Effective Localization of Humanoid with Fish-Eye Lens Using Field Line Detection

Pratyush Kar, Archit Jain
Dept. of Computer Science
BITS Pilani, India
e-mail: {pratyush.kar, amiarchit}@gmail.com

B. K. Rout
Dept. of Mechanical Engineering
BITS Pilani, India
e-mail: rout@pilani.bits-pilani.ac.in

***Abstract*—This paper presents a real-time technique for efficient localization of a humanoid robot in a soccer field using a fish-eye lens. We propose a novel method that uses node merging to provide a robust and an accurate estimate of positions of the intersection points in the field lines. The distances to these points are calculated using Inverse Perspective Mapping and fed to the particle filter for the purpose of localization. The proposed algorithms have been implemented and tested on the humanoid robot platform AcYut, being developed at the Centre for Robotics and Intelligent Systems (CRIS), BITS Pilani.**

Keywords-Localization; monte carlo; x-t detection; fish-eye lens; inverse perspective mapping; robocup

I. INTRODUCTION

Localization is a key task that an autonomous humanoid robot must perform to be successful as a soccer playing agent. The humanoid must be able to assert its position prior to it being able to perform tasks like path planning, obstacle avoidance, etc. This problem becomes more and more challenging as RoboCup increasingly discourage the use of colour based markers to assist in this endeavor. With the field becoming more symmetric and similar to a traditional football field, there is a need to come up with efficient and robust techniques to solve this problem. Due to the absence of markers on the field, it is essential to utilize localization cues such as the structure of the field lines to facilitate in the localization process. The use of line intersections in the localization is beneficial in two ways. Firstly, it improves the number of localization cues in the environment, thereby making the system more robust, as it reduces the dependence on any particular landmark. Secondly, it mitigates the errors caused due to the uncertainty in the position estimate of distant landmarks. This is particularly essential when precise positioning is required.

In humanoid vision systems, a rectilinear lens is generally used in the camera assembly. It has low distortion but also provides a low Field of Vision (FOV). This low FOV demands unnecessary actuator motion of the head and leads to inaccuracies (in distance estimation) generated due to the motion of the camera. Moreover, the number of objects of interest visible in a particular frame are relatively few. A fish-eye lens, on the other hand, offers a larger FOV, in the order of 185°, but at the same time has a large amount of radial distortion. This FOV alleviates the need for frequent head motion but gives rise to additional challenges in identifying the line intersection points. Due to the significant amount of distortion in the image the intersection points in the field lines appear severely skewed (see Fig. 1A). Although, there exist techniques to undistort the distorted image captured using a fish-eye lens [1], [2]. These techniques prove to be computationally expensive when applied to the entire image and often the undistorted image lacks the clarity to identify the line intersection points with accuracy. This paper proposes a method based on node merging to identify the line intersection points which is robust against the distortion caused due to the optics of a fish-eye lens. A process termed as digestion, explained in Section IV, is used to compute the node graph. This is utilized for the purpose of identifying the intersection points in the field lines.

The rest of the paper is organized as follows. Section II cites some works that have been pursued in this area. The robot test platform has been described in detail in Section III. For the purpose of explanation of the algorithm the cognition machinery has been divided into three independent sub-modules namely, Feature Detection, Feature Management and Localization. Sections IV, V and VI elaborates these modules respectively. Section VII discusses about the observations and results obtained in the experiments. Finally, Section VIII concludes the paper, with future perspectives.

II. RELATED WORK

The task of utilization of field lines for localization of a mobile robot can be broadly broken into three sub-problems: - identification of the candidate line segments, merging the line segments to form a meaningful representation of the field lines, and matching the identified features to the corresponding cues on the known map for the purpose of localization.

The candidate line segments are usually identified by using an appropriate mask and then convolving across the image. Previous research has utilized filters based on green-white-green transition [3] and techniques that perform vertical scans of the image [4]. Other approaches have used techniques based on Canny or Sobel edge detectors [5]. In our approach we first apply the aforementioned edge detection techniques followed by morphological dilation and skeletonization of the image to extract the characteristic structure from the input image.

978-1-5090-1362-3/16/$31.00 ©2016 IEEE

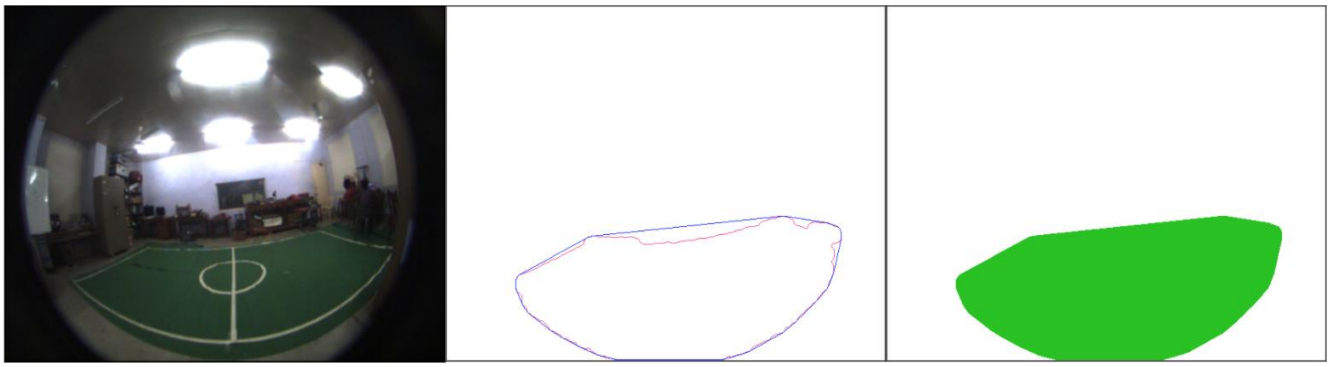

Figure 1. Computation of convex hull. (A) Image of the field captured using a fish-eye lens. (B) The red outline shows the contour of the green blob having the largest area. Its corresponding convex hull is drawn in blue. (C) Final region of interest (ROI) of the field.

Next these line segments need to be merged in order to obtain a more meaningful representation of the field lines. Traditionally this is done using Hough based techniques [6], [7] by transforming the detected line segments into the parameter space. This technique, however, requires a large accumulator array to be maintained and will suffer when used on images captured using a fish-eye lens, due to the excessive amount of radial distortion. The approach adopted by the Nao team NUManoid [3], where the lines are formed by repeated addition of detected points to a line across a gradient, also suffers from similar issues. The technique proposed by Team NimbRo [8] alleviates some of the issues of the previously mentioned techniques by computing the node graph on the skeletonized image but, relies on effective placement and connection of nodes. We have observed that errors in the detection of line segments, impact the connection of nodes which results in discontinuity in the computed node graph. This paper builds on this technique and proposes a new, robust method for the calculation of the node graph that is less susceptible to errors due to detection of spurious line segments.

The intent of this paper is to provide a generic technique for computing the node graph, for the field lines, that can be implemented on any camera system and is not restricted to a fish-eye lens-based vision system.

III. Humanoid Test Platform

The algorithm proposed in this paper has been implemented and tested on the humanoid robot platform AcYut being developed at the Centre for Robotics and Intelligent Systems, BITS Pilani. The latest version, AcYut VII is a 24-DOF humanoid robot and is equipped with a 1.7 GHz, 4th generation Intel Core i3-4010U processor. For the vision system, a 752 X 480 resolution camera (IDS UI-1221LE) is employed along with a fish-eye lens (Lensagon BF2M12520) and is capable of capturing images at 48 frames per second (fps). A 5m X 3m football field was used in the experimentation, in which the field lines were marked using 10cm wide white ribbon.

IV. Feature Detection

This module deals with the image acquisition from the camera and detection of features for the localization of the robot in the field. Features are defined as elements relevant for regional localization. Examples of features might be the ball, field lines, their intersections, opponents, and teammates. For brevity, only the feature detection of the field lines is explained here.

A. *Image Buffer*

The image frame captured from the camera, instead of being directly processed upon, is added to an image buffer that operates in a First in First out (FIFO) manner. When a new camera frame is added to the buffer, the oldest image stored in the buffer is released. The images stored in the buffer are averaged and the resultant image is used as an input to the algorithm. This method has several advantages, such as, it counteracts the variation caused due to the indoor lighting, there is no tearing between frames, and it also provides a smooth transition between two frames in which any object is displaced significantly. Furthermore, this technique is robust against errors in image frame acquisition from the camera. This averaged image is then subsampled to reduce the size to a 320 X 240 resolution image. This is done to reduce the computational overhead in the subsequent steps of the algorithm. For the purpose of illustration the image shown in Fig. 1A is considered as a sample input image for the rest of the paper.

B. *Field Detection and Boundary Extraction*

Before the image can be used for feature detection, a Region of Interest (ROI) needs to be computed where all the objects of interest are present. Since, all the objects of interest (field lines) are present on the field, the field must be identified and the boundary needs to be computed prior to the feature detection step.

The averaged image produced in the image acquisition step is converted from the standard RGB colour space to an HSV colour space. The converted image is made to pass through a min-max threshold cut-off filter with the colour range of green. The thresholded image is first morphologically eroded to remove small blobs that appear in the image and then it is morphologically dilated to fill up small holes. The contours of the image are calculated to produce polylines around each of the blobs. The contours for

the sample input image are shown in red in Fig. 1B. The area of the blobs is computed and the largest one is selected as the field contour. A convex hull enclosing this selected polyline is computed, which serves as the field boundary of the input image. The calculated field and its boundary are used as a mask to define the ROI for the process of line isolation. The convex hull for the sample input image is shown in Fig. 1C.

C. *Line Detection*

In the ROI computed above, a whitespace search and canny edge detection is used to identify any non-smooth or extremely steep changes in the image. These features are assumed to be field lines. The identified line segments are shown in Fig. 2A in blue. These features are morphologically dilated to make both edges of the field lines merge, remove any non-smoothness in the lines and merge any breaks in the detection. The image is then morphologically eroded and skeletonized to a pixel width using the Zhang-Suen thinning algorithm [9]. The usage of Zhang-Suen thinning algorithm ensures that the end points and the pixel connectivity in the 3X3 pixel neighborhood is preserved. The skeletonized image for the sample input image is shown in Fig. 2B.

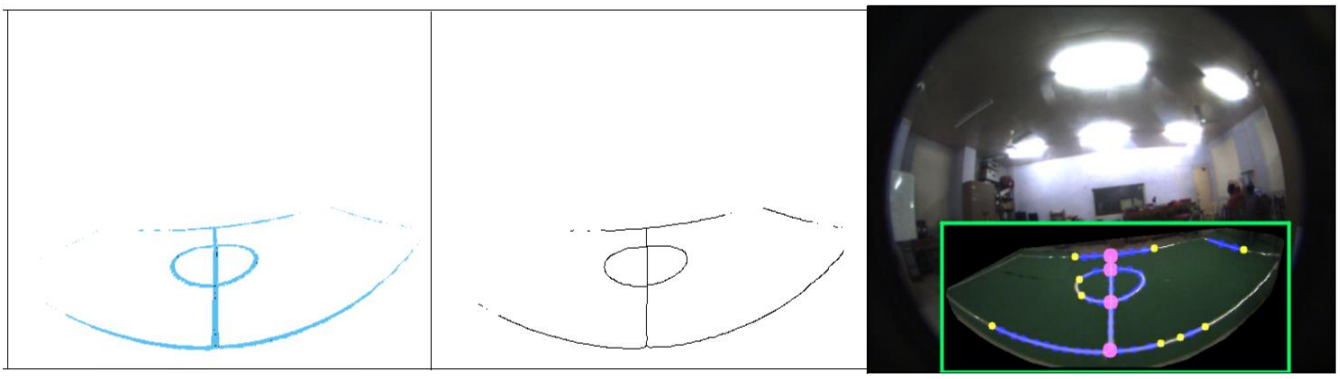

Figure 2. Calculating the node graph G. (A) Thresholded image showing the field lines in blue. (B) Skeletonized image of the field lines. (C) Final computed node graph overlaid on the original input image. Blue circles denote the major nodes, pink circles denote the detected feature points, and yellow circles show the end points of the connected line segments. The connection between the major nodes is shown in blue.

D. *Algorithm – Identifying Major Nodes*

To localize itself the robot needs to identify certain key points that can be used as localization cues. Easiest of all are the crossing over of field lines. To detect these points, we first assume that all the points on the skeleton are possibly key points, each of these points being called a node. Then we assign a certain weight to each node based on how many neighbouring pixels are node themselves (*low_wt*, *med_wt* and *high_wt* in order). To identify the major nodes we begin a process termed as digestion. In this process each node starts digesting all nodes, having a lower digestion weight, within a certain radius termed as digestion radius. When the node performing digestion encounters any other node, the node with the lesser weight is deleted and the other node continues the digestion with the weight of the deleted node added to its own weight. Algorithm 1 describes the technique for identification of the major nodes in additional detail.

Algorithm 1 Identify Major Nodes

```
img ← Input skeletonized image
sz ← Size(img)
white_cnt ← new Mat(sz)
digest_wt ← new Mat(sz)
cij ← new Mat(sz)
for all non-zero pixels (i, j) in img do
    cij[i][j] ← 1
    white_cnt[i][j] ← white count in the 3x3 neighborhood of pixel (i, j)
    if white_cnt[i][j] ≤ 1 then
        digest_ct[i][j] ← 0
    else if white_cnt[i][j] is 2 then
        digest_ct[i][j] ← low_wt
    else if white_cnt[i][j] is 3 then
        digest_ct[i][j] ← med_wt
    else
        digest_ct[i][j] ← high_wt
    end if
end for
for all non-zero pixels (i, j) in img do
    N ← all pixels inside the digestion radius of (i, j)
    (mi, mj) ← pixel ∈ N having max digest_wt
    digest_wt[mi][mj] ← sum of all digest_wt ∈ N
    for all (ki, kj) ∈ N – {(mi, mj)} do
        digest_wt[ki][kj] ← 0
    end for
    cij[mi][mj] ← 2
end for
```

Here *white_cnt*, *digest_wt* and *cij* are 2D vectors having size equal to the size of the image and are all initialised to 0. *white_cnt* is used to maintain the white count in the 3x3 neighbourhood of the pixel, *digest_wt* keeps track of the digestion weights in the digestion process and *cij* is used as a flag to differentiate between major nodes and secondary node (that are deleted in the digestion process) having values 1 and 2 respectively. After the digestion, we find that

the skeleton is simplified to include only major nodes and now these nodes can be connected. Digestion is tremendously advantageous as it inherently takes care of any aberrations in the skeleton and the generated major nodes tend to be in agreement with points of intersection of the field lines. In this algorithm, the digestion process has the most computation overhead. As the size of digestion radius is increased this computation overhead increases. But, through experimentation we have observed that the value of the digestion radius does not need to be big for the successful operation of the algorithm. A moderate value of the digestion radius gives us frame rates comparable to the frame rate at which the images are captured from the camera. The value of the digestion radius used in our implementation is 5. This value can be further reduced, but it was observed that at certain camera angles, due to excessive skew in the field lines the major nodes were not getting identified properly.

E. Algorithm – Building the Node Graph

After the identification of the major nodes in the skeletonized image, we need to connect these nodes before we can use these features for localization. The key nodes need to be connected in order to compute the degree of the node for the purpose of identification of the feature. This is achieved by traversing along the skeletonized line segment starting from the major node (through the secondary nodes) till a neighbouring major node is encountered. When another major node is reached, the two major nodes along with the intermediate secondary nodes are connected together and added to the node graph *G*. This process is repeated for all the major nodes until the entire node graph is computed as shown in Algorithm 2. In the preprocessing step, Zhang-Suen algorithm [9] is applied on the image, hence, the pixel connectivity between the nodes of the graph is ensured. Thus, we can use common graph algorithms like depth-first search (DFS) for traversing the graph. DFS is asymptotically linear in the number of nodes in the graph, hence, this stage has a very small computation overhead on the overall algorithm.

Algorithm 2 Building the Node Graph

1: *img* ← Input skeletonized image
2: **for all** pixels *(i, j)* having *cij*[*i*][*j*] = 2 in *img* **do**
3: look for nearby major nodes *(mi, mj)*
4: **repeat**
5: traverse the line connecting *(i, j)* to *(mi, mj)* via the secondary nodes
6: **until** any neighboring major node *(mi, mj)* is found
7: join *(i, j)* and *(mi, mj)* through the secondary nodes in the Node Graph *G*
8: **end for**

Now that the connected node graph is available to us, we can identify the respective features by computing the degrees of the key points in the node graph. Points having degree 4 are X's, formed between the half line and the central circle. Points having degree 3 are T's, formed between half line and the field boundary. The corresponding feature points for the sample image are shown in pink in Fig. 2C. Corner L's can also be identified by finding degree 2 points that have a sufficiently large gradient between the two connecting line segments.

V. Feature Manager

Features are a common link between real world locations and virtual space landmarks which are used to localize the robot. This module receives the potential feature points and validates them for spurious detections. The relative distance of the robot is calculated from each of the identified features is calculated and passed on to the localization module for estimation of the current pose of the robot.

A. Feature Validation

Information regarding every feature type is stored in a form of a dynamic list that gets updated as and when new features are detected by the Feature Detection module. Whenever a potential feature point is sent to the Feature Manager, it decides whether it is a realistic one, based on the current pose of the robot. If the feature is valid, it is pushed onto the corresponding dynamic list. With every feature type, a corresponding confidence is associated with it, which is calculated based on the likelihood of encountering the said feature and its pixel distance from the polar axis of the camera. This confidence c is decayed at each frame exponentially as given is Eq. (1), and features having a confidence less than a certain threshold are discarded from the list. This aids in reducing the effects of bogus detection of feature points and makes the overall localization more robust. A feature that is not likely to be seen will be given a lower value of c_o and hence, it will not have a significant impact on the localization estimate. On the other hand if a feature is getting detected intermittently, it will not be discarded immediately in the frames in which it is not detected.

$$c = c_o e^{\frac{-n}{\gamma}} \tag{1}$$

here c_o is the initial confidence, n is the frame number and γ is the decay constant. Whenever a new feature is encountered the value c_o for the corresponding feature type is initialized to 1. If the feature point had been detected earlier, c_o is updated by adding 1 to the corresponding value of c in the previous frame. The value of γ has been set to 50 in our implementation.

B. Distance Calculation

Relative distance determination of the features is essential for the purpose of localization. Inverse Perspective Mapping (IPM) [10], a geometrical transformation technique where the image is transformed from one perspective to another, can be used to generate a bird's eye view of the image, thus removing the perspective effect. Application of IPM however, requires a rectilinear input image like in a pinhole camera. Therefore, the coordinates of the feature points need to be undistorted before the distance is estimated.

There are several methods for undistorting the images captured using a fish-eye lens. Out of these, two of them have been considered in this work. First one assumes a simple barrel distortion and exploits this assumption to provide a geometric estimate for the undistorted coordinates [2]. The second method uses polynomial regression to map the distorted coordinates to the undistorted coordinates [1].

1) Barrel Distortion: In barrel distortion, the image magnification decreases as we move radially outward from the center of the image. This results in an image with an apparent spherical effect. The distortion correction algorithm assumes a simple barrel distortion and calculates the corrected image coordinates $\overrightarrow{r_c}$ given the distorted image coordinates $\overrightarrow{r_d}$ by the following equation:

$$\overrightarrow{r_c} = \frac{\overrightarrow{r_d}}{1-\alpha\|\overrightarrow{r_d}\|^2} \quad (2)$$

here α depends on the optical system and needs to be identified beforehand. The value of α used in our implementation for the camera setup explained in section III is approximately 8.1E-06.

2) Polynomial Regression: Best possible camera to carry out quantitative measurements is the pinhole camera where the distances are calculated using the properties of similar triangles. In this technique, the coordinates in the fish-eye image are found. Then the respective pinhole camera coordinates are estimated. Finally polynomial regression is used to map the coordinates of the fish-eye image to the coordinates in the rectilinear image. It has been found experimentally [1] that polynomials of degree 5 and 7 provide the best results. Since fish-eye lens has a distortion which is radial in nature, polar coordinates have been used to represent the various sample points. The mapping can be represented by the following equations:

$$r_c = a_0 + a_1 r_d + a_2 r_d^2 + a_3 r_d^3 + a_4 r_d^4 + a_5 r_d^5 \quad (3)$$

$$\theta_c = b_0 + b_1 \theta_d + b_2 \theta_d^2 + b_3 \theta_d^3 + b_4 \theta_d^4 + b_5 \theta_d^5 \quad (4)$$

where the constants $a_0, \dots, a_5$ and $b_0, \dots, b_5$ are calculated using the techniques of polynomial regression. The values of constants calculated for our vision setup are shown in Table I. It is evident from the values calculated for $(b_0, \dots, b_5)$ that only the linear term (b_1) is significant. This demonstrates that the distortion is mainly in the radial direction and the angular distortion caused by the optics is negligible. This property enables us to detect features on the distorted image because the skew caused due to the angular distortion can be neglected.

Both the techniques provide comparable results, however, the polynomial regression technique provides additional versatility, as we can change the complexity of the mapping model to make the polynomial function more flexible. For this reason, we have used the polynomial regression model in our implementation.

TABLE I. POLYNOMIAL REGRESSION CONSTANTS

r_c	Values	θ_c	Values
a_0	0.0336	b_0	0.0001
a_1	-1.6658	b_1	0.9987
a_2	0.0759	b_2	3.95E-17
a_3	-0.0008	b_3	5.27E-04
a_4	3.61E-06	b_4	-7.56E-18
a_5	-5.78E-09	b_5	-3.87E-05

VI. LOCALIZATION

We have used Monte Carlo Localization (MCL) [11] for estimating the 3D pose of our robots. The pose is a tuple (x, y, θ), where (x, y) is the position of the robot in the field in Cartesian coordinates and θ is the orientation of the robot. The particle filter is updated at each frame using the Bayes' theorem based on the combination of beliefs of both the motion model and the observation model. The motion model denotes the probability that the robot is in state x_t given that the robot executes an action a_t in state x_{t-1} and is updated at each frame by the walk module. The observation model is the likelihood that the robot makes observations z_t given that the robot is in statex_t. It uses the identified landmarks and estimates the probability of a particular pose based on the distances measured to the said landmarks.

Initially, a fixed number of particles (poses) N are spread randomly throughout the field. At each frame, the observation model estimates the likelihood of a particle p by comparing the observed distance with the expected distance using the following equation:

$$P(p) \propto \prod_{\forall l \in L} e^{\frac{\|s_e - s_o\|}{2\sigma_i^2}} \quad (5)$$

here s_e and s_o are the expected and observed distances respectively. σ_i^2 is the variance of the distances between landmark l and all the particles. L is the set of all landmarks.

The particles are then resampled using the P as the probability of selecting a particle. This process is repeated for each frame till the pose is stabilised. A few additional particles are added randomly in place of some particles at each frame in order to avoid the "kidnapped robot problem". If the pose certainty drops suddenly below a certain threshold, then additional particles are used to rectify the situation, by augmented MCL [12]. Since the RoboCup field has symmetric landmarks, the MCL algorithm identifies 2 possible poses for the robot. One of the poses is eliminated using the yaw reading of the Inertial Measurement Unit (IMU). The yaw reading of the opponent's goal is fed into the robot prior to the match. Fig. 3 shows the localization estimate of the robot on a sample image. Only the field lines were used as the localization cues, the goal posts in yellow were not used in this illustration.

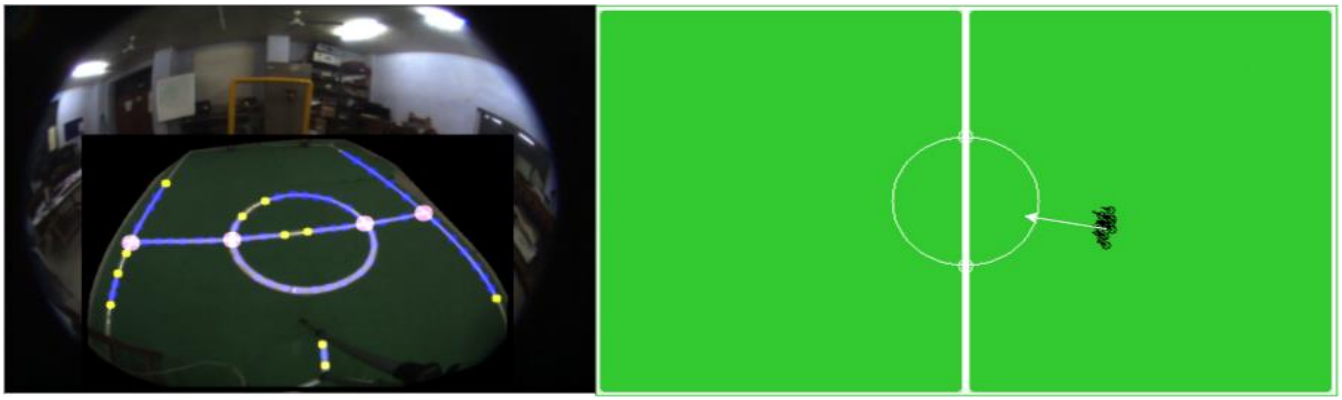

Figure 3. Localizing the robot in the field. (A) Features identified by the feature manager. Pink circles denote the feature points, blue circles represent the major nodes. (B) Localization estimate of the robot. The opponent's goal is on the left hand side of the image. The localization estimate of the bot is shown in black. The orientation of the robot θ is depicted by the white arrow.

VII. Results

For comparing the results of the localization techniques, two situations were considered. In the first situation only the goal posts were used to localize the bot. In the second case only the field lines were used as localization cues. The field lines are meant to augment the localization along with the goal posts, for the purpose of comparison, only field lines were used to estimate the robot's location in the field. In both the situations 200 particles were used in the particle filters. An IMU was used in both the cases in order to differentiate between the two halves of the field as the field is symmetric. 20 random poses were considered on the field, 10 in our own half, the other 10 in the opponent's half. Due to the symmetric nature of the field all the 20 poses were chosen such that the opponent's goal post was visible to the robot.

In case of goal post based localization 8 out of the 20 poses were within 40cm of the ground truth. Out of these, majority of the cases were localized accurately when the goal posts were within 2 - 2.5m of the robot. This is because the distance estimate starts to become erroneous when point of interest is at the periphery of the camera frame. On the other hand when only field lines were used for the localization 17 out of 20 poses were within the stipulated error radius of 40cm. One of the sample pose is shown in Fig. 3 along with the localization estimate. The black cluster of points in Fig. 3B represent the position (x, y) of the robot and the white arrow represents its orientation θ. Table II reports the experimental results in an organized manner.

TABLE II. Experimental Results

Mode	Number of Poses in Error Radius (40cm)	
	Own Half	Opponent's Half
Only goalposts	2	6
Only field lines	9	8

VIII. Conclusion

In this paper we have presented a new method of identifying the field lines by computing the node graph through a process of digestion. This method provides us additional landmarks that assist in the localization. We observed that the majority of errors in the belief occur due to the inaccuracies in the distance estimation using IPM due to the large curvature of the fish-eye lens. The use of field lines as landmarks help in mitigating these errors and provide additional landmarks to make the localization more robust. Currently the implementation is not optimized and uses a matrix to store the digestion weights. Our future works will include implementations which use the disjoint-set data structure for the process of digestion to reduce this overhead. As shown in Fig. 3A the computed node graph captures the structure of the field lines effectively. This information can be used for more complex feature detections like arcs in the central circle that can be used for the precise positioning of the robot.

Acknowledgment

The authors would like to thank Department of Electronics and Information Technology (DeitY), Government of India for financially supporting the project. The experiments were performed at the Centre for Robotics and Intelligent Systems (CRIS), BITS Pilani.

References

[1] S. Shah, J. K. Agarwal, "Intrinsic parameter calibration procedure for a (high-distortion) fish-eye lens camera with distortion model and accuracy estimation", Pattern Recognition, vol. 29, no. 11, pp. 1775-1788, 1996.

[2] M. Missura, M. Schreiber, J. Pastrana, C. Münstermann, M. Schwartz, S. Schueller,, S. Behnke, "NimbRo TeenSize 2013 Team Description", 2013.

[3] N. Henderson, P. Nicklin, A. Wong, J. Kulk, K. Chalup, R. King, H. Middleton, S. Tang, A. Buckley, "The 2008 NUManoids Team Report", RoboCup SPL Team Descriptions, Suzou, China, 2008.

[4] M. Sridharan, P. Stone, "Real-time vision on a mobile robot platform", IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2148-2153, 2005.

[5] R. C. Gonzalez, R. E. Woods, "Digital Image Processing", Prentice Hall, 2002.

[6] A. Baist, R. Sablatnig, G. Novak, "Line-based landmark recognition for self-localization of soccer robots", Proceeding of the IEEE International Conference on Emerging Technologies, pp. 132-137, 2005.

[7] H. Strasdat, M. Bennewitz, S. Behnke, “Multi-cue localization for soccer playing humanoid robots,” RoboCup 2006: Robot Soccer World Cup X. Springer Berlin Heidelberg, pp. 245-257, 2006.

[8] H. Schulz, S. Behnke, “Utilizing the structure of field lines for efficient soccer robot localization”, Advanced Robotics, vol. 26, no. 14, pp. 1603-1621, 2012.

[9] T. Zhang, C. Suen, “A fast parallel algorithm for thinning digital patterns”, Communications of the ACM, vol. 27, no. 3, pp. 236-239, 1984.

[10] T. E. Schouten, Egon L. van den Broek, “Inverse perspective transformation for video surveillance,” Electronic Imaging 2008. International Society for Optics and Photonics, 2008.

[11] T. Agarwal, D. Gopinath, “Localization using relative mapping technique for mobile soccer robots,” in International Conference on Communications and Signal Processing (ICCSP), pp. 265-269, 2013.

[12] S. Thrun, W. Bugard, D. Fox, “Probabilistic Robotics”. Cambridge, Mass.: The MIT Press, 2005.