View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

FACTA UNIVERSITATIS
Series: **Electronics and Energetics** Vol. 30, N^{o} 3, September 2017, pp. 403 - 416
DOI: 10.2298/FUEE1703403P

ANALASYS OF TWO LOW-COST AND ROBUST METHODS FOR INDOOR LOCALISATION OF MOBILE ROBOTS *

Miloš Petković, Vladimir Sibinović, Dragiša Popović, Vladimir Mitić, Darko Todorović, Goran S. Đorđević

University of Niš, Faculty of Electronic Engineering, Niš, Serbia

Abstract. *This paper presents two simple and cost effective indoor localisation methods. The first method uses ceiling-mounted wide-view angle webcam, computer vision and coloured circular markers, placed on the top of a robot. Main drawbacks of this method are lens distortion and sensitivity to lighting conditions. After solving these problems, a high localisation accuracy of ±1cm is achieved at about 5 Hz sampling rate. The second method is a version of trilateration, based on ultrasound time of flight distance measurement. An ultrasonic beacon is placed on a robot while wall detectors are strategically placed to avoid an excessive occlusion. The ZigBee network is used for inter-device synchronisation and for broadcasting measured data. Robot location is determined as a solution to the minimisation of measurement errors. Using Nelder-Mead algorithm and low-cost distance measuring devices, a solid sub 5 cm localisation accuracy is achieved at 10Hz.*

Key words: *Robot localization, Nelder-Mead, Gnu Scientific Library, USB camera, OpenCV*

1. INTRODUCTION

The robot or objects indoor localisation is a vital research area, intrinsically important in expanding competences of future low-cost home robots. A comprehensive research overview is best gained by browsing applications in Microsoft's Indoor Localisation Competition, held three years in a row [2], starting with 2014. The best scores are often achieved through engagement of expensive components such as LIDAR's. However, when it comes to a low-cost mobile robot, it is demanded that localisation is both reliable and inexpensive. Consequently, a compromise is reduced to the ratio of positioning accuracy and the costs of producing and implementing localisation. This is not difficult to

Received October 7, 2016; received in revised form December 15, 2016
Corresponding author: Miloš Petković
Faculty of Electronic Engineering, University of Niš, Serbia, Aleksandra Medvedeva 14, 18000 Niš, Serbia
(E-mail: milos.petkovic@elfak.ni.ac.rs)
* An earlier version of this paper received Best Section Paper Award at Electronics Section at 3rd International Conference on Electrical, Electronic and Computing Engineering
IcETRAN 2016, Zlatibor, Serbia, 13-16 June, 2016 [1]

achieve for service robots. For example, home cleaning robots do not require high precision localisation for wandering. However, if servicing an arbitrary point in workspace is required, a comprehensive research would be needed in order to stay below the price tag.

Furthermore, the indoor localisation is especially challenging [3] due to a problem with weak or non-existing GPS signal, and due to occlusion problems as a result of variety of objects and their placement within a room. Thus, usage of any method that needs a straight line visibility between two parts would require a redundant solution. On the other hand, such increasing of complexity leads to the increase of the overall costs. Therefore, a careful consideration has to be made before choosing the right method.

The localisation is based on a low-cost, ultrasonic, time-of-flight, distance measuring system. It is similar to Cricket [4, 5]. The robot emits an ultrasonic beacon signal, while fixed wall-mount devices measure Time-of-Flight. This kind of system is often inexpensive, so increasing redundancy by adding more of wall devices is not increasing the overall system cost considerably.

Use of straightforward trilateration imposes few problems. The first one appears when, due to a measurement error, three or more spheres do not intersect at a single point. For smaller measurement errors this could be neglected and considered as a rounding error. Since our system had better than ± 10cm accuracy, this could not be the case. The other problem, a special case of the first one, is absence of intersection between spheres in case of negative errors. Mathematically speaking, a solution of trilateration is imaginary. Arguably, accuracy could be improved by calibrating each wall unit separately, and ensuring their precise coordinates. However, in cases of occlusion and reflections, these kinds of problems would reappear. Therefore, we seek a solution through a criterion-based optimisation to get as close as possible to the point that minimises the measurement error.

Further improvement could be achieved by using a secondary, more accurate, localisation system. When these two systems run in parallel, the second system would be a good reference for the calibration of the initial one. For this purpose, localisation rate does not even need to be high. Therefore, we decided to base the secondary system on computer vision and recognition of passive markers. Low-cost requirement was priority as well, so overcoming typical drawbacks of such an image processing methods was important. Fisheye lens distortion was removed by using known geometry [6], and complexity of object recognition was avoided by simplification and colour coding of markers [7]. The rest of them will be presented in details in the following section.

2. VISUAL FEEDBACK MAPPING FOR LOCALISATION

2.1. Materials and method

We have placed a fish-eye webcam on the ceiling in the middle of the test room. In order to make this system affordable, we based it on a full HD webcam, Genius F100, with 120° view angle lens, and moderate power PC of AMD Athlon II X3 455 3.30GHz, ATI Radeon HD 6450 and 4GB DDR3 RAM.

The distance between the camera lens and the floor is 3.1 metres Therefore, the camera with 120° wide view angle lens can cover the area of 4×3 m. A grid of 0.5 × 0.5 m was drawn on the floor to ease calibration and provide a visual clue during the measuring. The grid is highly accurate, with only 5 mm distortion error over the diagonals of 5 m.

Rectification was crucial for this system because a wide-angle lens that is used has intrinsic distortion. Its removal is easy since the camera itself is stationary and marker height was supposed to be constant. Sampling images of the marker at different positions reveals levels of distortion. This data is then used to invert the effects. We gathered those samples at drawn greed points. As an aid we used a tripod, and as a marker we used an orange ball, as shown in Fig. 1. Height of this customized calibration tool was set to 1.1 m which reduced the distance between the camera lens and the markers to exactly 2 m. After relocating the tripod around the grid, and overlaying all images one on top of the other, we generated Fig. 1. The central part of the grid, which aligns with the middle of the camera, is free from the lens distortion. That is why we dropped out some middle points but left one on the edges and corners, where the distortion is at its largest.

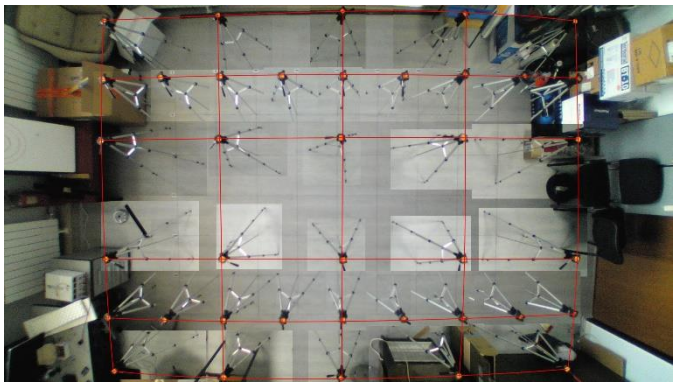

Fig. 1 Overlay of tripod with marker as calibration points in our test room.

We found it fitting to divide the frame to 9 regions and linearize them independently. This keeps rectification simple and calibration easy. Number of pixels between sampled points was manually counted and converted to centimetres. Later on, calibration constants and offsets for each region were calculated, and embedded in the positioning algorithm.

Distortional displacement within the camera image is not the same for close and distant objects. Obviously, an additional calibration is required if height of the marker is changed. However, there is no need for this if its placement is optimal. The best place for the marker is on the top of the tracked object, where chances for occlusion are negligible. We should note that markers placed higher do require more linearization sectors, as the difference between the real position of the object on the floor and the camera frame varies.

An important part of the simplification of the marker recognition is its colour coding. This makes identification easy. In addition, extracted marker shape is more accurate, which enhances precision in marker centre calculation. We implemented this extraction through pixels classification. The classification of pixels generates a black and white image, where white pixels are originally in adjacent colour space of the marker. This new image contains slightly etched shapes of markers with some artefacts as well. Another layer of smoothing filter corrects this. We suggest Gaussian blur, as it produced quite useful results for us. Larger artefacts, if they happen to persist, are filtered out by shape and size classification. We opted for a circular marker design.

Marker colour distinction also enables multi object tracking, or orientation recognition by engaging two markers per object. In particular we used the larger, orange coloured, marker for tracking position, while the smaller one which was green, was an aid in tracking robot heading. This marker combination proved to be the most desirable with respect to the program execution time.

After marker positions in pixels are extracted, in our case after the centre of the only remaining circle is calculated, its conversion to absolute position in centimetres comes in place, by using formula (1) and calibration constants.

$$MPc = \left(\frac{MP - os_1}{C_{calib}} \right) + os_2 \tag{1}$$

MP is the marker position in pixels while os_1 is the marker offset in pixels for the region it belongs to. C_{calib} and os_2 are linearity gain and offset in centimetres for the region. Their values, for all nine calibration regions, are given in Table 1. Finally, MPc is marker position in centimetres, in coordinate system which centre is placed at the bottom left calibration point of Fig. 1.

Table 1 Calibration Constants and offsets for conversion into cm

Marker osition	os_1		C_{calib}		os_2	
	X	Y	X	Y	X	Y
Upper left	285	28	3.44	3.64	0	0
Centre left	285	28	3.44	3.43	0	0
Lower left	285	900	3.36	3.23	0	250
Upper middle	620	20	2.29	2.26	100	0
Centre	620	200	3.5	3.5	100	50
Lower middle	620	1319	3.6	3.6	50	0
Upper right	1319	28	3.44	3.43	300	0
Centre right	1319	28	3.44	3.43	300	0
Lower right	285	900	3.36	3.23	0	250

2.2. Implementation and results

The program was done under Window 10 with Microsoft Visual Studio Community 2015 with inclusion of OpenCV library version 3.0. At the start up of the program, camera parameters, such as brightness, contrast, saturation, hue, gamma, sharpness and exposure, are pre-set to suitable values. This parameters tweaking enhances proper pixel colour classification at given lighting conditions. We experimentally determined them for our Neon light test room, with west facing windows. Prior to the pixel classification, the image is converted from RGB intoHSV. After this, the inRange function is used, as classifier, to generate black and white image. As already stated, we used GaussianBlur for BW image smoothing and smaller artefacts removal.

In the next step we calculate the marker position by data extraction. We used SimpleBlobDetector in this process. Parameters of this function are set to ignore everything but circles of particular size, thus filtering any larger artefacts. It is the middle point of a found blob, that is considered as the marker position, pixel-wise.

To speed up the program we decided to trim sampled frames only to Region-Of-Interests (ROI). This way, computationally intensive functions like SimpleBlobDetection shall execute faster. During the initialisation phase, the program searches the whole frame for marker, until it is found. Afterwards, the ROI is extracted from frames based on previous marker position and the maximum expected movement.

This ROI trimming not only shortens calculation time but also filters out other objects of similar visual properties as the marker's. Precaution that needs to be taken into account is that these kinds of objects are not present during the start of the program. In such cases it could happen that some other object is recognised for tracking, instead of the marker, and then the wrong ROI would be extracted.

In the last step, marker position in pixels is converted into actual position in centimetres, in absolute coordinate frame attached to the floor. Approximately, one centimetre corresponds to 2.5 pixels.

Initial verification of the system includes repetitive measurements with the marker, placed on a tripod, at an arbitrary point in workspace. This tests calibration accuracy and system repeatability. Upon consecutive large number of measurements, we can confirm that the system is reliable and repeatable at the acceptable level. The number of 1572 location samples of a still marker was acquired. On average, it required 235 ms to complete one localisation cycle. With 4.26 Hz localisation rate, such system is not suitable for localising high speed mobile platforms. Nevertheless, a robot that travels at comfortable speed of 0.3 m/s would be localised at points 7 cm apart. This can be considered acceptable in applications such as fetching objects to the customer or telepresence, but not in precise object handling. Repeatability for all 1572 measurements was within one-centimetre range which corresponds to 2 to 3 pixels of the camera. Due to small variations in lighting and inherent camera noise, there exists a jitter in marker position, found by a simple blob detector. When position in pixels is converted into position in centimetres, and rounded, the jitter passes to marker position in centimetres. An improvement is possible with the increase of camera resolution, or perhaps with the increase of the number of linearization sections. However, we find this system static performance quite satisfactory for calibration and support of low-cost, ultrasound based, time-of-flight localisation system.

For the dynamic testing of camera localisation system, we have decided to make several circular motions in the centre of the test room. There are two reasons for this. The first is simplicity of trajectory equations, which allows easier data analysis later on. The second is trajectory length that should provide sufficient time for acquisition of a sufficient amount of data. Since the test room was not large enough for straight line movements, the most logical trajectory then was circular. Also, it can be easily performed without the need for an expensive setup. For example, a simple remotely driven mobile platform, like more powerful homemade RC car, suffices. Another proposal is a motor driven rotating stand. At our disposal was a small, student grade, robotic platform. After attaching the marker to it, we have initiated the localisation and made 30 laps, with approximately constant speed of 20 cm/s. The programme was set to log the marker positions with the time stamps of frame acquisitions. The time stamps are expressed in milliseconds and the local time is measured from the beginning of the test. Fig. 2. shows plotted positions of the marker. As it can be noted, the trajectory is circular but there exists some slight movement of the centre.

Fig. 2 Logged trajectory of circular motion of marker. Number of repetitive cycles is 30.

In the next step, we have done a time stamp analysis of about 426 s long measurement streak. This was necessary for the later analysis of trajectory. The logged time seemed rather linear when plotted. An average time between processed frames is 236 ms, with standard deviation of 22.9 ms. Differences on the histogram of time between two successively grabbed frames are an interesting observation, which is shown in Fig. 3.

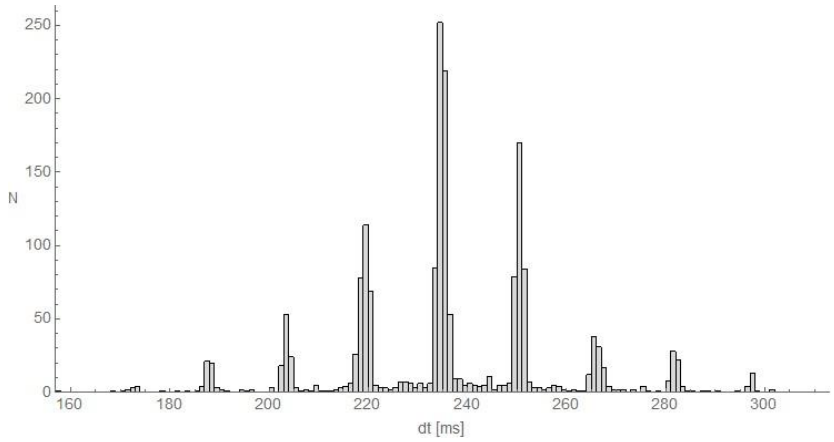

Fig. 3 Histogram of time differences, *dt*, between two successively grabbed frames.

Histogram peaks are at an equal distance of approx. 15 ms. Since the camera streams at about 30 fps, this 15 ms seems like a half of a frame time. An average period of 236 ms is then correlated to 7 frames. Considering a slight variance in stream frame rate and code execution, it could lead to a frame grabbing jitter. The jitter would be only one frame. Its effect would be increase in localisation uncertainty of one frame time multiplied by the speed of marker. If speed is low, uncertainty increase is only a few centimetres. In our case, for speed of just under 20cm/s, it is evaluated to 0.6cm. When time stamps are

converted to integer number of frames from the beginning of test, and time difference is recalculated, the histogram looks like in Fig. 4. Now it is much clearer that almost half of the samples are taken with 7 frame difference. From the remaining samples, about one third is with 6 frame difference and one third with 8 frame difference. In other words, standard deviation is 0.77 frames. To conclude, as far as the timing analysis is concerned, since no real time OS were used, a variance in processing frames and sampling does exist. However, it is not more than 10 frames or one third of a second.

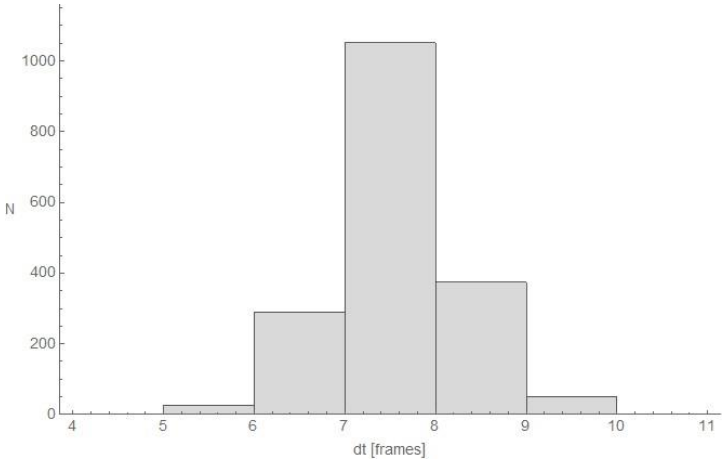

Fig 4 Histogram of time stamp differences,
when time is converted to frames with 30fps rate.

In parallel with the dynamic performance test we have done an additional timing analysis. We wondered whether this kind of localisation system could be integrated as small localisation device capable of broadcasting tracked object location via Wi-Fi. Thus, the image processing PC was set to send position via UDP packets to PC within the same wireless network. Comparing the time difference of localisation frame sampling time and time of the UDP arrival, we got 236 ms of time difference between location information. On the other hand, a standard deviation is now 133 ms, which is almost 6 times more than for the localisation alone. The main culprit is packet buffering, and wireless signal quality. Due to them, considerate number of packets was late. Note also that this differential analysis excludes fixed amount of latency from Wi-Fi, as it did with camera frame grabbing. Since we are using low-cost off the shelf components, it is not possible to determine accurately this kind of delays. At least not without the use of special setups. Conversely, we find sending location via UDP packets and Wi-Fi for control purposes plausible, however, control algorithms must either be rugged enough for variable time delays or take advantage of frame time stamp and perform small corrections of received location.

In the following step, we have done trajectory analysis in two stages. Firstly, we have found trajectory radius r and centre (x_0, y_0), as well as speed of centre movement (v_x, v_y). This was achieved by finding the best fit for function (2).

$$f(x,y,t) = r - \sqrt{(x_0 + v_x t - x)^2 + (y_0 + v_y t - y)^2} \tag{2}$$

Basically, function (2) represents difference in radius of acquired location and the estimated one. For any measured point it should be equal to zero. The best fit result gave r of 41.9 cm, (x_0, y_0) of (191.1, 149.6) cm, as well as (v_x, v_y) of (0.264, -0.096) mm/s. The best fit average error is 6E-16, while the standard deviation is 0.633cm. It is interesting to note that the standard deviation is on the level of mentioned frame jitter, for an object with speed of 20 cm/s. Nevertheless, we state that accuracy of this system for moderate speed of tracked marker is ±1.5cm, or ±2.25cm if absolute limits are applied. So performance of system for tracking a moving object does not go far off from the static measurements.

Now, if we take into consideration that speed of the marker was constant, we can assume that coordinates (x, y) change as in (3), where ω is constant angular velocity and ϕ is initial angular offset. The formula (3) is our ideal mathematical model of real trajectory.

$$(x(t), y(t)) = (x_0 + v_x t + r\cos(\omega t + \varphi), y_0 + v_y t + r\sin(\omega t + \varphi)) \tag{3}$$

Difference of trajectory given with the formula (3) and measured data is given with function (4). Ideally, it equals zero.

$$g(x,y,t) = \sqrt{(x_0 + v_x t + r\cos(\omega t + \varphi) - x)^2 + (y_0 + v_y t + r\sin(\omega t + \varphi) - y)^2} \tag{4}$$

The best fit result gives angular velocity of -0.439 rad/s, which translates to 18.4 cm/s peripheral speed, and angular offset of 3.163 rad. Negative velocity comes from the clockwise direction of trajectory. Average fitting error is 2.8 cm and standard deviation is 1.8 cm. Since this result seems much worse than the one from trajectory path analysis, we conclude that this method is accurate for localisation within a frame. However, when a tracked object is moving, due to unsynchronised frame grabbing, larger margin of error occurs. Indeed, when we calculated travelled distances between successive sampled frames, we got 4.4 cm in average and standard deviation of 0.5 cm. This seems like a great variance, considering the fact that marker speed was pretty constant. After calculating temporal velocities, we got the result that average speed is 18.6 cm/s and standard deviation is 0.6 cm/s. So generally, due to variance in precise image capturing, we get very rough velocity approximation based only on two samples. However, after filtering, this information seems quite right.

3. TIME-OF-FLIGHT LOCALISATION METHOD

3.1. Materials and method

A simplified block diagram of time-of-flight distance measurement system is presented in Fig. 5. There is a beacon that emits ultrasound on the left and a wall mount device on the right. The minimum number of wall devices necessary for successful trilateration is three. Before the beacon fires a streak of waves, it notifies a wall device via radio module, and it starts the counter. When the wall device detects emitted sound, it stops the counter. Information about time of flight is then sent via radio. Distance is calculated after the time

of flight is multiplied by the speed of sound. Since the device is for indoor use only, speed changes due to temperature variations are neglected. Multiple ultrasonic transducers are used in both devices. Beacon covers 360 degrees horizontally and about 45 degrees vertically. The Wall device covers about 140 degrees horizontally and 45 degrees vertically. Therefore, a proper redundancy is needed for specific coverage. Currently we use 4 wall devices placed in corners of a rectangle, with an orientation toward common centre. We made sure to do the measurements only in areas covered with more than 3 wall units. Although devices are low-cost to make, this is only an initial accuracy testing and we find it irrelevant to have coverage of any preferred size or shape.

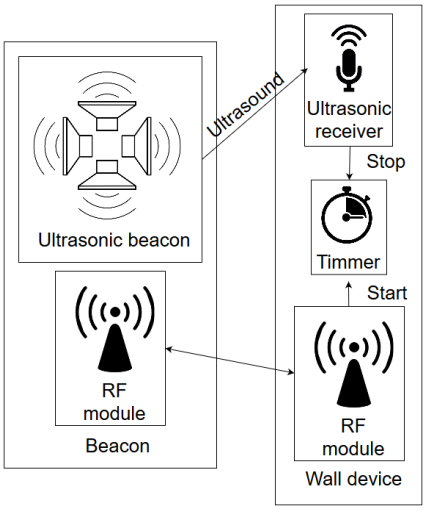

Fig. 5 Simplified block diagram of system: ultrasound emitting beacon on the left and time-of-flight measuring wall mount device on the right.

In order to overcome the problem of trilateration when using low-accuracy, but also low-cost, distance measuring system, we have based solution calculation through minimisation of the sum of squares of measurement errors. In the minimisation function

$$F = \sum_{i=1}^{n} (|p_i - p_r| - d_i)^2, \tag{5}$$

n represents number of wall devices that responded to ultrasonic beacon. Position vector of beacon p_r and position vectors of wall devices p_i are defined in 3D and in regard to some ground reference point. Again, vectors p_i, where i is from 1 to n, are known, as they are measured during localisation system installation. The x and y axes are in the plane of the floor while the z axis is oriented toward the ceiling. Measured distances d_i are obtained short after the beacon signal is emitted. The function minimum is located around the beacon's position. This function is equal to zero when no measuring error is present. Otherwise, a small precision uncertainty will occur in the case of measurement errors. When measured data noise is of random nature, there is no possibility to narrow down solution search area, at least not statically.

In order to test this method, we have created a Wolfram Mathematica script. It simulates a system of 3 or 4 wall devices and a beacon. Distance measuring error is randomly generated and added to the precise value. We set the x and y plane to correspond to the floor and the z axis to point to the ceiling. Although this method allows finding position of beacon in 3D, we are more interested in keeping its height constant. This would be most probable use-case in mobile robotics. Therefore the script visualises 2D plane of the z axes at the fixed height of beacon of 1.3 m, as in Fig. 6. Possible beacon positions in that plane are circles, designated with thick circular arcs in Fig. 6. Note that both positive and negative measurement errors were introduced. The dot represents calculated position, while the short lines, that connect it to the arcs, are estimated measurement errors. The squares represent projection of wall devices on the plane. They are also centres of the circles. The lower left part contains magnified detail around the dot.

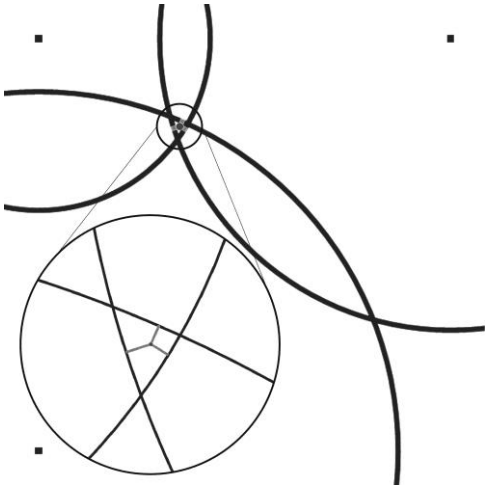

Fig. 6 A plane, where the z coordinate is constant 1.3 m, that contains calculated robot position which is shown with a dot. Possible beacon positions, for that plane, according to the measured data are circles, are shown partially with thick arcs. The short lines represent estimated measurement error. The squares represent projection of wall devices on the plane. They are also centres of the circles. The zoomed detail around solution point is presented at the bottom left.

Visual checks were only used as an aid, for better understanding of behaviour of solution in response to errors and device placement. For example, actual and calculated positions are identical when there is no measurement error. Equal errors in all wall devices tend to cancel each other. Numeric evaluation is done as well.

We used *NMinimize* function for minimization. Available minimisation methods are Nelder-Mead [8], Differential evolution [9], Simulated annealing [10] and Random search [11]. We used them all simultaneously in order to compare them with respect to efficiency and accuracy. Wall devices were placed in rectangular pattern with same height, as they might be used commonly. We generated random beacon positions, calculated accurate distances to wall devices, and then added a Gaussian error in range of ±10 cm. Beacon

location found by minimisation of function (5) was accurate enough, mostly bellow 5 cm error. However, in some cases, the error went up to extremes of almost 20 cm. That occurs in situation when two adjacent wall devices have maximal error of +10 cm while the opposite two have –10 cm of error. Probability for this is rather low and general conclusion is that this method works quite nicely. It shows robustness to both positive and negative measurement errors. Solution exists independently from the number of wall devices. Increasing their number to overcome temporary occlusion problems does not affect solution calculation, neither in complexity nor in time.

Comparison of results of four minimisation methods showed no significant difference between them. Difference in accuracy was well below 1 cm. The same could be said about efficiency. So we chose the Nelder-Mead for practical implementation.

3.2. Implementation and results

After successful method of validation in Wolfram Mathematica, we have built C++ code. We have chosen to use Nelder-Mead solver from the Gnu Scientific Library. The program was used on the MinnowBoard computer with non-commercial Ubuntu OS. The MinnowBoard is an open-source, 64-bit Intel® Atom™ based mini/embedded PC.

Initial tests were done with pre calculated examples, generated with Mathematica script. Execution time was about 1 ms, in average. Though sometimes it reached 3ms However, this was not the only program running. Nevertheless, we find this quite satisfactory. For service type robot speed, this introduces a localisation error less than one millimetre. Delays in distance measuring system are much greater and position sampling is below 10Hz. If by any chance execution time has to be reduced it could be done by lowering solver precision. We noticed that in most cases 10 to 20 iterations were enough to get the right position of centimetre resolution.

As in the Visual Feedback Localisation in Section 2, we initially verified the system, through repetitive measurement with beacon fixed at arbitrary position in the workspace. This verification helps understanding repeatability in measurement and also gives reasonable confidence in usability for further implementation on a mobile robot. Upon consecutive large number of measurements, we can confirm that the system is reliable and repeatable at an acceptable level. The beacon firing rate was fixed, with the period of 150 ms, which is frequency of 6.67 Hz. Although we could set it up to 10Hz, we did not want to use it at its limits. A number of 1172 measurements at fixed position is presented as histogram in Fig. 7. The average point is (213cm, 169cm) and standard deviation is 0.62, or 0.38 for x axis data and 0.49 for y axis data. In general, only 0.26%, or 3 points, is outside of ± 1.5cm accuracy region.

These data show a satisfactory initial accuracy of the method. Although it returns a bit more scattered location than the camera based method, it works faster.

For dynamic testing of ultrasonic based localisation system, we have done the same test as with camera based localisation system. Furthermore, we decided to do both tests in parallel. This would make the comparative analysis easier. So the ultrasonic beacon was placed on the same platform as the marker. Since the platform, which was in the centre of the test room, was making circular motions, both the marker and the beacon had the same centre of rotation. Since the beacon must not occlude the marker it was placed as close as possible to it. Nevertheless, there still existed a slight difference of almost 3 cm, in their radiuses. The

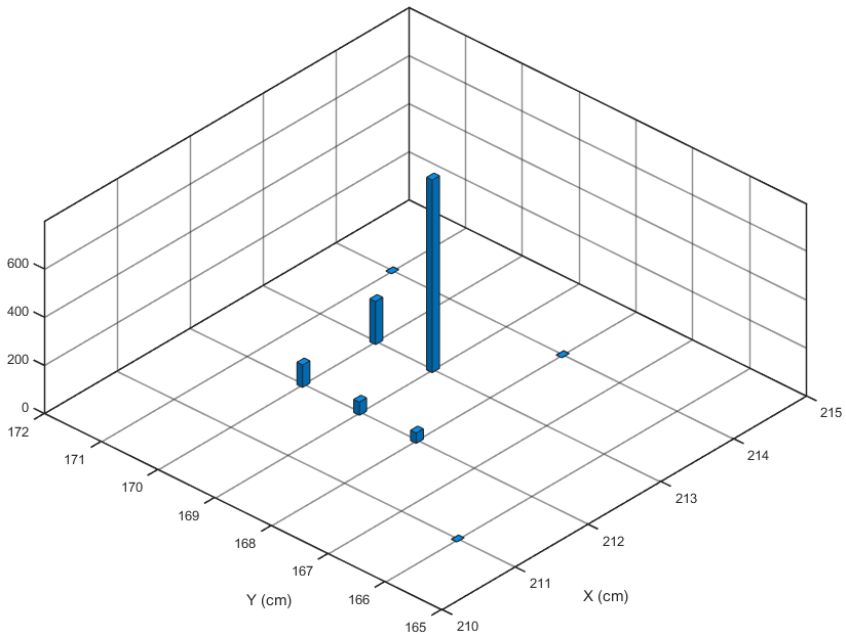

Fig. 7 Histogram of 1172 measurements at single beacon pint. Most often measured position is (213, 169) cm.

initial trajectory analysis confirmed a slightly lower localisation accuracy of this system compared to the camera based one. Therefore, we decided to use the centre of rotation (x_0, y_0) calculated from the camera based system trajectory analysis, as well as speed values (v_x, v_y), and repeat fitting process with (2). The best result gave r of 44.8 cm, an average error of -3E-14, and a standard deviation of 7.44 cm. This result looks a lot higher than the one for the static test. This stems from the poor choice of RF modules for the system. These are low power ZigBee modules. Several studies indicate low performance of ZigBee communication in presence of Wi-Fi signals. This is nicely summarised in [12]. There it is clearly stated that Wi-Fi signal can corrupt ZigBee signal on bit level or cause drastic increase in retransmission. Since our setup room had one Wi-Fi router and there were plenty more distributed in nearby offices, we have noticed both effects. When we analysed time of arrival of packets from single wall device we discovered that latency between packets is quite drastic. Instead of having packets at regular beacon firing intervals of 150 ms, plus or minus time of flight of ultrasound up to 5 m, there were packet buffering where packets came with less than 30 ms difference. Since packets with distance information were not time stamped at transmitter side, it was impossible to determine whether the wall device failed to transmit after one beacon firing or the measured distance information came after the following beacon firing. In such cases mixing of data occurred. It could be otherwise interpreted like higher inaccuracy in distance measurement, which leads to higher localisation error. At some rare moments, packets from unknown wall unit address were received, which we interpret like obvious pollution of data. It is quite possible that lower performance of ZigBee modules is even due to its quality, since they were one of the cheapest on the market.

Problems associated with ZigBee modules could perhaps be overcome by using better and more reliable modules, and by implementation of some better protocol for sending data over ZigBee as suggested in [12]. Another solution could be using modules that avoid overcrowded 2.4 GHz region at all.

Since we had already identified the problematic latency in our system, we skipped the second part of trajectory accuracy analysis that we did with the camera based system. Simply, it would not add any value to the results.

4. CONCLUSION

We have implemented two methods for indoor localisation, and tested them against each other under identical conditions in our testing facility. After initial static testing and validation of systems accuracy, with laser range finder, we have determined that the first method, the camera-based one, has better accuracy. Although it has half of localisation speed than the time-of-flight method, we have decided to use it as referent system during dynamic testing. Since mobile service robots have moderate speeds, then the localisation rate of visually based system is quite adequate. Dynamic test showed that ultrasonic based localisation system has lower accuracy and success rate of measurement, due to ZigBee modules communication glitches that require additional attention and improvements. On the other hand, the first method has its own pitfalls. It is, foremost, sensitivity to changes in lighting condition. It also requires a comprehensive calibration which should be automated in order to make it an off-the-shelf localisation solution. The standard PC could be easily replaced with embedded type PC, for example, with any of newer Raspberry Pi series. Nevertheless, both systems showed simplicity in setting up and use. Their low implementation cost makes them affordable for use in education and some less demanding real life applications, such as service robots.

In conclusion, camera-based system is better for laboratory conditions due to its high accuracy. The other system, although less accurate, is more suitable for a variety of other locations.

REFERENCES

[1] M. Petković, V. Sibinović, D. Popović, V. Mitić, D. Todorović and G. S. Đorđević, "Robust indoor localisation methods of mobile robots: direct visual feedback and time-of-flight trilateration", In Proceedings of the 3rd International Conference on Electrical, Electronic and Computing Engineering (IcETRAN 2016), Zlatibor, Serbia, June 13 – 16, 2016, ELI2.6 1-6.

[2] "Microsoft Indoor Localisation Competition". research.microsoft.com. N.p., 2016. Web. 15 Apr. 2016.

[3] J. Borenstein, et al, "Mobile Robot Positioning - Sensors and Techniques", Invited paper for the *Journal of Robotic Systems, Special Issue on Mobile Robots*, vol. 14, no. 4, pp. 231 – 249, 1996.

[4] "The Cricket Indoor Location System: An NMS Project". cricket.csail.mit.edu. N.p., 2016. Web. 15 Apr. 2016.

[5] N. B. Priyantha, A. Chakraborty and H. Balakrishnan, "The Cricket Location-Support system", In Proceedings of the 6th ACM MOBICOM, Boston, MA, August 2000.

[6] C. Hughes, et al., "Wide-angle camera technology for automotive applications: a review", *IET Intelligent Transport Systems*, vol. 3, no. 1, pp. 19-31, 2009.

[7] Z. Garofalaki, et al, "Object Motion Tracking Based On Color Detection for Android Devices", *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 9, no. 4, pp. 970-973, 2015.

[8] J. A. Nelder and R. Mead, "A simplex method for function minimization", *Computer Journal*, no. 7, pp. 308–313, 1965.
[9] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces", *Journal of Global Optimization*, no. 11, pp. 341–359, 1997.
[10] S. Kirkpatrick, C. D. Gelatt Jr and M. P. Vecchi, "Optimization by Simulated Annealing", *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[11] L. A. Rastrigin, "The convergence of the random search method in the extremal control of a many parameter system", *Automation and Remote Control*, vol. 24, no. 10, pp. 1337–1342, 1963.
[12] C. M. Liang, N. B. Priyantha, J. Liu and A. Terzis "Surviving Wi-Fi Interference in Low Power ZigBee Networks", In Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems, ACM NY, 2010, pp. 309-322.