petersburg, Russia*, pp. 249–257, 2016.
- [29] P. Davidson, M. Mansour, O. A. Stepanov, and R. Piché, "Depth estimation from motion parallax: Experimental evaluation," in *Proc. 26th Int. Conf. on Integrated Navigation Systems, St. Petersburg, Russia*, 2019.
- [30] O. A. Stepanov, "Optimal and sub-optimal filtering in integrated navigation systems," in *In Aerospace Navigation Systems; Nebylov, A., Watson, J., Eds.*, pp. 392–446, John Wiley & Sons, Inc.: New York, NY, 2016.
- [31] S. Särkkä, *Bayesian filtering and smoothing*. Cambridge University Press, 2010.
- [32] Y. Oshman and P. Davidson, "Optimal observer trajectories for passive target localization using bearing-only measurements," in *Proc. AIAA Guidance, Navigation, and Control Conference, San-Diego, CA*, p. 3740, 1996.
- [33] Y. Oshman and P. Davidson, "Optimization of observer trajectories for bearings-only target localization," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 35, no. 3, pp. 892–902, 1999.
- [34] J.-Y. Bouguet, *Camera Calibration Toolbox for Matlab*. California Institute of Technology, Pasadena, CA, 2006.
- [35] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 3936–3943, 2012.
- [36] J. Shi *et al.*, "Good features to track," in *Proc. Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 593–600, IEEE, 1994.
- [37] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of Imaging Understanding Workshop, Vancouver, BC, Canada*, pp. 121–130, 1981.
- [38] J.-Y. Bouguet, "Pyramidal implementation of the affine Lucas-Kanade feature tracker: description of the algorithm," *Intel Corporation*, vol. 5, no. 1-10, p. 4, 2001.
- [39] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng, "Complete solution classification for the perspective-three-point problem," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, pp. 930–943, Aug 2003.

APPENDIX

This appendix provides derivation of the Jacobians for the motion model Equations (26), (27).

$$A_{k-1} = \frac{\partial}{\partial X} \Psi \left(X_{k-1}, \tilde{V}_{z_{k-1}}^c, \tilde{\omega}_{y_{k-1}}^c, q_{k-1} \right)_{X=\hat{X}_{k-1}, q_{k-1}=0},$$

$$A_{k-1} = \begin{bmatrix} 1 - \frac{2(x - c_x)\tilde{\omega}_y^c \Delta t}{f} + \tilde{V}_z^c \xi \Delta t & 0 & \tilde{V}_z^c (x - c_x) \Delta t \\ -\tilde{\omega}_y^c (y - c_y) \frac{\Delta t}{f} & 1 + \tilde{V}_z^c \xi \Delta t - (x - c_x) \tilde{\omega}_y^c \frac{\Delta t}{f} & \tilde{V}_z^c (y - c_y) \Delta t \\ -\tilde{\omega}_y^c \xi \frac{\Delta t}{f} & 0 & 1 + 2\tilde{V}_z^c \xi \Delta t - (x - c_x) \tilde{\omega}_y^c \frac{\Delta t}{f} \end{bmatrix}_{X=\hat{X}_{k-1}},$$

$$G_{k-1} = \frac{\partial}{\partial q} \Psi \left(X_{k-1}, \tilde{V}_{z_{k-1}}^c, \tilde{\omega}_{y_{k-1}}^c, q_{k-1} \right)_{X=\hat{X}_{k-1}, q_{k-1}=0},$$

$$G_{k-1} = \begin{bmatrix} f + \frac{(x - c_x)^2}{f} & -\xi(x - c_x) \\ \frac{(x - c_x)(y - c_y)}{f} & -\xi(y - c_y) \\ \frac{\xi(x - c_x)}{f} & -\xi^2 \end{bmatrix}_{X=\hat{X}_{k-1}}.$$