The Artificial Landmark Design for Mobile Robots Localization and Mapping

Arthur Huletski, Dmitriy Kartashov
The Academic University
Saint-Petersburg, Russia
hatless.fox@gmail.com, dmakart@gmail.com

Kirill Krinkin
St.Petersburg State Electrotechnical University "LETI"
St.Petersburg, Russia
kirill.krinkin@fruct.org

***Abstract*—Nowadays the big challenge in simultaneous localization and mapping (SLAM) of mobile robots is the creation of efficient and robust algorithms. Big amount of SLAM methods use computer vision for extraction of unique environment features from camera images (feature-based approach). In some cases, artificial landmarks are being used to simplify environment markup (landmark-based approach). In this paper we introduce a new QR-like color artificial landmark design and algorithm for fast detecting and tracking them in arbitrary images.**

I. Introduction

Landmarks are generally defined as passive objects in the environment that provide a high degree of localization accuracy when they are within the robot's field of view [1]. Artificial landmarks may carry additional information about the environment and may be used to assist a robot in localization and navigation. Although this approach requires environment preprocessing, it makes developers free to choose a type of landmark and information it holds. Similar approaches based on localization with artificial landmarks are widespread in human life: numbers on hotel rooms, catchy shop signs, etc. It this report authors analyse existent approaches for landmarking and introduce new kind of color label for fast and robust detection.

The paper is organized in the following way. In this section quick response codes (QR codes) [2] and detection methods for them are discussed. In Section II. new design for color landmark is introduced and detection algorithm is suggested. In Section III. tracking approach is introduced. Rest of the paper contains evaluation and conclusion.

Technically, a developer could choose any design for landmark that can be sensed in small (with respect to entire environment) location. Since artificial landmark is chosen to simplify extraction of location features, appropriate landmark should satisfy next requirements:

- can be reliably detected in a given environment. Detection should be robust for bad lighting conditions, glares, wide spectrum detection angles;
- can be identified;
- can be easily created and fixed in a given environment.

One possible option is a visual printed landmark, e.g. QR codes. This kind of landmarks has several advantages comparing to RFID and other technologies: it doesn't consume power and requires only camera which most mobile robots are already equipped.

There are several ways to detect QR codes in an image. In [3] the Viola-Jones framework based on Haar features is used to detect QR finder patterns (FIPs). Found FIPs are aggregated into a graph by their size and distance between them. The graph is searched for 3-cycles that satisfy the orientation criterion and represent QR codes. Unfortunately, the QR code plane must be almost orthogonal to the camera axis to reach the claimed 90% detection rate. In addition, this algorithm gives no information about the QR code position in the space.

Another approach that is described in [4] utilizes a special line parametrization called PClines which is a variant of the Hough transform. This parametrization uses a parallel coordinate system and allows faster accumulation than the basic Hough-transform method. The image is searched for the specific parallel lines pattern that is declared as QR code. The algorithm can handle various orientations of QR codes in images, tolerant to uneven illumination and allows real-time processing but is unable to detect QRs from far away or in blurred images. It's also unclear whether the algorithm can detect several QRs on the same image.

Some other algorithms (e.g. [5], [6]) assume that there is only one QR code in the image, so they are not suitable for our problem as the robot may see several landmarks simultaneously.

In order to estimate detection speed and resource requirements authors implemented and tested a variant of the QR detection algorithm described in [3], that also uses cascade classifiers to find the QR alignment pattern (AP) in the detected QR code. The last is necessary since 4 points are needed in order to compute position of the landmark in the space, but the existing algorithm gives only 3 points. Experiments have showed that using bare QR codes as landmarks for marking up the environment have following significant drawbacks:

1) it's rather difficult to detect and extract bare QR code when it is far enough (more than 1 meter, which is quite often condition for indoor mobile robots);
2) QR code detection quality turns out to be very sensitive to angle between camera and QR code planes. Only if these planes are almost parallel then acceptable quality and robustness are reachable.

Based on this preliminary experience it was decided to create a new type of landmark that can be easily and reliably

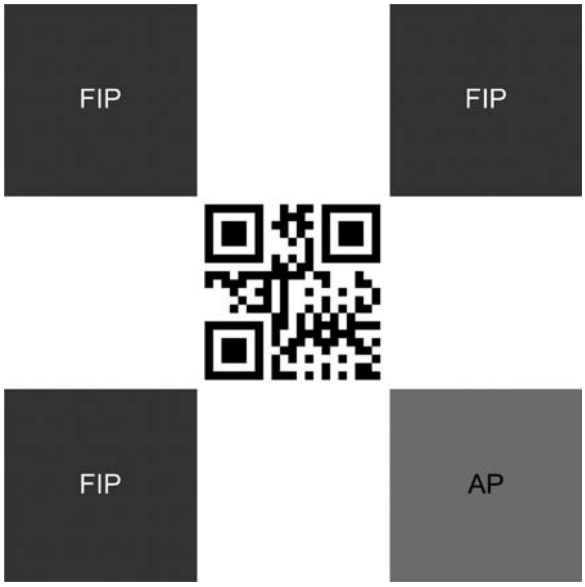


Fig. 1. The landmark example

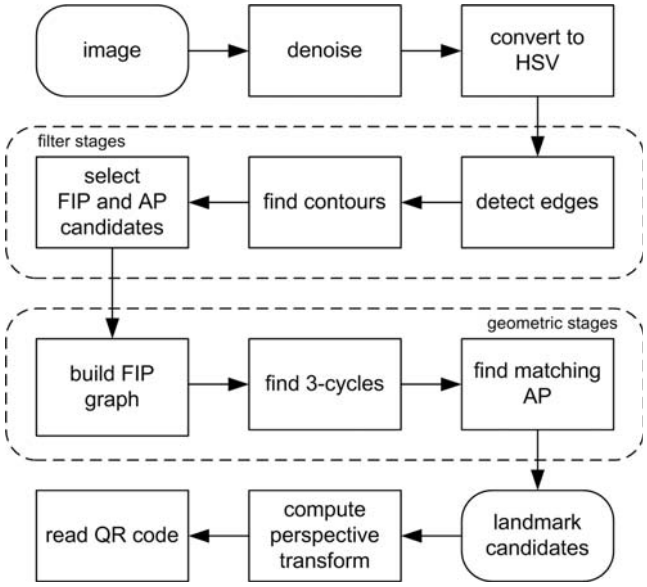


Fig. 2. Detection algorithm pipeline

detected (i.e. chance of detection doesn't depend much on the camera position) and allows to carry extra information (like position or point identifier in the environment, or even instructions for mobile robots).

Various types of such visual landmarks are discussed in [7]. Besides good discussion of existent landmark types, this work introduces a novel landmark design based on QR code. According to it QR code is placed into blue rectangle that has three colored circles around it. Landmark of proposed design is scalable, special algorithm is introduced to extract inner area from outer circle for further QR code detection. Provided evaluation claims that landmark with diameter of outer circle equal to 20 cm can be reliably detected from 2 meters while its horizontal angle lies in range [-60; 60].

The aim of our work is to propose a landmark that can be detected in broader horizontal angle range and has smaller size at the same time. The novel design was inspired by QR code detection algorithm described in [3] and [8]. The last one introduces artificial striped landmark. Two colors are used for stripes, each stripe has neighbours of different color.

II. Label design and detection

A. Landmark layout

A general idea for a new landmark is based on QR code layout extended with color markup. The suggested layout consists of 3 blue squares called finder patterns (FIP) and one red square – alignment pattern (AP) – on the white background (we use these particular colors but actually any 2 distinguishable colors are also acceptable). They are located on the landmark like FIPs and APs in QR codes (see Fig. 1). This landmark layout looks like 4 light squares on a dark background in the saturation channel of the HSV color space in daylight (Fig. 3b). This allows to run some edge detection algorithm, e.g. Canny edge detector [9], on the saturation channel to find contours in the image (Fig. 3c). Some of the detected contours are chosen to be FIP candidates for the landmark.

B. Detection algorithm

The full algorithm pipeline is shown in the Fig. 2. Below we discuss each stage of the algorithm.

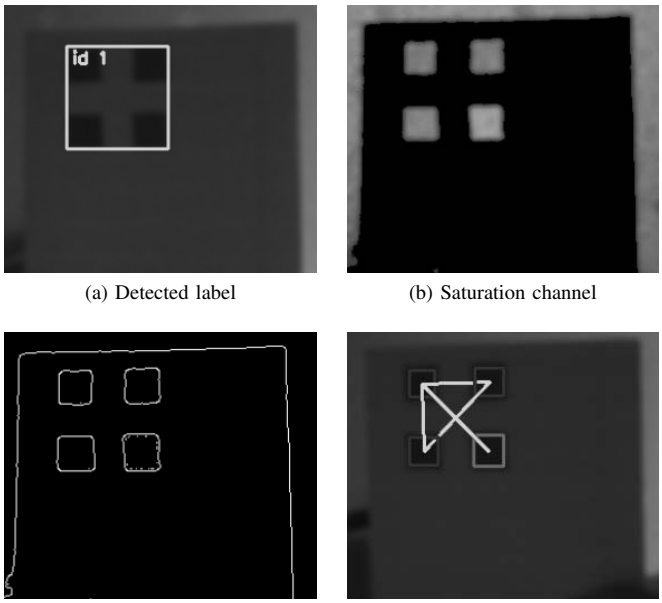
To select FIP candidates consider bounding rectangles of the detected contours in the RGB color space. For each candidate all pixel values within the bounding rectangles are accumulated separately for each channel. Then the following conditions should be checked:

$$\sum blue > a \cdot \sum red \quad \text{and} \quad \sum blue > b \cdot \sum green$$

If these conditions are satisfied then the contour is pushed into the list of FIP candidates. If the similar conditions are met for the red channel then the contour is stored as an AP candidate. The coefficients a and b in the formula above are called a color ratio and determine significance of the color component within the contour bounding rectangle: higher coefficient values discard more candidate contours at this stage. On the other hand, the farther from the camera the landmark is the lower the coefficient values should be to detect a FIP. The coefficient values from the range [1.25, 1.6] are reasonable to use in practice. The output of this stage is shown in the Fig. 3d.

Note that the found contour is not checked to be rectangular since it is not required in practice: due to optical distortions or non-orthogonality of the camera axis and the landmark plane FIPs can be of any shape. In our case, discarding contours by color is more robust than by geometric features, though at a great distance this method might discard some real FIPs.

After obtaining the lists of FIPs and APs quadruples (3 FIPs and 1 AP) are formed from candidates using landmark geometric features. At first, the graph is constructed in the following way: vertices are FIPs and an edge connects FIPs if they satisfy two types of constraints:



(a) Detected label (b) Saturation channel

(c) Edge detector output (d) Detected FIPs and APs

Fig. 3. Detector output

1) the minimum and maximum distance between FIPs:

$$\begin{cases} D_{min} \cdot width < |X_{FIP1} - X_{FIP2}| \\ D_{min} \cdot height < |Y_{FIP1} - Y_{FIP2}| \\ |X_{FIP1} - X_{FIP2}| < D_{max} \cdot width \\ |Y_{FIP1} - Y_{FIP2}| < D_{max} \cdot height \end{cases} \quad (1)$$

where $width$ and $height$ – mean size of a FIP pair; X, Y – FIP position; D_{min}, D_{max} – constraint coefficients;

2) the difference in height and width:

$$W_{min} < width_{FIP1}/width_{FIP2} < W_{max}$$

$$H_{min} < height_{FIP1}/height_{FIP2} < H_{max}$$

where $width$ and $height$ – FIP size; H, W – size ratio coefficients;

Parameters in these constraints may vary and their actual values depend on the layout of the landmark. In our case a distance constraint is from 1 to 3 FIP sizes and a FIP size ratio is from the range [0.5, 1.5]. The resulting graph is searched for 3-cycles that are considered to be landmark candidates.

Finally, the last component of the landmark is added – AP. From all of the AP candidates we choose one that meets the constraints on the location (the top-left corner or center must be inside the bounding rect of the three selected FIPs) and which size is closest to the selected FIPs. Note that the constraint on the top-left corner doesn't allow detecting upside down landmarks, but in our use case these landmarks may be simply ignored. If an appropriate AP is found then the landmark candidate is accepted. If several selected candidates have one or two vertices (FIPs) in common then the candidate that has a triple of FIPs that forms a right triangle with the smallest error is accepted.

The output on each stage of detection is shown on the Fig. 3. Note that a small saturation threshold is used on Fig. 3b. It amplifies the difference in intensity of light and dark regions on an image that slightly improves the edge detection quality.

The figures 4 and 5 show different detector settings. In the second case the lower values for filter parameters are used (but the same geometric constraints) and it's results in many false positive FIP detections, but the label still can be accurately detected.

Given the location of FIPs and APs the perspective transformation (homography) can be computed in order to get orthogonal projection of the landmark and it's position in the space. A QR code located in the center of the landmark can be extracted using that orthogonal projection and can be decoded using any QR code detection and decoding algorithm (e.g. [3], [4] or even [5]). It's clear that it still can't be done from far away, but the robot can always drive up to the landmark if it knows where the landmark is. An example of QR code extraction is shown in the Fig. 6. Binarization, erosion and dilation have been applied to the extracted QR code to get the image in the Fig. 6b. Most bar-code readers, e.g. ZXing online decoder [10] or ZBar library [11], can decode the resulting QR, though the additional rectification, like the one that is described in [12], may be applied.

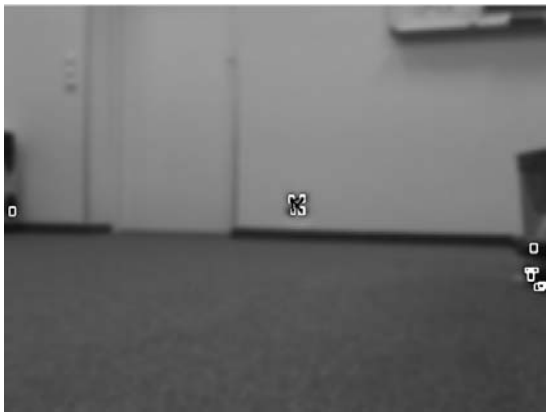


Fig. 4. Detected landmark with some false positive FIP candidates. The landmark is reliably detected

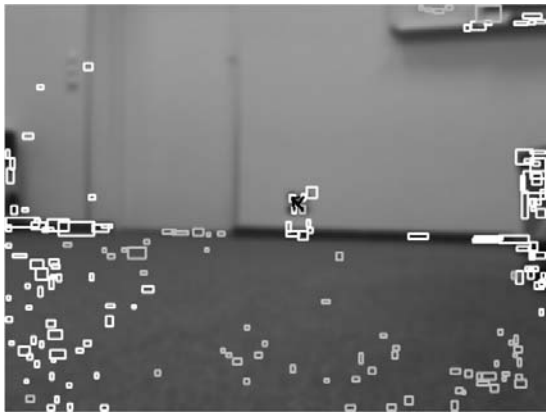
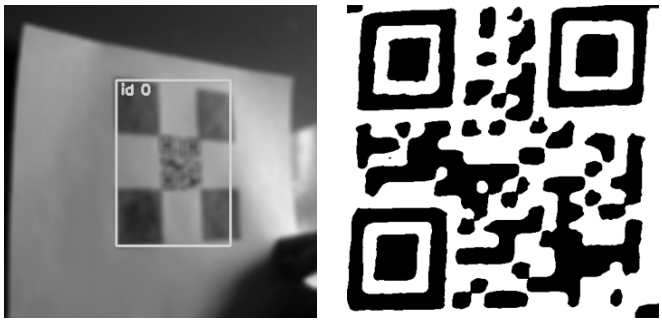


Fig. 5. The same image but different detector settings. The landmark still can be detected



(a) Detected label (b) Extracted QR code

Fig. 6. QR code extraction

There are many parameters that can be adjusted in the detection algorithm. Let's sum them up:

1) geometric constraints – the distance between FIPs and APs and their size ratio;
2) color constraints – the color ratio used in FIP detection;
3) filter parameters – the Canny's threshold, saturation threshold, blur kernel size.

In the further research the methods for adaptive parameters adjusting according to the current observation conditions should be investigated.

III. Label tracking

Landmarks tracking is the next step for increasing detection robustness and quality. A robot can change the physical position of observation relatively smooth so the detected landmarks in the camera video stream will be relatively close on the two consecutive frames and therefore the detected landmarks may be tracked in a sequence of images. This can be done in the following way:

1) Initially there is an empty object (landmark) pool. Lifetime is assigned to every landmark and allows to keep the landmark in the pool for some time even if it isn't detected in several frames. The landmark is kept in the pool while its lifetime is less than the fixed maximum value.
2) Processing the next frame obtains a new set of landmarks. There are 3 different cases are possible. They are shown in the Fig. 7.
3) If the object pool is empty then all new landmarks are stored in the pool and assigned ids.
4) If the new set of landmarks is empty and the pool is not then the lifetime of all landmarks is increased and obsolete landmarks are removed.
5) If both the pool and new landmark set are non-empty then it's required to solve the assignment problem where the pool landmarks are agents and the new landmarks are tasks (or vice versa), and the cost is the distance between a pool landmark and new one. So the total distance between old and new landmarks is minimized.
6) The obtained solution is checked to meet certain restrictions:
   - mapped landmarks are of similiar sizes,
   - distance between them doesn't exceed the maximum value.

   If the mapped landmarks meet these restrictions then the position of the pool landmark is updated and its lifetime is reseted. Otherwise the lifetime of the pool landmark is increased and the new landmark is added to the pool.
7) All new landmarks that doesn't map to the pool landmarks are added to the pool.

Landmark tracking example is shown in the Fig. 8. This algorithm is pretty standard and has 3 parameters that can be adjusted:

1) the landmark maximum lifetime depends on camera frame rate, but total time of 1 second looks appropriate;
2) the maximum distance between landmarks depends on robot speed;

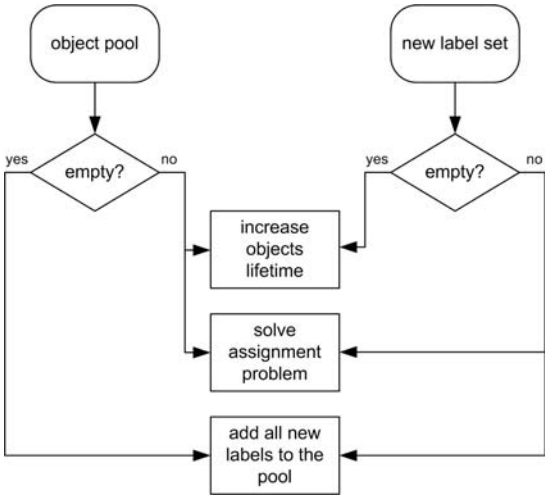


Fig. 7. Merging object sets in the tracking algorithm

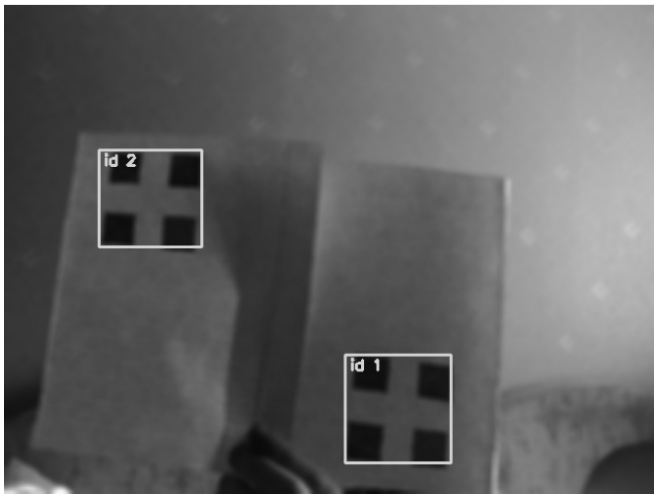


Fig. 8. Tracking example

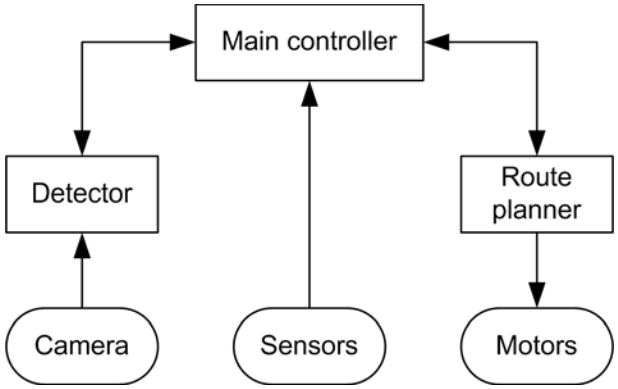


Fig. 9. ROS nodes of the robot

3) the landmark size ratio also depends on the robot speed and it's tuning based on the specific external parameters.

IV. Evaluation

The proposed landmark detection algorithm suffers from image noise like any other algorithm based on edge detection. Various denoising algorithms can be applied (Gaussian blur in our case), but there is a trade-off between overall performance and image quality. For example, large blur kernels can affect edge detection quality, but good denoising algorithms like non-local means [13] run extremely slow.

Another disadvantage of the algorithm is that it relies on color features that depend on external lighting. The landmark looks like light squares on a dark background in daylight and completely different in dim light. So detection algorithm parameters should be adjusted depending on the ambient conditions.

A. Performance

The detection algorithm is based on OpenCV framework [14] and handles 30-50 images of 640x480 pixels per second on 2.5 GHz Core i5 CPU depending on environment conditions and algorithm settings, so it can process frames in real-time and may use a video stream to track landmarks.

Actually, detection algorithm is supposed to be used on simple mobile robots with limited resources. Authors has created testing environment with Robot Operating System (ROS) and popular low cost, credit-card sized computer Raspberry Pi.

ROS (Robot Operating System) [15] is a framework for robot software development that provides various system services such as hardware abstraction, low-level device control, implementation of commonly used functionality and message-passing between processes. Set of ROS processes is represented in a graph architecture where processing takes place in nodes that may receive and post sensor, control, planning and other messages.

Simplified structure of our robot is shown in the Fig. 9. The detector node works as a service that is polled by the main controller from time to time. Note that the ROS itself is not a real-time OS and message passing introduces some overhead. Our proof-of-concept robot uses Raspberry Pi [16] that utilizes 400 MHz ARM CPU as the main brain and the performance of the detection algorithm itself on this hardware is around 1 FPS.

When the detector is integrated in the whole robot-control system, it handles only 0.5 frames per second. In [17] authors suggest to use adaptive threshold algorithm instead of Canny edge detector and optimized OpenCV library in the pose estimation algorithm that is also based on edge detection – it improves performance twice but in our case it's still insufficient. So the detector node requires a separate and more powerful CPU.

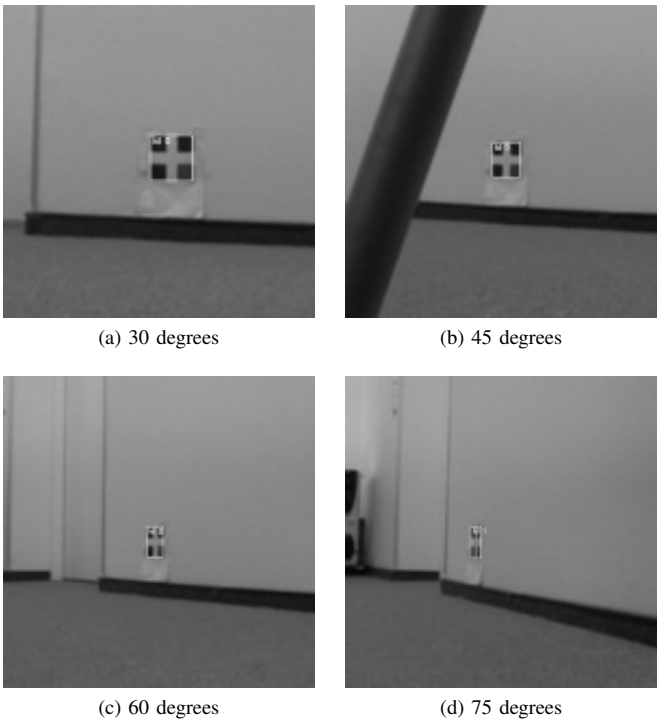
B. Detection robustness

Our measurements shows that landmark with 11 cm side can be reliably detected in [-65; 65] horizontal angle range. It also can be detected if angle exceeds given limit, for example equals to 75 degrees, but detection rate in this case varies from 10% to 80%, so the estimate requires further investigation. Comparing with landmark design proposed in [7] which uses 20 cm landmark and is able to detect it reliably in [-60; 60] horizontal angle range the algorithm proposed in this work slightly better.

Fig. 10 shows examples of label detection when angle is varying.

V. Conclusion

The proposed landmark fits good landmark criteria: its easy to create, set up, detect and identify. Although identification often requires approaching the landmark to read QR code,



(a) 30 degrees (b) 45 degrees

(c) 60 degrees (d) 75 degrees

Fig. 10. Landmark detection for different angles

the fact of landmark(s) presence narrows amount of potential positions during localization process.

Described design and detection algorithm have various opportunities for enhancement:

- adaptive filters can be added to adjust their own parameters based on environment parameters;
- video stream can be used for image stabilization, noise reduction and label tracking;
- experiments with colored FP and AP width can be performed to increase reliable angle range;
- landmark detection rate should be estimated for vertical angle variations (useful for UAVs such as quadcopters);
- measurement and precise estimates for errors should be done when horizontal angle is in (-90, -65) (65, 90) range.

All programs and algorithm implementations are published as Open Sources Software and can be accessed by the following link: http://github.com/OSLL/landmark-detection.

ACKNOWLEDGMENT

Authors would like to thank JetBrains company for provided support and materials for working on this research. The paper has been prepared within the scope of project part of the state plan of the Board of Education of Russia (task # 2.136.2014/K).

REFERENCES

[1] R. Siegwart, I. R. Nourbakhsh, D. Scaramuzza, Introduction to Autonomous Mobile Robots. *MIT Press*, 2011, p. 453.

[2] International Standard ISO/IEC 18004:2000 Information technology – Automatic identification and data capture techniques – Bar code symbology – QR Code, 2000.

[3] Luiz F. F. Belussi, Nina S. T. Hirata, "Fast Component-Based QR Code Detection in Arbitrarily Acquired Images", *Journal of Mathematical Imaging and Vision*, vol.45, no.3, Mar. 2013, pp. 277-292.

[4] Gabriel Klimek, Zoltan Vamossy, "QR Code Detection Using Parallel Lines", *Computational Intelligence and Informatics (CINTI), 2013 IEEE 14th International Symposium*, Nov. 2013, pp. 477-481.

[5] Yunhua Gu, Weixiang Zhang, "QR Code Recognition Based On Image Processing", *International Conference on Information Science and Technology*, Mar. 2011, pp. 734-736.

[6] Yue Liu, Mingjun Liu, "Automatic Recognition Algorithm of Quick Response Code Based on Embedded System", *In. proc. of the Sixth International Conference on Intelligent Systems Design and Applications*, Oct. 2006, pp. 783-788.

[7] Hao Wu, Guohui Tian, Peng Duan, Sen Sang, "The Design of a Novel Artificial Label for Robot Navigation", *Proceedings of 2013 Chineseintelligent Automation Conference*, pp. 479-486.

[8] Kuk-Jin Yoon, Gi-Jeong Jang, Sung-Ho Kim, In-So Kweon, "Fast Landmark Tracking and Localization Algorithm for the Mobile Robot Self-Localization", *IFAC Workshop on Mobile Robot Technology*, 2001, pp. 190-195.

[9] John Canny, "A Computational Approach To Edge Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol.8, no.6, Nov. 1986, pp. 679-698.

[10] ZXing Decoder Online, Web: http://zxing.org/w/decode.jspx.

[11] ZBar bar code reader, Web: http://zbar.sourceforge.net.

[12] Kong Suran, "QR Code Image Correction based on Corner Detection and Convex Hull Algorithm", *Journal of Multimedia*, vol.8, no.6, Dec. 2013, pp. 662-668.

[13] Antoni Buades, Bartomeu Coll, Jean-Michel Morel, "A non-local algorithm for image denoising", *In proc. of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2005, pp. 60-65.

[14] OpenCV website, Web: http://opencv.org.

[15] ROS.org — Powering the world's robots, Web: http://www.ros.org.

[16] Raspberry Pi, Web: http://www.raspberrypi.org.

[17] Sunil Shah, "Real-time Image Processing on Low Cost Embedded Computers", *Techincal report No. UCB/EECS-2014-117*, 2014.