In 19th Portuguese Conference on Pattern Recognition (RecPad 2013)
Lisboa, Portugal, November 2013

Homing a Teleoperated Car using Monocular SLAM

Nuno Ribeiro, Ricardo Ferreira, José Gaspar
http://www.isr.ist.utl.pt

Institute for Systems and Robotics / IST
Lisbon, Portugal

Abstract

Personal, inexpensive, easy-to-use, teleoperated, autonomous, mobile robots are starting to be a reality and are expected to be widespread within some few years. Fostering the development of these robots implies testing various hardware and software technologies. In this work we propose developing the communications with the robot through an IP wireless camera. The rational is that the required bandwidth for motion commanding and reporting is much lesser than the one required for video streaming. Therefore, robot commanding can be embedded non-disruptively within the video streaming. In this paper we detail the base components forming the robot, show navigation experiments based in MonoSLAM, and propose a methodology to regulate the accumulation odometry error associated with map-less MonoSLAM.

1 Introduction

Current home personal robots are starting to have affordable prices. These robots can be vacuum cleaners such as the iRobot Roomba, can be telepresence robots, as the Anybots' QA, or can be simply Mobile Webcams, such as the WowWee's Rovio. Most of these robots have in common the combination of mobile robotics, video cameras and wireless communications. Communications are in many aspects the bottleneck of the robots.

In this work we propose using the wireless network of surveillance cameras as a basis to build networked mobile robots. The objectives of this work are threefold: (i) assembling one Axis 207w camera and one Arduino Uno; (ii) developing software to make possible the communication between the user's pc, the Axis camera and the Arduino; and (iii) Designing a homing strategy which enables the robot to return autonomously to its initial position.

2 Hardware and Software Setup

A car is assembled based in a network (IP) camera, one Arduino Uno, one Motor Shield and two DC motors for steering and propulsion (see fig.1(a)). A Java GUI from [2] is used to control the car in real time using a laptop. The camera gives a live feed from the car (robot) point of view and extract images to be processed on the user's computer. The camera has a Linux OS which runs a server with the objective of processing user's instructions and sends the corresponding signals to the Arduino board. Once a complete instruction is received by the Arduino, it will execute the commands for motors control. The only sensor that the car has available is the camera. The program on the user's computer runs in MATLAB, that calls the Java GUI and runs MonoSLAM [1] in parallel. The hardware setup is sketched in fig.1(b).

3 Autonomous Homing

We use MonoSLAM [1], so that the camera is the single sensor necessary to compute the pose of the vehicle. Basically, MonoSLAM consists of an Extended Kalman Filter (EKF) with Random Sample Consensus (RANSAC) embedded. The EKF makes a prediction of the vehicle's pose using a constant velocity model. The update step is made by using the most voted hypothesis in RANSAC which generates hypothesis from candidate features matches. To detect local features in the images a corner extraction procedure is used. MonoSLAM is initialized assuming that an a priori probability distribution over the model parameters is known.

MonoSLAM provides an estimation of the pose of the car. One extracts from the vector *state* the position of the car

$$^{W}t_C = state(1:3) = [x\ y\ z]^T \quad (1)$$

and compute its rotation matrix in the world frame

$$^{W}R_C = q2r(state(4:7)). \quad (2)$$

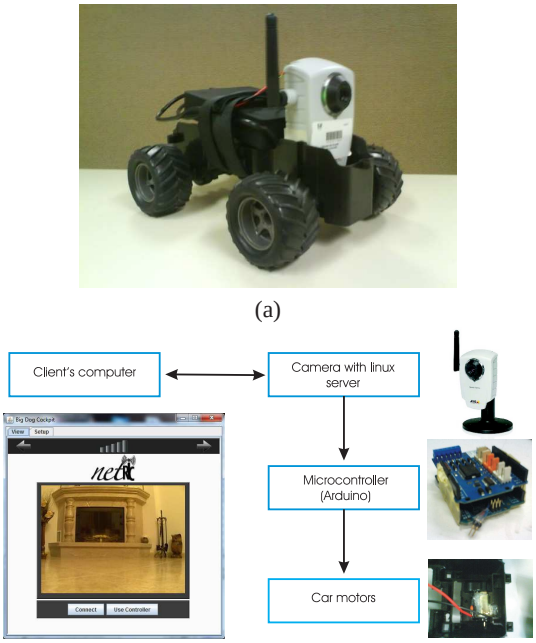


Figure 1: (a) Car used in experiments. (b) Basic project design.

where $q2r$ denotes the conversion for a quaternion to a rotation matrix. Defining $c_2 = \sqrt{^{W}R_C(1,1)^2 + {}^{W}R_C(1,2)^2}$, i.e. the cosine of the rotation angle over the z axis, one has the the robot orientation angle is

$$\theta_2 = \text{atan2}(-{}^{W}R_C(1,3), c_2). \quad (3)$$

MonoSLAM also gives us the inverse depth coordinates of every feature, from which we can compute the euclidean coordinates in the world reference frame.

The program saves the car's position and orientation in every iteration in order to backtrack (home) to its initial position by the same way it reached its actual position. The backtracking is triggered by the user at any point he wants.

After the user activates the homing procedure, the car reverses and the controller actions take place. To calculate the vehicle's pose in each iteration of homing, a simulator using the vehicle's kinematic is being used where its reference orientation its calculated as

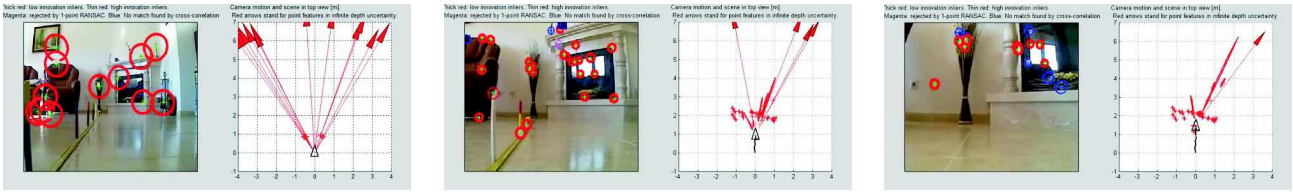
$$\theta_{ref} = a\tan 2\left(\frac{y_{ref} - y_{actual}}{x_{ref} - x_{actual}}\right) \quad (4)$$

This strategy is being used in real-time experiments, once the features matching in Mono SLAM running in MATLAB fails with abrupt movement.

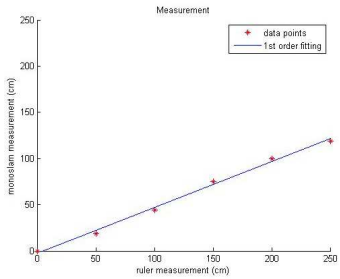
4 Experiments and Results

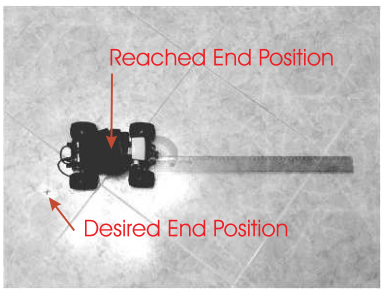
In order to have an idea of the errors magnitude when the car moves, we have chosen the hard case where the car has just *Forward Motion*. As you can see in fig. 2(b) and fig. 2(c), the Mono SLAM acknowledges that the car is moving straight and the orientation is no different from the real one. When a considerable number of features moves its position on the image plan from one frame to the next, the algorithm understands the movement and realizes if the camera is sliding right, left, up, down, or if it is actually moving forward or backward. That can be verified on fig. 2(b) and fig. 2(c). As we can see in fig.2(d), the difference between ruler and monoslam position estimate is very large.

In a second experiment the car moves in a straight line and some few meters later, while still reading the pose from the MonoSLAM, a homing



(a) (b) (c)





(d) (e)

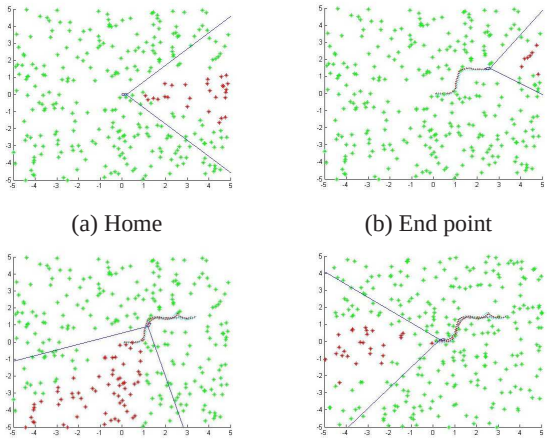
Figure 2: (a) Car at the beginning of straight line trajectory experiment. (b) Car at the half of straight line trajectory experiment. (c) Car at the end of straight line trajectory experiment. (d) Ruler vs Mono SLAM measurement at straight line trajectory experiment. (e) Complete Homing system test

	real	estimated
X (cm)	22	5
Y (cm)	-16	-8
θ (^{o})	-55	-41

Table 1: Real and MonoSLAM measurements comparison.

request is issued. The difference between real and ours program pose estimation is shown in table 4. As you can see in table 4, the results are considerably different. The final vehicle's pose is shown in fig.2(e).

From the experiments made, we can see that MonoSLAM easily detects changes in direction of motion, but it's not very accurate at determining the magnitude of the camera's forward motion, especially if there is no significant movement in more directions. The high computational cost of running MonoSLAM in Matlab makes the job at hand difficult. No abrupt movements can be done as otherwise the difference between two consecutive images will lead to no features matches. Using the kinematic model as the only prediction of pose while homing is possible, but leads also to a rapid increase of the pose prediction errors.



(a) Home (b) End point

(c) Returning home (d) Arrived home

Figure 3: (a) Beginning of simulation of homing using camera calibration for pose computation. (b) Beginning of homing procedure. (c) Homing procedure at middle. (d) Final homing position.

5 Future Work

As was already expected, visual odometry introduces error accumulation, even for short trajectories. One possible way to reset error accumulation is by modifying the scenario either by adding landmarks or fixed cameras informing the car location. An alternative way is to reset the error accumulation by comparing the actual image with the ones stored in an *images-map* [3], which is more interesting as the scenario is not required to be changed.

In an *images-map* based approach we are considering saving just image features that are registered (and 3D-reconstructed) along time. In each homing iteration we compute the camera matrix P from the 3D points stored previously and their current images. Given P, we extract the position t and orientation R of the camera with respect to a global coordinate system [4]. First we apply QR factorization to the first three columns of P, then transform from the QR to the RQ factorization and correct the sign of K. In the end we obtain $P = K[R \mid t]$. Figure 3 shows a simulated homing procedure encompassing calibration, pose computation and control.

6 Conclusion

This work started with the integration of the various hardware and software components into a mobile robot. This is still an ongoing project. Various positive conclusions can be extracted already. The first conclusion is that wireless IP cameras are practical and functional tools to help building teleoperated robots. Another conclusion is that it is possible and plausible to use odometry measurement using MonoSLAM but, as expected, we have considerable error accumulation. Further work is necessary to improve our strategy, from the Mono SLAM robustness to a better homing procedure. As a general conclusion, we can say that is already possible to build a teleoperated vehicle with autonomous features at a reasonable cost.

Acknowledgments

This work has been partially supported by the FCT project PEst-OE / EEI / LA0009 / 2013, by the FCT project PTDC / EEACRO / 105413 / 2008 DCCAL, and by the FCT project EXPL / EEI-AUT / 1560 / 2013 ACDC.

References

[1] J. Civera, O. Grasa, A. Davison, and J. Montiel. 1-point ransac for ekf filtering. application to real-time structure from motion and visual odometry. *J. Field Robot.*, 27(5):609–631, 2010.

[2] James Crosetto, Jeremy Ellison, and Seth Schwiethale. Control an rc car from a computer over a wireless network. `https://code.google.com/p/networkrccar`.

[3] J. Gaspar, N. Winters, and J. Santos-Victor. Vision based-navigation and environmental representations with an omni-directional camera. *IEEE Transactions on Robotics and Automation*, 16:890–898, 2000.

[4] N. Leite, A. Del Bue, and J. Gaspar. Calibrating a network of cameras based on visual odometry. In *Proc. of IV Jornadas de Engenharia Electrónica e Telecomunica cões e de Computadores, Portugal*, pages pp174–179, 2008.