Visual Vehicle Localization System for Smart Parking Application

Hicham Lahdili(✉) and Zine El Abidine Alaoui Ismaili

ENSIAS/Information, Communication and Embedded Systems (ICES) Team, University Mohammed V, Rabat 10010, Morocco
dh.lahdili@gmail.com, z.alaoui@um5s.net.ma

Abstract. With a vision of proposing a fully automated parking management solution for smart parking, in which all the operations in the process of parking will be automated. And as a first step we will focus on vehicle localization inside parking based on image processing theory. Video based localization algorithms present an important interest in the field of intelligent video surveillance, the integration of such functionality in the surveillance system will revolt their classic roles. Navigation tool and other amazing systems can easily build based on such feature. This paper describes an implementation of a FPGA based real-time visual system for vehicle localization. Vehicle in the video frames are extracted after the application of the background subtraction method on the input image using a background reference image. The dynamic threshold used is computed by the Otsu method. Finally, the object mask resulting from the segmentation process is used to compute the relative distance to the camera based on the relation between the ratio of the size of a vehicle on the camera sensor and its size in real life which is a function of the camera focal length and distance between the vehicle and the camera. The experimental results show that the proposed system is sufficiently satisfying the real time constraint (under the 100 MHz frequency a 32 frames per second is achieved for the 1440 * 1080 resolution, and under 50 MHz frequency a 41 frames per second is achieved for the 640 * 480 resolution) with an accuracy error around the centimeter level.

Keywords: Smart parking · Vehicle localization · Image processing
Background subtraction · Otsu method · Real time · FPGA · HDL

1 Introduction

In nowadays, cities are constantly growing and with this swelling appear certain problems, one of these major challenges is the problem of parking management. In a vision of proposing a fully automated parking management solution, in which all the operations in the process of parking will be automated, comes our paper for presenting a visual vehicle localization system. The parking management system based on the vehicle localization information, and the available free place will generate commands with speed, direction and the angle of rotation information. The vehicle that will be equipped with a dedicated system designed for this purpose will execute the commands and move through the parking to its reserved place. As a result, the whole of the parking process will be done without human intervention.

M. Ben Ahmed and A. A. Boudhir (Eds.): SCAMS 2017, LNNS 37, pp. 366–378, 2018.
https://doi.org/10.1007/978-3-319-74500-8_34

The localization information generated by the visual localization system can be used beside of vehicle localization to build amazing tools for different purpose as guiding people during navigating in unfamiliar buildings like airport, museum, office building. And as our system is based on movement detection we can easily limit data storage for surveillance system to active area (The data issued from a camera will be stored only when moving objects are detected).

Despite the remarkable progress realized in image processing discipline, the complexity of the algorithms used (the increase in image resolutions, as well as the need to implement increasingly complex techniques) on the one hand, and the resources required (computational and memory) on the other hand, have made their implementation in embedded systems a challenging task. This complexity increases with the introduction of the constraints imposed by the standards according to the fields of application, ex: real time constraints, safety standards…

In this paper, we will present our proposal for a visual vehicle localization system (VLS), developed for static camera. Vehicles in the video frames are extracted after the application of the background subtraction method on the input image using a background reference image. The dynamic threshold used is computed by the Otsu method. Finally, the object mask resulting from the segmentation process was used to compute the relative distance to the camera based on the relation between the ratio of the size of the vehicle on the camera sensor and its size in real life which is a function of the camera focal length and distance between the vehicle and the camera.

The remainder of this work is organized as follows; Sect. 2 presents the related works, where the implemented algorithm and description of its parts is given in Sect. 3, in Sect. 4 the implementation board was presented in addition to the syntheses summary. Execution time characteristic of the implemented algorithm, as well as some experimental results are covered in Sect. 5. And finally Sect. 6 draws the conclusion.

2 Related Work

Positioning systems are more and more used in our daily life to perform more complex tasks such as navigation (everywhere in the globe and in all-weather conditions) or simply for pleasure like augmented reality games. The Global Positioning System (GPS) [8] which is the most widely used navigation system in the world receives signals from multiple satellites and employs a triangulation process to determine the physical localizations with an accuracy error around the meter level. The latter is in general acceptable for outdoor applications. Unfortunately, this system reaches its limitation within the buildings and closed environment because of the attenuation of electromagnetic waves. This limitation in addition to the low accuracy of traditional positioning system present the motivation to conduct various researches on different physical signal in a vision to build new positioning systems.

Wireless technologies such as ultrasonic [1, 2], infrared [3], radio-frequency based systems which may be radio-frequency identification [4], received signal strength of RF signals, Bluetooth and WIFI those systems are based on concepts like Time of arrival, Angle of arrival and Received signal strength indication to calculate the distance between the transmitters and the receiver [5–7]. The main disadvantage of the Wireless technologies that only object equipped with a receiver can be located.

Non-radio technologies like visual system. Different approaches are used in this category, visual marker based system [9]: markers are placed at specific locations, and when a device (mobile robot) identifies a marker, it can be localized thanks to the markers database. Map-based visual localization [10]: first a collection of successive image of an area (building, road, …) is used to build a dataset of images about this environment, then any device wants to be localized in this area will need just to take an image of its environment which will be compared to the dataset to find its current position. And finally real time visual localization system [17, 18]: the main idea behind those systems is based on the use of a camera network to localize generic objects such vehicle or people. Their major disadvantage is their low accuracy (0.37 m in the best case).

3 Proposed Algorithm

The main idea behind the proposed algorithm is that vehicle size in an image is a function of some camera characteristics (focal length and the sensor size) and vehicle's features such as weight, height and its distance to the camera. This means that for any vehicle with a known size in the field of view of a camera, we can estimate its distance to this one, if we can compute their dimensions in pixel in the image. And as we are only interested in moving objects, we will try firstly to find such objects, then extract their mask in the image, so their dimension, and finally compute the relative distance. The implemented system is described in the following diagram block (Fig. 1), which is composed of two main parts:

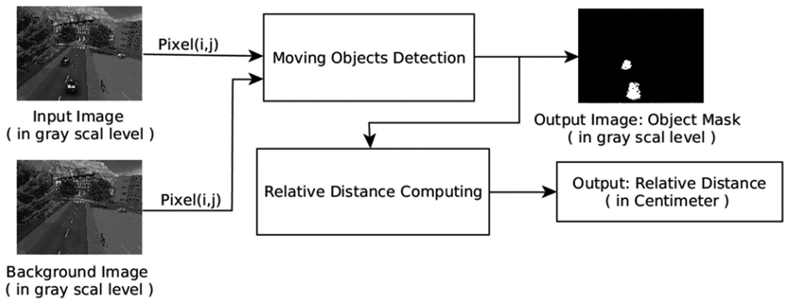

Fig. 1. Block diagram of the implemented VLS.

- Detection of vehicles in the scene.
- Calculation of the relative distance between the vehicle and the camera, then this distance will be used to compute the absolute distance.

The first step will be performed using the Background subtraction method, since our system will consist only of static cameras. The second step will be carried out by

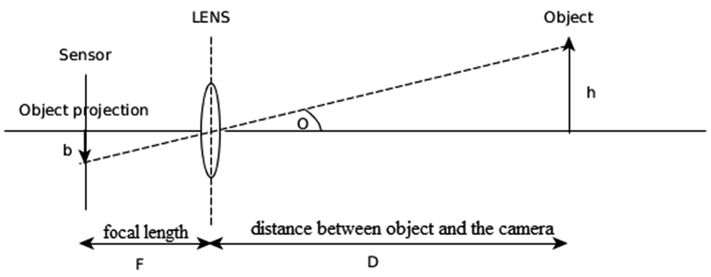

Fig. 2. Object projection on the camera sensor.

computing the size (in pixel) of the vehicle in motion and using some technical information about the vehicle and the camera we can deduce the distance between the vehicle and the camera.

3.1 Vehicle Detection

In the literature, several approaches are used for object detection, the most used are based on feature descriptor such the histogram of oriented gradient (HOG) [15, 16], in this approach the input image is converted to a feature vector, which simplifies the image by extracting useful information and throwing away extraneous information.

The computed feature vector is used by classification algorithms like Support Vector Machine and based on training data of positive (image with the object to be detected) and negative (image without the object to be detected) datasets. Objects are detected with a good accuracy. Unfortunately, object detectors can be painfully slow, especially when expensive features such as HOG need to be computed, it can really kill the performance. The background subtraction method presents another alternative, especially for indoor environments, where the lighting condition is approximately constant. And by taking into account their minimal implementation cost and its sufficient accuracy, as a result, we make the choice to go for background subtraction method.

Background Subtraction. The main idea of this method is based on the subtraction of the current image (in which the moving object is present) pixel-by-pixel from a reference background image as described in Eq. (1).

$$O(x,y,t) = |I(x,y,t) - B(x,y)| \tag{1}$$

Where:

O(x,y,t) is the subtracted image.
I(x, y, t) is the object image.
B(x, y) is the background image.

This technique is designed for static camera, where the background is approximately the same in all frames, and by applying Eq. (1) background is removed from object image. As a result the histogram of the object image O(x, y, t) will be composed of two main pixel classes, background pixel class (near to the 0 gray scale level) and the object pixel class.

Otsu threshold. The second step in the background subtraction method is segmentation in order to get the foreground mask. The object image will be compared to a global threshold as presented in Eq. (2).

$$O(x, y, t) \leq T \tag{2}$$

Where:

O(x,y,t) is the subtracted image.
T is the threshold value.

If the pixel O(x, y, t) verifies Eq. (2), then it is considered as a foreground pixel, else it is a background pixel. T is a one global threshold, for all pixels in the image. And it needs to be a function of time, in other case the segmentation can easily be impacted by the environmental conditions change. The Otsu method will be used in order to meet this objective. The Otsu algorithm [13] is a popular dynamic thresholding method for image segmentation. Based on the idea that the image histogram can be divided into two classes, so it looks for a threshold that minimizes the variance for both classes. This way, each class will be as compact as possible. Only pixel value is taken into account for the Otsu algorithm the spatial relationship between pixels has no effect on the algorithm result, different regions with similar pixel value are treated as one region.

In Otsu method we exhaustively search for the threshold that minimizes the intra-class variance (the variance within the class), defined as a weighted sum of variances of the two classes:

$$\sigma w^2(t) = w(t) \,.\, \sigma^2(t) + w'(t) \,.\, \sigma'^2(t) \tag{3}$$

Where;

σw^2(t) is the intra-class variance
w and w′ are the probabilities of the two classes separated by a threshold t
σ^2(t) and σ'^2(t) are variances of these two classes.

$$\sigma^2 = \sigma w^2(t) + \sigma b^2(t) \tag{4}$$

Where,

$$\sigma b^2(t) = w(t).w'(t).[\mu(t) - \mu'(t)]^2 \tag{5}$$

The Algorithm 1, which is based on the Eq. (5) step through all possible thresholds and keep the threshold value that maximizes the inter-class variance σb^2. The whole system computing the Background subtraction in addition to the Otsu method is described in the Fig. 3.

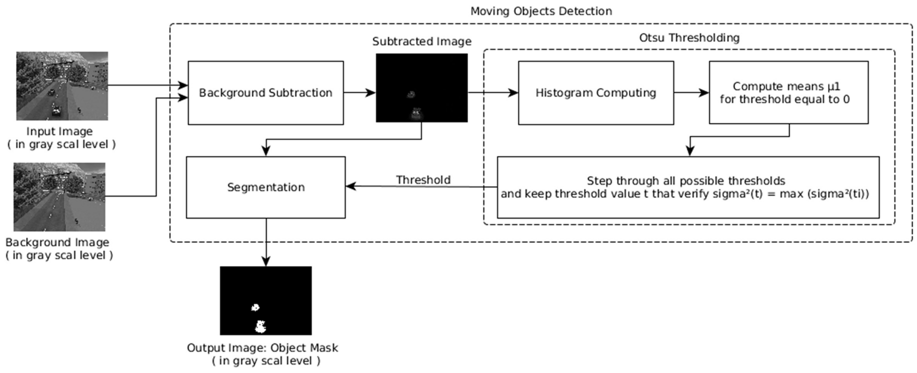
Moving Objects Detection
Subtracted Image
Otsu Thresholding
Input Image (in gray scal level)
Background Subtraction
Histogram Computing
Compute means µ1 for threshold equal to 0
Segmentation
Threshold
Step through all possible thresholds and keep threshold value t that verify sigma²(t) = max (sigma²(ti))
Background Image (in gray scal level)
Output Image: Object Mask (in gray scal level)

Fig. 3. Diagram block of the moving object detection module.

```
Algorithm 1: Otsu threshold
Input: grayscale_image
Output: Threshold
I = grayscale_image.weight * grayscale_image.height
σb² , σb_max, s, s' , w, w', µ,µ' ← 0
intensity ←255
histogram[255]
for j in 0 to intensity
      histogram[j] ← 0
for j in 0 to grayscale_image.weight
    for k in 0 to grayscale_image.height
         histogram[grayscale_image(j,k)] ← histo-
gram[grayscale_image(j,k)] +1
for j in 0 to intensity
       s ← s + histogram[j] * j
for j in 0 to intensity
       w ← w + histogram[j]
       if w = 0 then
            continue
       w'← I- w
       s' ← s' + histogram[j] * j
       µ ←  s'/w0
       µ' ← (s-s')/w'
       σb² ← w*w'*(µ-µ')²
       if  σb²>= σb_max then
            Threshold ← j
            σb_max ← σb²
Return: Threshold
```

3.2 Relative Distance Between the Vehicle and the Camera

Vehicle projection on camera sensor depend on three parameters vehicle dimensions, distance to the camera and the focal length as explained in Fig. 2. Equation 6 explains that the ratio of vehicle size on the sensor and the focal length is the same as the ratio between vehicle size in real life and distance to the vehicle. And in Eq. 7 vehicle size on the sensor is expressed as the vehicle size in pixels, divided by the image size in pixels and multiply by the physical size of the sensor.

The focal length and the sensor size are technical characteristic of the used camera. The vehicle size in pixels can be extracted from the segmented image (the output of the movement detection vehicle module). So we need just the vehicle size in real life to compute its relative distance, and as a result the absolute distance using the camera position information.

$$\tan(o) = \frac{b}{F} = \frac{h}{D} \rightarrow D = \frac{F.h}{b} \tag{6}$$

Where,

b	is the object size on the sensor.
H	is the object size in real life.
F	is the focal length (technical characteristic of the used camera).
D	is the distance between the object and the camera.

$$b = \frac{O.S}{I} \tag{7}$$

Where,

O	is the object size in pixels.
I	is the image size in pixels.
S	is the size of the sensor.

So, the whole equation can be rewritten as:

$$D = \frac{F.h.I}{O.S} \tag{8}$$

4 Implementation Using FPGA Board

Before going for the hardware implementation, a software implementation was already performed using a 650 MHz dual-core Cortex-A9 processor with an embedded Linux, but due to the limitation in the execution time 3 frame per second (fps), where a minimum of 30 fps is needed for real time video processing, we go for hardware acceleration, which presents an interesting choice for execution time improvement.

4.1 Board Description

With a vision to take the advantage of the parallelism feature provided by the field-programmable gate array (FPGA), we make the choice of implementing our system using the Artix-7 FPGA. The available fast RAM block space will be used to implement a dual - port RAM, which will be accelerate the treatment and give us more flexible architecture for our real-time image processing application (Fig. 5 and Table 1).

4.2 Simulation and Syntheses

The grayscale inputs images, foreground and background of the visual localization module are stored in two dual ports RAM, the process is started with computing the Otsu threshold for the subtracted image, then the segmentation is performed and the resulting object mask image is stored in a third dual port memory. The ISE Simulator (ISIM) was used to simulate the implemented system, the testbench (Fig. 4) reads the pixel's value of the foreground and background images (the ASCI PGM image format is used due to its simple manipulation) at each clock tick, and save the system result as an image in the same format. The inputs images Fig. 6(a) and (b) used in this example are part of the Background Models Challenge (BMC) [14], the resulting image from the background subtraction is presented in Fig. 6(c) and the object mask image is presented in Fig. 6(d). The simulation results of the presented example are presented in the Table 2.

Table 1. Implementation cost.

Logic utilization	Register	LUT	RAMB	BUFG
Used	837	448	41	1
%	1%	1%	29%	3%

Table 2. Simulation results.

Image resolution	640 * 480
Computed Otsu threshold	12
Computed object weight	49 pixels

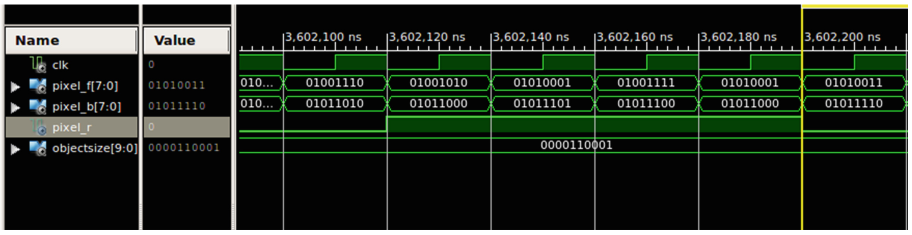

Fig. 4. System simulation using ISIM.

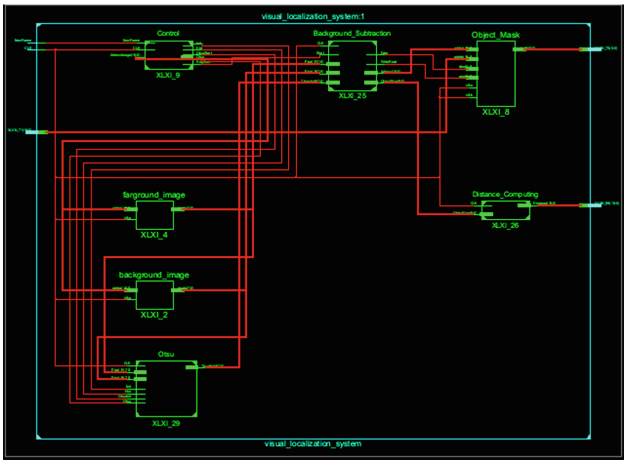

Fig. 5. RTL view.

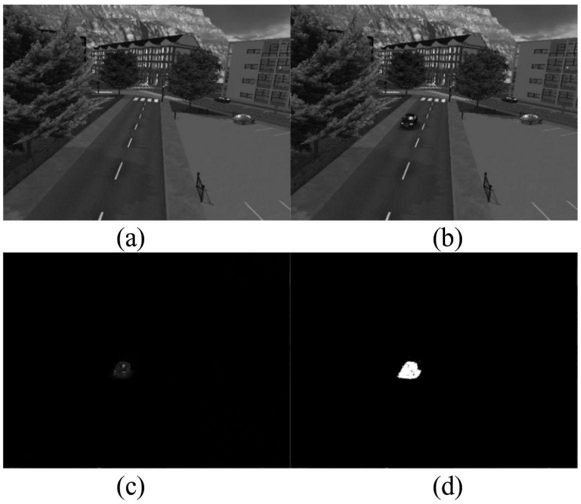

Fig. 6. (a) background image, (b) foreground image, (c) subtracted image, (d) object mask image.

5 Results and Evaluation

5.1 Temporal Analysis

The imaging device such as video cameras use the frames per second to explain the frequency (rate) at which an imaging device displays consecutive images called frames. The Phase Alternating Line (PAL) [12] and the National Television System Committee (NTSC) [11], which are the most used color encoding system for analogue television will be used as reference in our study. In the NTSC standard 30 frames are transmitted each second, each frame is made up of 525 individual scan lines. And in PAL 25 frames are transmitted each second, each frame is made up of 625 individual scan lines. Where a normal motion picture film is played back at 24 frames per second (fps). Thus we can consider that a real-time system is running at 30 fps.

This part, will evaluate the execution time for our system with different image resolution and frequency. The execution time characteristic of each block is presented in the Table 3. The execution time will be computed in function of the image resolution weight * height (W * H) using the operation number (clock ticks) unit Table 5. The detail of the Otsu threshold execution time is given in the Table 4.

Table 3. Block execution time in function of resolution and clock ticks.

Blocks	Background subtraction	Otsu	Segmentation	Relative distance
Operation number	W * H	W * H + 1792	W * H	W * H
Total	4 * W * H + 1792			

Table 4. Otsu block execution time in function of resolution and clock ticks.

Blocks	Histogram computing	Mean μ0 computing	Max σb^2 computing
Operation number	W * H + 256	256	1280
Total	W * H + 1792		

Table 5. Frames per second computing for different resolution in different clock frequency.

	640 * 480	1280 * 720
50 MHz	1 230 592 * 20 * 10^$^{-9}$ s = 0.024 s (>41 fps)	3 688 192 * 20 *10^$^{-9}$ = 0.074 s (>13 fps)
75 MHz	1 230 592 * 13.3 * 10^$^{-9}$ s = 0.016 s (>62 fps)	3 688 192 * 13.3 *10^$^{-9}$ = 0.049 s (>20 fps)
100 MHz	1 230592 * 10 * 10^$^{-9}$ s = 0.012 s (>83 fps)	3 688 192 * 10 *10^$^{-9}$ = 0.036 s (>27 fps)

After some improvement of the implementation taking advantage of parallelism execution, the Background Subtraction and Histogram computing, Segmentation and Relative distance are grouped in the same block. Which means an optimization of 2W * H (Table 6). The new simulation values are given in Table 7. As a result, we notice that for 640 * 480 resolutions in the worst case frequency 50 MHz we achieve 83 fps more than the double of the needed fps. For 1280 * 720 resolutions a real time system can be built in frequency superior than the 56 MHz. And bigger resolution such 1440 * 1080 can be implemented under 100 MHz frequency.

Table 6. Block execution time in function of resolution and clock ticks.

Blocks	Background subtraction + Histogram computing	Otsu	Segmentation + Relative distance
Operation number	W * H + 256	1536	W * H
Total	2 * W * H + 1792		

Table 7. Frames per second computing for different resolution in different clock frequency.

	640 * 480	1280 * 720
50 MHz	0.012 s (>83 fps)	0.036 s (>27 fps)
75 MHz	0.008 s (>125 fps)	0.024 s (>41 fps)
100 MHz	0.006 s (>166 fps)	0.018 s (>55 fps)

5.2 Accuracy

In order to measure the real accuracy of our proposed system, we put a car in different distance from a fix camera Fig. 7 inside a small parking, and by using a fix camera the relative distance between the car and the camera is computed in real time from different distances. The real and computed distances are given in Table 8. Compared to similar work as [17, 18] as shown in Table 9, the maximal accuracy error of 0.1 M for the proposed system present an interesting improvement.

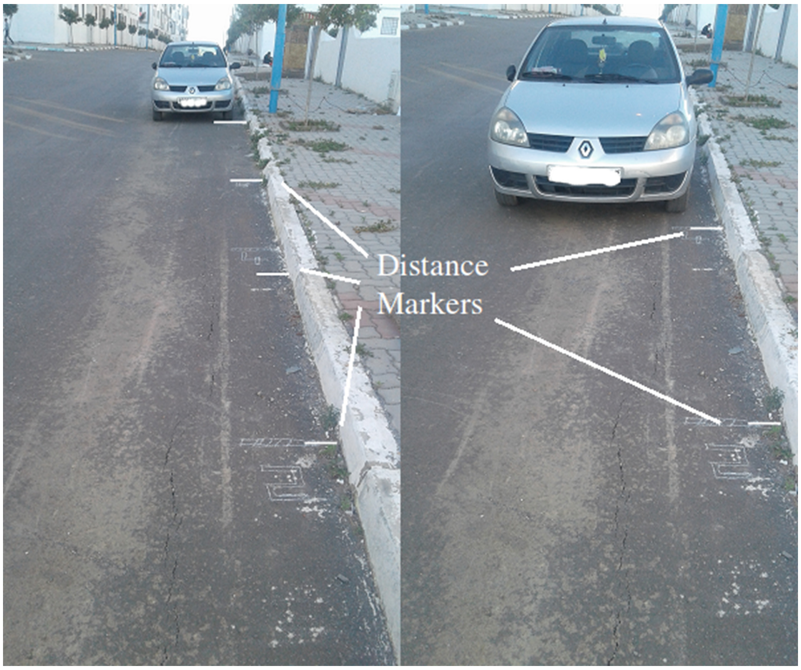

Fig. 7. Testing environment.

Table 8. Real and computed distance.

Real distance	Computed distance	Real distance	Computed distance
0.9 M	0.89 M	10 M	9.97 M
2 M	1.99 M	15 M	14.95 M
5 M	4.98 M	20 M	19.9 M

Table 9. Accuracy error compared to similar work.

Proposed system	System [18]	System [17]
0.1 M	0.34 M	0.8 M

6 Conclusion and Outlook

This paper presents an FPGA based Indoor real-time visual location system. The input image is segmented in order to extract the vehicle, this step is achieved using the background subtraction method, with a dynamic threshold computed using the Otsu method. The segmented image is then used to compute the relative distance to the camera. The proposed system is sufficiently satisfying the real time constraint 32 frames per second for the 1440 * 1080 resolution under 100 MHz frequency. In the next step, the presented algorithm will be improved and adapted in a vision of proposing a high accuracy visual navigation system for parking areas. This latter present one of the main parts in our approach of a fully automated parking solution, which will be presented in our future papers.

References

1. Ureña, J., Gualda, D., Hernández, Á., García, E., Villadangos, J.M., Pérez, M.C., García, J. C., García, J.J., Jiménez, A.: Ultrasonic local positioning system for mobile robot navigation: from low to high level processing. In: 2015 IEEE International Conference on Industrial Technology (ICIT), pp. 3440–3445, 17–19 March 2015. https://doi.org/10.1109/ICIT.2015.7125610
2. Ijaz, F., Yang, H.K., Ahmad, A.W., Lee, C.: Indoor positioning: a review of indoor ultrasonic positioning systems. In: 2013 15th International Conference on Advanced Communications Technology (ICACT) (2013)
3. Huang, C.-H., Lee, L.-H., Ho, C.C.: Real-time RFID indoor positioning system based on Kalman-Filter drift removal and Heron-Bilateration location estimation. IEEE Trans. Instrum. Meas. **64**(3), 728–739 (2015). https://doi.org/10.1109/TIM.2014.2347691
4. Wang, C.-S., Chen, C.-L.: RFID-based and Kinect-based indoor positioning system. In: 2014 4th International Conference on Wireless Communications, Vehicular Technology, Information Theory and Aerospace & Electronic Systems (VITAE) (2014). https://doi.org/10.1109/vitae.2014.6934458
5. Xu, H., Ding, Y., Wang, R., Shen, W., Li, P.: A novel Radio Frequency Identification three-dimensional indoor positioning system based on trilateral positioning algorithm. J. Algorithms Comput. Technol. **10**(3), 158–168 (2016)

6. Wang, Y., Yang, X., Zhao, Y., Liu, Y., Cuthbert, L.: Bluetooth positioning using RSSI and triangulation methods. In: 2013 IEEE 10th Consumer Communications and Networking Conference (CCNC) (2013). https://doi.org/10.1109/CCNC.2013.6488558
7. Yang, C., Shao, H.: WiFi-based indoor positioning. IEEE Commun. Mag. **53**(3), 150–157 (2015). https://doi.org/10.1109/mcom.2015.7060497
8. Hofmann-Wellenhof, B., Lichtenegger, H., Collins, J.: Global Positioning System Theory and Practice, vol. 1. Springer, Vienna (2001)
9. La Delfa, G.C., Catania, V., Monteleone, S., De Paz, J.F., Bajo, J.: Computer vision based indoor navigation: a visual markers evaluation. In: Mohamed, A., Novais, P., Pereira, A., Villarrubia González, G., Fernández-Caballero, A. (eds.) Ambient Intelligence - Software and Applications. AISC, vol. 376, pp. 165–173. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19695-4_17
10. Wolcott, R.W., Eustice, R.M.: Visual localization within LIDAR maps for automated urban driving. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (2014). https://doi.org/10.1109/iros.2014.6942558
11. https://en.wikipedia.org/wiki/NTSC
12. https://en.wikipedia.org/wiki/PAL
13. Vala, M.H.J., Baxi, A.: A review on Otsu image segmentation algorithm. Int. J. Adv. Res. Comput. Eng. Technol. (IJARCET) **2**(2), 387 (2013)
14. Background models challenge. http://bmc.iut-auvergne.com/
15. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2005). https://doi.org/10.1109/cvpr.2005.177
16. Malisiewicz, T., Gupta, A., Efros, A.A.: Ensemble of exemplar-SVMs for object detection and beyond. In: 2011 International Conference on Computer Vision (2011)
17. Einsiedler, J., Sawade, O., Schäufele, B., Witzke, M., Radusch, I.: Indoor micro navigation utilizing local infrastructure-based positioning. In: 2012 IEEE Intelligent Vehicles Symposium (2012). https://doi.org/10.1109/ivs.2012.6232262
18. Ibisch, A., Houben, S., Schlipsing, M., Kesten, R., Reimche, P., Schuller, F., Altinger, H.: Towards highly automated driving in a parking garage: general object localization and tracking using an environment-embedded camera system. In: 2014 IEEE Intelligent Vehicles Symposium (IV), 8–11 June 2014, Dearborn, Michigan, USA (2014). https://doi.org/10.1109/ivs.2014.6856567