A global, camera-based mobile robot localization

ENGEDY, István

Budapest University of Technology and Economics, Department of Measurement and Information Systems,
Magyar tudósok körútja 2. H-1117, Budapest, Hungary
e-mail:engedy@mit.bme.hu

HORVÁTH, Gábor

Budapest University of Technology and Economics, Department of Measurement and Information Systems,
Magyar tudósok körútja 2. H-1117, Budapest, Hungary
e-mail: horvath@mit.bme.hu

Localization is a main problem in most real mobile robot navigation systems. No other parts of the system can be used in the real world without it. There are different approaches to this problem, from local collision avoidance to global positioning systems. In this paper we will present a camera- and image processing-based global localization system that can be used in indoor environment. We will also present that the system is able to locate different objects marked with simple marker shapes, and it is able to locate any other objects in the working area too. It is also able to determine the size and orientation of the objects. We will show the basic steps of this image processing algorithm and also how Fourier transformation is used to determine the main localization parameters.

1 Introduction

The field of mobile robotics is a developing industry nowadays. There are numerous applications of mobile robots. They are sortable based on their operation environment: there are land, aerial, water and underwater mobile robots. They can differ in the level of autonomy they have during operation. Some mobile robots need no human interaction or supervision even in unknown environments, while others are only able to follow a previously painted line or curve on the ground, and there are remote controlled robots, which always need human supervision. In the sequel, we narrow the scope of this paper to the discourse of an

autonomous land mobile robot, which is able to navigate in the specified environment.

Mobile robot navigation systems have a number of well separable subtasks. Each has also numerous different solutions of different approaches. One of the most important functions of a navigation system is to sense the surrounding environment and to determine the location of the robot, which is the localization of the robot and the surroundings. If the entire working area should be known, we were talking about mapping. The other important task is motion planning. This can be divided into at least two separate subtasks, the path planning, that is planning a route from one point to another, and planning the movement itself, how to follow actually the path. The latter, in some real-world system, can be rather difficult due to the kinematic constraints of the robot [1].

1.1 Localization

Localization is determining the position of the robot and other items in a reference system. It is separable to three different problems: tracking of a known object, finding an unknown object, and kidnapping of the robot, which is to take it to an unknown place. To solve these problems, sensors that provide information of the physical reality are needed for the robot. The various sensors determine the type of the localization techniques.

There are sensors that operate only in a short range. Robots using such sensors can observe only the immediate environment. Collision avoidance can be solved more or less easily, but for the task of mapping the robot must walk through the entire working area. Different sensors can be used in this task e.g. tactile sensors like sensor mustache, or bumper, or more active sensors such as laser, ultrasonic, or radar distance sensors. A camera mounted on the robot may also belong to this category, the robot has the ability to monitor the environment, and identify obstacles.

Long-range distance sensors typically require some other device, or the sensor is not mounted on the robot. The GPS system is such a sensor, which requries the signals of GPS satellites to calculate its own position. A robot can be easily localized using this sensor, but the surrounding environment can not be observed. Another solution would be the application of camera or cameras, which monitor the entire workspace, so the robot can see all the obstacles in the workspace, thereby providing an accurate map for the robot. In this case localization would be carried out by using image processing algorithms. Disadvantage of these methods are that they do not work under ground or under water, the satellite signal is detected only under the open sky, while the cameras can not see through walls [3].

There are a number of localization methods, which use the above-mentioned sensors. It is common in them that they all require some recognition algorithm,

which is based on the sensor signals, by which it is able to locate the robot and any objects in space. In cases when only the immediate surroundings of the robot is detected by the sensors, the exact location of the robot can be determined by the surrounding environment and an a priori known map. Often there are ambiguous solutions (for example, there are several similar intersections in a maze), so it is necessary to process not only the currently measured, but the previously measured sensor data, so the more information may make the localization of the robot unambiguous. Such methods include the Monte Carlo localization, where the possible locations are represented by particles, or a Kalman filter solution, where recursive state estimation is used in simultaneous localization and mapping (SLAM) [4].

When the entire work space is known, for example, through camera surveillance, there could be also cases when the localization of the robot or other object is ambiguous, such a case might be the robot soccer, where robots are looking the same, and they can not be distinguished from one another merely by an image. The solution for this problem may be to distinguish the robots from each other with markers.

In this paper we will present the localization part of a mobile robot navigation system. The navigation system is based on a regularized version of the back-propagation through time method, and its main purpose is to navigate a robot car among moving obstacles [2]. It is tested on a real car-like robot, and as a real-life experiment, all parts of the navigation system has to be functioning in a real-world system, including the localization system.

2 Localization system architecture

As it was shown in the introduction, a number of solutions exist for the problem of localization. One of them is to observe the entire working space by a camera, and use an image processing algorithm to obtain important informations for the robot from camera image. Such a system is described and implemented in this paper, where only a single camera is watching vertically down from the ceiling to the workspace. The workspace, in which the robot can move, corresponds to the field of vision of the camera. This camera system is responsible for the localization of the robot, the target, and the mapping of obstacles.

Such a camera-based localization system has many advantages, including the fact that as the entire workspace is seen, not only the nearby environment of the robots will be known, but the whole working space, including all objects positions. This is particularly important in a rapidly changing environment, because the current state of the terrain is always taken into account during path planning, thus there is much less danger of collision with the obstacles.

The disadvantages of such a system are that it is unable to locate objects that cannot be seen directly by the camera; which renders this method unusable in underground or underwater applications, as well as in terrains having great falls, such as an environment among buildings. The problems of mounting of the camera also reduce the number of possible fields of application.

2.1 Preprocessing of the camera images

The system is built up from a webcam and an image processing algorithm. A webcam is mounted on the ceiling, monitoring an area on the floor (Figure 1). the camera provides 15, 320x240-pixel, colored image to the computer each second. Those are acquired through the DirectShow multimedia framework in the implemented application. The images are processed in real time by an algorithm, which is responsible of locating all the objects in the working area, and recognizing some special objects, like the robot itself, or an object determining the target position. These special objects are marked with markers. The possible markers are given to the image processing algorithm, so it is able to recognize them. All other objects, which can not be matched with any markers, are considered as barriers.

Figure 1
The situation of the camera and the working area

Image processing takes place in short as follows (Figure 2). The image is noise-fildered after it has been captured. In the next step a previously recorded background image is subtracted from the image to determine the shapes of the objects laying on the work area, then after another noise filtering the contour of each shape is determined so all shapes are separated by cutting them out from the original image. This makes possible to process them independently. Another contour–finding step is made on the binary version of the cut-out images where binarization is based on the intensity values. The contour of this marker then processed further. The size, location and orientation of the marker are calculated based on the Fourier transform of the contour. Then the contour is normalized using these parameters, so it can be compared to the pre-defined contours of the reference markers, to check if it is a predifined type of marker.

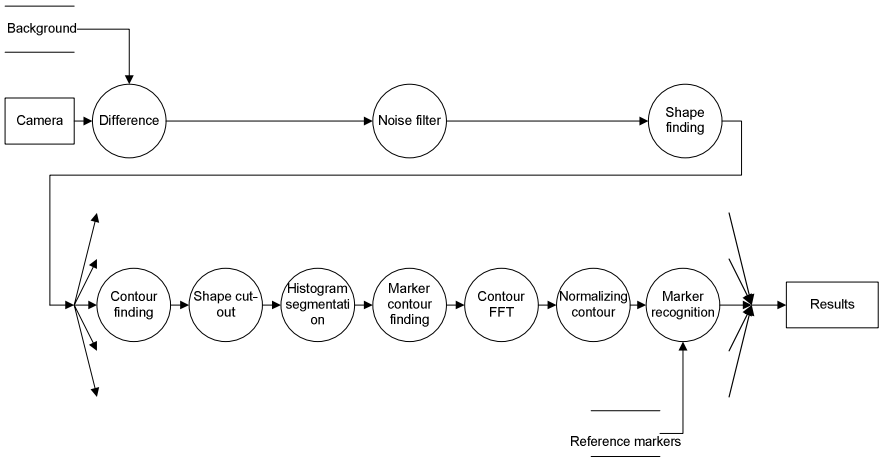

Figure 2
The data flow of the image processing

2.1.1 Differntiating from the background image and noise filtering

To find objects in the working area a simple subtraction is made to detect changes in the image. The actual camera image is compared to a pre-recorded background image, which was captured when there was no objecs on the working area. Any deviation from this picture could mean an object, as the camera and the floor compared to each other's position does not change. The only errors could origin from the changes of the lighting, which cannot be considered to be static. To overcome this problem, not the RGB values are subtracted from the actual image and the background image, but first the RGB values of each pixels are converted to HSL (hue, saturation and lightness) representation, and the lightness value is discarded. This way the effect of different lighting can be eliminated or at least reduced, as the saturation and hue of the objects are independent of the intensity of the lighting, especially when artificial lighting is applied, like in indoor environment.

It should be noted that due to the subtraction objects that have similar color as the background, or which are transparent, can only vaguely found by the system, if it can be found at all, so using such objects in the workspace should be avoided. For an object, seen by the camera the same as the background, there is no way for the image processing algorithm to detect, so the object is invisible to the system.

The signal of the camera is of course noisy. The noise would make the subtraction based object finding less accurate. For denoising median filtering followed by tresholding is applied. If the sum of the differences between the actual camera image and the background in a 3x3-pixel window is larger than a predefined treshold value, the pixel in the middle of the window is considered as part of an

object, so it will be a binary value 1 on the result image, in any other case it will be the value of 0.

Despite the median filtering and tresholding, there could still be some point defects on the image, mainly along the boundary of the objects because of shadows, or in places where the object has the same color as the background. Even different lighting conditions could cause such errors. Due to these problems additional filtering, smoothing is needed. The morphological operations, opening and closing are used to solve this problem in 3x3 windows (Figure 3). Opening (erosion, dilation) clears all the point defects, and then closing (dilation, erosion) will patch the gaps at the edge of the object, and the point defects in objects. (Dilatation: value of the pixel will be 1 if there is at least one pixel of value 1 in a 3x3 neighborhood, erosion: the pixel value will be 0 if there is at least one pixel of value 0 in a 3x3 neighborhood.)

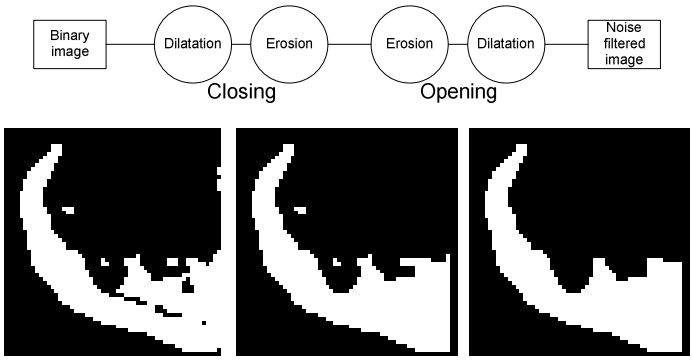

Figure 3

Denoising the difference image, using closing and opening operations. Upper figure: the process of noise filtering, Lower figures: before closing and opening; after closing and after closing and opening.

2.1.2 Localization of the shapes

After all the shapes of the objects are available on the image, the exact position, size, shape and orientation of these objects should be determined. To do this, we use a contour finder algorithm, which not only displays the binary image of the contour, but also gives the coordinates of the contour points. Since there are several shapes on the picture, there will be multiple contours, which should be returned in a list.

There are two parts of this algorithm: the shape finder, and the contour following algorithms. The task of the shape finder is to find each shape on the previously denoised binary image. To do this, the whole image should be scanned row by row, and once a pixel of value 1 is found, the contour following algorithm starts, as it is sure that the point just found is part of a contour. After the contour

following algorithm is finished, the shape is simply removed from the image, ensuring that each shape is processed exactly once.

2.1.3 The contour following algorithm

The contour following algorithm receives as input the image itself, and one point of the contour. It starts from that point to follow the edge of the shape. It is known in which direction this contour point was found. This direction is important to walk clockwise around the shape. The point surely is found in the upper left "corner" of the shape and this will be the first tested point. The algorithm does the following at each test points: it tests all eight neighbouring points, clockwise, starting from the last found contour point (Figure 4). If it finds a pixel with the value of 1, then it goes to this pixel which will be a contour point too. The algorithm will stop when the next contour point would be the same as the first one.

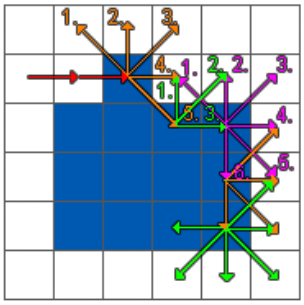

Figure 4
The process of contour following

```
function CONTOUR_FOLLOW (image, column, row) returns contour
   DIRECTIONS = {{-1,-1}, { 0,-1}, { 1,-1}, { 1, 0},
                 { 1, 1}, { 0, 1}, {-1, 1}, {-1, 0}}
   x = column
   y = row
   dirindex = 0
   dir = DIRECTIONS[dirindex]
   repeat
      contour.ADD(POINT(x, y))
      while image.value[x + dir.x, y + dir.y] == 0
         dirindex = (dirindex + 1) mod 8
         dir = DIRECTIONS[dirindex]
      end while
      x = x + dir.x
      y = y + dir.y
      dirindex = (dirindex + 5) mod 8
   until x == column and y == row
   return contour
end function
```

Figure 5
The algorithm of contour following

2.2 The localization of special objects

The image processing algorithm is expected to recognize certain objects. Such an object would be the robot itself, or the target object. The recognition could be carried out by recognizing merely the object, or it can be done with the help of some kind of marking on the object, and recognizing this marker. This image processing system uses the latter.

Any object that the system must recognize, should have a rather simple shape on it, called marker. The marker is a dark shape on a light background, and sharp in outline. A further requirement of the markers is that they must have only one axis of symmetry, to ensure that their direction can be determined definitely. However, it is good if it really has one axis of symmetry, we will show the reason for this later. Such potential markers are visible in the figure below (Figure 6).

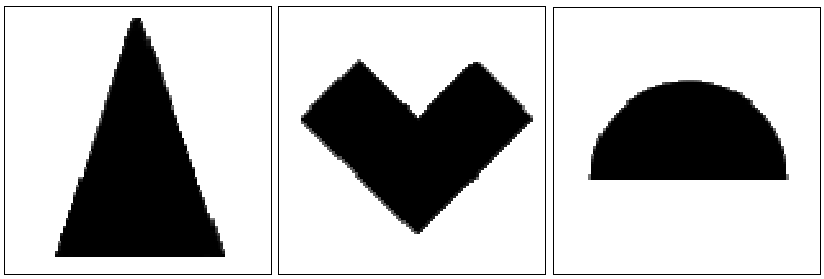

Figure 6
Possible markers. All has exactly one axis of symmetry.

However not all objects are needed to be marked with a marker. Those shapes, on which the system can not find a marker, are automatically classified as obstacles.

2.2.1 Extracting the markers from the shapes

The image processing steps so far has extracted the contours of each shape. In order to determine which marker can be seen on the shapes, the algorithm uses the original camera image to cut out the shapes using their contours. The process of cutting out is done with a use of a mask, which can be easily created using the contour of the examined shape.

The resulting cut-out shape is already in RGB space, so it has to be segmented to get a binary image. It has to be done so that on the resulting binary image the marker is separated from the background. This can be achieved easily if the image is segmented according to its intensity. To do this, first the algorithm calculates the image intensity histogram. Then using this histogram, the algorithm finds the treshold value, which separates the light and dark points from each other the best. The algorithm is iteratively try to fit two Gaussian distribution function to the light and dark pixels on the histogram. The treshold will be the value which is half-way

from the centers of the Gaussian distributions. Using this treshold, the image is easily segmentated by the intensity of its pixels.

After segmentation of the cut-out image is done, it is necessary to perform the opening and closing operations again, as it was done at the initial processing of the camera image. If there is a marker of the cut-out image, that must be the only shape visible on the resulting binary image. To recognize it, first the contour of the marker must be found. Based on the contour it is much more easier to recognize the marker. The contour finding works just like as it was described above.

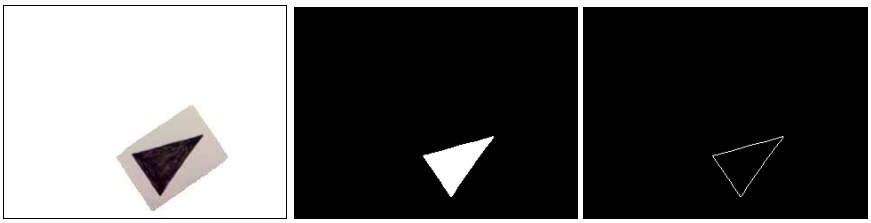

Figure 7
The cut-out image, the segmentation of that image, and the contour finding of the shape.

2.2.2 Determining the type, position, size and orientation of the marker

The markers are not only needed to be recognized, but it is also necessary to determine its location on the image, its orientation and its size. Furthermore, this information make easier the recognition of the marker.

There is a simple way of calculating all these parameters based on the contour. If the points of the contour are used as complex numbers $(x + j \cdot y)$, the contour can be Fourier-transformed, using the FFT algorithm. To have the same number of Fourier-coefficients for each markers, the contour must be resampled using fixed number of samples, in this case 32 samples.

The first Fourier coefficient, F_o is the center of gravity of the shape, so it is the position of the object (1). The absolute value of the second Fourier coefficient, F_1 determines the size of the shape, and its square is in relation with the area (2). Furthermore the orientation of the shape (3) and the phase of the sense of rotation (4) can be determined from the phase of the second and last Fourier coefficient. The shape can be rotated by multiplying the coefficients with $e^{j \cdot \varphi_1}$ (5), and the phase of the sense of rotation can be shifted by multiplying the coefficients with $e^{i \cdot j \cdot \varphi_2}$ (6), where j is the imaginary unit.

$$F_0 = \frac{1}{N} \sum_{k=0}^{N-1} c_k ; F_0^{'} = F_0 + \Delta \tag{1}$$

$$\forall i \longrightarrow F_i^{'} = F_i \cdot \lambda \tag{2}$$

$$\varphi_1 = \arg(F_1) + \arg(F_{N-1}) \tag{3}$$

$$\varphi_2 = \arg(F_1) - \arg(F_{N-1}) \tag{4}$$

$$\forall i \longrightarrow F_i^{'} = F_i \cdot e^{j \cdot \varphi_1} \tag{5}$$

$$\forall i \longrightarrow F_i^{'} = F_i \cdot e^{i \cdot j \cdot \varphi_2} \tag{6}$$

2.2.3 Normalizing the contours of the markers

The contour can be translated anywhere, rotated and scaled by changing these values. By knowing these parameters, the contour can be normalized, that is setting these values to predefined constants. The center of gravity of the contour is translated to the origo, by changing only one the first Fourier coefficient to zero. The area of the shape can be normalized easily both in the contour point representation, and in the Foruier-spectrum too. The shape can be rotated so that the longest diameter will be parallel to the imaginary unit vector. To do this, the orientation must be rotated by the calculated $-\varphi_1$, and the phase of the sense of rotation must be shifted by $-\varphi_2$. The shape can be normalized to identical contours applying inverse FFT to the altered coefficients, whatever direction was it previously aligned, whatever position it was translated on the image, and whatever size it was scaled (Figure 8).

There is however, a defect in this method: there is a 180-degree rotation uncertainty, so it can not be determined if the normalized contour is oriented properly, or it is rotated by 180 degrees. This ambiguity is resolved during the marker recognition, by comparing the normalized contour with the reference markers, and also with their 180-degree rotated counterparts. The phase of the sense of rotation should be changed with integer multiples of $2\pi/N$, otherwise the shape of the marker is distorted.

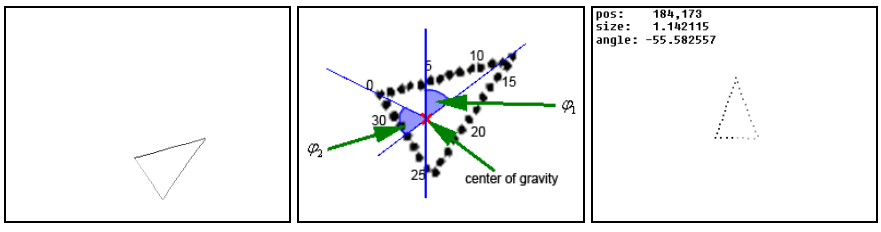

Figure 8

The contour of the marker, its center of gravity, its orientation, its phase of sense of rotation (from dark to light), and the resampled and normalized contour

The similar contours are transformed to numerically similar contours using the normalizing method above, the marker recognition can be carried out pretty easily.

The unknown normalized contour is compared with all the normalized contours of the reference markers, and the most similar marker is selected as the recognized marker. If the difference of the unknown marker and the most similar reference marker excess a specified treshold value, the contour is considered as an unrecognized shape.

The contours are compared by the quadratic distance of their points. This is possible because the contours have the same number of sample points as they were resampled during the normalization, and the phase of sense of rotation (φ_2 in Figure 8) is also identical. The contours can be compared using this method with sufficient accuracy even in noisy cases, moreover, it takes about an order of magnitude less computing time, as if the bitmap of the markers would have been compared. It is also better than a comparison method which would calculate numerous features from the images, because this method not only recognizes the markers, but it also determines their location, orientation and size, which are also needed during localization.

3 Results and conclusion

The localization system was built in a real navigation system, so it has been tested in real-life experiments. The results have shown that the segmentation of the camera image using a background image works perfectly in practice, which means that all the shapes and only the shapes of objects in working area is determined. Low-light conditions degradate the image signal-to-noise ratio, this also causes some segmentation errors, which can be compensated by the following nosie filtering step. Problem only occurs when an object has very similar color as the background, in such case it is not detected by the system.

The contour finder algorithm and the contour normalization using Fourier-transfrormation also work well. Every error of the contour finding can be originated from the error of the input of the contour finding, so the algorithm itself is performing well, only the errorous input could lead to wrong results.

The segmentation of the cut-out image based on the intensity histogram gives good results if the brightness of the cut-out images are adequate. It is important that the light and dark parts are well separated from each other. Unfortunately in most cases poor lighting conditions are the origins of the errors during localization.

The contour recognition step is the origin of the largest uncertainty during the image processing. The detection and recognition of a marker is correct, there are very few misrecognized markers. The recognition of obstacles however, which does not have markers on them, is worse. There are a lot of false positive marker recognitions. To overcome this problem, the markers are filtered by their size

parameter. In most false positive cases, the markers are much smaller or larger than the markers used on the special objects, like the robot or the target object, so these can be considered as obstacles too, despite there is a recognized “marker” on them. This improves the performance a lot.

In this paper we have presented a working localization system. Its weakness is that it can be aplied only when we have a fixed position camera, but it can be used well in indoor applications, like in real experiments of the testing of other navigation system parts, where the localization is considered a solved problem, or in applications like robot soccer. This localization method has also some problems as mentioned above, but with further development, these problems can be resolved. One of these could be the following of each object in the working area, so when it is misrecognized, or not recognized at all, it can be temporarily replaced with a forecasted value. An other improvement would be to use better classification method for the marker recognition, like artificial neural networks, or support vector machines.

References

[1] J.-C. Latombe, Robot Motion Planning. Boston , MA : Kluwer, 1991.

[2] I. Engedy, G. Horváth: Artificial Neural Network based Mobile Robot Navigation. IEEE International symposium on Intelligent Signal Processing, 2009

[3] Stuart Russel, Peter Norwig: Mesterséges Intelligencia modern megközelítésben. Panem kiadó 2005, ISBN:963-545-411-2

[4] Sebastian Thrun, Wolfram Burgard, Dieter Fox: Probabilistic Robotics MIT Press 2005, ISBN:0-262-20162-3