Semi-automatic parking slot marking recognition for intelligent parking assist systems

Ho Gi Jung

Department of Automotive Engineering, Hanyang University, 222 Wangsimni-ro, Seongdong-gu, Seoul 133-791, Republic of Korea
E-mail: hogijung@hanyang.ac.kr

Published in *The Journal of Engineering*; Received on 23rd October 2013; Accepted on 7th November 2013

Abstract: This paper proposes a semi-automatic parking slot marking-based target position designation method for parking assist systems in cases where the parking slot markings are of a rectangular type, and its efficient implementation for real-time operation. After the driver observes a rearview image captured by a rearward camera installed at the rear of the vehicle through a touchscreen-based human machine interface, a target parking position is designated by touching the inside of a parking slot. To ensure the proposed method operates in real-time in an embedded environment, access of the bird's-eye view image is made efficient: image-wise batch transformation is replaced with pixel-wise instantaneous transformation. The proposed method showed a 95.5% recognition rate in 378 test cases with 63 test images. Additionally, experiments confirmed that the pixel-wise instantaneous transformation reduced execution time by 92%.

1 Introduction

In general, automatic parking systems consist of three components: target position designation, path planning and path tracking by active steering [1–3]. The methods of target position designation can be divided into four categories [1–3]: user interface-based [4, 5], parking slot marking-based [2, 6–9], free space-based [1, 3, 10–14] and infrastructure-based [15]. There are three reasons why the parking slot marking-based type is expected to be the most useful in a practical application: (i) the parking assist system is used almost always in urban parking lots, which generally have parking slot markings; (ii) the parking slot marking-based type is simpler and more robust than the free space-based variety using a vision sensor [11]; and (iii) the parking slot marking-based type can be fused with other approaches to enhance the reliability and accuracy of the target position.

Parking slot marking-based target position designations can be divided into two categories: automatic and semi-automatic [2]. Automatic recognition systems use various kinds of constraints to enhance the efficiency of parking slot marking recognition. Xu *et al*. [6] extracted pixels belonging to parking slot markings by applying restricted Coulomb energy neural-network-based colour segmentation assuming that parking slot markings are drawn with a specific colour. In [8], assuming that parking slot markings consist of line-segments with a fixed width and two borders of the line-segment form a discriminative pattern in the Hough space, specially designed one-dimensional filtering in the Hough space was devised. Semi-automatic approaches utilise the user's input assuming that the user is willing to provide useful information to reduce the search range and improve the recognition performance. I proposed a semi-automatic parking slot marking-based target position designation method which requires the driver to initiate parking slot marking recognition by placing a finger on each side of the entrance of the target parking slot [2]. Additionally, I proposed another semi-automatic parking slot marking-based method for the rectangular type, with which the driver designates a target parking position by touching the inside of a parking slot [9]. Although the method shown in [2] can deal with various types of parking slots, the method in [9] seems to be preferable in the case of the rectangular type considering the fact that touching the inside of the target parking slot is more intuitive and convenient in a motion-constrained cockpit unit than pointing out two end-points of the entrance of the target parking slot. Additionally, the method in [2] is too heavy to be implemented in an embedded environment without support of any specific hardware as it consists of local bird's-eye view image construction, neural-network-based junction classification, histogram analysis-based binarisation, skeleton extraction, distance transform image construction and genetic algorithm-based template localisation.

This paper completes the method shown in [9] and improves the execution speed by adopting pixel-wise instantaneous transformation. Fig. 1 shows the user interface of the proposed method: the driver observes a rearview image captured by a rearward camera installed at the rear of the vehicle through a touchscreen-based human machine interface (HMI) and touches the inside of the target parking slot. Using the point inputted by the driver, a marking line-segment separating the parking area from the roadway is detected, and along the marking line-segment two marking line-segments dividing neighbouring parking slots are detected. To make the proposed system operate in an embedded environment using a general purpose processor, pixel-wise instantaneous transformation was introduced, which means that the pixel intensity of the bird's-eye view image is calculated only if the value is asked by the remaining

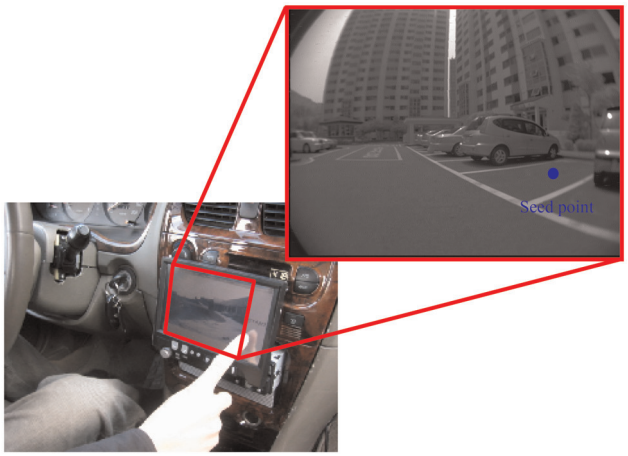


Fig. 1 *After the driver observes a rearview image captured by a rearward camera installed at the rear of the vehicle through a touchscreen-based HMI, a target parking position is designated by touching the inside of the parking slot*

doi: 10.1049/joe.2013.0120

algorithm. This is in contrast to the previous implementation found in [9], image-wise batch transformation, which constructs a bird's-eye view image of the input image before the start of the major algorithm. Since the required pixel number of a bird's-eye view image is very small compared with the whole image size, such a simple change of memory access method can drastically reduce the execution time and memory consumption.

2 Semi-automatic parking slot marking recognition

The proposed method consists of three steps: bird's-eye view construction, guideline recognition and parking slot separating line-segment recognition. As the only information needed for automatic parking control is two end-points of the entrance of the target parking slot [4, 15], the proposed method recognises only the entrance of the target parking slot.

A bird's-eye view image is constructed using fisheye lens characteristics and constant camera pose with respect to the ground surface. In general, the rearward camera of a parking assist system uses a fisheye lens as it should be able to capture a horizontally wide area of the rearward area. Although a wide field of view is acquired thanks to the fisheye lens, inherent image distortion hinders image understanding. Therefore an undistorted image is constructed by compensating the fisheye lens distortion as shown in Fig. 2. To obtain undistorted images, distances from a pixel to the distortion centre before and after the radial distortion (r_u and r_d) are modelled with a fifth-degree polynomial [16] as follows

$$r_d = a_1 r_u^5 + a_2 r_u^4 + a_3 r_u^3 + a_4 r_u^2 + a_5 r_u + a_6 \quad (1)$$

where a_1–a_6 are the coefficients of the polynomial. After the radial distortion is removed from the original fisheye images, the undistorted images should be transformed into bird's-eye view images. For this task, projective transformation (homography) between the image of the ground plane and the real-world ground plane is utilised [17]. Hence, a point in the undistorted image coordinates ($\boldsymbol{x}_u$) can be transformed into a point in the bird's-eye view image coordinates ($\boldsymbol{x}_b$) by using a 3×3 matrix as follows

$$\boldsymbol{x}_b = \begin{bmatrix} h\cos\theta_y & h\sin\theta_y\sin\theta_p & -fh\cos\theta_p\sin\theta_y \\ h\sin\theta_y & -h\cos\theta_y\sin\theta_p & fh\cos\theta_y\cos\theta_p \\ 0 & \cos\theta_p & f\sin\theta_p \end{bmatrix} \boldsymbol{x}_u \quad (2)$$

where h, θ_y and θ_p are height, yaw angle and pitch angle of the rear-view camera, respectively. f is the focal length of the camera in terms of pixel dimensions. $\boldsymbol{x}_b$ and $\boldsymbol{x}_u$ are in homogeneous coordinates. Therefore one set of coordinates of the input image corresponds to the coordinates of the undistorted image, which also corresponds to the coordinates of the bird's-eye view image. It is notable that the parking slot markings seem correct although vehicles, which are higher than the ground surface, seem to be stretched outward from the camera.

If the driver can find an empty parking slot in the input image and touch its interior, it means there is no obstacle on a line between the seed-point and the camera. Furthermore, it means that the guideline separating the parking area from the roadway and its junctions with line-segments separating neighbouring parking slots can be observed. Fig. 3*a* depicts a seed-point inputted by the driver on the constructed bird's-eye view image and draws a line connecting the seed-point and the camera position.

Owing to the structural characteristics of the rectangular type, the line between the seed-point and the camera meets once or twice with the parking slot markings. By detecting the peaks of the intensity gradient along the line between the seed-point and the camera and verifying whether the edge following from the peak is possibly longer than a certain length, cross-points with parking slot markings are detected. $dI(\boldsymbol{p}, \boldsymbol{u})$ denotes the directional intensity gradient of a point $\boldsymbol{p}(x, y)$ in the direction of the unit vector $\boldsymbol{u}$ as follows

$$dI(\boldsymbol{p},\ \boldsymbol{u}) = \frac{1}{(W/2)}\sum_{i=1}^{(W/2)} I(\boldsymbol{p} - i\,\boldsymbol{u}) - \frac{1}{(W/2)}\sum_{i=1}^{(W/2)} I(\boldsymbol{p} + i\,\boldsymbol{u}) \quad (3)$$

where $I()$ denotes the intensity value of a certain pixel coordinate. Since the camera maintains a certain height and angle with respect to the ground surface, marking line-segments painted on the ground surface will appear with a fixed width W. The directional intensity gradient using the average intensity of $(W/2)$ interval makes the edge detection robust to noise. Fig. 3*b* depicts two detected cross-points on the line of Fig. 3*a*.

Fig. 3*c* is the result of edge following from the detected cross-points in the direction orthogonal to the intensity gradient of the point. The edge following iteration continues when the edge strength is beyond a threshold. If the detected cross-point is caused by noise or stains, edge following starting from the point will not progress longer than a threshold. Therefore falsely detected cross-points can be deleted by examining the successful edge following length. The edge following is implemented by iterating prediction and verification. The prediction of edge position of $(n+1)$th iteration, $\hat{\boldsymbol{e}}[n+1]$, is calculated by extending the edge by step length s in the edge direction $\boldsymbol{a}[n]$

$$\hat{\boldsymbol{e}}[n+1] = \boldsymbol{e}[0] + (n+1)\,s\,\boldsymbol{a}[n] \quad (4)$$

where $\boldsymbol{e}[0]$ is the edge position of 0th iteration, that is, the detected cross-point. Then, the verification refines the prediction by finding the maximum directional intensity gradient in the direction normal to the edge $\boldsymbol{n}[n]$. The edge position of $(n+1)$th iteration, $\boldsymbol{e}[n+1]$, is updated using the result

$$t_{max} = \arg\max_t dI\left(\hat{\boldsymbol{e}}[n+1] + t\,\boldsymbol{n}[n], \boldsymbol{n}[n]\right), \quad -\frac{W}{2} \le t \le \frac{W}{2} \quad (5)$$

$$\boldsymbol{e}[n+1] = \hat{\boldsymbol{e}}[n+1] + t_{max}\,\boldsymbol{n}[n] \quad (6)$$

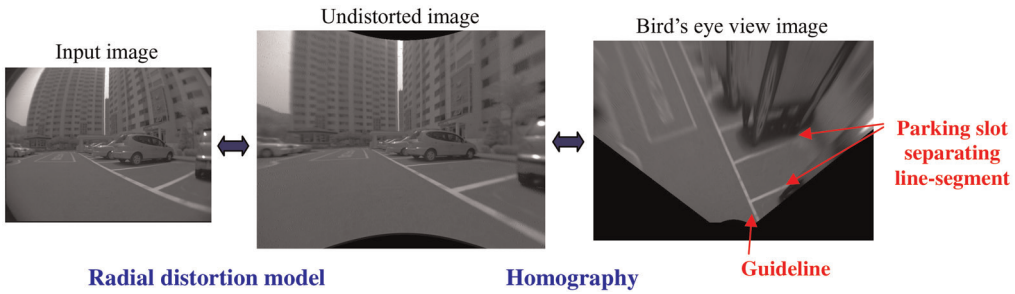


Fig. 2 *Three coordinate systems that satisfy one-to-one correspondence*

J Eng 2013
doi: 10.1049/joe.2013.0120

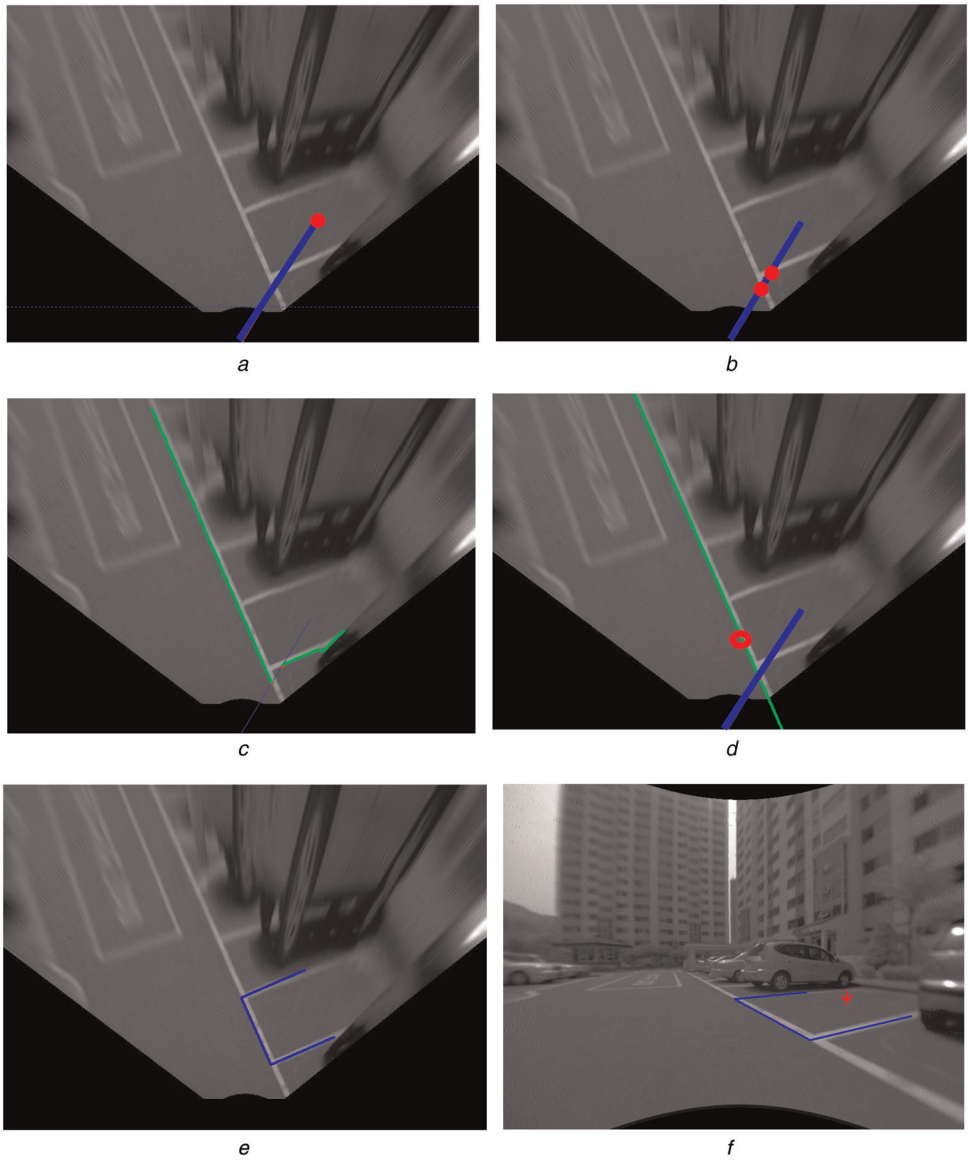


Fig. 3 *Procedure of parking slot marking recognition*
a Seed-point on the bird's-eye view
b Two cross-points detected by peak detection in the intensity gradient
c Validation by edge following
d Recognised guideline
e Recognised target parking slot
f Target parking slot projected onto the undistorted image

The bidirectional search range is limited by half of line-segment width W. Once the edge position is determined, the edge direction and normal direction of $(n+1)$th iteration, $\boldsymbol{a}[n+1]$ and $\boldsymbol{n}[n+1]$, are updated using $\boldsymbol{e}[0]$ and $\boldsymbol{e}[n+1]$.

Among cross-points validated by the edge following test, a guideline is set to the one nearest to the camera. Fig. 3*d* shows the guideline recognised using the nearest cross-point and its edge following result. A point depicted by a hollow circle is the projection of the seed-point onto the recognised guideline. As the point always exists on the entrance of the target parking slot, the search range for the parking slot separating line-segment can be established around the point. By examining whether the intensity of an area slightly apart from the guideline is similar to the intensity of the guideline, parking slot separating line-segments are recognised. As the intensity comparison is conducted along the guideline, local intensity variations such as a shadow can be ignored. Once the guideline and two parking slot separating line-segments are recognised, the target parking slot can be recognised as shown in Fig. 3*e*. Fig. 3*f* is the recognised target parking slot projected onto the undistorted image.

3 Pixel-wise instantaneous transformation-based improvement

Although the proposed method is relatively simple, it required some improvements to be able to operate in real-time on an embedded system. After the basic idea of the proposed method was verified by an off-line test [9], it was ported into an electronic control unit (ECU) with a 32 bit embedded processor with an operating frequency of 400 MHz [18]. Even though input image coordinates corresponding to all bird's-eye view image coordinates were calculated in advance and stored in a table to shorten the execution time, the average execution time is above 5 s. Consequently, the proposed method could not be used in a practical situation.

By measuring and analysing the execution times of every programme component, interpolation of the bird's-eye view image construction is recognised as the bottle-neck. As image analysis and understanding is conducted using the bird's-eye view image, the image quality should be maintained sufficiently high. In general, the undistorted image coordinates corresponding to the bird's-eye view image coordinates and the input image coordinates corresponding to the undistorted image coordinates are not integer. If the intensity of the pixel nearest to the input image coordinates is simply copied, block pattern distortion cannot be avoided because of quantisation error. Therefore a pixel of a bird's-eye view image is calculated by linear interpolation using four pixels neighbouring the corresponding undistorted image coordinates and a pixel of the undistorted image is calculated by linear interpolation using four pixels neighbouring the corresponding input image coordinates. Although such a calculation for one pixel is simple and could be done quickly, the total burden becomes too heavy and the execution becomes too slow if it is applied to every pixel of a bird's-eye view image. To address this bottle-neck problem, hardware-based implementation using a field programmable gate array was developed [19, 20]. However, such a hardware-based implementation makes the ECU complicated and significantly increases the cost.

A change of the memory access method resolves the bottle-neck: image-wise batch transformation is replaced with pixel-wise instantaneous transformation. The pixel intensity of the bird's-eye view image is calculated only if the value is asked by the remaining algorithm. This is in contrast to the previous implementation of [9], which constructs a bird's-eye view image of the input image before the start of the major algorithm. This solution was initiated by the fact that although the whole bird's-eye view image might be required for the developer's understanding, only a small portion of the image was actually used. Fig. 4 depicts the pixel-wise instantaneous transformation. When the system needs the intensity of a pixel of the bird's-eye view image, the bird's-eye view image coordinates are converted to undistorted image coordinates, and then to the input image coordinates. As the input image coordinates are not integers, it is depicted to be located between the dotted line grids that represent the integer coordinates. The intensity of the point is calculated by linear interpolation using four pixels neighbouring the point. Such a memory access method can be easily implemented by simple changes to the previous implementation of [9]. In other words, memory access of the previous implementation is replaced with the call of the function accessing the input image. If the pixel-wise instantaneous transformation is applied, there is no need to construct the undistorted image and the bird's-eye view image in advance. Therefore there is no need to use large memory blocks for the undistorted image and the bird's-eye view image. Furthermore, the execution time related to the bird's-eye view image construction can be drastically reduced. This point will be proved in the experimental results shown in Section 4.2.

4 Experimental results

4.1 Performance evaluation

The proposed method was validated with 378 cases: 63 test images captured in various places and illumination conditions; for each test image, 6 trials or seed-points, which are randomly and evenly distributed inside the target parking slot; 21 of 63 test images are when the target parking slot is between parked vehicles, and the others are when only one side is occupied or there are no adjacent vehicles. When two end-points of the entrance of the target parking slot are localised within five pixels (physically, 10 cm) from manually designated ground truth, the trial is regarded as a success. Consequently, the proposed method succeeded in 361 cases and the recognition rate measured 95.5%.

Figs. 5–7 show that the proposed method can operate robustly in various places and conditions. A red arrow depicts the seed-point inputted by the driver and three blue line-segments depicted the entrance of the recognised target parking slot. Fig. 5 shows that the proposed method is irrespective of the subjective pose with respect to the target parking slot. In Fig. 5*a*, the target parking slot is relatively near, and is to the right side of the subjective vehicle. Contrarily, in Fig. 5*b*, the target parking slot is relatively far, and to the left side of the subjective vehicle.

Fig. 6 shows that the proposed method is irrespective of the existence and type of neighbouring objects. Fig. 6*a* shows a case when the target parking slot is between parked vehicles. It shows that the proposed method successfully recognises the target parking slot if the guideline and its junctions with the parking slot separating line-segments are observable.

As the proposed method mainly uses the borders of parking slot markings, it is relatively robust to the illumination conditions significantly influencing object surface brightness and texture. Furthermore, as the proposed method uses average intensity values before and after the investigated pixel, border detection has few influences from noises. Additionally, by checking whether the edge following from a candidate border could be

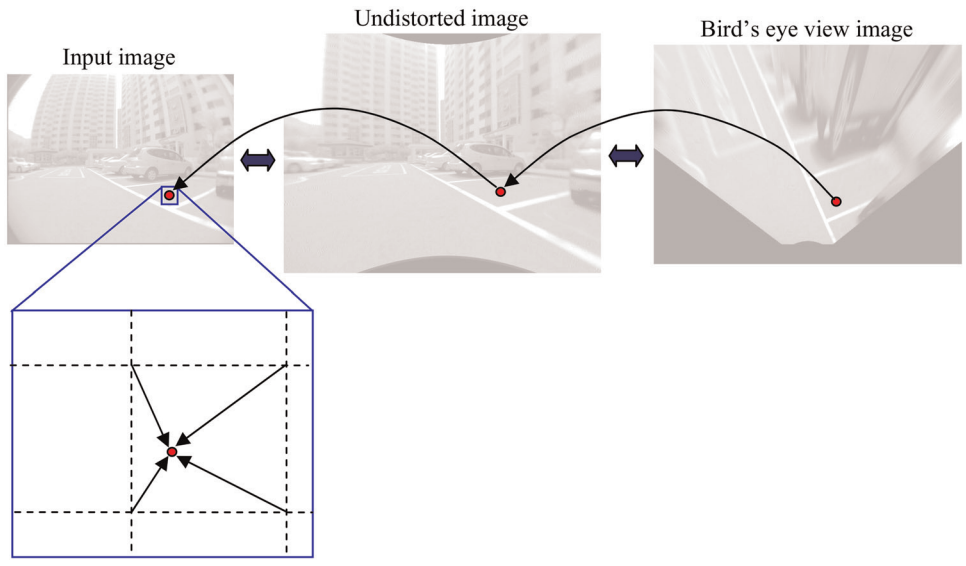
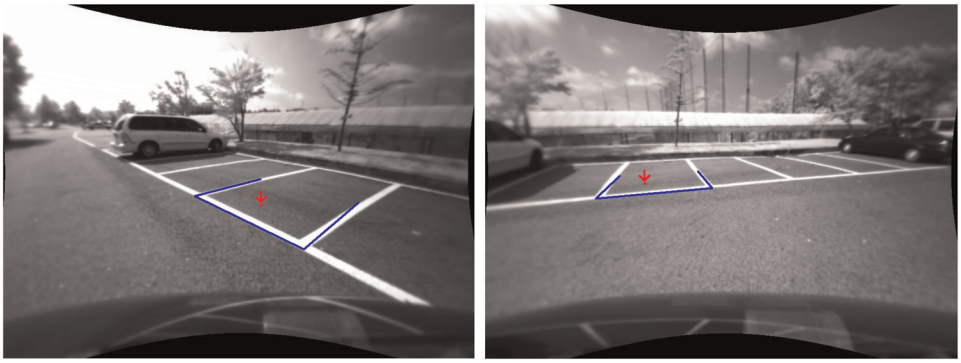
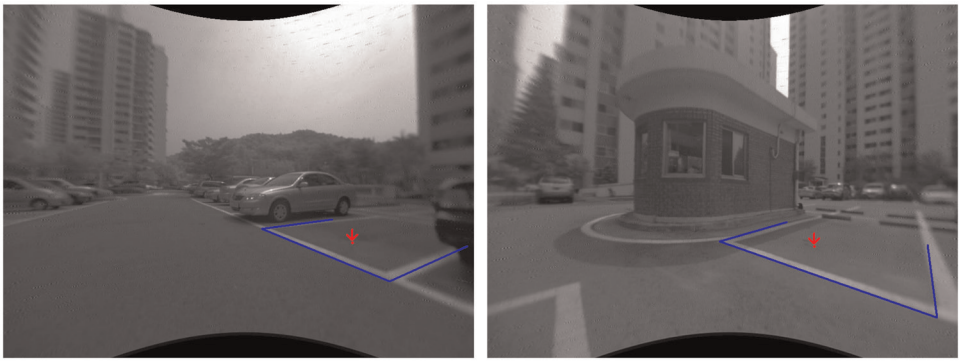


Fig. 4 *Pixel-wise instantaneous transformation*



a b

Fig. 5 *Pose of the target parking slot is irrespective*
a When the target parking slot is near and to the right side of the subjective vehicle
b When the target parking slot is far away and to the left side of the subjective vehicle



a b

Fig. 6 *Existence and type of neighbouring objects is irrespective*
a When the target parking slot is between parked vehicles
b When the target parking slot is next to a building

longer than a threshold in a restricted direction, the proposed method reduces any error caused by stains. Fig. 7 shows that the proposed method successfully recognises the target parking slot even when there is a shadow or stain inside the slot. Particularly, in Fig. 7*c*, the subjective vehicle stands with the sun at its back; the intensity contrast of the captured image is very large and a subjective dark shadow is cast in front of the camera. Contrarily, in Fig. 7*d*, the camera is against the sun; the intensity contrast of the captured image is very large and the ground surface glistens.

The cause of the failed cases can be classified into two categories: (i) when the markings of the target parking slot is so worn as to be indistinct and (ii) when there are other kinds of markings such as emergency parking. Fig. 8*a* shows an example of the first cause, worn markings. In Fig. 8*b*, the bird's-eye view of Fig. 8*a*, the marking line-segment is indistinct where it meets the line between the seed-point and the camera. Fig. 8*c* shows an example of the second cause, other kinds of markings. In Fig. 8*d*, or the bird's-eye view of this case, markings drawn in front of the parking area to denote special locations for emergency vehicles are recognised as the guideline.

However, it is notable that the proposed method does not fail only because other kinds of markings exist. In Fig. 9*a*, although additional markings are observed, they do not disturb the recognition because they do not meet the line between the seed-point and the camera. Fig. 9*b* is the bird's-eye view of this case. In Fig. 9*c*, although an arrow marking exists between the seed-point and the camera, it can be ignored if it is farther than a general vehicle's longitudinal length from the seed-point (marking border searching is conducted by the length of a general vehicle's longitudinal length assuming that the seed-point is inside a parking slot).

4.2 *Execution time evaluation*

It was confirmed that the change of memory access method from image-based batch transformation to pixel-wise instantaneous transformation reduced the execution time by 92%. The proposed method was tested 100 times with 28 test images and the execution times were recorded. To exactly measure the execution times of consisting modules and the memory access number, experiments were conducted on a personal computer (PC) environment. The central processing unit of the PC was Intel Core2Quad operating at 2.66 GHz and random access memory size was 3.5 GB. The programme was developed with Visual C++ 6.0 and execution times were measured with multimedia timer functions [21], whose resolution was 1 ms.

At first, while providing the same seed-point, it was confirmed that two versions outputted the same result. Then, execution times and memory access numbers were compared. The average execution time of the image-wise batch transformation version measured 161.95 ms and the bird's-eye view image construction spent 98.39%. In other words, it was found that the bird's-eye view image construction spent almost all of the execution time in the case of the image-wise batch transformation version. The average execution time of the pixel-wise instantaneous transformation version measured 12.69 ms, and is just 7.84% of the previous version. Furthermore, the memory access number was 6.65% of the previous

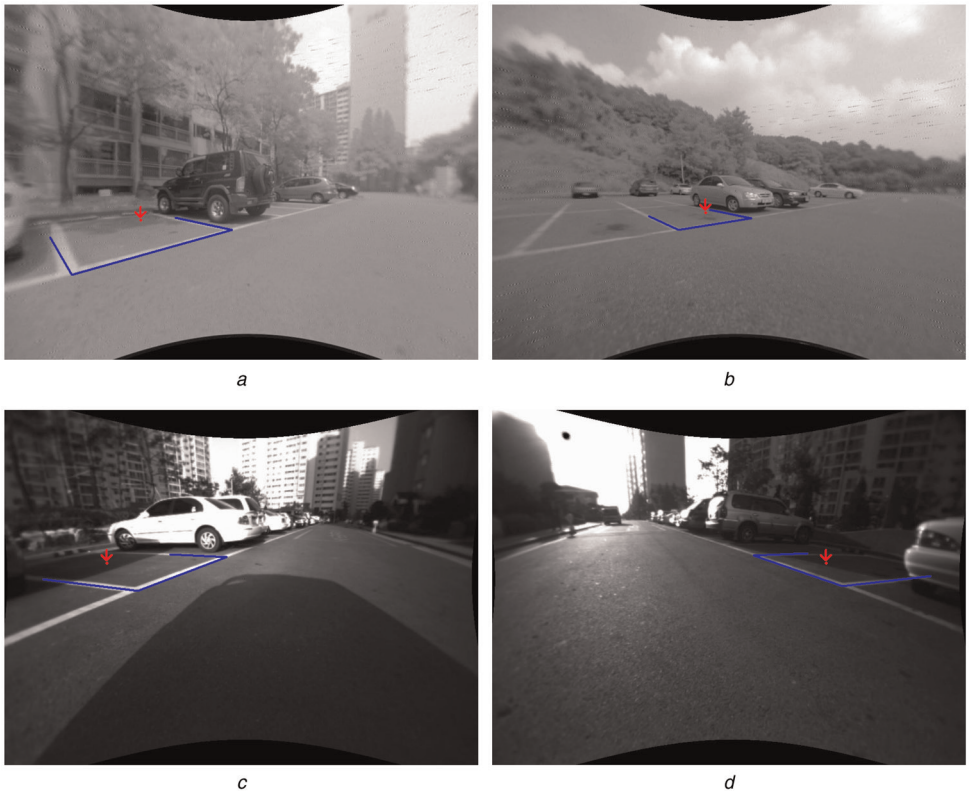


Fig. 7 *Proposed method is robust to shadow and illumination conditions*
a When there is a shadow inside the slot
b When there is a strain inside the slot
c When the intensity contrast is very large and subjective dark shadow is cast in front of the camera
d When the camera is against the sun

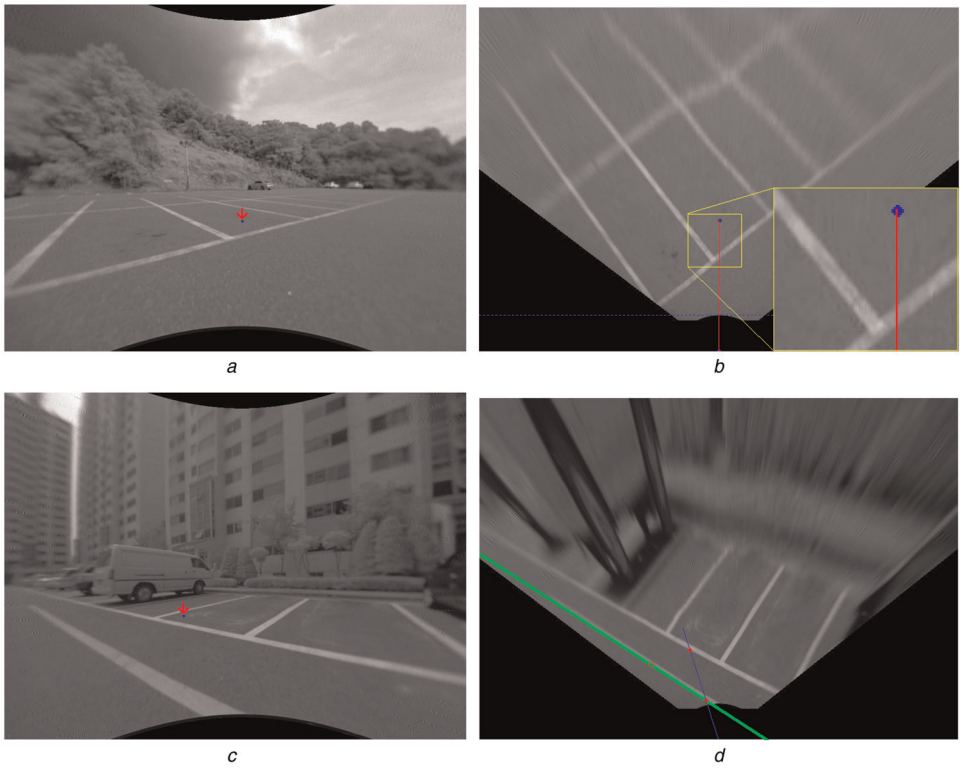


Fig. 8 *Typical examples of a failed case: worn markings and other kinds of markings*
a When the parking slot markings are worn
b The bird's-eye view of *a*
c When there are other kinds of markings
d The bird's-eye view of *c*

doi: 10.1049/joe.2013.0120

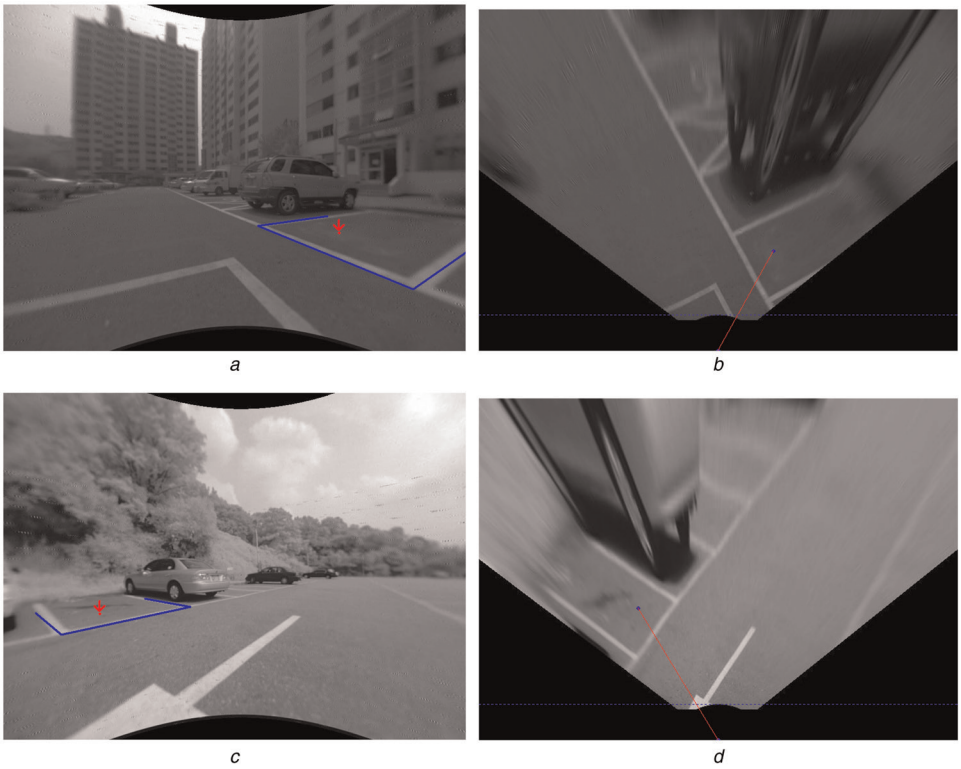


Fig. 9 *Success even with other kinds of markings*
a When the additional markings do not meet the line between the seed-point and the camera
b The bird's-eye view of *a*
c When the additional markings are further than a general vehicle's longitudinal length from the seed-point
d The bird'-eye view of *c*

version. Such results mean that the memory access number required for the proposed method is very small compared with that of the entire bird's-eye view image construction. Although the entire bird's-eye view image is required for the initial development when developers use it for a new invention, it might include a large number of pixels that are of no use for a specific algorithm. Therefore by changing the memory access method, that is, removing construction of pixels not accessed by a recognition algorithm, the execution speed could be significantly improved. Although such an improvement strongly depends on the characteristics of the algorithm, it might be a promising candidate for efficiency improvement and has proven to be valid for the algorithm used in this paper. Consequently, as the explained change of memory access method can reduce the execution time from 5 to 0.4 s on the 32 bit embedded processor, it is believed to solve the bottle-neck problem of the previous implementation [9].

5 Conclusion

This paper proposes a semi-automatic parking slot marking-based target position designation method for a parking assist system and its efficient implementation. The contributions of this paper are as follows: (i) it proposes a simple and easy-to-use semi-automatic parking slot marking recognition method which can be used with the most common parking slot marking type, or the rectangular type. Furthermore, it confirms the performance of the proposed method by quantitative evaluation; the recognition rate measures 95%. (ii) It shows that the execution time of the proposed method can be reduced by 92% by changing the memory access method from image-wise batch transformation to pixel-wise instantaneous transformation. The memory access method is related with when and how many pixels of the bird's-eye view image will be calculated from the input image. Particularly, the second contribution gives an important implication for the development of an embedded computer vision system: when trying to implement an algorithm requiring image transformations on an embedded system without hardware additions or modifications, pixel-wise transformation only when requested might be a good solution for real-time operation.

The major limitation of the proposed method is that it can be used only for rectangular type parking slot markings. Although its feasibility could be accepted considering the fact that the rectangular type is the most common parking slot marking type in urban areas, its extension to various types of parking slot marking is required.

6 References

[1] Jung H.G., Cho Y.H., Yoon P.J., Kim J.: 'Scanning laser radar-based target position designation for parking aid system', *IEEE Trans. Intell. Transp. Syst.*, 2008, **9**, (3), pp. 406–424
[2] Jung H.G., Lee Y.H., Kim J.: 'Uniform user interface for semi-automatic parking slot markings recognition', *IEEE Trans. Veh. Technol.*, 2010, **59**, (2), pp. 616–626
[3] Jung H.G., Kim D.S., Kim J.: 'Light stripe projection-based target position designation for intelligent assist system', *IEEE Trans. Intell. Transp. Syst.*, 2010, **11**, (4), pp. 942–953
[4] Hiramatsu S., Hibi A., Tanaka Y., Kakinami T., Iwata Y., Nakamura M.: 'Rearview camera based parking assist system with voice guidance'. SAE Paper 2002-01-0759, Warrendale, PA, SAE
[5] Jung H.G., Choi C.G., Yoon P.J., Kim J.: 'Novel user interface for semi-automatic parking assistance system'. Proc. 31st FISITA World Automotive Congress, 22–27 October 2006
[6] Xu J., Chen G., Xie M.: 'Vision-guided automatic parking for smart car'. Proc. IEEE Intelligent Vehicle Symp., 3–5 October 2000, pp. 725–730

[7] Tanaka Y., Saiki M., Katoh M., Endo T.: 'Development of image recognition for a parking assist system'. Proc. 13th World Congress Intelligent Transport Systems Services, 8–12 October 2006
[8] Jung H.G., Kim D.S., Yoon P.J., Kim J.: 'Parking slot markings recognition for automatic parking assist system'. Proc. IEEE Intelligent Vehicles Symp., 13–15 June 2006, pp. 106–113
[9] Jung H.G., Kim D.S., Yoon P.J., Kim J.: 'Structure analysis based parking slot marking recognition for semi-automatic parking system'. Joint IAPR Int. Workshops on Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition, Hong Kong, China, August 2006, pp. 384–393
[10] Jung H.G., Cho Y.H., Kim J.: 'ISRSS: integrated side/rear safety system', *Int. J. Automot. Technol.*, 2010, **11**, (4), pp. 541–553
[11] Suhr J.K., Jung H.G., Bae K., Kim J.: 'Automatic free parking space detection by using motion stereo-based 3D reconstruction', *Mach. Vis. Appl.*, 2010, **21**, (2), pp. 163–176
[12] Heilenkötter C., Höver N., Magyar P., Ottenhues T., Seubert T., Wassmuth J.: 'The consistent use of engineering methods and tools', *AutoTechnology*, 2007, **6**, pp. 52–55
[13] Pohl J., Sethsson M., Degerman P, Larsson J.: 'A semi-automated parallel parking system for passenger cars', *Proc. Inst. Mech. Eng. D, J. Automob. Eng.*, 2006, **220**, (1), pp. 53–65
[14] Satonaka H., Okuda M., Hayasaka S., Endo T., Tanaka Y., Yoshida T.: 'Development of parking space detection using an ultrasonic sensor'. Proc. 13th World Congress Intelligent Transport Systems Services, 8–12 October 2006
[15] Wada M., Yoon K.S., Hashimoto H.: 'Development of advanced parking assistance system', *IEEE Trans. Ind. Electron.*, 2003, **50**, (1), pp. 4–17
[16] Jung H.G., Lee Y.H., Yoon P.J., Kim J.: 'Radial distortion refinement by inverse mapping-based extrapolation'. Proc. 18th Int. Conf. Pattern Recognition, Hong Kong, 20–24 August 2006, pp. 675–678
[17] Ehlgen T., Pajdla T., Ammon D.: 'Eliminating blind spots for assisted driving', *IEEE Trans. Intell. Transp. Syst.*, 2008, **9**, (4), pp. 657–665
[18] Freescale Semiconductor: 'MPC5200: 32-bit power architecture microcontrollers', Available at http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MPC5200&nodeId=0162468rH3bTdG06C10898, accessed on 16 July 2010
[19] Oh S., Kim G.: 'An architecture for on-the-fly correction of radial distortion using FPGA'. Real-Time Image Processing 2008, San Jose, CA, USA, Proc. SPIE, 26 January 2008, vol. **6811**, 68110X, pp. 1–9
[20] Yang S.M., Kim G.: 'Bird's eye view image enhancement using modified EWA'. Proc. KIISE 2007 Fall Conf., Korean, October 2007, vol. **34–2**, pp. 443–446
[21] Microsoft: 'timeGetTime function', Available on http://www.msdn.microsoft.com/en-us/library/dd757629(v=VS.85).aspx, accessed on 27 July 2010