A Low-Cost, Practical Localization System for Agricultural Vehicles

Gustavo Freitas, Ji Zhang, Bradley Hamner,
Marcel Bergerman, and George Kantor

Electrical Eng. Dept., Federal University of Rio de Janeiro, Brazil
Field Robotics Center, Robotics Institute, Carnegie Mellon University, Pittsburgh, US
gfreitas@coep.ufrj.br, zhangji@andrew.cmu.edu, bradley.hamner@gmail.com,
marcel@cmu.edu, kantor@ri.cmu.edu
http://cascrop.com/

Abstract. This paper addresses the refactoring of an agricultural vehicle localization system and its deployment and field-testing in apple orchards. The system enables affordable precision agriculture in tree fruit production by providing the vehicle's position in the orchard without the use of expensive differential GPS. The localization methodology depends only on the wheel and steering encoders and the laser rangefinder already on the vehicle for row following, thus adding zero hardware cost to the overall setup. It employs an Extended Kalman Filter to integrate the information from the sensors, with the pose being predicted via encoder odometry and updated via point and line features detections. The objective of this paper is to describe the complete refactoring of the initial proof-of-concept localization system, with the goal of making it robust, modular and reusable. Field test results indicate that the final system has sufficient accuracy for deployment of autonomous vehicles in tree fruit orchards.

Keywords: Autonomous Agricultural Vehicles, GPS-Free Localization, Extended Kalman Filter.

1 Introduction

Specialty crops are defined in US as fruits, vegetables, tree nuts, dried fruits and nursery crops, including floriculture. In 2007 they accounted for almost 17% of the US agricultural market, or US$50 billion, with fruit and tree nut production alone generating about 13% of all farm cash receipts [6]. Within specialty crops, tree fruit production is particularly challenged by the large cost of labor and its seasonal needs-for example, seven times more apple orchard workers are needed in the state of Washington during harvest season than during the winter pruning season. Today, there is a real opportunity to introduce automation solutions into tree fruit production to lower labor costs, smooth out labor requirements, and increase production efficiency. This opportunity is compounded by the introduction, in the past twenty years, of high-density "fruit wall" planting architectures.

C.-Y. Su, S. Rakheja, H. Liu (Eds.): ICIRA 2012, Part III, LNAI 7508, pp. 365–375, 2012.

Autonomous vehicles driving down along these rows of structured trees can mow and spray, as well as carry workers pruning, thinning, performing tree maintenance, and harvesting. Figure 1 shows two of the four vehicles we developed and deployed since 2008. Together, they logged a combined 350 km in research and commercial apple orchards in several US states. The vehicle on the left, Laurel, is a development platform where we test new row following and turning and localization methods and algorithms before integrating them on the other vehicles. It is equipped with wheel and steering encoders, laser range finders, cameras, and a differential GPS receiver used to generate ground truth for driving and localization experiments. The vehicle on the right, Cascade, is a barebones version with only the encoders and the laser. Cascade and its "twin" Allegheny are equipped with a scissors lift from where workers prune, thin, tie trees to wires, place pheromone dispensers, etc. They are used on a weekly basis by Extension educators and growers in time trials comparing the performance of workers on the platform versus workers on ladders.

Fig. 1. (Left) Autonomous orchard vehicle "Laurel." This experimental vehicle is equipped with encoders, laser rangefinder, cameras, and a differential GPS receiver for ground truth. (Right) "Cascade" in use by workers at Allan Bros. Orchards, in Prosser, WA, to thin green fruit. Time trials show that work on the top half of the trees, when conducted from the autonomous platform, can be more than twice as efficient than from ladders.

Vehicle localization is key to introduction of autonomous vehicles in tree fruit orchards, for various reasons. First, it enables precision agriculture applications similar to those performed today in field crops (corn, wheat, soy, etc.). For example, with a vehicle that knows how much fertilizer or herbicide each tree received, growers can correlate future yield with past maintenance records, and thus be able to manage production at a much finer scale. Second, it enables practices that are prohibitively costly today, such as counting and sizing fruit on the tree and creating yield maps. Last, but not least, accurate vehicle localization may increases accuracy of row following. While our vehicles' row following performance

is satisfactory, the current navigation system employs a pure-pursuit approach [1] and does not rely on the vehicle's pose inside the orchard. This information could improve the navigation reliability and safety.

In four years of work developing autonomous orchard vehicles for the apple industry, we concluded that a key requirement is to be able to localize the vehicle within a half-meter of its true position. Such sub-metric accuracy can be obtained with differential GPS receivers, but their cost is unrealistic when compared to the target cost of the vehicle itself. Standard GPS is cheap, but may provide position errors of up to 20 m, or the equivalent to four of five rows of trees. Besides, GPS signals get degraded under the heavy canopies adopted by some growers.

The original localization system [3,4] is a proof-of-concept implementation designed to validate the hardware and the extended Kalman filter (EKF) algorithm. It presented the accuracy necessary for operations in a variety of typical orchards. This paper describes the refactoring of the localization software in a way that makes it robust, modular and reusable. Our approach is to list the main shortcomings of the initial implementation and demonstrate, via actual experimental results, how they were addressed in the refactored software. The new system was validated over 5 km of driving in orchards in Pennsylvania and Washington.

The paper is structured as follows. Section 2 presents in more detail the autonomous orchard vehicle used in this work. Section 3 presents the localization system from a functional perspective. Section 4 discusses the original implementation and the results it affords. Section 5 presents the refactored software, and Section 6 presents the experimental results obtained. Section 7 presents an analysis of the results and a discussion on future work.

2 Autonomous Orchard Vehicle

The base vehicle used in this work is Laurel (Figure 1, left). It is based on the Toro MDE eWorkman electric utility vehicle, retrofitted to function either in manual or drive-by-wire mode. Laurel is a research vehicle, where we implement and test orchard navigation technologies before they are ported to others. It is important to note that, while Laurel is equipped with a high-accuracy Applanix POS 220 LV GPS-assisted inertial navigation system, we do not use it for the localization estimation proposed here. The Applanix data is used only during the EKF setup, including the process of mapping the orchard block where we operate. It also provides ground truth so we can assess the performance of the localization system. The presence of the Applanix does not constitute an operation limitation, as explained in Section 7.

The relevant sensors for this work are: steering and wheel encoders with linear resolution of 2.33×10^{-5} m/tick and angular resolution of 0.38°/tick; one SICK LMS 291 laser rangefinder with 180° field-of-view, 1° resolution, and maximum scanning range of 80 m; and a SICK LMS 111 laser rangefinder with 270° field-of-view, 0.5° resolution, and maximum scanning range of 30 m. The onboard computer is a rugged, waterproof, industrial unit with an Intel Core 2

Duo 1.6 GHz CPU with 4GB DDR2 DRAM from Small PC. The localization software runs on Ubuntu Linux, with the message passing provided by the Robot Operating System (ROS).

3 Localization Methodology

The localization system estimates the position of the vehicle in the orchard using an Extended Kalman Filter. The orchard terrain is assumed to be locally flat, and the pose is defined as $x = [x_r, y_r, \theta_r]^T$, where x_r, y_r represent the vehicle's planar position and θ_r its orientation with respect to an initial predefined configuration.

In the prediction step, the EKF uses the wheel and steering encoders to estimate the vehicle's pose. In the update step, it uses point and line features from the environment, obtained from laser data, to correct the initially estimated value. The system's simplified functional architecture is illustrated in Figure 2.

The proposed solution assumes the vehicle starts in a known initial position and orientation with respect to a previously-built reference map that contains the tree rows' ending positions. The current procedure used to create the map

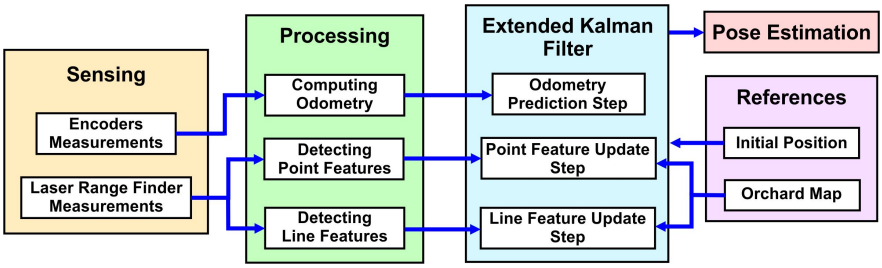

Fig. 2. Functional architecture of the localization system. The Extended Kalman Filter uses data from encoders and laser rangefinder to estimate the vehicle's pose. The software uses as reference the initial position and an orchard map obtained off-line.

Fig. 3. (Left) Apple orchard with reflective tape installed on the row ends for mapping purposes. (Right) The resulting orchard map, used for vehicle localization.

consists on driving the vehicle around the orchard and combining the measurements from the Applanix and lower laser to obtain the position of landmark reflective tape placed on the row ends (Figure 3).

4 Original Localization Estimation System

All of the orchard vehicle's software runs on Willow Garage's ROS, an open source framework for robotic system development, which in turn runs on Ubuntu Linux [5]. ROS is a useful tool because it provides a structured communication layer above Linux. A particularly useful ROS feature is the recording of messages in "bag" files. With them, it is possible to playback the messages to recreate past experiments, which is specially important during software implementation and debugging.

Each software module is implemented as a ROS node. The nodes exchange information by subscribing and publishing messages to ROS topics. The localization node subscribes to the sensor topics, receiving input measurements to run the EKF. After processing the information, the node publishes the pose estimation into another topic.

The localization node begins operation by loading the orchard map. It then acquires the vehicle position and subscribes to different sensor topics. After initialization, the node enters a polling mode running at 100 Hz. At every cycle, it checks for new measurement messages. As they arrive, it calls the respective EKF steps - prediction for encoder measurements and update for point and line features. The odometry messages are published at 45 Hz, laser measurements

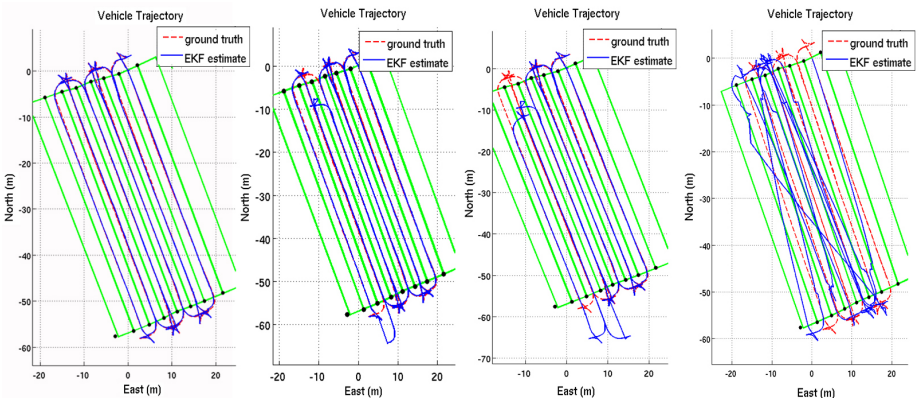

Fig. 4. Original localization system results using playback data. The left image shows a good result where the system achieved the necessary sub-metric accuracy. The other images contain results with progressive faults, with playback data yielding different results every time it is processed off-line. The errors are a result of the manner with which ROS handles messages.

for landmark detection at 75 Hz and line feature detections at 15 Hz. The localization node publishes a new pose message at each polling cycle that invokes the Kalman filter.

To evaluate the performance of the original localization system, we used playback data from bag files recorded during experiments conducted in September 2010 in our half-acre experimental nursery. This is a block with eight tree rows 54 m long and 3.5 m wide. The original implementation is able to reach the required sub-metric accuracy, as presented on the left in Figure 4.

The original software, however, is not always consistent, presenting different results when processing off-line the same playback data (Figure 4, center and right). The different results are caused by problems related to the ROS messaging mechanism, which may lose messages or process them out of chronological order. Despite achieving the accuracy required for the task we are pursuing, the original localization system implementation is not robust to ROS's asynchronous messaging scheme.

5 Refactored Localization Estimation System

Once the localization approach was validated, we set out to refactor the code so as to deal with the ROS message passing inefficiencies and also to make it modular and reusable in future projects. The refactored localization system implements the EKF through object-oriented software written in C++.

In the new version, the localization software is still implemented as a ROS node. As before, it begins operation loading the orchard map, acquiring the vehicle's position and subscribing to different sensor topics.

The difference is after initialization, when the new localization software operates similarly to an interruption-based system, in comparison to the original program. This is implemented using the command *ros::spinOnce()* inside a while loop running at 1 KHz. Whenever the node receives a ROS message, it calls the correspondent EKF step. As before, the odometry messages trigger the prediction step, and the point and line features activate the update step.

The software main algorithm is briefly described in the following pseudo-code:

```
global applanixDataMsg, encoderDataMsg, pointFeatureDataMsg, lineFeatureDataMsg;

void treatRosApplanix(applanixDataMsg);
void treatRosEncoder(encoderDataMsg);
void treatRosPointFeat(pointFeatDataMsg);
void treatRosLineFeat(lineFeatDataMsg);

int main () {

 poseEstimation = [x, y, theta];

 loadMap();
 getInitialPosition(applanixDataMsg);

 ros::Subscriber applanixDataSubscriber => treatRosApplanix(applanixDataMsg);
 ros::Subscriber encoderDataSubscriber => treatRosEncoder(encoderDataMsg);
 ros::Subscriber pointFeatDataSubscriber => treatRosPointFeature(pointFeatDataMsg);
 ros::Subscriber lineFeatDataSubscriber => treatRosLineFeature(lineFeatDataMsg);
 ros::Publisher poseEstimationDataPublisher => poseEstimationDataMsg;
```

```
 while(true){
  ros::spinOnce();
  loop_rate.sleep(1KHz);

  if encoderDataMsg
   odometryPredictionStep(encoderDataMsg, poseEstimation);

  if pointFeatureDataMsg
   pointFeatUpdateStep(pointFeatDataMsg, poseEstimation);

  if lineFeatureDataMsg
   lineFeatUpdateStep(lineFeatDataMsg, poseEstimation);

  poseEstimationDataPublisher(poseEstimation);
 }
}
```

Considering the frequencies at which each component of the localization system runs (see Section 4), we expected it would be capable of processing all sensor measurements. During debugging procedures, however, we identified that the node loses 5% of the odometry and line feature messages. The problem is aggravated for landmark detection, with 50% of the messages not being processed. We also noticed that the localization software does not receive the different sensor messages in chronological order.

These problems were reported to the ROS mailing list (see `http://bit.ly/LggVu7`). In response, we were informed by Willow Garage that the time when messages buffers are serviced can vary greatly, independently from the frequency of the main node. They also confirmed that when receiving messages from different sources, ROS does not guarantee the order in which they will be received and processed.

To solve the problems, we used vectors to store messages coming from different sensors. An algorithm looks into the vectors and selects which message should be sent to the localization node. The code does not lose measurements, and also selects the messages in chronological order before calling the EKF steps.

During the software refactoring, the code was divided into auxiliary functions and files, increasing its modularity, and better documented. The final software retains the accuracy of the original one and is simpler, clearer, easier to understand, and easier to adapt to other applications.

Figure 5 presents the result obtained with the refactored software when processing the same playback data in Figure 4. After the refactoring, the localization system presents consistent results and yields the exact same results every time it is processed off-line.

Because of the line feature corrections, the maximum crosstrack error is about 0.5 m. The small errors are caused by the line detection process. The line feature parameters corresponding to the tree rows are estimated by a particle filter [2]. Due to the canopy shape, the parameters are constantly changing. When the vehicle drives in straight line, the dead reckoning longtrack error accumulates due to small errors propagation. The maximum longtrack error of 0.55 m corresponds to 1% of the row length, compatible with the expected wheel slippage deviation. The error is corrected at the row ending, when the vehicle identifies a landmark.

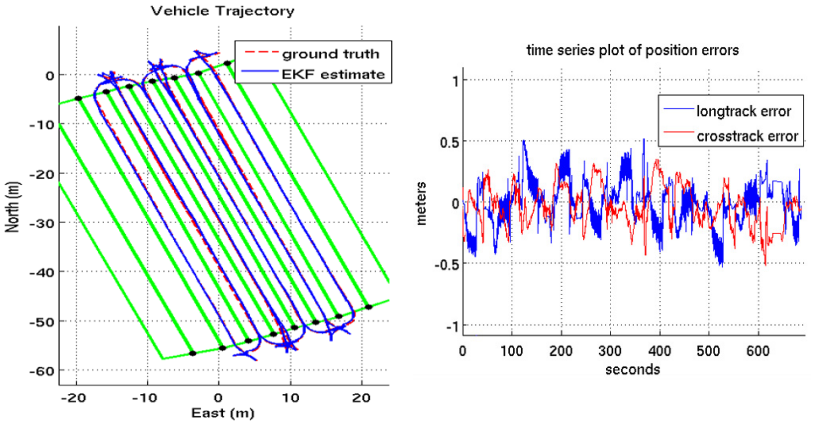

Fig. 5. Trajectory and position error obtained by the refactored localization software processing playback data acquired during field tests

Figure 6 presents the distribution of errors for the experiment in Figure 5. The crosstrack error is within 30 cm of the ground truth for more than 95% of the time. The 3σ (99.7% of final value) interval is 0.45 m. The longtrack error is also lower than 0.5 m during almost the entire operation, except at a few isolated locations. The position estimate is within 30 cm of ground truth for more than 90% of the time. The 3σ (99.7% of the steady state) interval is 0.5 m.

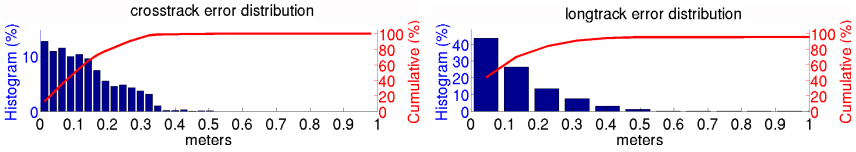

Fig. 6. Crosstrack and longtrack error distributions obtained with the refactored localization software. The maximum crosstrack error is about 0.5 m and the localization estimate is within 30 cm of ground truth for more than 95% of the time. The longtrack error is also within 30 cm of ground truth for more than 90% of the time.

6 Experiments in Apple Orchards

To assess the performance of the refactored localization system, we conducted several field tests at Washington State University's Sunrise Orchard in Rock Island, WA, and Ridgetop Orchards in Fishertown, PA. At both locations we drove the vehicle manually inside the tree rows, with the goal of verifying the localization system's robustness and functionality over long-term operations in real crops.

Figure 7 presents two representative results at Sunrise; Table 1 summarizes three of them. The vehicle traveled a total of 5,924 m in blocks 9A, 9B and 9C, and its position was estimated with sub-metric precision. The mean errors range from 0.17 to 0.23 m, and all the 3σ distances are less than 1 m. The good results are in part due to the flat, dry terrain, which don't generate much wheel slippage, and short tree rows, which brings the landmarks into view more quickly, reducing odometry-associated errors. All were obtained on-line as the vehicle drove in between rows.

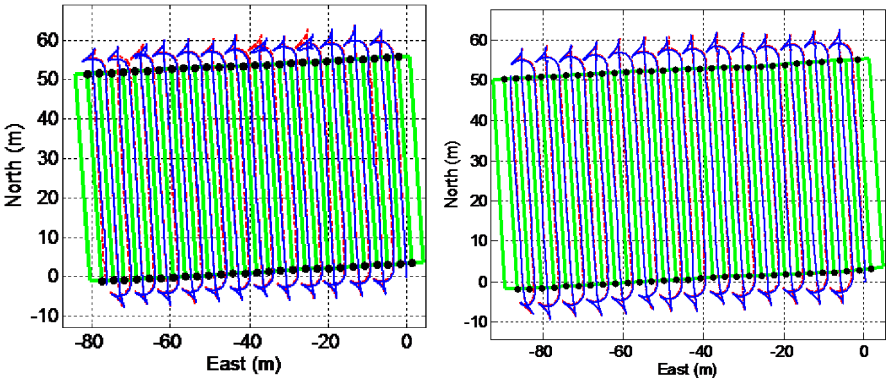

Fig. 7. Trajectory obtained by the localization system at Sunrise Orchard's blocks 9A and 9B. The estimated position is plotted in blue, and the Applanix ground truth data is plotted in red.

Table 1. Summary of experiments conducted at Sunrise

Test site	Block 9A	Block 9B	Block 9C (*)
Total distance [m]	1898	2136	1893
Mean longtrack error [m]	0.22	0.19	0.21
Mean crosstrack error [m]	0.23	0.17	0.22
3σ longtrack error [m]	0.91	0.65	0.98
3σ crosstrack error [m]	0.76	0.51	0.72

(*) Not shown in Figure 7.

Figure 8 presents the results obtained at Ridgetop Orchards. This is a much more challenging environment, with rows that are 300 m long and very steep (up to 12.5° of inclination). At Ridgetop we traversed four rows. When the robot is driving inside the rows, there is no correction for the longtrack error and the encoder odometry drifting accumulates. The maximum long track error is about 2 m when the vehicle is going downhill and up to 6 m when going uphill, again compatible with wheel odometry.

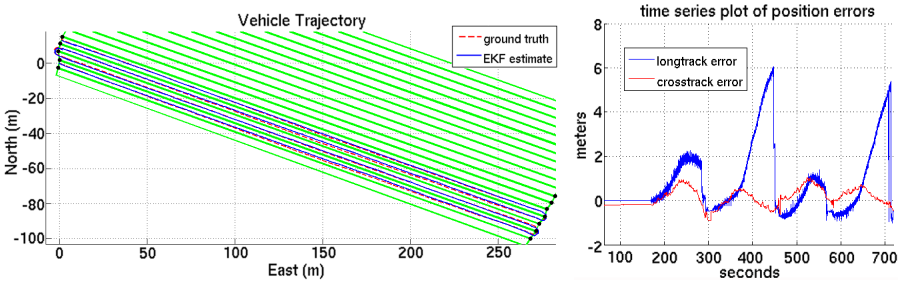

Fig. 8. Trajectory and position error obtained by the localization system at Ridgetop Orchards

7 Conclusion and Future Work

The localization system described in this paper is part of the larger goal of demonstrating the feasibility of operating autonomous orchard vehicles in commercial tree fruit production environments year-round. We started with a system that provided the necessary accuracy for most operations and refactored it to end with a more reliable, modular, and reusable version. The documentation includes three user manuals with detailed instructions on how to generate the orchard map, execute the localization software, and analyze the results. Fifteen experimental data sets are included in the documentation for development and test of future improvements

Experiments conducted in apple orchards showed that the new localization system presents satisfactory results, obtaining sub-metric precision in locations with relatively flat terrain. It does not meet the required accuracy, however, when operating in very long or very steep rows. The former cause EKF updates to take too long to kick in, leading to odometry drift; the latter causes wheel slippage that confound the odometry. In the worst case, when driving uphill on a very long and steep row, the localization error reached 6 m, or 2% of the traveled linear distance.

One possible solution to improve accuracy consists on computing odometry using cameras, because visual odometry is not affected by wheel slippage. We have begun work in this direction and expected to enhance the localization system accuracy by one order of magnitude.

The localization methodology was tested with data from two laser rangefinders, one detecting point and the other detecting line features. In an actual field deployment, only one laser rangefinder would have to provide data to both the navigation and localization. Also, Laurel is the only vehicle using the Applanix high-accuracy localization system. The others rely only on the system presented here to obtain position estimation. The vehicles do not have GPS, requiring alternative solutions to obtain the initial position and also the orchard map.

For initialization purposes, the vehicle can start the operation in a known position. Regarding the mapping requirement, one procedure already in use consists

on placing a low-cost GPS receiver at each landmark, or even only the four corners of the block, for about one hour, and correcting the position data with RTK post-processing data from the USGS. In the future, the mapping procedure may even be eliminated entirely and replaced with a simultaneous localization and mapping (SLAM)-type method.

Acknowledgments. We would like to thank all members of the Comprehensive Automation for Specialty Crops. We also thank the crew from Sunrise and Ridgetop orchards were we tested the APM and the localization system.

This work is supported by the US Department of Agriculture under the Specialty Crop Research Initiative, award number 2008-51180-04876. Gustavo Freitas is funded by a grant from the Brazilian National Council for Research and Development (CNPq).

References

1. Coulter, R.: Implementation of the pure pursuit path tracking algorithm. DTIC Document, Tech. Rep. (1992)
2. Hamner, B., Bergerman, M., Singh, S.: Autonomous orchard vehicles for specialty crops production. In: ASABE Annual International Meeting, Louisville, Kentucky (2011)
3. Libby, J., Kantor, G.: Accurate GPS-free positioning of utility vehicles for specialty agriculture. In: ASABE Annual International Meeting, Pittsburgh, PA (2010)
4. Libby, J., Kantor, G.: Deployment of a point and line feature localization system for an outdoor agriculture vehicle. In: IEEE Int. Conf. on Robotics and Automation, Shangai (May 2011)
5. Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
6. Singh, S., Bergerman, M., Cannons, J., Grocholsky, B.P., Hamner, B., Holguin, G., Hull, L., Jones, V., Kantor, G.A., Koselka, H., Li, G., Owen, J., Park, J., Shi, W., Teza, J.: Comprehensive Automaton for Specialty Crops: Year 1 results and lessons learned. Journal of Intelligent Service Robotics (July 2010)