https://doi.org/10.2352/ISSN.2470-1173.2017.9.IRIACV-259

Real-time mobile robot navigation based on stereo vision and low-cost GPS

Soonhac Hong, Ming Li, Miao Liao, Peter van Beek
Sharp Laboratories of America
Camas, Washington, USA

Abstract

As most robot navigation systems for large-scale outdoor applications have been implemented based on high-end sensors, it is still challenging to implement a low-cost autonomous ground-based vehicle. This paper presents an autonomous navigation system using only a stereo camera and a low-cost GPS receiver. The proposed method consists of Visual Odometry (VO), pose estimation, obstacle detection, local path planning and a waypoint follower. VO computes a relative pose between two pairs of stereo images. However, VO inevitably suffers from drift (error accumulation) over time. A low-cost GPS provides absolute locations that can be used to correct VO drift. We fuse data from VO and GPS to achieve more accurate localization both locally and globally, using an Extended Kalman Filter (EKF). To detect obstacles, we use a dense depth map that is generated by stereo disparity estimation and transformed into a 2D occupancy grid map. Local path planning computes temporary waypoints to avoid obstacles, and a waypoint follower navigates the robot towards the goal point. We evaluated the proposed method with a mobile robot platform in real-time experiments in an outdoor environment. Experimental results show that the mobile vision and control system is capable of traversing roads in this outdoor environment autonomously.

Introduction

Tremendous advances have been made in the area of self-driving cars since the DARPA Grand Challenges over 10 years ago. However, most autonomous navigation systems to date have been designed based on high-end sensors, such as laser scanners, LIDAR, high accuracy augmented GPS receivers and high-cost inertial sensors [1][2][3][4]. These all add up to the significant cost of such vehicles and robots, making them unaffordable for many applications. At the same time, continuous high reliability is not guaranteed even for high-end sensors, such as augmented GPS or RTK-GPS, in a variety of environments. For example, GPS is not sufficiently reliable in dense urban environments. Thus, it is both attractive and challenging to implement an autonomous navigation system that is both low-cost and robust for a wide range of applications.

This paper presents a real-time autonomous navigation system using a stereo camera and a low-cost GPS. Although a couple of related works have presented robot navigation systems with low-cost sensors [5][6][7][8], these systems are often evaluated in indoor or small-scale environments only. We demonstrated real-time stereo vision-guided navigation in outdoor environments on long trajectories with a mobile robot.

The sensors in our system include a forward-facing stereo camera and a low-cost GPS. The proposed method includes 3D Visual Odometry (VO), pose estimation by Extended Kalman Filter (EKF), obstacle detection, local path planning, and a waypoint follower as shown in Figure 1. VO computes relative poses between consecutive pairs of stereo images. EKF pose estimation combines VO data with GPS data to estimate position and orientation. The algorithm combines the advantages of both VO and GPS; namely, GPS provides accuracy in a global sense, and VO provides locally precise and smooth position updates between GPS readings. To detect obstacles in the driving path, we generate a dense 3D point cloud using stereo image pairs. After removing 3D points in the ground plane, remaining 3D points are projected into a 2D occupancy grid map. A path planning algorithm computes temporary waypoints to avoid obstacles, based on the local 2D occupancy grid map and the estimated robot pose. After avoiding an obstacle, a waypoint following algorithm navigates the robot back to the desired route and goal point.

The following sections explain VO, EKF pose estimation, obstacle detection, and local path planning. We describe a waypoint follower in the experimental results section.

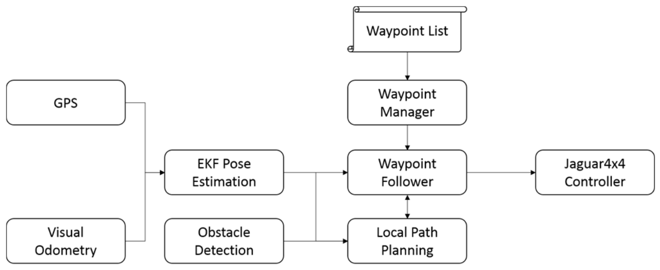

Figure 1. Overview of real-time mobile robot navigation with stereo vision and low-cost GPS

Visual Odometry

We followed algorithms described in [13] and [14] to develop a robust visual odometry system. The first step is to get the intrinsic and extrinsic parameters of the cameras. Here we estimate both camera intrinsic and extrinsic parameters with Camera Calibration Toolbox for Matlab [11]. Camera calibration requires capturing multiple images of the calibration pattern at different poses, in order to accurately estimate the intrinsic camera parameters.

Second, when images from both left and right cameras are received, we need to remove the image distortion introduced by the lenses. Usually, wide angle lens has large distortion close to image boundaries. The lens distortion parameters are contained in the output of the camera calibration [12]. Therefore, we simply use these distortion parameters to reverse the distortion process in order to remove image distortion.

Third, since stereo VO involves feature matching between left and right images, it is necessary to rectify left and right image after undistortion to make the same feature point lie on the same horizontal line. By doing this, the feature searching space is reduced from 2D to 1D, largely improving the algorithm efficiency.

Image rectification involves projecting both left and right image onto a common image plane. This new image plane must be parallel to the line that connects left and right camera center, which guarantees the projections of a scene point are on the same horizontal line on both left and right rectified images. It also guarantees that rectified left and right images have the same intrinsic camera parameters. Once the new image plane is defined, we can simple rectify both images by a homography warping.

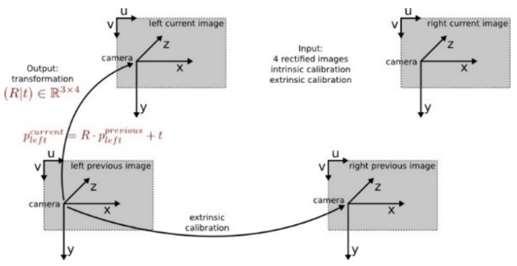

Figure 2. Transformation between 2 neighboring frames (image courtesy of Libviso [13]).

As shown in Figure 2, we use two stereo pairs to estimate the robot movement between time s-1 and s. First we establish feature correspondences across these 4 images. Feature matches between left and right images are used to reconstruct 3D positions of those features in the scene. For the same set of feature points, we can find their 3D positions at both time s-1 and s. Under the assumption of rigid body transformation, the transformation of those 3D points is actually inverse of the robot transformation.

Thus, we estimate the rigid transformation (rotation $\boldsymbol{R}$ and translation $\boldsymbol{t}$) of those 3D points by minimizing the following term in the least squares sense:

$$\sum_{i=1}^{n} \left\| \boldsymbol{R}\boldsymbol{p}_{i,s-1} + \boldsymbol{t} - \boldsymbol{p}_{i,s} \right\|$$

where we assume total number of 3D points is n. $\boldsymbol{p}_{i,s-1}$ and $\boldsymbol{p}_{i,s}$ are 3D positions of point p at time s-1 and s respectively. If we define the coordinate system at center of left camera, the 3D position of feature points can be easily obtained once left and right image are rectified:

$$z = \frac{fB}{d}, x = \frac{uz}{f}, y = \frac{vz}{f}$$

where f is camera focal length, B is baseline of left and right cameras, d is disparity of the feature on left and right images. [u, v] is 2D coordinate of the feature on left image.

Since the transformation of 3D point is inverse of that of robot, we can simply get robot transformation ($\boldsymbol{R}_s$ and $\boldsymbol{t}_s$) by:

$$\boldsymbol{R}_s = \boldsymbol{R}^{-1}$$
$$\boldsymbol{t}_s = -\boldsymbol{R}^{-1}\boldsymbol{t}$$

In order to estimate the current pose of robot, we need to accumulate frame-to-frame transformations from a known start point. We here assume that the pose of the first frame is world origin [0, 0, 0], then the pose of robot at time s can be obtained by following equation:

$$\begin{bmatrix} \boldsymbol{O}_s & \boldsymbol{P}_s \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_s & \boldsymbol{t}_s \\ 0 & 1 \end{bmatrix} * \begin{bmatrix} \boldsymbol{O}_{s-1} & \boldsymbol{P}_{s-1} \\ 0 & 1 \end{bmatrix}$$

where $\boldsymbol{O}_{s-1}$ is a 3x3 matrix recording robot orientation at time s-1 and $\boldsymbol{P}_{s-1}$ is a 3x1 vector recording 3D position of the robot at time s-1.

VO algorithm inevitably suffers from drift after long distance traveling, because error is also accumulated when frame-to-frame transformations are concatenated. In order to minimize the drift, we need to minimize the number of frames used for VO estimation. That means we have to keep neighboring frames as distant as possible but not lose feature tracking. In our system, we compute the distance between two neighboring frames from $\boldsymbol{t}_s$, which is the translation of the camera center in 3D space. We also compute rotation angle from $\boldsymbol{R}_s$. $\boldsymbol{R}_s$ is first converted to a rotation vector using Rodrigues algorithm. The normal of the result vector is the rotation angle we need. If normal of $\boldsymbol{t}_s < 5$ cm or rotation angle of $\boldsymbol{R}_s < 5$ degrees, we simply drop current frame and capture new frame to calculate robot transformation to last frame.

Up to this point, the result robot pose is defined in the coordinate system of left camera. In particular, the coordinate system origin coincides with the camera center of left camera when it captures the first frame. Z axis of the coordinate system is aligned with the looking direction of the camera. This may cause some issues when the camera looking direction is not parallel to the ground. In order to measure the distance it traveled on the ground, we need to transform the robot pose into a coordinate system that is parallel to the ground.

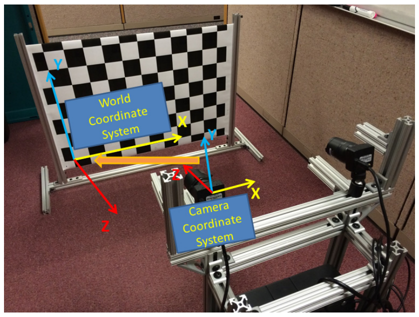

Figure 3. Transformation from camera coordinate system to world coordinate system.

As shown in the Figure 3, we define a world coordinate system using a calibration checkerboard. We put the checkerboard perpendicular to the ground, so that its x and z axis are both parallel to the ground. Once we get the robot pose in camera's coordinate system, we transform that pose into world coordinate system and then project it onto the xz plane to get the distance traveled on the ground. The transformation between camera coordinate system and world coordinate system can be easily obtained through camera calibration.

EKF Pose Estimation

Although VO accurately computes a relative pose between two pairs of stereo images, VO suffers from drift accumulation over time. A low-cost GPS provides absolute locations that can be used to correct VO drift. We fuse data from VO and GPS to achieve more accurate localization both locally and globally, using an EKF.

EKF pose estimation is composed of two steps: prediction and correction. In the prediction step, the robot pose $\hat{\boldsymbol{x}}_s$ and its covariance $\hat{\boldsymbol{P}}_s$ at time *s* can be predicted by

$$\hat{\boldsymbol{x}}_s = f(\boldsymbol{x}_{s-1}, \boldsymbol{u}_s)$$
$$\hat{\boldsymbol{P}}_s = \boldsymbol{F}_x\hat{\boldsymbol{P}}_{s-1}\boldsymbol{F}_x^T + \boldsymbol{F}_u\boldsymbol{Q}_s\boldsymbol{F}_u^T$$

where $\boldsymbol{u}_s$ is the relative robot pose computed by VO between two robot poses. As the robot pose $\hat{\boldsymbol{x}}_s$ can be represented by $(\hat{x}_s, \hat{y}_s, \hat{\theta}_s)$ in the 2D environment, the prediction function $f(\boldsymbol{x}_{s-1}, \boldsymbol{u}_s)$ can be represented by

$$\hat{x}_s = \hat{x}_{s-1} + \delta_x$$
$$\hat{y}_s = \hat{y}_{s-1} + \delta_y$$
$$\hat{\theta}_s = \hat{\theta}_{s-1} + \delta_\theta$$

where δ_x and δ_y are the relative translations in x-axis and y-axis respectively. δ_θ is the relative orientation between two robot poses. These relative translations and orientation are computed by VO.
$\boldsymbol{F}_x$ and $\boldsymbol{F}_u$ are the Jacobians of the robot pose with respect to the previous pose and relative pose respectively. In the 2D environment, $\boldsymbol{F}_x$ and $\boldsymbol{F}_u$ can be represented by

$$\boldsymbol{F}_x = \begin{bmatrix} 1 & 0 & -d\sin\hat{\theta}_s \\ 0 & 1 & d\cos\hat{\theta}_s \\ 0 & 0 & 1 \end{bmatrix}$$
$$\boldsymbol{F}_u = \begin{bmatrix} \cos\hat{\theta}_s & -d\sin\hat{\theta}_s \\ \sin\hat{\theta}_s & d\cos\hat{\theta}_s \\ 0 & 1 \end{bmatrix}$$

where d is the Euclidean distance between $\hat{\boldsymbol{x}}_s$ and $\hat{\boldsymbol{x}}_{s-1}$.

In the correction step, the robot pose $\boldsymbol{x}_s$ and its covariance $\boldsymbol{P}_s$ can be corrected by

$$\boldsymbol{x}_s = \hat{\boldsymbol{x}}_s + \boldsymbol{K}_s\boldsymbol{v}_s$$
$$\boldsymbol{P}_s = \hat{\boldsymbol{P}}_s - \boldsymbol{K}_s\boldsymbol{H}_x\hat{\boldsymbol{P}}_s$$

where $\boldsymbol{K}_s = \hat{\boldsymbol{P}}_s\boldsymbol{H}_x^T(\boldsymbol{S}_s)^{-1}$ (Kalman gain), $\boldsymbol{S}_s = \boldsymbol{H}_x\hat{\boldsymbol{P}}_s\boldsymbol{H}_x^T + \boldsymbol{H}_w\boldsymbol{W}_s\boldsymbol{H}_w^T$, and $\boldsymbol{v}_s = \boldsymbol{z}_s - h(\hat{\boldsymbol{x}}_s, \boldsymbol{w})$. $\boldsymbol{H}_x$ and $\boldsymbol{H}_w$ are the Jacobians of the robot position with respect to the measured robot position by GPS and measurement noise of GPS. $h(\hat{\boldsymbol{x}}_s, \boldsymbol{w})$ can be represented in the 2D environment by

$$h(\hat{\boldsymbol{x}}_s, \boldsymbol{w}) = \begin{bmatrix} \hat{x}_s \\ \hat{y}_s \end{bmatrix} + \begin{bmatrix} w_x \\ w_y \end{bmatrix}$$

where $\boldsymbol{w} = (w_x, w_y)$ is the measurement noise of GPS.
Thus, $\boldsymbol{H}_x$ and $\boldsymbol{H}_w$ can be represented by

$$\boldsymbol{H}_x = \frac{\partial h}{\partial x}|_{w=0} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$
$$\boldsymbol{H}_w = \frac{\partial h}{\partial w}|_{w=0} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

and $\boldsymbol{z}_s$ obtained by GPS can be represented by

$$\boldsymbol{z}_s = \begin{bmatrix} x_g \\ y_g \end{bmatrix} + \begin{bmatrix} w_x \\ w_y \end{bmatrix}$$

The prediction and correction steps are executed repeatedly whenever new data is available from either VO or GPS. If new data is available from both VO and GPS at time *s*, EKF estimates the robot pose by running the prediction step and the correction step sequentially. If new data is available from only VO at time *s*, EKF updates the robot pose by running the prediction step only.

Obstacle Detection

This section presents the obstacle detection using dense stereo matching. The algorithm is built based on two assumptions which are reasonable in terms of the robot and automobile application. First, the cameras are pre-calibrated and are mounted on the platform with known pitch angle and height to the ground plane. Second, the cameras have a smooth trajectory which can avoid heavy computation related to rotation, scale invariant feature descriptors. The assumptions allow us to project between world coordinate system, camera coordinate system and image plane.

The first step of the algorithm is dense stereo matching. In this system, we used the Library for Efficient Large-scale Stereo Matching (LibELAS) [9] for stereo matching. The library is efficient enough to achieve real-time matching result. Using the disparity map, 3D coordinate can be calculated with camera matrix from calibration. Given focal length *f*, principle point $[x_c, y_c]$ and baseline *B*, the 3D coordinates of a matched point in camera coordinate system can be computed by the following equations:

$$X_c = \frac{f(x - x_c)}{d}$$
$$Y_c = \frac{f(y - y_c)}{d}$$
$$Z_c = \frac{fB}{d}$$

With $[X_c, Y_c, Z_c]$ coordinate in camera coordinate system, the point can be projected to world coordinate system using the extrinsic camera matrix as follows:

$$\begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ 0 & 1 \end{bmatrix}\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -sin\theta & -h \\ 0 & sin\theta & cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} X_c \\ Y_c \\ Z_c \\ 1 \end{bmatrix}$$

where θ is the pitch angle of the cameras facing down, h is the height of the cameras from the ground plane.

When the point cloud is in the world coordinate system, the next step is to remove the points that represent the road region. For road point removal, RANdom SAmpling Consensus (RANSAC) [10] algorithm is used. In the outdoor application, the most dominant plane in a scene is the road plane. Based on this assumption, RANSAC is used to fit just one plane. This algorithm used the RANSAC algorithm in the Point Cloud Library (PCL) to estimate planes. While it is powerful, it also requires a heavy computation time considering the number of points in a point cloud. So the plane estimation process was designed to work periodically, instead of being active every frame in order to be able to run online. To make plane estimation more computation efficient, some optimizations are made to the algorithm. The primary reason of the costly computation time is the large number of points as input to the RANSAC algorithm. Before, we used all the points generated from disparity map. Because the random sampling process, it requires all the points are included in error measurements. The large number of outliers also increases the number of iterations intensively. The optimization here is to limit the input size.

Since the goal of the algorithm to estimate ground plane, we can limit the input points to be in a certain region in the 3D space. Using the points in this region can still estimate the ground plane accurately, because the ground is still the dominant plane in the subsample of the point clouds. The sub sample point cloud also has less outliers, such as sky, trees, thus the algorithm can converge in fewer iterations. It can reduce the computation time of RANSAC algorithm and achieve real-time speed.

After the ground plane is estimated, the inliers of this plane are removed. The remaining points are then project the points into an occupancy grid. Occupancy grid is a 2D grid map with each grid cell representing a block in the 3D space. Points are projected to the ground plane and into different cells on the grid map based on their location. A cell is considered as an obstacle once the number of points in the cell reaches a pre-defined threshold.

Local Path Planning

To avoid obstacles and regenerate the path, we implement local path planning. Local path planning uses EKF pose estimation and Obstacle Detection (OD) presented in the previous sections. EKF pose estimation provides the robot pose for local path planning to localize the robot in the world coordinate. OD generates an occupancy grid map which is used for local path planning to avoid obstacles. We implement a Tangent Bug (TB) algorithm for local path planning.

TB is capable of finding the shortest path to the goal when a robot has a finite range sensor. TB has two basic behaviors: 1) moving toward the goal; 2) following the boundary of a concave obstacle. When the robot encounters obstacles, TB finds a waypoint which minimizes the travel path to the goal. In addition, TB can follow the boundary of a concave obstacle until it clears that obstacle and resume a path toward the goal point.

TB can be described as shown in Algorithm 1, 2 and 3. Whenever a new image pair (i.e., left and right images) is obtained from a stereo camera, TB computes a robot pose using EKF pose estimation and generates an occupancy grid map using OD (Line 3 and 4 in Algorithm 1). In 2D environments, a robot pose can be parameterized as $\boldsymbol{x}= [x, y, \theta]$, where $[x, y]$ is the translation and θ is the orientation in the world coordinate W. The occupancy grid map can be presented as $\boldsymbol{M} = \{m_1, m_2, \ldots, m_n\}$, where m_n is the nth grid cell in the robot coordinate. In the occupancy grid map, each cell has 1 or 0. If a cell is occupied by obstacles, it has 1. Otherwise, the cell has 0. Figure 4 depicts a robot pose in the world coordinate and an occupancy grid map in the robot coordinate.

If the occupancy grid map has obstacles (Line 6-8 in Algorithm 1), TB finds a temporary waypoint $\boldsymbol{n}$ which minimizes the following cost function

$$h(\boldsymbol{x},\boldsymbol{n}) = d(\boldsymbol{x},\boldsymbol{n}) + d(\boldsymbol{n},\boldsymbol{g})$$

where $d(\boldsymbol{n}, \boldsymbol{g})$ is the Euclidean distance between a temporary waypoint $\boldsymbol{n}$ and the goal $\boldsymbol{g}$. $d(\boldsymbol{x}, \boldsymbol{n})$ is the Euclidean distance between the current robot location $\boldsymbol{x}$ and a temporary waypoint $\boldsymbol{n}$. Temporary waypoints $\boldsymbol{N} = \{\boldsymbol{n}_1, \boldsymbol{n}_2, \ldots, \boldsymbol{n}_k\}$ are selected by Algorithm 2. $\boldsymbol{n}$ is selected among unoccupied grid cells around the occupied grid cells. If there is no unoccupied grid cells in the nearest row of the map, the right location of the most right occupied grid cell in the nearest row is selected as a temporary waypoint because we assume the area out of the grid map is obstacle-free.

If the cost function $h(\boldsymbol{x}, \boldsymbol{n})$ increases, TB begins to follow the boundary of a concave obstacle (Line 10 and 17 in Algorithm 1). In the boundary following algorithm (Algorithm 3), TB continues to follow the boundary of the concave obstacle and updates d_{reach} and $d_{followed}$, where d_{reach} is the shortest distance between the robot and the goal and $d_{followed}$ is the shortest distance between the sensed boundary and the goal, respectively (Line 1-2 in Algorithm 3). If $d_{reach} < d_{followed}$, TB terminates the boundary following (Line 3-4 in Algorithm 3) and moves the robot toward the goal.

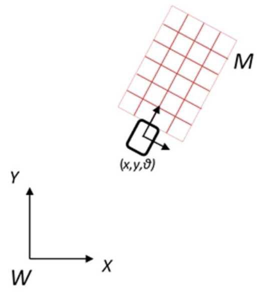

Figure 4. Robot pose relative to world coordinate system, and occupancy grid map (based on robot obstacle perception) relative to local robot coordinate system.

Algorithm 1 Tangent_Bug
```
 1: Set boundary_following_mode off.
 2: while Goal is not reached do
 3:     Compute a robot pose by using EKF pose estimation
 4:     Compute an occupancy grid map by using obstacle detection
 5:     if boundary_following_mode is off then
 6:         if Obstacles are detected then
 7:             $Select\_waypoints(occupancy\_grid\_map)$
 8:             Select a waypoint $n$ minimizing $h(x, n) = d(x, n) + d(n, g)$
 9:             Move a robot toward a waypoint $n$
10:             if $h(x, n)$ begins to increase then
11:                 Set boundary_following_mode on
12:             end if
13:         else
14:             Move a robot toward the goal.
15:         end if
16:     else
17:         $Boundary\_following(occupancy\_grid\_map)$
18:     end if
19: end while
```

Algorithm 2 Select_waypoints(occupancy_ grid_map)
```
 1: for Each grid cell in obstacles do
 2:     if The left grid cell is free space then
 3:         Add the left grid cell into waypoints
 4:     else if The right grid cell is free space then
 5:         Add the right grid cell into waypoints
 6:     end if
 7: end for
 8: if All grid cells in the nearest row are obstacles then
 9:     Add the right grid cell of the most right grid cell into waypoints
10: end if
11: return waypoints
```

Algorithm 3 Boundary_following(occupancy_ grid_map)
```
1: Move a robot along the boundary in same direction as before repeating
2: Update $d_{reach}$ and $d_{followed}$
3: if $d_{reach} < d_{followed}$ then
4:     Set boundary_following_mode off.
5: end if
```

Experimental Results

We implemented the proposed autonomous navigation system using ROS [22] and evaluated the autonomous navigation system with a mobile robot as shown in Figure 5. The mobile robot consists of a Jaguar4x4 wheel [23], two Pointgrey Chameleon cameras [24] for a stereo vision, a u-blox GPS evaluation Kit [25], and a mini-computer. VO updates a relative robot pose at 15 Hz and GPS updates a absolute robot position at 3 Hz. The mobile robot runs at 1 m/s of speed in the experiment.

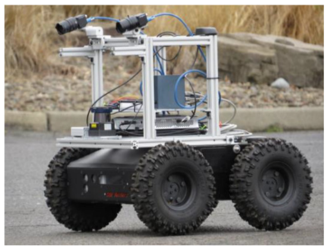

Figure 5. Mobile robot

We evaluated the real-time stereo vision-guided navigation system in outdoor environments on long trajectories (e.g. ~430 meter) as shown in Figure 6. We generate a waypoint list which includes three waypoints. Two waypoints are generated at the corner of the trajectory and one waypoint is generated at the end point of the trajectory. Each waypoint is composed of the waypoint location and the target orientation of the robot at the waypoint.

Once the robot starts, a waypoint follower drives the robot toward the first waypoint. The waypoint follower computes the translational and rotational velocity of the robot using the distance and orientation difference between the estimated robot pose and the first waypoint. The translational and rotational velocity is used for Jaguar4x4 controller to run the robot. If the robot detects an obstacle, local path planning generates a temporary waypoint to avoid the obstacle and the waypoint follower drives the robot toward the temporary waypoint. If the robot reaches the temporary waypoint, the waypoint follower resumes the path toward the first waypoint. Once the robot reaches the first waypoint, the waypoint follower drives the robot toward the second waypoint. After passing the first and second waypoints, the robot stops at the last waypoint.

The experimental results show that: 1) the proposed autonomous system allows the robot to run along the path and arrive at the end point successfully; 2) the final position error at the end point is about 3 meter as shown in Figure 7; 3) stereo vision-based obstacle detection is able to detect low curbs at the side of the road; 4) local path planning successfully generates a temporary waypoint for avoiding low curbs; 5) autonomous navigation in this environment is not yet robust, due to limited precision of location and pose estimation and due to limited mapping capability.

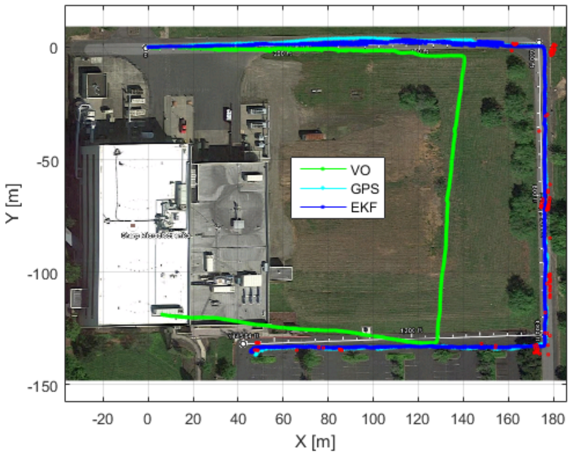

Figure 6. Trajectories of VO, GPS and EKF pose estimation and detected obstacles. Red rectangles represent detected obstacles along the trajectory.

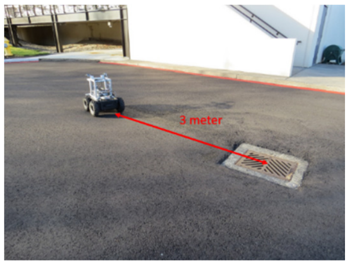

Figure 7. Final position error at the end point

Conclusion

We proposed the real-time autonomous navigation system using only a stereo camera and a low-cost GPS. The proposed method is composed of VO, EKF pose estimation, obstacle detection, local path planning, and a waypoint follower. EKF pose estimation computes the robot pose by fusing VO and GPS. Obstacle detection and local path planning avoid obstacles on the path. The waypoint follower generates transitional and rotational velocity to drive the robot toward a waypoint or a temporary waypoint. The proposed method is successfully demonstrated in outdoor environments.

In the future study, we will improve pose estimation by using additional a low-cost sensor such as IMU. In addition, we will try to build a map to improve accuracy of pose estimation and robustness of autonomous navigation.

References

[1] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, J. Dolan, D. Duggins, D. Ferguson, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. Howard, A. Kelly, D. Kohanbash, M. Likhachev, N. Miller, K. Peterson, R. Rajkumar, P. Rybski, B. Salesky, S. Scherer, Y. Woo-seo, R. Simmons, S. Singh, J. Snider, A. Stentz, W. R. Whittaker, J. Ziglar, J. Struble, and M. Taylor, "Tartan Racing: A Multi-Modal Approach to the DARPA Urban Challenge," Defense, vol. 94, no. 4, pp. 386–387, 2007.

[2] C. Stiller and J. Ziegler, "3D perception and planning for self-driving and cooperative automobiles," International Multi-Conference System Signals Devices, pp. 1–7, 2012.

[3] C. Siagian, C. K. Chang, and L. Itti, "Autonomous Mobile Robot Localization and Navigation Using a Hierarchical Map Representation Primarily Guided by Vision," Journal of Field Robotics, vol. 31, no. 3, pp. 408–440, 2014.

[4] M. Bibuli, M. Caccia, and L. Lapierre, "Autonomous Driving in Urban Environments: Boss and the Urban Challenge," IFAC Proc., vol. 7, pp. 81–86, 2007.

[5] F. Dayoub, T. Morris, B. Upcroft, and P. Corke, "Vision-Only Autonomous Navigation Using Topometric Maps," IEEE International Conference on Intelligent Robots and Systems, pp. 3-7, 2013.

[6] H. Morioka, S. Yi, and O. Hasegawa, "Vision-based mobile robots SLAM and navigation in crowded environments," IEEE Int. Conf. Intell. Robots and Systems, pp. 3998-4005, 2011.

[7] K. Konolige, E. Marder-Eppstein, B. Marthi, "Navigation in hybrid metric-topological maps," IEEE International Conference on Robotics and Automation (ICRA), pp. 3041-3047, 2011.

[8] M. Agrawal, K. Konolige, and R. C. Bolles, "Localization and mapping for autonomous navigation in outdoor terrains: A stereo vision approach," IEEE Workshop Applications of Computer Vision (WACV), 2007.

[9] A. Geiger, M. Roser and R. Urtasun, "Efficient Large-Scale Stereo Matching," Asian Conference on Computer Vision (ACCV), 2010.

[10] M. Fischler and R. C. Bolles, "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," Communications of the ACM, vol. 24, pp. 381-395, 1981.

[11] Camera Calibration Toolbox for Matlab https://www.vision.caltech.edu/bouguetj/calib_doc/

[12] J. Heikkila and O. Silven, "A Four-step Camera Calibration Procedure with Implicit Image Correction," Conference on Computer Vision and Pattern Recognition (CVPR), 1997.

[13] A. Geiger, J. Ziegler and C. Stiller, "StereoScan: Dense 3D Reconstruction in Real-time," Intelligent Vehicles Symposium, 2011.

[14] B. Kitt, A. Geiger and H. Lategahn, "Visual Odometry based on Stereo Image Sequences with RANSAC-based Outlier Rejection Scheme," Intelligent Vehicles Symposium, 2010.

[15] J. Engel, J. Sturm, D. Cremers, "Semi-Dense Visual Odometry for a Monocular Camera," IEEE International Conference on Computer Vision (ICCV), 2013.

[16] K. Konolige, M. Agrawal, R. C. Bolles, C. Cowan, M. Fischler, and B. Gerkey, "Outdoor mapping and navigation using stereo vision," International Symposium on Experimental Robotics (ISER), pp. 179-190, 2006.

[17] V.J. Lumelsky and A.A. Stepanov, "Path-planning strategies for a point mobile automation moving amidst obstacles of arbitrary shape," Algorithmica, pp. 403-430, 1987.

[18] J. Borenstein and Y. Koren, "The vector field histogram- fast obstacle avoidance for mobile robots," IEEE Journal of Robotics and Automation, vol. 7, pp. 278-288, 1991.

[19] C. Siagian, C. Chang and L. Itti, "Mobile robot navigation system in outdoor pedestrian environment using vision-based road recognition," IEEE International Conference on Robotics and Automation, pp. 564-571, 2013.

[20] K. Sabe, M. Fukuchi, J. Gutmann, T. Ohashi, K. Kawamoto, T. Yoshigahara, "Obstacle avoidance and path planning for humanoid robots using stereo vision," IEEE International Conference on Robotics and Automation, vol. 1, pp. 592-597, 2004.

[21] S. Quinlan and O. Khatib, "Elastic bands: Connecting path planning and control," IEEE International Conference on Robotics and Automation (ICRA), pp. 802-807, 1993.

[22] Robot Operating System, http://www.ros.org/

[23] Jaguar4x4 wheel, http://jaguar.drrobot.com/specification_4x4w.asp

[24] Pointgrey chameleon camera, https://www.ptgrey.com/chameleon-usb2-cameras

[25] u-blox 7 GNSS evaluation kits, https://www.u-blox.com/en/product/evk-7p

Author Biography

Soonhac Hong received his BS and MS in mechanical engineering from Hanyang University (1994,1996), MS in computer science from Columbia University (2010) and PhD in engineering science and systems from University of Arkansas at Little Rock (2014). Since then he has worked at Sharp Laboratories of America, WA. His work has focused on visual Simultaneous Localization and Mapping (SLAM) and autonomous robot navigation.

Ming Li received his BS from Wuhan University in China (2011), MS in imaging science from Rochester Institute of Technology (2015). He is a researcher and engineer at Sharp Laboratories of America. His work mainly focuses on autonomous robot navigation and computer vision.

Miao Liao received his BS in school of software from Tsinghua University (2005), and PhD in Computer Science from University of Kentucky (2011). Currently, he is a Senior Researcher at Sharp Labs of America. His research interests include image processing and image-based 3D sensing & modeling.

Peter van Beek is a group manager and project leader at Sharp Labs of America. His interests are in signal / image / video processing, computer vision and machine learning, From 1996 to 1998, he was a research associate with the Department of Electrical and Computer Engineering at the University of Rochester. Peter received the M.Sc.Eng. and Ph.D. degrees in Electrical Engineering from the Delft University of Technology, the Netherlands (1990 and 1995).