World Academy of Science, Engineering and Technology
International Journal of Physical and Mathematical Sciences
Vol:4, No:4, 2010

Hardware Centric Machine Vision for High Precision Center of Gravity Calculation

Xin Cheng, Benny Thörnberg, Abdul Waheed Malik and Najeem Lawal

***Abstract*—**We present a hardware oriented method for real-time measurements of object's position in video. The targeted application area is light spots used as references for robotic navigation. Different algorithms for dynamic thresholding are explored in combination with component labeling and Center Of Gravity (COG) for highest possible precision versus Signal-to-Noise Ratio (SNR). This method was developed with a low hardware cost in focus having only one convolution operation required for preprocessing of data.

***Keywords*—**Dynamic thresholding, segmentation, position measurement, sub-pixel precision, center of gravity.

I. Introduction

AUTOMATIC indoor- and outdoor navigation techniques for robots and vehicles are of greatest importance for the development of future smart products and household robots.

Fig. 1 depicts the schematic principle for navigation using a video camera assembled on top of a vehicle and light beacons sending light into the camera. This smart camera is able to identify a number of beacons in its neighborhood and measure the spatial positions of the light spots projected on the sensor. The angle of the incoming light rays relative to the vehicles direction can then be computed. Every beacon can be identified using coded light such that the position and direction of the vehicle can be calculated. This measurement technique is very similar to nautical navigation on sea. Larsson et al. presents a similar automatic navigation system based on a rotating laser [1]. The machine vision system used for navigation shown in Fig. 1 has motivated us to investigate how to get high precision on the light spot position measurement at real-time performance for a reasonable low hardware cost.

Most of the computations for a machine vision system are preferably done on a computational platform closely located to the image sensor. The smart camera, see Fig. 2A, constitutes such a configuration [2][3].

Xin Cheng is with the Department of Information Technology and Media, Mid Sweden University, Holmgatan 10, 851 70 Sundsvall, Sweden (corresponding author to provide phone: +46-60148917; fax: +46-60148456; e-mail: xin.cheng@miun.se).

Benny Thörnberg, Abdul Waheed Malik and Najeem Lawal are with the Department of Information Technology and Media, Mid Sweden University, Holmgatan 10, 851 70 Sundsvall, Sweden (e-mail: {benny.thornberg, waheed.malik, najeem.lawal}@miun.se).

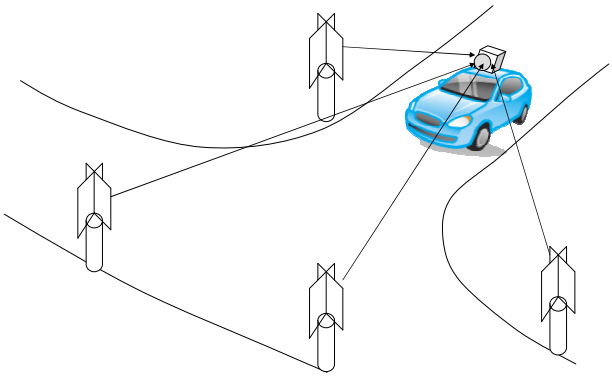

Fig. 1 Navigation of vehicle using smart camera and light beacons.

Machine vision algorithms are typically divided into the following steps [4]: Video is acquired from the image sensor at *Image acquisition*. Image objects are extracted from the pre-processed video data at *Segmentation*, see Fig. 2B. During *Labelling*, pixels belonging to the same image component are assigned a unique label. At *Feature extraction* an image component is described for example in terms of region features such as area, ellipse-, square- or circle parameters. Components can also be described in terms of gray value features such as mean gray value or position. This feature information can then be used for *Classification* of image components. Information about recognised objects in the camera's observation area can then be transmitted at the camera output using typically a very low bandwidth.

Field Programmable Gate Array (FPGA) is a computational platform that offers massive parallelism, on-chip memories, arithmetic units and is therefore found to be most suitable for front end video processing [3][5].

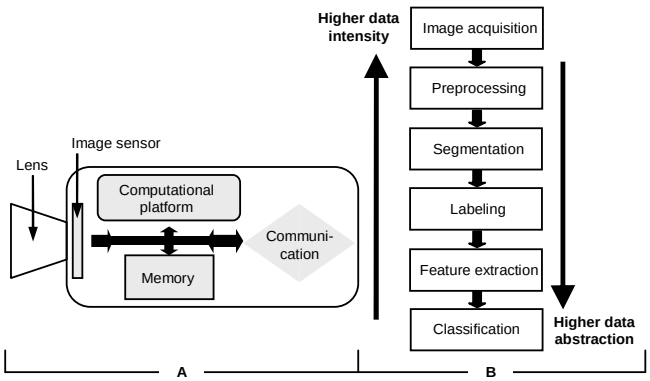

Fig. 2 A) Smart camera. B) Fundamental steps of machine vision.

However, due to the limited resources, it becomes essential to consider hardware costs such as memory storage requirements, bit-widths and complexity of arithmetic operations when selecting and designing algorithms for implementation in real-time FPGA systems.

When used for image processing, COG is an algorithm for calculating the mass center of an image object [6]. The image objects of interests must first be separated from the background at an image segmentation step before the COG can be applied separately on all objects [4][7]. This image segmentation is at its simplest form a threshold applied globally on the grey levels of the image. The precision of the calculated COG is dependent on image noise as well as the selected threshold. This is, because thresholding is the method to exclude pixels with low SNR from the COG calculation.

We have found several articles on COG [6][8][9][10][11] and several books and articles on image segmentation [4][7]. But to our knowledge, we have not found any article that combines image segmentation with component labeling and COG that investigates the accuracy and robustness of the calculated object positions and its dependency of the SNR. We propose in this work an enhanced segmentation method for efficient extraction of light spots from a non uniform background to be combined with component labeling and COG for high precision measurement of image object positions. This enhanced image object position measurement algorithm is developed with a low hardware cost in focus. We strongly believe that algorithms for real-time video processing and machine vision must be developed in combination with an analysis of a possible computational hardware platform.

II. Center Of Gravity

From the definition of COG [6], coordinate $c(x,y)$ of a light spot in a gray-level 2D image can be calculated over a neighborhood Ω of pixels,

$$c(x,y) = \left(\frac{\sum_{x,y\in\Omega} xw(x,y)}{\sum_{x,y\in\Omega} w(x,y)}, \frac{\sum_{x,y\in\Omega} yw(x,y)}{\sum_{x,y\in\Omega} w(x,y)}\right) \quad (1)$$

The coordinate weights $w(x,y) = a\cdot(f(x,y)-m)$ $m < \min_{x,y\in\Omega}(f(x,y))$ when $a>0$ or $m > \max_{x,y\in\Omega}(f(x,y))$ when $a<0$ are input data to the COG calculation. In the first case, the COG is attracted to the brighter pixels in the neighborhood or else darker pixels. Since the light spots in our study are brighter pixels, we apply the first case by simply selecting $m=0$ and $a=1$.

We can describe the approximate variance of the COG based on the assumption of an input image with additive noise having zero mean and variance σ^2 [6],

$$\mathrm{var}(c(x,y)) \approx \frac{\sigma^2\cdot\sum_{x,y\in\Omega} x^2}{(N\cdot\hat{\mu}_w)^2}, \frac{\sigma^2\cdot\sum_{x,y\in\Omega} y^2}{(N\cdot\hat{\mu}_w)^2}. \quad (2)$$

N is the number of pixels in a neighborhood Ω and $\hat{\mu}_w$ estimates the local mean value of the weight signal. Equation (2) is found to be a good variance estimator under the conditions that the neighborhood Ω has a mass center that coincides with the light spot's mass center. We define the signal-to-noise ratio for an object as,

$$SNR = \frac{\hat{\mu}_w}{\sigma}. \quad (3)$$

This allow us to rewrite the expression for the COG variance from equation (2) as,

$$\mathrm{var}(c(x,y)) \approx \frac{1}{N^2}\cdot\left(\sum_{x,y\in\Omega} x^2, \sum_{x,y\in\Omega} y^2\right)\cdot\frac{1}{SNR^2} \quad (4)$$

The reversed standard deviation of the COG output can be interpreted as the sub-pixel precision of the determined position of an object. The reversed standard deviation is simply developed from equation (4) as,

$$\frac{1}{std(c(x,y))} \approx N\left(\frac{1}{\sqrt{\sum_{x,y\in\Omega} x^2}}, \frac{1}{\sqrt{\sum_{x,y\in\Omega} y^2}}\right)\cdot SNR. \quad (5)$$

Thus, according to equation (5) we can expect that the standard deviation of the calculated position of a light spot is reversely proportional to the light spot's SNR. But we can also conclude that the same standard deviation will be dependent on the neighborhood Ω of pixels. This neighborhood of pixels is determined by the *segmentation* as depicted in Fig. 2B.

The simplest method for segmentation is to apply a threshold on the pixel gray values. This threshold determines which pixels are to enter into the neighborhood Ω. For bright image components such as light spots on a dark background and using smaller threshold, more pixels will enter into the neighborhood Ω.

III. Image Segmentation by Thresholding

Thresholding plays an important role in segmenting objects from background in image processing due to its intuitive properties and simplicity of implementation. For a threshold T_{xy}, this operation can be formalized into,

$$B(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T_{xy} \\ 0 & \text{others} \end{cases} \quad (6)$$

B(x,y) is the resulting binary image and *f(x,y)* is the input gray level image. In this section, we will discuss different

methods for selecting the threshold T_{xy}. We will also show how preprocessing can be applied to suppress the influence of a non homogeneous image background as well as how we have improved this preprocessing compared to referenced literature. At the end of this section, based on existing work, we propose an FPGA based hardware architecture to be used for the image segmentation.

A. Optimized global threshold using Otsu's criterion

Otsu's criterion [7] selects a global threshold T such that it maximizes the in between variance of classes belonging to objects and background. If the image has objects with equal intensity and a distinct background such that its image histogram becomes bimodal, we can get an optimized threshold by using Otsu's method. However, an image having a non uniform background and multiple light spots with different range and illumination intensities, Otsu's criterion cannot be applied directly with good results.

B. Variable- or dynamic thresholding

We know the local standard deviation σ_{xy} denotes the local contrast and the local mean m_{xy} denotes the local average intensity. By comparing the pixel intensity with its local average intensity, we can segment it even if the background is non uniform [4][7]. If the local contrast is also considered, it is named *Variable thresholding* [7] and the local threshold T_{xy} can be determined by the following formula,

$$T_{xy} = a\sigma_{xy} + bm_{xy}. \tag{7}$$

The non negative constants a and b are selected on experimental basis.

Dynamic thresholding as described in [4] is very similar to equation (7) but the threshold is now defined as $T_{xy}=m_{xy}+ D_{diff}$. In this case the thresholding is based on a global nonzero constant D_{diff} selected on experimental basis. From this we conclude that *Dynamic thresholding* can alternatively be described as two subsequent steps:

1) *Image preprocessing*: Calculate an estimation of the image background by using the local mean values and subtract this background from the original image. This operation corresponds to a high-boost filter [7].
2) *Thresholding:* Apply a global threshold T on the preprocessed image according to equation (6).

The *Image preprocessing* in step 1 constitutes one single convolution operation, suitable for real-time video processing. This is a 2D filter, preferable illustrated in the frequency domain. The amplitude characteristic for an 11x11 pixels filter mask is shown in Fig. 3. It is the combination of mean value and subtraction as described in step 1 results in a high-boost filter. The stop band for the low frequencies will suppress the image background while higher frequencies for the small light spots will pass this filter.

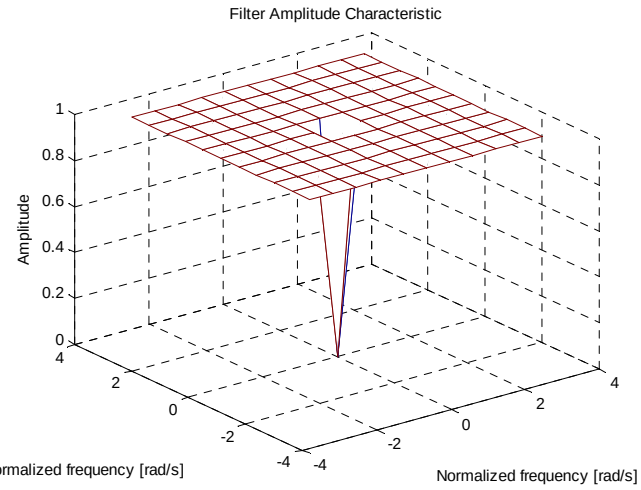

Fig. 3 Ampl. characteristics for pre-processing filter of Algorithm 1.

Since the image background is suppressed at step 1, the global threshold T used at step 2 could possibly be optimized using Otsu's criterion [7]. However the global statistics required for this operation will have a high hardware cost associated with it. A much simpler computation of the global threshold T is,

$$T = P \cdot \max\big(f(x,y)\big). \tag{8}$$

P is another global threshold specified in percentage of the maximum pixel value for a whole video frame (image). P was experimentally selected to $P=0.10$ for all experiments in this work.

C. Improved preprocessing

Equation (5) estimates the stability of the COG output under the influence of noise assuming a fixed neighborhood Ω centered over the image object. However, the thresholding operation is also sensitive to noise which means that the neighborhood Ω will shrink, grow and alter its shape in a non deterministic manner. In order to reduce the noise sensitivity, we suggest that the filter's response to the high frequencies must be limited. Fig. 4 shows the frequency response for a modified preprocessing filter. This filter was generated by simply convolving the filter mask corresponding to amplitude characteristic in Fig. 3 with a Gaussian function $g(x,y)$ defined for an 11x11 pixel neighbourhood Φ.

$$g(x,y) = \frac{h(x,y)}{\sum_{u,v\in\Phi} h(u,v)} \wedge h(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}. \tag{9}$$

The parameter σ was experimentally selected to σ=1.2. We see in Fig. 4 that the resulting amplitude characteristic now corresponds to a two dimensional band pass filter. For this simple addition to the preprocessing filter, the improvement of the subpixel precision for the calculated light spot positions will be shown in section V.

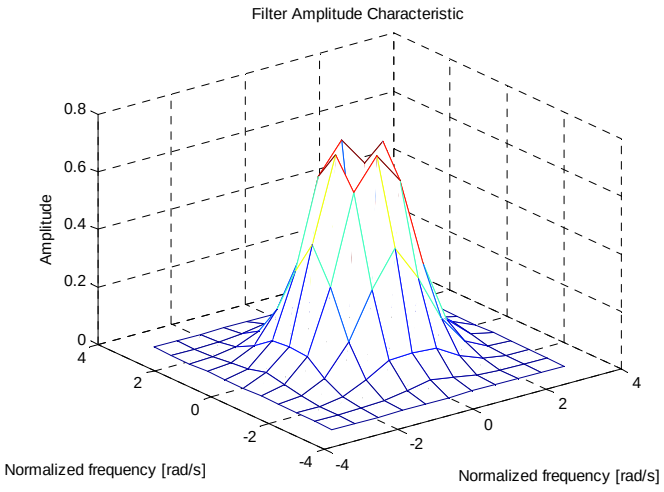

Fig. 4 Pre-processing filter of Algorithm 2 and 3.

D. *Hardware architecture for image segmentation*

The improved preprocessing, previously described in section III.C combines a high-boost filter with a Gaussian smoothening filter in one single convolution operation. For real-time processing on FPGA, convolving is preferable implemented using the hardware architecture shown in Fig. 5. This is a memory hierarchy where data reuse is exploited. There are First-In-First-Out (FIFO) registers used for delaying the video data equal to the number of clock cycles corresponding to one line of progressive video. These FIFO registers are preferably implemented in on-chip block-RAMs [12]. Pixel delays are implemented as registers close to the data path. The computational logic is preferably pipelined such that a throughput of one pixel per clock cycle is achieved.

IV. Measurement of Image Object Positions

In previous section, we proposed a method to be used for segmentation of light spots from image background. In this section, we also include image component labeling and computation of light spot's position by COG.

A. *Three alternative algorithms*

We have developed three alternative algorithms from which we will choose the one that gives the best performance. These algorithms are represented as Signal Flow Graphs (SFG) divided into two main parts, *Preprocessing* and *Object processing*. See Fig. 6 and Fig. 7. *Preprocessing* corresponds for Algorithm 1 in Fig. 6 to the filter illustrated in Fig. 3.

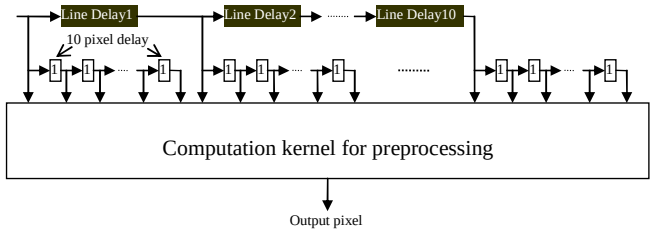

Fig. 5 Hardware architecture for the preprocessing filter.

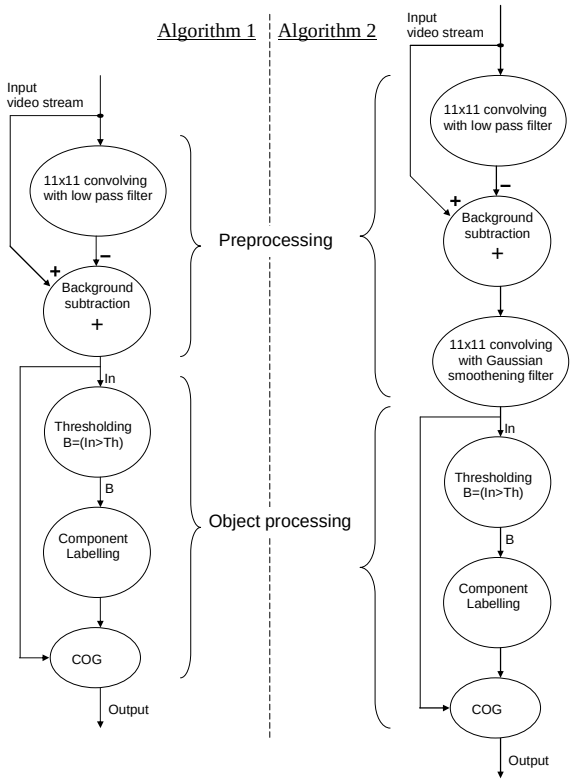

Fig. 6 Signal flow graph for Algorithm 1 and 2.

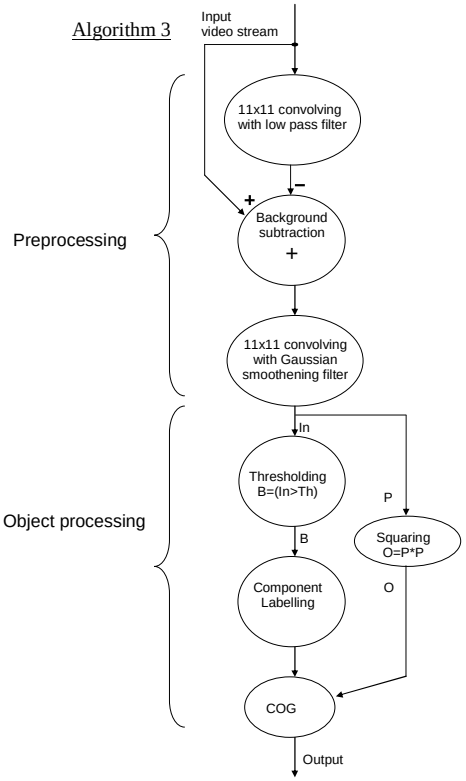

Fig. 7 Signal flow graph for Algorithm 3.

The method for segmentation used for Algorithm 1 equals the dynamic thresholding as described in [4] and presented in section III.B.

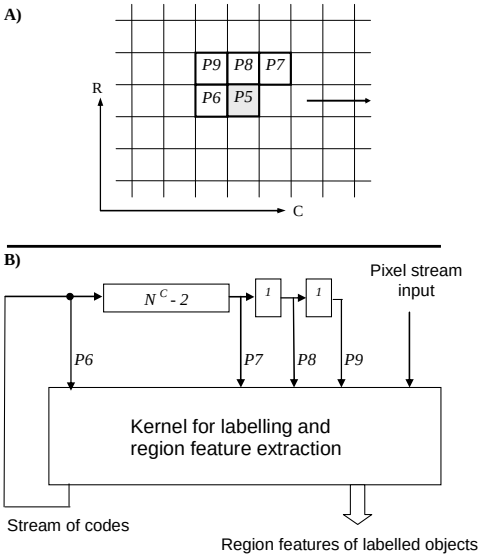

Fig. 8 A) Neighbourhood of labeled codes. B) Kernel and delay line.

Algorithm 2 shown in Fig. 6 and Algorithm 3 shown in Fig. 7 are both based on the improved dynamic thresholding using the preprocessing filter shown in Fig. 4. *Object processing* corresponds for all three algorithms to thresholding, labeling and COG calculation. The only difference is that for Algorithm 3, COG is calculated on the squared pixel data.

B. HW architecture for component labelling and COG

Based on previous research, we propose the following hardware architecture as suitable for implementation of component labelling and COG calculation on FPGA [13].

A neighbourhood of labels that can be used for both 4- and 8-connectivity labelling is shown in Fig. 8A. A delay line of one FIFO-buffer and two registers holds the recursive data dependency arising from the neighbourhood of previously labelled pixels, see Fig. 8B. Pixels are assigned labels at *P5* based on the neighbouring labels in *P6* to *P9*. We assume that the latency of the labelling is exactly one clock cycle. The length of the FIFO buffer is $N^C - 2$ numbers of elements where N^C equals the length of one image row [13]. The kernel for labelling and COG calculation (feature extraction) is further described in Fig. 9.

The *Labeller* assigns label codes. Label pairs *(A,B)* are sent to the *Equivalence table* whenever neighbouring labels are found equal and must be merged. Label merging is targeted to either *Table A* or *B* dependent on odd or even frame (*O/E*). Resolving of linked lists of labels is thus frame interleaved with labelling and label merging.

In parallel with assigning label codes for connected image components, data is accumulated in data table A or B. This accumulation corresponds in the case of COG calculation to the numerators and denominators for each connected components, see equation (1). When the *Resolver* is ready, the division of numerator and denominator according to equation (1) is done.

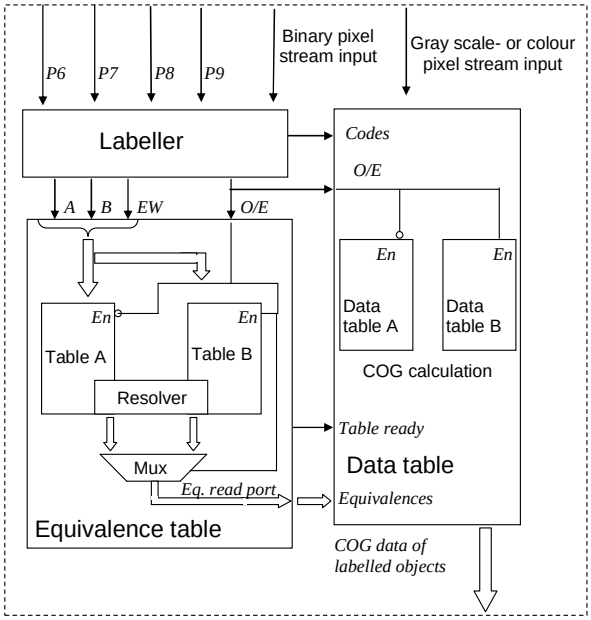

Fig. 9 Kernel for labeling and calculation of COG.

This final calculation of COG will require access to the resolved equivalence table. This division is preceded by accumulation of numerators and denominators in *Data table A* or *B*. Accumulation of data and final calculation of COG is thus frame interleaved.

V. Experimental Setup

The experiments carried out to analyze the performance of Algorithm 1 to 3 are described in this section.

A. Image acquisition

A board was used as a laminate for the assembly of 90 infrared Light Emitting Diodes (IR-LED). Se Fig. 10A.

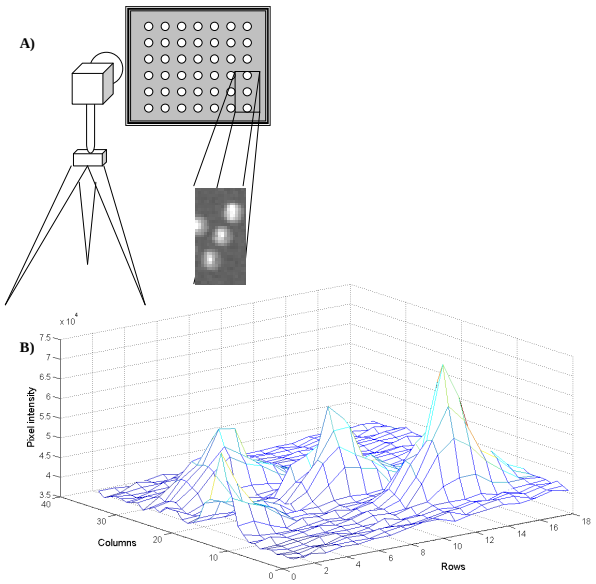

Fig. 10 A) Camera setup for image acquisition of 90 LED light spots.
B) Intensity mesh plot of three light spots.

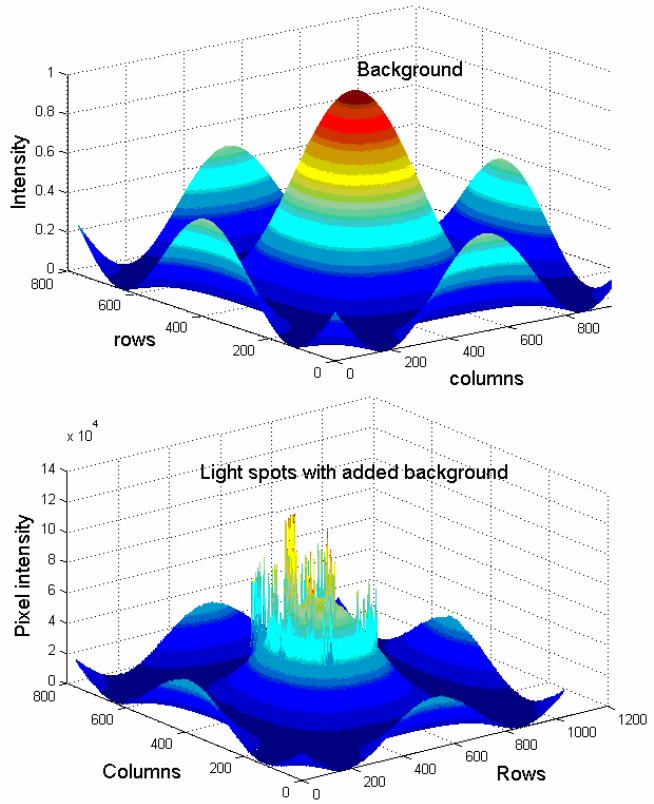

Fig. 11 Synthetic background added to original image of light spots.

A camera equipped with an optical IR band pass filter was set up in front of this board using a rigid tripod. The IR filter matches the optical wavelength of the LEDs such that the influence from visible stray light was suppressed. A series of images having the same condition for lightening and exposure were acquired. A mesh plot of the intensity for four out of 90 LEDs is shown in Fig. 10B.

B. Simulations

A simulation script was written in Matlab to read the 250 images previously acquired and stored in a data file. Algorithms 1 to 3, see Fig. 6 and Fig. 7, were all modeled using Matlab. The acquired images were then used as simulation stimuli in order to analyze the subpixel precision versus SNR for the 90 light spots under the influence of image noise.

In addition, we also wanted to efficiently verify the ability of the image segmentation to suppress the influence of a background shade. For this purpose we created a synthetic background shown in Fig. 11. This background was then added to all images to allow for a simple comparison with the results using no synthetic background. Subpixel precision versus SNR is expected to be unaffected by this synthetic background.

The plotting of subpixel precision versus SNR requires a method to calculate the SNR for each light spot within a series of images.

C. Signal-to-Noise-Ratio

First, we calculate the mean power frame *P(x,y)* from *N* number of frames,

$$P(x,y)=\frac{1}{N}\sum_{t=1}^{N} f_t^2(x,y). \tag{10}$$

A mean value frame *M(x,y)* is also calculated from the same N number of frames,

$$M(x,y)=\frac{1}{N}\sum_{t=1}^{N} f_t(x,y) \tag{11}$$

The mean power frame *P(x,y)* and the mean value frame *M(x,y)* is then used to calculate a frame of standard deviations *S(x,y)* for all pixels. This is a pixel wise measure of the temporal noise, estimated from N number of frames.

$$S(x,y)=\sqrt{P(x,y)-M^2(x,y)} \tag{12}$$

Let the mean value frame *M(x,y)* be a measure of the signal and the standard deviation *S(x,y)* a measure of the noise. Then the Signal-to-Noise-Ratio for a single light spot detected within a neighborhood Ω becomes,

$$SNR=\sqrt{\frac{\sum_{x,y\in\Omega} M_l^2(x,y)}{\sum_{x,y\in\Omega} S_l^2(x,y)}}. \tag{13}$$

The SNR was calculated at simulation according to equation (13) for all 90 light spots.

VI. Results

This section presents the simulation result from analyzing subpixel precision versus SNR for the 90 LEDs illustrated in Fig. 10A. Positions of LED light spots where calculated using Algorithm 1 to 3, described in section IV. SNR was calculated for each light spot according to equation (13).

Fig. 12 and Fig. 13 show the simulation results for Algorithm 1 and 2. Fig. 14 and Fig. 15 show the simulation results for Algorithm 3 with and without the synthetic background added.

Fig. 12 to Fig. 14 all show 90 values corresponding to the 90 LEDs in the experimental setup as illustrated in Fig. 10. In addition to the 90 values there is also a line fitted to data having least square error. The parameters for these lines are shown in Table I for all the diagrams. The first column in Table I shows the line intersection with the vertical axis at SNR=11. The second column shows the line slope.

VII. Analysis

Algorithm 2 in comparison with Algorithm 1 shows about 160 percent improvement of subpixel precision at SNR=11 and according to Table I. Algorithm 2 also has about 50 percent stronger slope.

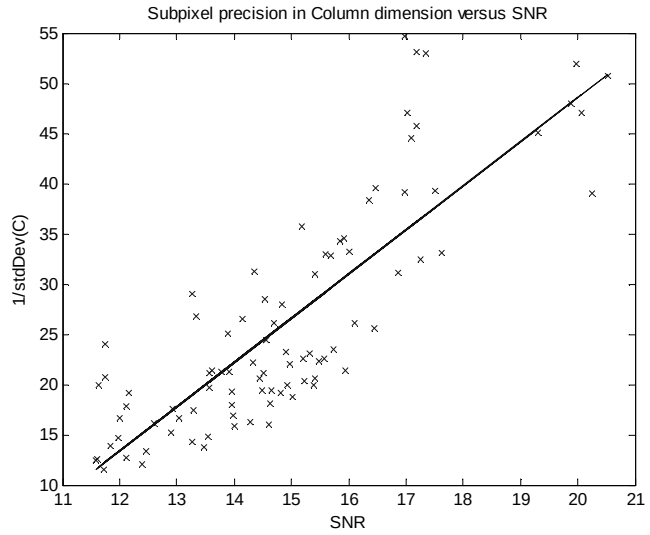

Fig. 12 Sub-pixel precision vs SNR for Algorithm 1 with data offset.

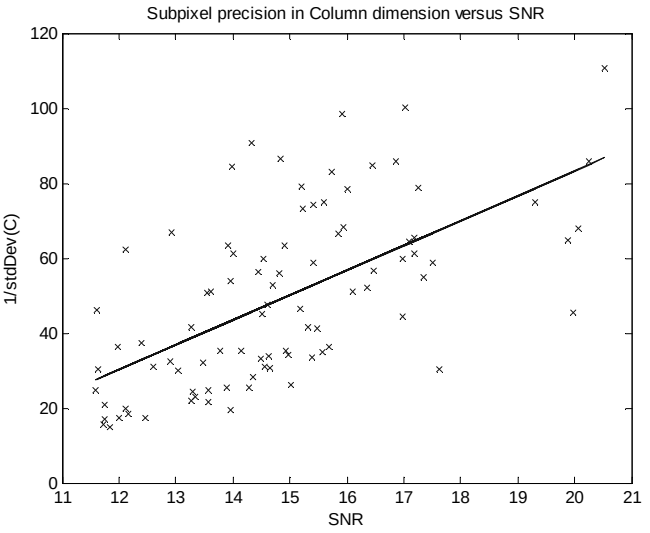

Fig. 13 Sub-pixel precision vs SNR for Algorithm2 with data offset.

This improved performance is obviously due to the changes we made to the preprocessing filter described in section III.C. However, the correlation between SNR and subpixel precision is weaker for Algorithm 2 when comparing Fig. 12 and Fig. 13. From equation (5) based on an estimator of COG variance published in [6] we expect a linear dependency of subpixel precision versus SNR for the light spots. This estimation is based on that a light spot is always centered within a neighborhood Ω. However, this can never be the case when Ω grows and shrinks as a result of image noise at segmentation. Equation (5) shows that subpixel precision is also dependent on Ω.

The only difference between Algorithm 2 and Algorithm 3 is that the latter computes COG based on squared image data. See Fig. 6 and Fig. 7. This difference obviously improves the correlation when comparing Fig. 13 and Fig. 14.

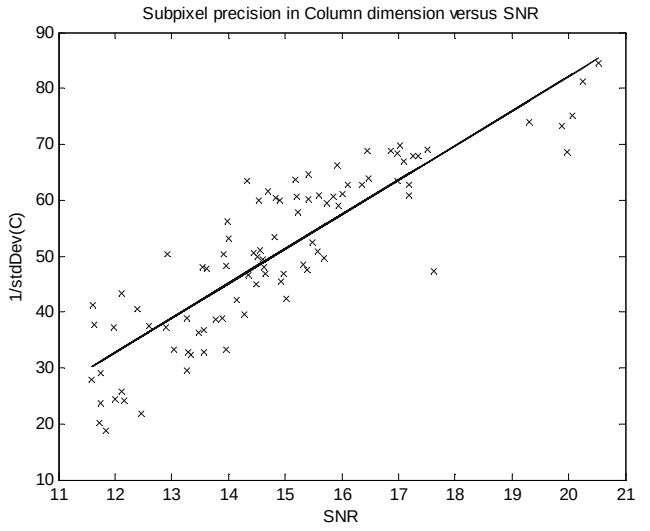

Fig. 14 Sub-pixel precision vs SNR for Algorithm 3 with data offset.

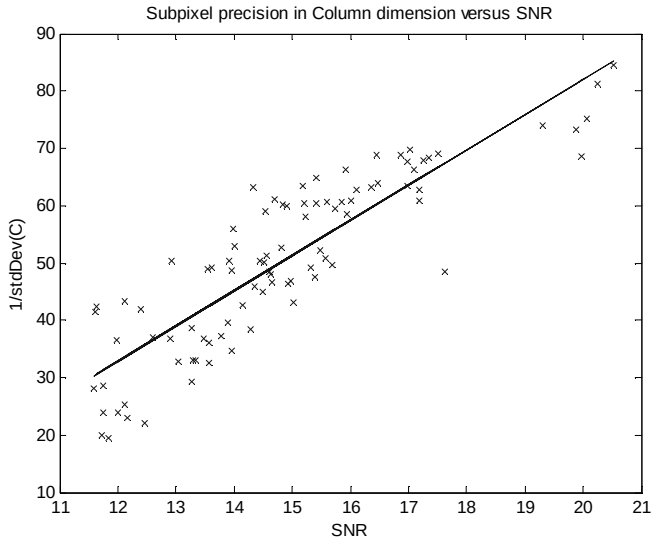

Fig. 15 Sub-pixel precision vs SNR for Alg. 3 without data offset.

TABLE I
PARAMETERS FOR LEAST SQUARE FITTED LINES

	Intersection with vertical axis at SNR=11	Slope
Algorithm 1	9.0	4.40
Algorithm 2	23.7	6.61
Algorithm 3	26.6	6.18
Algorithm 3 without synthetic background added	26.7	6.14

The improvement of subpixel precision at SNR=11 is 195 percent if compared with Algorithm 1. When making this improvement by firstly squaring data, we think it is important to also scale and truncate data for preserved fixed point precision. Otherwise the storage requirement for the data tables shown in Fig. 9 will explode in size. Remember that these data tables are used to accumulate the numerators and denominators according to equation (1). Exactly how the fixed

point precision will affect the hardware implementation and subpixel precision is beyond the scope of this paper and needs further investigation.

We can conclude that when Algorithm 3 is simulated with input images having a synthetic background added or not, it does not cause any detectable difference in subpixel precision. We take this as a proof that the segmentation really acts as dynamic and transparent to reasonable background shades.

In addition to the results generated from simulations shown in section VI, we have also analyzed and suggested hardware architectures for FPGA-based real-time computation. We strongly believe that algorithms developed for machine vision having real-time constraints must also be accompanied with a primary solution for hardware implementation. This is, because hardware resources are always limited and we think it is beneficial to have the computational platform in mind already at algorithm development.

VIII. Conclusions

Algorithm 3 has an improved method for segmentation of image components in comparison with state of the art dynamic thresholding. These improvements are as high as 195 percent for the subpixel precision at SNR=11 and 40 percent for the slope. This means that any improvements of SNR for the navigation system that we target caused by longer exposure times or more intensive illumination will pay off better in terms of improved subpixel precision when Algorithm 3 is used instead of Algorithm 1.

This analysis will be very useful when designing the navigation systems that we target. Decisions can be made on balancing exposure time, light intensity and subpixel precision for a given camera setup.

The real-time constraints for the kind of machine vision systems that we target require aggressive parallelization of the computation to be met. FPGAs are known to offer this parallelism. Therefore we also in this work, based on previous research, suggest hardware architectures suitable for computation of the developed algorithm.

Acknowledgment

The Swedish KK-foundation and Mid Sweden University is gratefully acknowledged for their financial support.

References

[1] Larsson U., Zell C., Hyyppä K., Wernersson Å.: Navigating an Articulated Vehicle and Reversing with a Trailer. Proceedings 1994 IEEE International Conference on Robotics and Automation, vol. 3, pp. 2398--2404, San Diego, USA (1994).

[2] Wolf W., Ozer C., Lv T.: Smart cameras as embedded systems. Computer, vol. 35, no. 9 (2002).

[3] Dias F., Berry F., Serot J., Marmoiton F.: Hardware, design and implementation issues on a fpga-based smart camera. Proc. First ACM/IEEE international conference on distributed smart cameras. pp 20--26, Vienna, Austria (2007).

[4] Carsten Steger, Markus Ulrich and Christian Wiedemann,Machine vision algorithms and applications, Wiley-VCH 2008.

[5] Wnuk M.: Remarks on hardware implementation of image processing algorithms. Int. journal of applied mathematics and computer science. Vol. 18, No. 1, pp105--110 (2008).

[6] H. C. van Assen, M. Egmont-Petersen, and J. H. C. Reiber, "Accurate Object Localization in Gray Level Images Using the Center of Gravity Measure: Accuracy Versus Precision, IEEE Transaction on Image Processing," Vol. 11, No.12 December 2002.

[7] R.C. Gonzales and R.E. Woods, Addison Wesley,Digital Image Processing, 2008, third edition.

[8] A. Patwardhan, Subpixel position measurement using 1D,2D and 3D centroid algorithms with emphasis on applications in confocal microscopy, Journal of Microscopy, Vol. 186,Pt 3, June 1997, pp. 246-257.

[9] Alexander Fish, Dmitry Akselrod and Orly Yadid-Pecht-Pecht, High Precision Image Centroid Computation via an Adaptive K-Winner-Take-all Circuit in Conjunction with a Dynamic Element Matching Algorithm for Star Tracking Applications, Analog Integrated Circuits and Signal Processing, 39, 251–266, 2004.

[10] G.A.W. West, & T.A. Clarke, 1990, "A survey and examination of subpixel measurement techniques.", ISPRS Int. Conf. on Close Range Photogrammetry and Machine Vision, SPIE Vol. 1395, pp 456 - 463, Sept. 3-7.

[11] Clarke, T.A. Cooper, M.A.R. & Fryer, J.G., 1993. An estimator for the random error in subpixel target location and its use in the bundle adjustment. Optical 3-D measurements techniques II, Pub. Wichmann, Karlsruhe:161-168.

[12] B.Thörnberg et al. "Bit-Width Constrained Memory Hierarchy Optimization for Real-Time Video Systems", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol26, No 4, pp 781-800, April 2007.

[13] B. Thörnberg and N. Lawal, "Real-time component labelling and feature extraction on FPGA", *Proc. of International Symposium on Signals, Circuits and Systems*, Iasi, Romania 2009,