Canny Edge Based Self-localization of a RoboCup Middle-sized League Robot

Yoichi Nakaguro*

*Sirindhorn International Institute of Technology, Thammasat University
P.O. Box 22, Thammasat-Rangsit Post Office, Pathumthani 12121 Thailand
Email: ynaka96@yahoo.com

I. INTRODUCTION

In the RoboCup middle-sized soccer league, autonomous robot players in the field are required to localize themselves in real time by utilizing the sensors mounted on themselves. There could be many different kinds of sensors that can serve as part of the solution to this task. Especially, vision sensors are one of the most promising because they are small, light-weight, inexpensive, and the most importantly, easy to be implemented and integrated as a real time subsystem in a mobile robot. Although the accuracy of vision sensors is subject to noise from camera configurations and environmental conditions, well-established image processing techniques are available and expected to overcome the difficulties.

In this project we adopted an omni-directional vision sensor attached on top of a RoboCup middle-sized league robot (Corbenic) as the only available sensor for the self-localization purpose. The vision sensor has a perceptive range which is enough to cover the most of the soccer field regardless of robot’s locations. From the omni-vision camera, spherical images are constantly captured at the rate of 30 frames per second. A GPU (Graphics Processing Unit) on the robot’s main board is capable of carrying out the Canny edge detection algorithm at the same rate of the image updates. The goal of this project is to design and implement a real time subsystem capable of self-localizing a robot in the soccer field by utilizing the Canny edge information.

We designed and implemented a localization method which takes Canny image data as input and provides a robot’s pose (2D position and orientation) as output. An experiment was conducted using omni-vision images captured in a real scale RoboCup soccer field in University of Ulm. The result showed that our method is highly feasible and expected to work even better with the help of estimation theories such as Kalman filtering and/or fusion with the other kind of sensors such as wheel encoders and polarization detectors.

II. METHODOLOGY

A. Vision Based Self-localization

Figure 1 shows the overview of our vision based self-localization system. The self-localization method requires two kinds of basic data, one is spherical Canny images from GPU and the other is polarization information. Both information is updated at the rate of 30 frames per second. Canny images are used to infer robot’s pose (2D position and orientation) with respect to the soccer field. However, the determination of robot’s position is up to the center line symmetry, i.e. we cannot decide which side of the half fields the robot is in. This is because the geometry of the lines which compose the soccer field is symmetric with respect to the center line. In order to remove out the ambiguity, polarization information should be provided. Polarization is treated as one of the other sensor modules which is out of the scope of this project and it is not discussed further here.

B. Canny Edge Based Self-localization

In this subsection we explain the flow of our robot’s pose estimation subsystem (Figure 2). The subsystem takes a spherical Canny edge image as input and provides robot’s pose (with the center line symmetry ambiguity) as output. A Canny edge image provided by GPU is parsed and the edge information is stored in an appropriate data structure for further processing. Edges are segmented into line segments with reasonable lengths. Each segmented line is then mapped into a 2D Robot coordinate based on the mirror formula of the omni vision camera system. At this moment all the image
distortion is removed (or the image is “rectified”) and the transformed line segments can be directly compared with the real geometry information of the soccer field.

To obtain the two principal orientations (vertical and horizontal directions) of the soccer field with respect to the robot, line segments are further mapped into the Hough space where each line segment is described by its orientation and distance from the principal point of the 2D robot coordinate. After noise removal in the Hough space, we determine the two principal orientations of the set of line segments by looking at the angular distribution of the line segments. Note that although we detect the two major orientations that are perpendicular to each other, we do not yet determine which orientation corresponds to vertical or horizontal one at this moment.

The center point of the soccer field is one of the most distinctive feature points in the soccer field model because it is surrounded by a circle with exactly one meter radius. The center circle is detected by analyzing the angular distribution of the rectified line segments in the Hough space. Successful detection of the center circle leads to the detection of the center point of the field, and then to the robot’s position relative to the center point.

The detected center point of the soccer field is used to solve the vertical and horizontal ambiguity of the principal orientations of the soccer field. To do so, we first detect the center line of the soccer field which is passing through the center point, and then see which of the two principal angles lines up with the orientation of the center line.

Given the robot’s position relative to the center of the soccer field and the orientation of the soccer field relative to the robot, it is now possible to calculate the robot’s pose with respect to the field.

III. IMPLEMENTATION

In this section each key process in our Canny edge based self-localization is described in more detail from the practical point of view. Corresponding functions implemented in the C programming language are found in file src/sensors/vision/localize/VisionLocalize.c within the Corbenic project source tree.

A. Parsing a Canny Image

Canny edge detection is carried out on GPU against spherical images captured by an omni-vision camera mounted on the robot. The processing speed is 30 frames per second. The output is a Canny image which is a sequence of bytes with a positive integer value if the pixel is part of an edge, or zero if not. In this step, the raw Canny image should be parsed and the resultant edge data should be stored in a data structure in memory for further processing. At the time of writing, this part is still under development.

B. Line Segmentation of Canny Edges

In this step each edge is segmented and linearly approximated by a set of line segments. Line segments with lengths shorter than a reasonable threshold are discarded. The resultant line segments are stored sequentially in memory, i.e. any two line segments next to each other in the pixel space are also stored successively within the line segment data structure. This process is done by function SupportImage2lnseg() which is defined in src/sensors/vision/localize/CorbenicSupport.h.

C. Rectification of a Spherical Image

Line segments are mapped from the spherical image coordinate to the 2D robot coordinate. PointTransform Rectify() does this task. The function maps two end points of a line segment to two points in the robot coordinate. The mapping function (mirror formula) is a nonlinear function which depends solely upon the pixel distance from the center of the spherical image. Although the mirror formula was theoretically derived and implemented in src/sensors/vision/VisionSensor.cpp, we found that it did not remove the distortion correctly especially for pixels away from the center of the image. Instead we fitted the nonlinear mirror formula with an eight dimensional polynomial function and used it for the rectification.

D. Principal Orientations Detection in the Hough Space

LinesCalcInfo() calculates the Hough parameters (vertical distance between a line segment and the center of an image, and the orientation of the line) of rectified line segments. A set of the rectified line segments typically contains many noisy ones because there are many edges seen in a Canny image that are not exactly lying on the ground such as robots, goal posts, objects outside of the soccer field etc. Those edges after rectified tend to produce long line segments passing through the center of the new coordinate system, however it is relatively easy to remove them. Function CorbeLineSegData CheckEveryCondition() excludes noisy line segments, such as ones out of the camera’s perceptual range, ones whose length is too short or long and ones that pass near the center of the rectified coordinate.
After the noisy line removal, function CorbeLineSegData.FindPrincipalAngle() finds the two principal orientations (vertical and horizontal orientations) of the soccer field. It is done by a simple bin counting algorithm. We quantize the total angle into a set of bins. For each quantized bin we count the number of line segments whose orientation fall into it or the other bin perpendicular to the corresponding bin. The two principal orientations are determined by taking the bin pair which has the largest counting score.

E. Center Circle Detection

The purpose of this step is to detect the unique center circle with one meter radius. After the line segmentation of the Canny edges and rectification, we can find some line segments which are linear approximating arcs of the center circle. If two of such line segments are “next to” each other (spatially close, originally belonged to the same edge and was stored successively in the line segment data structure), it is likely that the intersect of normal lines which passes through the middle points of the two line segments sits close to the center of the circle we are searching for.

CorbeLineSegData.FindCenterCircle() sequentially looks through the line segment data. For each line segment pair which is likely to be two approximated arcs of the same circle, it calculates the intersect of normal lines of the two segments. If the calculated intersect is approximately one meter away from the corresponding two line segments, the intersect is marked as a candidate of the center circle. The unique center point is chosen by a voting scheme. For each candidate of the center points, the function counts the number of candidates within its neighbor. It then chooses the candidate which has the largest number of neighbor candidates and calculates the mean position of the neighbor candidates, which becomes the estimated center of the center circle.

F. Calculation of Robot’s Pose

CorbeLineSegData.ChooseHorAngle() removes the vertical-horizontal ambiguity of the two principal orientations in the previous step. Given the estimated center of the soccer field, it compares the orientations of the center line (more specifically, orientations of line segments which compose the center line) which passes through the center point, and two of the principal directions. One of the principal orientations parallel enough to the center line becomes the horizontal direction of the field, and the other becomes the vertical direction.

CalcRobotPose() takes the robot position relative to the center point and its orientation from the horizontal direction of the field. In the function the original robot position relative to the center point is rotated back by the robot’s angular shift from the horizontal direction in order to get the real robot’s position relative to the soccer field.

IV. EXPERIMENTAL METHODS

To test our omni-vision based localization subsystem, we conducted an offline experiment using real images captured by an omni-vision camera mounted on Corbenic (a prototype robot for RoboCup middle-sized soccer league built by a team of Hochshule Ravensburg-Weingarten). We simulated by manual control typical robot moves that could happen in an actual game such as translation, rotation and both at the same time in the middle-sized soccer field hosted by a team of University of Ulm. The path of the robot was designed so that it would cover every possible robot position within the soccer field. During the run, we captured approximately 1200 omni-vision JPEG images (1024x768 pixels) at the rate of 30 frames per second (Figure 3). For visual inspection of the experimental result, we made an MPEG format video from the images.

Since the integration of the self-localization subsystem and the Canny image parser were not ready at the time of the experiment, we instead extracted Canny edges from the JPEG images by a utility function running on CPU and passed the edge data to the subsystem as input. To see the performance of our algorithm, we collected and analyzed estimated robot positions and orientations relative to the soccer field.

V. RESULTS

Since each robot’s pose was estimated per image basis, i.e. neither previous image nor previously estimated pose was not used for the estimation of the current pose, it is important to see whether estimated poses compose a consistent path though over 1200 frames. The path should be continuous and there should not be any significant number of missing points wherever the robot is running in the soccer field. To see that, it is more convenient to focus on the transition of the estimated center point relative to the robot (Figure 4) rather than seeing the estimated path of the robot in the soccer field coordinate (Figure 5). This is because the latter contains a geometric ambiguity due to the symmetry of the soccer field with respect to the center line. Figure 4 shows that the estimation of the center point was conducted successfully because the transition of the estimated positions is mostly smooth and continuous although a small degree of noisy estimations are observed.
Fig. 4. Transition of the estimated center point of the soccer field with respect to the robot.

Fig. 5. Transition of the estimated robot position in the soccer field.

Fig. 6. Transition of the estimated principal orientations of the soccer field.

Figure 6 shows the result of the estimation of soccer field orientation relative to the robot. The estimated principal directions of the soccer field are plotted. Using the MPEG video we visually verified that the transition of the angles are accurate. Since we already have the differentiation of the vertical and horizontal orientations of the field, robot’s orientation relative the soccer field is easily derived from the result.

Figure 5 is a plot of the estimated robot’s positions. Since the polarization information was not provided yet in the subsystem, the determination of a position is up to the center line symmetry of the soccer field. Therefore every point was plotted on the half of the actual soccer field for visualization of the result. However, We verified that the absolute values of the robot’s position parameters were estimated correctly.

VI. CONCLUSION

We have successfully designed and implemented a subsystem which is capable of providing self-localization information on a RoboCup middle-sized league robot. Although the estimation of robot’s position contains an ambiguity with respect to the center line symmetry of the soccer field, we expect that it will be easily resolved by combining the result with polarization information provided by the other sensor model.

It should be noted that our localization method is based only upon the geometry of the soccer field. Output images of the omni-vision camera has color information, but such information was never used for localization in this case.

Although the result of our per-frame based localization looks very robust in ordinary cases, there are some limitations of the algorithm. The first is the time cost of the localization process. The total RoboCup system is expected to run in real time. Considering the speed of the Canny edge detection on the GPU is 30 frames per second, our self-localization is expected to run at the rate of at least several frames per second. After the completion of the Canny image parser function, it is required to benchmark the time cost of the localization system, find possible bottlenecks and optimize them if necessary.

The second major limitation of the current algorithm is that a successful detection of the center circle is inevitably required. In highly dynamic RoboCup games, occlusions of the center circle by robots in the playground will happen very often. To realize robustness of self-localization in such a scenario, an application of some estimation theory such as Kalman filtering or particle filtering is strongly expected.
VII. EXTENSION OF THE WORK

In this section we list up some possible extensions regarding the project.

- Complete the Canny parser function.
- Integrate the self-localization subsystem (C code) under Corbenic source tree (C++ code). Currently offline experiments using JPEG images work, however we have to solve the name mangling problem when calling our localization functions from the upper system.
- Sensor fusion with polarization data to obtain the unique estimate of the robot’s position in the soccer field.
- Benchmark the algorithm and optimize bottleneck processes to make it real time.
- Introduce estimation theories (e.g. Kalman Filter and Particle Filter) to realize more robust inter-frame self-localization.

ACKNOWLEDGMENT

I would like to thank Dr. Wolfgang Ertel for supervising me and giving me this enjoyable and challenging project during my internship program at Hochshule Ravensburg-Weingarten. He was always patient with my progress on the project and kept showing me proper research directions.

The RoboCup team at University of Ulm kindly provided us with a precious opportunity to collect the real image data in their laboratory with a complete RoboCup middle-sized league soccer field. The success of our experiment was made possible thanks to their generosity.

My special thanks go to the folks in the RoboCup laboratory at Hochshule Ravensburg-Weingarten. They gave me many useful insights and suggestions when I got lost in the project. Especially, Mr. Arne Usadel supported me in both technical and management way.