Real-time and low-cost embedded platform for car's surrounding vision system

Sergio Saponara, Emilio Franchi

Sergio Saponara, Emilio Franchi, "Real-time and low-cost embedded platform for car's surrounding vision system," Proc. SPIE 9897, Real-Time Image and Video Processing 2016, 98970T (29 April 2016); doi: 10.1117/12.2228236

SPIE.

Event: SPIE Photonics Europe, 2016, Brussels, Belgium

Real-time and low-cost embedded platform for car's surrounding vision system

Sergio Saponara*[a], Emilio Franchi[b]
[a]Dep. of Information Engineering, University of Pisa, via G.Caruso 16, 56122, Pisa, I,
[b]R.I.CO. Italy, Via Adriatica 17, 60022, Castelfidardo (AN), I

ABSTRACT

The design and the implementation of a flexible and low-cost embedded system for real-time car's surrounding vision is presented. The target of the proposed multi-camera vision system is to provide the driver a better view of the objects that surround the vehicle. Fish-eye lenses are used to achieve a larger Field of View (FOV) but, on the other hand, introduce radial distortion of the images projected on the sensors. Using low-cost cameras there could be also some alignment issues. Since these complications are noticeable and dangerous, a real-time algorithm for their correction is presented. Then another real-time algorithm, used for merging 4 camera video streams together in a single view, is described. Real-time image processing is achieved through a hardware-software platform

Keywords: Car surrounding vision, fish-eye camera, ADAS (Advanced Driver Assistance Systems), real-time image processing, distortion correction, blind zones, video mosaic, image fusion

1. INTRODUCTION

The use of cameras for automotive applications is growing rapidly, probably because among all the car safety technologies that are spreading, those that stimulate the sight are considered the most immediate and reliable from the drivers. However, current proposed systems are still too expensive to be equipped on medium- and low-end vehicles. For this reason, in academia and industry there is a lot of research to develop advanced but low-cost ADAS solutions, offering technologies for the obstacle's detection, brake assist and some other products to improve vision [1-11]. Among all the dangerous situations that can be encountered while driving, the most frequent is the one determined by the blind spots [1]: the driver is unable to see the zones around the vehicle. The system presented in this work solve this issue, allowing for a real-time vision from the top of the car of the zones near the vehicle on the on-board display (e.g. the LCD display already mounted in the dashboard for navigation and infotainment subsystems). This solution has been designed to help the driver while parking, but can easily be adopted also for the obstacle detection and brake assist. The purpose of the project is to develop a low-cost product, so that it can spread even in the low-end of the automotive market. For this reason, four cheap VGA fish-eye cameras are used, which are obviously affected by radial distortion, but they have also a misalignment of the center of distortion. Many fish-eye's correction algorithms are available in literature [2-8]. However, real-time processing is needed for automotive driver assistance. To address computing-intensive algorithms, many hardware architectures for ADAS foresee a programmable core, often a low-power microcontroller as in [12], enhanced by dedicated co-processors as in [13-17]. The ADAS solution has to be also a low-power one, because power-efficiency is becoming a main issue in automotive applications. Furthermore, the platform must be low-cost to obtain widest diffusion in the large automotive market. Harsh automotive operating conditions have to be faced [18-20]. A real-time low-power correction of a fish-eye camera has already been proposed in literature [4, 5]. Unfortunately, many of these algorithms after radial correction, crop the scene to remove image's borders where the results of the correction are worst. To merge more videos, they can't be cropped because they have to overlap between them. Starting from these works, reviewed in Sections 2 and 3, a new hardware-software system is presented in Sections from 4 to 7 to solve lenses' alignment issues and to obtain wide images. Conclusions are drawn in Section 8.

Real-Time Image and Video Processing 2016, edited by Nasser Kehtarnavaz, Matthias F. Carlsohn, Proc. of SPIE Vol. 9897, 98970T · © 2016 SPIE
CCC code: 0277-786X/16/$18 · doi: 10.1117/12.2228236

2. STATE OF THE ART OF FISH-EYE EFFECT CORRECTION

The fish-eye optic is characterized by a field-of-view between 100 ° and 230 °. The distortion introduced by this lens is proportional to its field-of-view and is indicated by the two components of the polar reference system: radial and tangential. The tangential distortion is due to the poor quality of construction of the lens and, even if present, can be ignored because it has a negligible impact on the results. Instead, the radial distortion cannot be ignored, because its effect is considerable and represents the bottleneck for the development of video processing applications. There are different types of fish-eye lenses, each identified by a mapping function, which is a mathematical relation that associates the points of the projected image on the sensor, to the points of the real scene view from the lens. These functions are described starting from the model of a pinhole camera, which perfectly describes the relationship for a non-distorting lens. It should be noted that for the fish-eye lens, the focal length can be different from the physically measurable one, so we will talk about Apparent Focal Length. If θ is the angle between the optical axis and the ray of light emitted from a point P of the real scene, f is the focal length and finally r_{corr} is the distance between the projection of P on the sensor and the focal axis, these variables can be related between them by means of the following equation:

$$r_{corr} = f \tan\theta \quad (1)$$

The algorithms for the correction of radial distortion calculate the appropriate position of each pixel of the distorted image, and then they make a copy of these pixels to another empty image, at the position calculated. Distortion is limited to the radial axis, so the correct position of the pixels will still be along this axis, but at a different distance from the center. In the literature, there are two families of different correction algorithms: polynomial and non-polynomial [2, 5-8]. A comparison is described in [2]. The first type, determine r_{corr} using the polynomials, as shown in [5]. The simplest models only consider the terms of the polynomial of odd degree, like shown in Eq. 2. Their advantage is that the polynomial equations require a moderate computational cost.

$$r_{corr} = r_{fish} + \sum_{n=1}^{\infty} k_n r_{fish}^{2n+1} = r_{fish} + k_1 r_{fish}^3 + \cdots + k_n r_{fish}^{2n+1} \quad (2)$$

The non-polynomial algorithms [6-8] instead, take advantage of the equation of the mapping function associated with the lens. Often, the mapping function is characterized by logarithmic or trigonometric functions that are undoubtedly more expensive to evaluate than polynomial ones. The fish-eye lenses used in this system are represented by the mapping function called “Equisolid Angle”:

$$r_{fish} = 2f \sin\left(\frac{\theta}{2}\right) \quad (3)$$

Both methods can act in two ways: they can calculate the correct destination for each pixel in the distorted image, or they can calculate the correct source pixel to be taken from the image distorted for each pixel of the empty image. The first procedure is called “Forward Mapping” and may cause an incomplete picture due to missing pixels. The reverse mode is called “Backward Mapping” and it does not suffer from missing pixels, but it may have duplicate pixels, resulting in a very similar result to what can be achieved with a “Nearest Neighbor” interpolation algorithm. It should be specified that the algorithms characterized by polynomial equations are often not easily reversible, therefore it is not always possible to apply the backward mapping. The equation (3) could be reversed only if approximated with a finite number of monomials. In contrast, the non-polynomial algorithms are analytically invertible, without any approximation.

In [4] it’s used backward mapping approach for compiling a Lookup Table (LUT). The LUT stores the positions of the source pixels calculated initially, and then is used to rearrange the pixels of each video frame. The resulting frames can be reassembled in a video stream. The use of the LUT makes superfluous any computational advantage of polynomials algorithms. The procedure used in [4] uses (3) to determine the apparent focal length f. Then, substituting λ to θ in (1), it obtains the angle of the radial axis associated with the pixel currently being processed:

$$\lambda = \tan^{-1}\left(\frac{r_{corr}}{f}\right) \quad (4)$$

Finally, by substituting the equation (4) in (3), with λ in place of θ it obtains the equation for backward correction:

$$r_{fish} = 2f \sin\left[\frac{1}{2}\tan^{-1}\left(\frac{r_{corr}}{f}\right)\right] \quad (5)$$

3. STATE OF THE ART OF VIDEO FUSION ALGORITHMS

The process of merging graphical objects is a recent digital processing area, and it's still under study. A video stream is composed of a sequence of images, so it may think to be inspired by the algorithms for creating panoramic photos, like the one described in [9]. In this system, creators first locate the common parts of the images that have to be merged together. In a second time, the overlapping areas are compared to identify common elements between them. Those items are then used to apply perspective transformation. Finally, bilinear interpolation is applied to homogenize the result. What emerges is one picture of fair quality, in which there aren't any sort of discontinuities on the scene. However, an algorithm like the one described in [9] cannot be implemented in real-time, because it would require a very powerful computational unit, which is not low-cost. A product closer to the constraints of the system proposed here is described in [10], in which the authors use four fish-eye cameras to propose to the driver a top-view of the vehicle, with the possibility to vary the viewing angle. For the necessary calculations two processors are used: an ARM926EJ-S 333 MHz and a 166 MHz Graphics Display Controller that contains a graphics engine for 2D and 3D geometries. The algorithm makes use of OpenGL ES1.1 libraries. The results obtained are good, but far from the ones shown in [9]. In fact, despite a form of interpolation is applied, the border areas between the cameras are clearly recognizable.

4. HARDWARE-SOFTWARE ARCHITECTURE

The proposed Car's surrounding vision platform involves 4 cameras made by Rico. S.r.l. and equipped with Aptina MT9V128 sensors, capable of producing video streams at 30 fps with PAL resolution, and 4 fish-eye lenses with 173° FOV. The four video streams are combined into a single video stream, always with PAL resolution, using a development board Texas Instruments TVP5154 EVM equipped with a 4-channel multiplexer. The resulting video stream is divided into four quadrants, each of which is coupled to a camera. The quad video stream is then processed through a further development board, Texas Instruments DM642 EVM, equipped with a DM642 DSP at 720 MHz. This DSP, although obsolete because new families have been released, is designed for low-power automotive applications. An architectural representation can be seen in Fig.1. As regards the software programming, the multiplexer is configured by means of the proprietary software WinVCC, while the algorithms described below were realized in C, using the IDE Code Composer Studio.

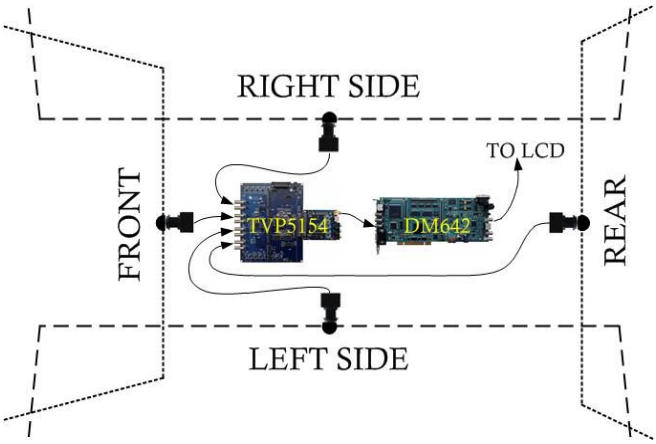

Figure 1. Interconnections between the 4 cameras and the other components. The dashed lines show the capture areas of each camera.

5. ISSUES OF FISH-EYE CAMERAS

As mentioned previously, the used cameras are affected by some manufacturing imperfections, which required parameterizing the algorithms and making them flexible. The worst of all defects, is that there is the misalignment between the focal axis of the lens and the center of the sensor, and it is due to the imprecise construction of the plastic support that connects the sensor to the lens. The point of the sensor that intersects the focal axis is called "Center of Distortion" (COD), as described in [2]. This issue requires to not apply the radial correction algorithm from the radial center of the sensor, but rather by the COD. Obviously, the corrected image need to be re-centered on this point. Therefore, it will be as asymmetric as is the deviation of the COD from the center of the sensor, both vertically and horizontally. In the proposed system, each low-cost camera may be affected by a different misalignment, as shown in

Fig. 2. Another problem not easily solvable by means of software, is that the thread pitch of these lenses is not perfectly compatible with the one of the mechanical support. Being a bit smaller, does not generate the necessary grip to lock the lens to the support. This causes a form of instability of the parallelism between the focal axis of the lens and the sensor, during the movement. This phenomenon is referred to as slack, and it is shown in Fig. 3

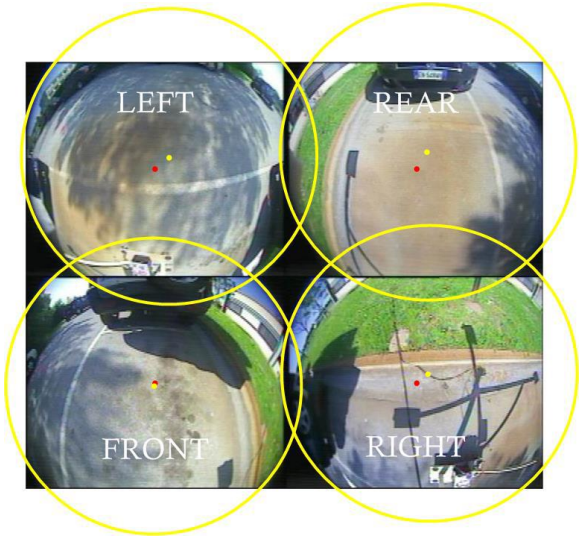

Figure 2. Different misalignment for each camera. The red point is the sensor center, yellow one is COD.

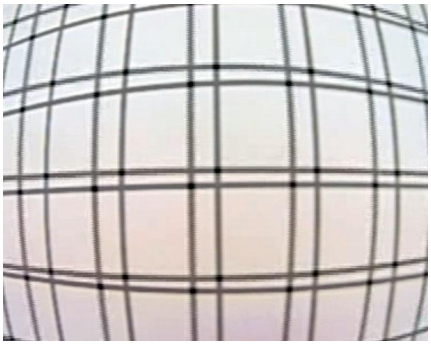

Figure 3. Slack effect on a calibration grid. Lens can move anyway. Here are shown two different images overlapped.

6. FLEXIBLE FISH-EYE CORRECTION ALGORITHM AND VIDEO FUSION

The fish-eye correction algorithm implemented is a hybrid version, in the sense that it is based on the principle of operation of non-polynomial algorithms, but refines the results using a polynomial approach. This choice was made analyzing the behavior of a non-polynomial algorithm through a calibration grid: what emerges is that the peripheral areas and the central one of the captured image does not represent the objects in the same way. In fact, an object framed centrally will have a reasonable size, but if you move the camera to reposition the same object at the edge of the image, it will be larger. This is due to the calculation of the apparent focal length, which is derived from Eq. (3). This mapping function does not care of the variation of FOV of the lens used. To solve this problem, the apparent focal length is still calculated using Eq. (3), but with a different multiplicative coefficient between the two coordinates. A separate elaboration is done on both sizes, one used for the vertical coordinates and another for the horizontal ones. Furthermore, this multiplication coefficient is dynamically adapted: it changes as they are processed more distant pixels. Here it comes the theory behind the polynomial algorithms: to obtain the solution, a polynomial is added or subtracted from the initial factor. These polynomials have as a variable one of the two coordinates. In addition, the proposed algorithm allows to specify two offsets, one vertical and the other horizontal, to indicate the misalignment of the center of distortion. Finally, additional parameters were included to scale the size of the corrected image, to maximize the overlapping areas of the cameras. Since each camera is affected by different imperfections, calculations for each radial corrections are made only for the first frame and stored in four different “Fish-Eye Correction Lookup Tables” (FCLUT).

The video fusion algorithm initially fills a primary Lookup Table (LUT), which contains the positions of the pixels of the source quad image. Later, this LUT is used cyclically to reorder the source pixels and compose new frames, offered in ordered sequence to the driver. This work focuses on the calculation step of the primary LUT. Because this algorithm must also be flexible and adaptable to different types of vehicles, it accepts as parameters the positions and tilt angles of the four cameras. Moreover, also the "Point of View" (POV) from the top of the car must be changeable. Therefore, other parameters identify its position in three-dimensional space. The first phase of this algorithm allows to derive the three-dimensional coordinates of the real environment that correspond to a given pixel to be displayed on the car's display to the coordinates (i, j). For this calculation it is assumed that the floor is perfectly flat and that the car's display is virtually positioned at the focal distance LCDFOCAL from POV, which will stay at a height POVHEIGHT, disposed parallel to the road surface as to form a pinhole camera in the virtual coordinates of the point of view. The expressions for the two coordinates are obtained by simple proportions, derivable from Fig. 4a. The following proportion describes the procedure for calculating the x coordinate:

$$x : i = POVHEIGHT : LCDFOCAL \rightarrow x = \frac{i * POVHEIGHT}{LCDFOCAL} \quad (6)$$

The second step finds from which of the four cameras is more appropriate to take the flooring area associated with the pixel being processed. The calculations are very similar to those of the third phase, in which it get the line and column numbers of the pixel that takes up the affected area on the pavement identified in the first phase, relative to the camera video stream selected in the second stage. Following the geometrical construction shown in Fig. 4b-c, and bearing in mind that the QUADHEIGHT and QUADWIDTH identify respectively the height and width (in pixel) of the quad of one of the four video streams, it is possible to derive the desired dimensions:

$$linPos : \frac{QUADHEIGHT}{2} = d_L : d_L Max \quad (7)$$

$$colPos : \frac{QUADWIDTH}{2} = d_A : d_A Max \quad (8)$$

The fourth phase, uses FCLUT associated with the current camera to convert the line and column of the selected pixels, so that they can be corrected by the radial distortion. A static image of a car is overlapped for marketing issues.

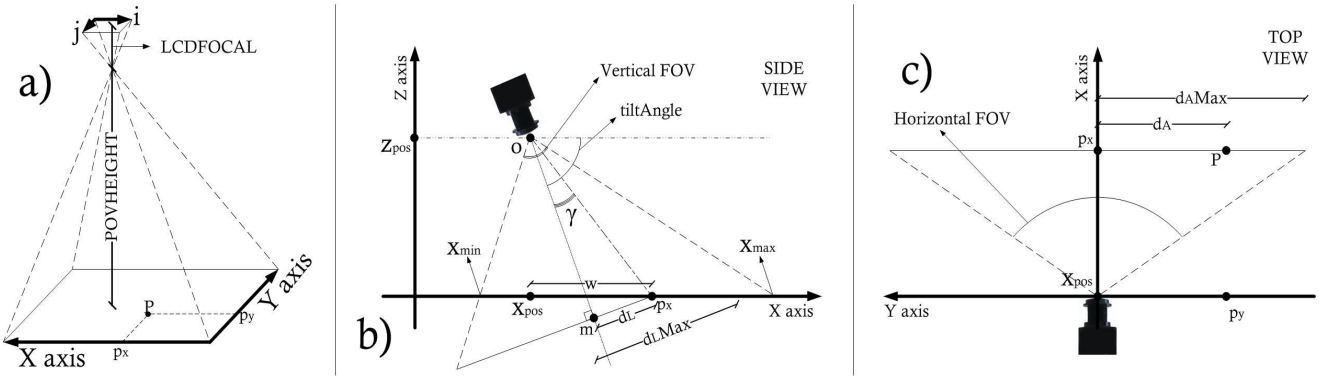

Figure 4. Geometric models involved in the video fusion algorithm.

7. EXPERIMENTAL RESULTS

The experimental results obtained are shown in Fig. 5. The two sequences of images show a parking simulation, done with a mobile support. The support is designed to keep constant the distance between the cameras, and has the typical dimensions of a small car. In the center of the video an image of a hypothetical car was also placed. The two sequences were shot by varying the height of the point of view. Regarding memory requirements, a FCLUT is small and needs 400KB, the primary LUT instead, it takes about 1700 KB. If we consider that the FCLUT are no longer needed once drawn the primary LUT, 2 MB of total memory are enough to run the code.

Figure 5. Two different parking simulations at different POV.

8. CONCLUSION AND FUTURE WORK

The proposed real-time embedded system could have a higher resolution, opting for a multiplexer that does not reduce it so drastically, as one of those for high-definition applications. The discontinuities between the different cameras are perceivable, also because of the difference in brightness and color between the zones. This defect can be eliminated by performing more accurate calibration of the fish-eye correction algorithm, and applying some kind of interpolation. However, if you consider that the system requires about 2 MB of memory to be run, and that the DSP DM642 at 720 MHz is used for about 40% of its total power, the results are satisfactory. Further work is on-going for visual quality improvement or further reduction of the cost, scaling on lower-frequency processors like the Texas Instruments DM368, which is an ARM926 processor at 400 MHz.

ACKNOWLEDGMENT

Discussions with Dr. Fontanelli and Prof. Fanucci from University of Pisa are gratefully acknowledged.

REFERENCES

[1] Hughes, C., et al. “Wide-angle camera technology for automotive applications: a review”, IET Int. Transport Sys., vol. 3, (2009)

[2] Hughes, C., et al. “Review of geometric distortion compensation in Fish-Eye cameras”, IEEE ISSC2008, pp. 162-167, (2008)

[3] Mallon, J. et al., “Precise radial un-distortion of images”, IEEE Int. Conf. on Pattern Rec., vol. 1, pp. 18–21, (2004)

[4] Turturici, M., et al., “Low-power DSP system for real-time correction of fish-eye cameras in automotive driver assistance applications”, Journal of Real-Time Image Processing, vol. 9, n. 3, pp. 463-478, (2014)

[5] Hughes, C., et al., “Validation of Polynomial-based Equidistance Fish-Eye Models”, IET ISSC 2009. (2009)

[6] Basu, A., et al. “Alternative models for fish-eye lenses”, Pattern Recognition Lett., vol.16, n.4, pp.433–441 (1995)

[7] Devernay, F., et al., “Straight lines have to be straight: automatic calibration and removal of distortion from scenes of structured environments”, Journal of Machine Vision and App., vol. 13, n. 1, pp. 14–24 (2001)

[8] Ishii, C., et al., “An image conversion algorithm from fish eye image to perspective image for human eyes”, IEEE/ASME Int. Conf. on Advanced Intell. Mechatronics, pp. 1009–1014 (2003)

[9] Lee, J., et al., “Fast panoramic image generation method using morphological corner detection”, Lect. Notes Computer Sci. vol. 3767, pp. 547 – 558 (2005)

[10] S. Shimizu, et al. “Wraparound view system for motor vehicles”, Fujitsu Sci. Tech. J., vol.46, n.1, pp.95-102 (2010).

[11] Marsi, S., et al., “Integrated video motion estimator with Retinex-like pre-processing for robust motion analysis in automotive scenarios: algorithmic and real-time architecture design”, Journal of Real-Time Image Processing, vol. 5, n.4, pp. 275-289 (2010)

[12] Fanucci L., et al., “Power optimization of an 8051-compliant IP microcontroller”, IEICE Transactions on Electronics, vol. E88-C, n. 4, pp. 597-600 (2005)

[13] Fanucci, L., et al., “A parametric VLSI architecture for video motion estimation”, Integration the VLSI Journal, vol. 31, n. 1, pp. 79-100 (2001)

[14] Saponara, S., et al., “Motion estimation and CABAC VLSI co-processors for real-time high-quality H.264/AVC video coding”, Microprocessors and Microsystems, vol. 34, n. 7-8, pp. 316-328 (2010)

[15] Fanucci, L., et al., “Parametrized and reusable VLSI macro cells for the low-power realization of 2-D discrete-cosine-transform”, Microelectronics Journal, vol. 32, n, 12, pp. 1035-1045 (2001)

[16] Fanucci, L., et al., “Data driven VLSI computation for low power DCT-based video coding”, IEEE International Conference on Electronics, Circuits, and Systems, ICECS 2002, vol. 2, pp. 541-544 (2002)

[17] Chimienti, A., et al., “VLSI architecture for a low-power video codec system”, Microelectronics Journal, vol. 33, n. 5-6, pp. 417-427 (2002)

[18] Costantino, N., et al., “Design and test of an HV-CMOS intelligent power switch with integrated protections and self-diagnostic for harsh automotive applications”, IEEE Transactions on Industrial Electronics, vol. 58, n. 7, pp. 2715-2727 (2011)

[19] Baronti, F., et al., “State-of-charge estimation enhancing of lithium batteries through a temperature-dependent cell model”, IEEE International Conference on Applied Electronics, pp. 29-34 (2011)

[20] Saponara, S., et al., “Sensor modeling, low-complexity fusion algorithms, and mixed-signal IC prototyping for gas measures in low-emission vehicles”, IEEE Transactions on Instrumentation and Measurement, vol. 60, n. 2, pp.372-384 (2011)