Pipelined Image Correspondence Analysis

Gyula Max

Department of Automatization and Applied Informatics
Budapest University of Technology and Economics
P.O.Box. 91, H-1521 Budapest, Hungary
Tel.: +36-1-463-2870, fax: +36-1-463-2871, e-mail: max@aut.bme.hu

Abstract: In this paper, a parallel calculation method is presented for an FPGA architecture which takes advantage of the data and logical parallel opportunities offered by a field programmable gate array to perform the correspondance of moving objects in image frames in real time. The architecture can process at least 25 frames per second, where the image resolution is 320 x 240. This method typically used to segment moving regions in image taken from a camera by comparing each new incoming frame to a master model.

1 Introduction

The task is simply. Follow somebody's moving. Today this task is possible to perform for a computer, though a human also can do it in half a second or less. This result becomes more shocking if we know that the 'processing time' of a typical neuron is about 5 ms. It seems to be fast, but a normal PC can do two-three hundred million operations in a second, and it means that the computer is one million times faster. The answer how our slow brain can solve this task is its special architecture and particular information representation and processing. It is our belief, however, that in order to make major breakthroughs, many parts of today's computer systems' architectures, and the way of information representation need to be changed. One of the ways of the changing is to replace hardware surface. During the last most months we decided to replace our PC system to FPGA. The new architecture is built up by numerous, simple computational elements that can perform only primitive functions like addition, subtraction, but they do it really quickly. These computational elements are connected to each other like the neurons in the brain. This architecture can be much more efficient in certain tasks than the complex, classical algorithms as in spite of the fact that thousands of simple operations are done, due to its special architecture, they can be performed in a fully parallel manner that tremendously reduce the calculation time. This paper introduces a typical solution on image recognition by FPGA that is developed on the analogy of the human vision

processing. The solutions show how can be solving the image recognition imitating the biological structures and how can be making real-time image processing. The rest of the paper is organized as follows. Section 3 gives an introduction to functions of the human visual pathway by the aid of epipolar geometry and Section 4 shows how can correspondence pixels of the image. Section 5 describes how this system works under PC control. In Section 6 we show the same functions controlled by FPGA.

2 Epipolar Geometry

With two cameras arranged arbitrarily, the general epipolar geometry is shown in Figure 2. The relative position of both cameras is known and C1 and C2 point out the optical centres of each camera. The straight line connecting both optical centres is called baseline. Each point M observed by the two cameras at the same time along with the two corresponding light rays through the optical centres C1 and C2 form an epipolar plane [5].

The epipole e is the intersection of the baseline with the image plane. The epipolar line is therefore defined as a straight line g through e and m that is the intersection of the line through M and the optical centre with the respective image plane. The point M in Figure 1 is projected as m1 in the left image plane. The corresponding point in the right image therefore lies on the previous described epipolar line g.

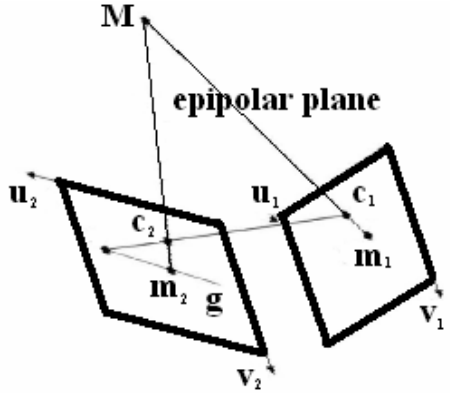

Figure 1
General Epipolar geometry

This reduces the search space from two dimensional, which would be the whole image, to one dimensional, a straight line only. A simplification of the general epipolar geometry is shown in Figure 2. Both cameras are arranged in parallel, their focal length is identical and the two retinal planes are the same. Assuming

these conditions all epipolar lines are horizontal within the retinal planes and the projected images m1 and m2 of a point M will have the same vertical coordinate. Therefore the corresponding point of m1 lies on the same horizontal line in the right image.

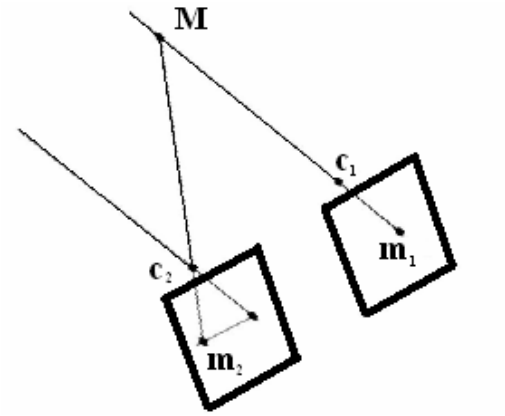

Figure 2
Stereo epipolar geometry

In order to assure the stereo epipolar geometry the rectification of both images is necessary. Therefore both cameras need to be calibrated first in order to get the camera parameters that are needed for the rectification. To avoid this long procedure as well as using two cameras we do not want to correspondence two different images. Let's take the first image as a master image. Using this method the second camera is not necessary. If we have already found what we wanted then the master image is being changed. The input pixels of our correspondence model are the master and the slave images. So in our neural model inputs come from a camera. Since the binocular cells are always orientation selective, the input of the model is not only the input image from the cameras, but also the orientation map of the image, which includes the edge orientations and magnitudes in each pixel.

3 Matching Pixels by Correspondence Analysis

Correspondence analysis tries to solve the problem of finding which pixels or objects in one image correspond to a pixels or objects in the other. This is known as the Correspondence Problem. The algorithms can be divided into feature-based and area-based, also known as region-based or intensity-based.

Area-based algorithms solve the correspondence problem for every single pixel in the image. Therefore they take values and/or intensities into account as well as a certain pixel neighbourhood. A block consisting of the middle pixel and its surrounding neighbours will then be matched to the best corresponding block in the second image. These algorithms result in dense depth maps as the depth is known for each pixel. But selecting the right block size is difficult because a small neighbourhood will lead to less correct maps but short run times whereas a large neighbourhood leads to more exact maps at the expense of long run times.

Feature-based correspondence algorithms on the other hand extract features first and then try to detect these features in the second image. These features should be unique within the images, like edges, corners, geometric figures, whole objects or part of objects. The resulting maps will be less detailed as the depth is not calculated for every pixel. But since it is much more unlikely to match a feature incorrectly because of its detailed description, feature based algorithms are less error sensitive and result in very exact depth maps.

There are a number of problems all correspondence analysis algorithms have to deal with. An object seen by one of the cameras could be occluded when seen by the other camera that has a slightly different point of view. This object will cause wrong correspondences when trying to match images. The cameras itself my cause distorted images due to lens distortion which will lead to wrong correspondences especially in the outer regions of the image.

Some more problems are caused by the objects themselves. Having lots of small objects that look alike or having a special pattern that iterates quite often makes it hard to find the matching object as there is more than one possible match. This is known as the aperture problem.

Another big problem is homogeneity. Big homogeneous regions are difficult to match when seen through a small window only. The same textures on different positions in the image will cause similar problems.

4 The Model with PC

The PC model is described in [2], [3], [4] and can be seen in Figure 4.

Our solution based on neural model and we also want to keep this model in the future. The goal of the proposed system is to adjust the camera on the centre of the image planes. The images provided by the camera will be referred to as master and slave images. The system adjusts the optical centre of the slave camera so that the same point M is projected on both of the optical centres, while the master camera is not moving at all. A window of 9 x 9 pixels is considered as a pixel surrounding, which corresponds to a set of receptive fields getting inputs from an

area of 81 pixels. On the master image the position of the 9 x 9 window is fixed to the optical centre. The final task is to find a window on the slave image that best matches the window of the master image according to all the image features taken into consideration. Not all the pixels on the slave image are taken into consideration. In stereo vision if the external and internal parameters of the cameras (such as their position, orientation, and focal length) are known, then for each point on one of the images a line can be defined on the other image which will contain the pair of the point seen on the first image. This line is referred to as the epipolar line, providing a constraint to the pixels to be taken into consideration during the matching process.

This constraint is also present in the biological system. Most of the animals can not move their eyes up and down independently. For this reason it is also supposed that in the proposed model there is no vertical rotation or translation between the master and the slave cameras, furthermore the window of sharp vision on the master image is always in the optical centre. This yields that the epipolar line on the slave camera's image plane is always horizontal and passes through the optical centre of the slave camera. As a result all possible windows of 9 x 9 pixels along the epipolar line of the slave image will be compared to the single window in the centre of the master image.

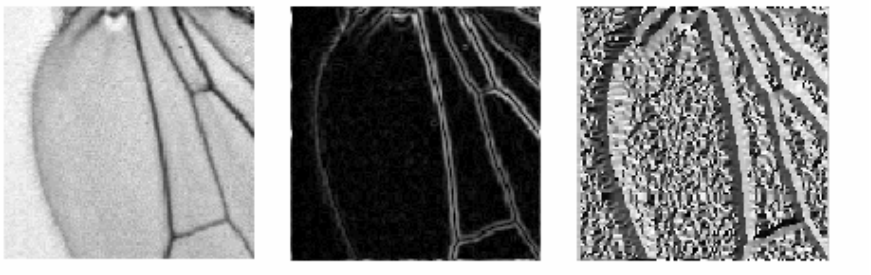

Figure 3
The intensity (left), edge magnitude (middle) and edge orientation (right) maps of an image

This implies that the image parts used as inputs to the model are the 9 x 9 square on the optical centre of the master image and an X_{max} times 9 stripe on the slave image, where X_{max} is the width of the slave image in pixels. In order to help comparison between master and slave images two auxiliary image matrixes are needed. An example is shown in Figure 3 where the same picture can be seen. The intensity (left), edge magnitude (middle) and edge orientation (right) matrixes of the same image are shown.

The camera makes images and tries to find the most correspondence image in its neighbourhood using an X_{max} = 320 x 9 pixels slave image. The result, the new camera position is transferred through the serial port of the PC which is connected to the controller of the camera. We must know that need of the memories is high

enough related to the capacity of FPGA. If we want to make a solution in FPGA somehow size of the memory must be reduced because in our FPGA the quantity of the memory is limited. We do not have such a problem in PC. The processor is fast, the memory is almost endless, but efficiency of the system is too low, and the size is too big. Figure 4 shows how this camera controlled computer system works. The explanation below separates the phases of the system into layers.

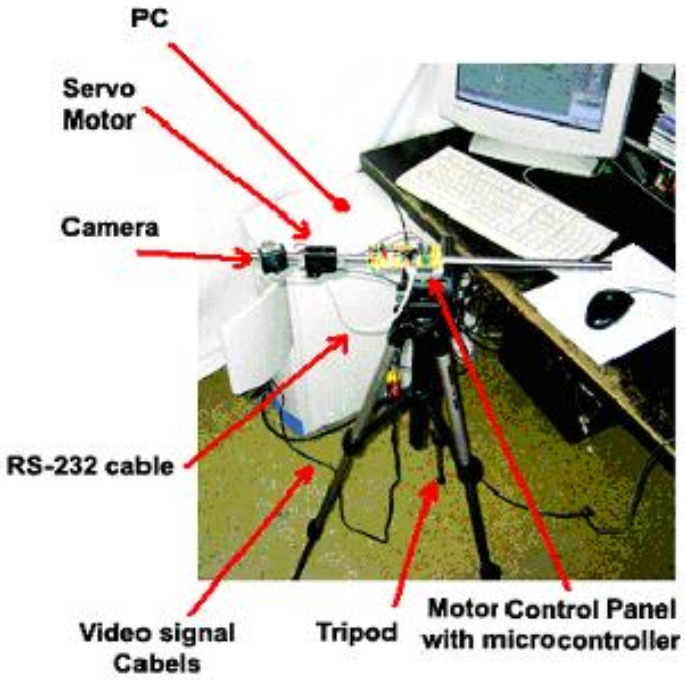

Figure 4
Camera controlled by PC

Layer I: PC input

Step 1: The camera makes an image

In the recent position camera makes an image frame as it is shown in figure 5. Because of technical parameters of the motor 4-6 image frames would be enough in every second.

Figure 5
The camera and the motor

Step 2: The image is sent to the PC

The communication between the camera and the PC is made by two FireWire ports. 8 bits greyscale pixels are transferred through this cannel. Each frame is size of 320 x 240 pixels. 25 frames are made per second.

Level II: Edge detection in PC

Step 3: Making D and O matrixes

After having arrived all information of the original image stripe to the PC, the intensity (I), the edge magnitude (D) and the edge orientation (O) matrixes can be calculated using the following methods:

$$D_x(i, j) = (-I(i-1, j-1) - 2I(i, j-1) - I(i+1, j-1) + I(i-1, j+1) + 2I(i, j+1) - I(i+1, j+1)) \quad \text{(II.)}$$

$$D_y(i, j) = (-I(i-1, j-1) - 2I(i-1, j) - I(i-1, j+1) + I(i+1, j-1) + 2I(i+1, j) - I(i+1, j+1)) \quad \text{(III.)}$$

$$D = \sqrt{D_x^2 + D_y^2} \quad \text{and} \quad \text{(IV.)}$$

$$O = arctg \frac{D_y}{D_x},$$

$$\text{where } -\frac{\pi}{2} <= O <= \frac{\pi}{2} \text{ converted to } [0:255] \quad \text{(V.)}$$

Level III: Correspondence Pixels

Step 4: Matching Pixels

Using figures of I, D and O matrixes the master image is compared to each 9 x 9 image of the slave stripe. The comparison level (t) is an edge magnitude (D) value. This system input figure is a gradients value which shows from that value we want to distinguish the edges.

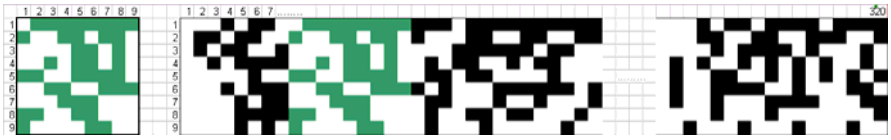

Figure 6
Matching pixels

The process of the calculation of correspondence is the following and shown in Figure 6:

To calculate the different matrix (R) of master and slave matrixes:

The edge orientation value of the same pixel of the master and a 9x9 matrix of the slave image are compared to the comparison level.

- If one of them is less than t and the other is higher than t, then for each pixel

$$If$$
$$((D_{master(i,j)} < t)\ and\ (D_{slave(i,j)} > t))\ xor\ ((D_{master(i,j)} > t)\ and\ (D_{slave(i,j)} < t))$$
$$then\ R_{i,j} = \left| D_{master(i,j)} - D_{slave(i,j)} \right| \quad \text{(VI.)}$$

- If both pixels are less than t, which means no sharp edges, the R is the absolute value of the intensity figures.

$$If\ \ ((D_{master(i,j)} < t)\ and\ (D_{slave(i,j)} < t))\ \ then\ \ \ R_{i,j} = \left| I_{master(i,j)} - I_{slave(i,j)} \right| \quad \text{(VII.)}$$

- If both pixels are greater than t, which means too sharp edges, the R is the absolute value of the orientation figures.

$$If\ \ ((D_{master(i,j)} > t)\ and\ (D_{slave(i,j)} > t))\ \ then\ \ \ R_{i,j} = \left| O_{master(i,j)} - O_{slave(i,j)} \right| \quad \text{(VIII.)}$$

The result of comparison of these two 9x9 matrixes is the average of $R_{i,j}$.

$$R = \frac{\sum R_{ij}}{81} \quad \text{(IX.)}$$

Level IV: Rotating motor

Step 5: Finding minimum value

When all existing 9x9 slave matrixes are compared to the master matrix, 312 results are calculated. Let's find which result is the minimal value of these 312 ones. This value shows the least different between the master and the slave image.

Step 6: Position of the minimum value

We must find the position of the least value and calculate the new position value of the camera.

Step 7: Communication between PC and motor

Using the slow serial cable (9600 Baud) connection between the PC and the control panel of the camera the new position must be transferred.

Step 8: Turn to the new position

Because of the mechanical part of the motor of the camera maximum 4-6 new values can be transferred to the camera in a second.

5 Tasks of FPGA

In this section tasks of the FPGA are described. Before writing about these tasks get acquainted with out FPGA.

In the former solution the computer itself is main problem [1], [2]. It takes too much time to calculate rotation. This long time is coming from the serial data process of the computer. First the computer waits for a total image. Then it calculates a new position and finally it turns camera. Recently maximum four images can be evaluated in each second. If the computer is replaced by a FPGA then the maximum number of images can be increased.

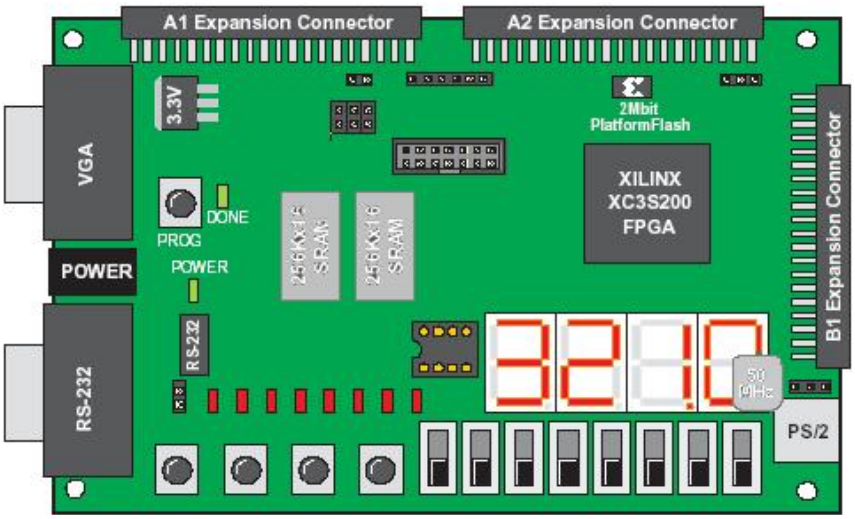

Figure 7
Xilinx Spartan3 Starter Kit with XC3S200FT256 FPGA

Spartan-3 Starter Kit board [6] was chosen included the following components and features as well as can be seen in Figure 7:

- 200,000-gate Xilinx Spartan-3 XC3S200FT256 FPGA
- 4,320 logic cell equivalents
- Twelve 18K-bit block RAMs (216K bits)
- Twelve 18x18 hardware multipliers
- Four Digital Clock Managers (DCMs) - 50 MHz Quartz
- Up to 173 user-defined I/O signals

Tasks of the FPGA are divided into four parts:

- data communication between camera and FPGA

- data pre-processing
- decision making
- controlling

Data communication between camera and FPGA

First the problem of the transmission must be solved. Camera of the former solution remained in our system. The features of the PL-A6xx cameras according to the technical reference are the following:

- The camera is equipped with a 2/3" monochrome 1280 x 1024 CMOS chip (1.3 Megapixel image resolution) with adjustable
- 8-10 bits of RGB (colour) or monochrome pixels
- Integrated infra-red (IR) filter over the image sensor
- Two FireWire connectors
- Lens mount for a standard C-mount lens (1" × 32 tpi)

Each PixeLINK camera has two FireWire connectors (ports), allowing several devices to the number of cameras that may be managed simultaneously depends on the total bandwidth and may be limited by power availability and the processing capabilities of the host computer. The FireWire bus requires that the sum of the packet sizes of the attached cameras be less than 4800. In case of 640 × 480 window size and 8 bit/monochrome pixels the total data amount is between 61.44 mega pixels/second. This figure changes into 262.144 megapixels/second in case of window size of 1280 x 1024 using 8 bit greyscale pixels. As it was shown in the former chapter not all pixels of the image are necessary for us. If 25 frames come in each second then we have got approximately 40 milliseconds to evaluate an image. It means 2 million ticks in our FPGA which works with 50 MHz. During this time the requested part of the image must be got, the different matrix (R) must be set up and the output signal of the motor of the camera must be calculate. Our situation is not so bad. Taking into consideration the physical parameters of the camera we can see in Figure 8 that the camera can be rotated only 4-6 times per second.

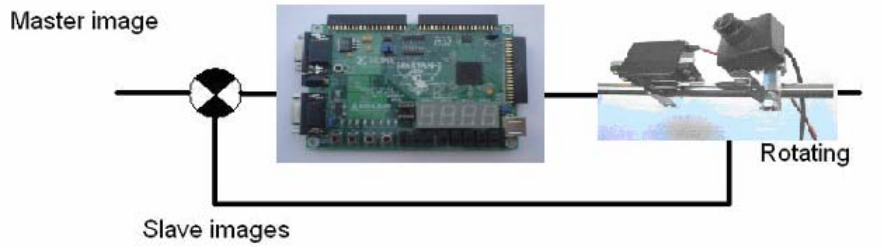

Figure 8
Camera controlled by FPGA

A new date receiver program is set up in the FPGA which can receive values of the pixels coming from the camera. Its highest speed is 192 MHz. Using this speed almost all frame size can be received by our FPGA.

The capacity of the FPGA was the second problem. We need 72 kbit memories for the matrixes. If we could use this amount no space would remain for the controller. Real problems are awake after getting the master matrixes. One picture coming trough the incoming channel takes 320 x 9 pixels, namely 23040 bits. If I, D and O pixels of a picture are stored, no free space remains for the further calculations in the FPGA. This result explains why the serial data processing was chosen. Every incoming three bytes value gives a piece of result in our result calculation. A special shifting calculation method was stood up to be able to avoid the standard, neural based multidimensional calculation method and we must use outer memory with maximum speed of 100 MHz.

In this case we have space for the controller but not enough for the additional mathematical functions. As we can see the weakness of Verilog is the multidimensional matrixes. The upper range of a multidimensional matrix is two. Two dimensional matrixes can be declared, but a three dimensional one can not be. Since a two dimensional 8 bit matrix is a three dimensional matrix in Verilog, this matrix can not be declared for our FPGA. To solve this problem a simply linearization method was used for transforming the master matrixes.

The third problem is coming from the calculation method. After having arrived matrix I, D and O matrixes must be set up as soon as possible. Circuits of our FPGA do not contain higher mathematical elements. During our first trial a simply divider occupied almost 20% of the resources of the FPGA.

After received master matrixes of the recent pictures the serial calculation method is started. The incoming pixels are compared to the adequate master pixels. In this section recent pixel means knowing recent pixel of I, D and O. In order to do this with the recent pixel we do not have to know any other pixel. We have to know the master pixels and the recent pixel (MD, MI, MO, I, D and O). Each appearance of the recent pixel is calculated for. It gives nine results which are stored in nine different variables. After each incoming 81 recent pixels the result is sent to the decision maker block. After 2880 recent pixels one turn of the calculation is finished and a new position of the camera is fixed.

The bottleneck of our neural is the ‘huge’ memory needs. In order to follow the neural method all incoming information of a picture must be saved. To do this the minimal memory need is 320x9+81 bytes, namely 23688 bits. Some inbuilt memory of the FPGA can be used, but reading and writing processes take too much time, if we want to use memories organized by bytes and not bits. Another problem in case of memory organized by bytes is the massive additional hardware using bytes instead of bits. To put these memories into the FPGA, at least a device with 1 million gates can be used. During our experiment device with 200

thousands can be used / reached. Unfortunately this size is small enough to solve the total process.

In the first step collects the necessary figures only. In this case these are the master pixels. Parallel reading a new recent pixel, the calculation part of our system calculates particularities of the result vector. For this reason this calculation method is very fast, because after having read the last recent pixel we need only a few clock signals to produce the new position of the camera. Easy to recognize that needs of the elements of mathematics are great enough. The minimal functions besides the existing ones are the following:

- Multiplication, division
- Square root
- Arc tangent

As it was described above the higher is the resolution the bigger is the throughput over the communication channel. Beside some resolutions and transfer speeds the maximum numbers of transferable frames are shown in Table 1.

	Pixels/frame		
Transfer speed	**320 x 240**	**640 x 480**	**1280 x 1024**
24 Mbaud	**39**	9	2
25 Mbaud	**40**	10	2
48 Mbaud	**78**	*19*	4
50 Mbaud	**81**	*20*	4
96 Mbaud	**156**	**39**	9
192 Mbaud	**312**	**78**	18

Table 1
Number of frames/second using different transfer speeds and frame sizes

Since the maximum clock speed of our FPGA panel is 200 MHz table 1 shows that throughput of VGA size frames are available on our channel if at least 25 frames are wanted to be evaluated in every second. The bold figures show the frame numbers/second over 25 frames per seconds. If no any change on clock speed of the FPGA that is 50 MHz clock is used than 81 frames could be transferred in a second. Our camera makes 25 frames per second. 20 frames/second could also be good. This value is the theoretical frame rate, in practice if the image is acquired in the form of a video stream from the camera, 15 frames per second can be reached. If more than one camera is attached to the same interface, the frame rate drops significantly, since the controller can not accommodate the data flow of cameras. In case of 10 bits/pixel the numbers of the frames are decreased by 20%.

Data pre-processing

One of the main tasks of the FPGA is the data pre-processing. Using matrixes of intensity (I) i.e. incoming data trough the communication channel FPGA has to calculate the edge magnitude (D) and the edge orientation (O) matrixes.

Pixels are coming trough the communication channel with 192 Mbaud. Pixels are sent by the camera row by row. A frame time is about 3.2 milliseconds.

Calculation can not be started after the very first pixels. During transferring the first row of the frame the zeroth line is fulfilled by the figures of the first row. It is necessary because of the Sobel filter. When the second element of the second row has already arrived the calculation starts.

To calculate the gradient and orientation values of pixels the Sobel filter was chosen. D and O matrixes are prepared by using the calculation method written in (II.), (III.), (IV.), (V.), in Section 5. Form of the filters are shown in Figure 9.

Direction: X

-1	-1	-1
0	0	0
1	1	1

Direction: Y

-1	0	1
-1	0	1
-1	0	1

Figure 9
The Sober filter is used the calculate D and O matrixes

In order to use Sober filter additional cells are prepared around the matrixes as it is shown in Figure 10. For example the most upper, the zeros line contains the same value as they are in the first line. The corners may contain either zeros or the closest values.

-1	-1	-1	p13	p14									p1320	p1320
0	0	0	p13	p14	…	…	…	…	…				p1320	p1320
1	1	1	p23	P24									:	
p31	p31	p32	p33	p34									:	
	:												:	
					…	…	…	…	…					

Figure 10
Incoming frame and additional cells to calculate D and O matrixes by using Sobel filter

To make the calculation method some mathematical operation like multiplication, division, square root and arc tangent are necessary. These methods are not difficult but we must load them to the FPGA. So we must choose the smallest size solutions. CORDIC (Coordinate Rotation Digital Computer) method was chosen to calculate figures. CORDIC methods describe essentially the same algorithm that with suitably chosen inputs can be used to calculate a whole range of scientific functions including; sin, coos, tan, arc tan, arc sin, arc coos, sin, cosh, tan, arc tan, log, exp, square root and even multiply and divide. CORDIC is a method for computing elementary functions using minimal hardware such as shifts, adds, subs and compares. CORDIC works by rotating the coordinate system through constant angles until the angle is reduces to zero. The angle offsets are selected such that the operations on X and Y are only shifts and adds.

Worst case calculation

Considering that camera sends one pixels in 10 bits as well as the field indicated for us is at the bottom of the incoming frame.

In this case an incoming frame contains 240 x 320 = 76800 pixels in 768 kbits. The first relevant information comes by the 232nd line. Camera sends 25 frames in a second. This is ≈20Mhz transfer speed. It takes 38.5 msec. The calculation can start when the second pixel of the 233rd is arrived in the 38.67th msec. The calculation must be finished less than 40 msec. Using our normal (50 MHz) frequency it means about 1.9 million clock signals.

Considering that each step is done step by step and not parallel. During creation of D and O matrixes the following procedures must be done by Table 2:

Task	**Result of operation**	**Number of clock signals**
Read adequate pixels from memory	$[I_{1,1};I_{3,3}]$	18
Add up adequate pixels of Sober filter	D_x	1
Add up adequate pixels of Sobel filter	D_y	1
Add up D_x, D_y	$D_x + D_y$	1
Second power of D_x	$D_x * D_x$	16
Second power of D_y	$D_y * D_y$	16
Square root of $D_{x*} D_x + D_{y*}D_y$	D	16
Divide D_y by D_x	D_y / D_x	16
Arc tangent of D_y / D_x	$arctg(D_y / D_x)$	16
Convert angle to [0,255]	O	16
Save result after each step		7
Total time for one pixel		**124**

Table 2
Serial worst case calculation

The same tasks are shown by using parallel calculation method in the Table 3.

Task	**Result of operation**	**Number of clock signals**
Read adequate pixels from memory	[I1,1;I3,3]	18
Add up adequate pixels of Sobel filter	D_x , D_y	1
Save result	D_x , D_y	1
Sum of second power of D_x and D_y **and** division D_y by D_x	D_* D D_y / D_x	16
Save result	D_* D and D_y / D_x	1
Square root of $D_{x*} D_x + D_{y*}D_y$ **and** arc tangent of D_y / D_x	D O’	16
Save result	D and O’	1
Convert angle (O’) to [0,255]	O	16
Save result	O	1
Total time for one pixel		**71**

Table 3
Serial worst case calculation

For our 9 x 320 pixel window the total time of this procedure is 357120 clock signal in case of serial method and 204480 clock signal in case of parallel method. During this time transmission of recent frame has finished and the new frame has started. If the transfer speed is 20 MHz between the camera and FPGA as well as our clock is used by 50 MHz the 37th line of the next frame is arriving in worst case. So if the master frame is changed than the distance between the former and the new master frame must be more than 50 lines. Over 50 lines FPGA has enough time to finish calculation of D, and O matrixes.

The 1 MByte (2 x 256k x 16 bits) memory of FPGA can contain calculation results of two frames of 240 x 320 pixels or one VGA frame. For bigger frames just the requested window size and its neighbourhood are saved.

When the matrixes are made we have to correspondence the master and slave frames. The calculation of minimum space of the vector R takes calculation of 312 elements. The index of least value of vector R gives the new position of the motor of the camera. Since this calculation can start when the first D,I,O values are made and each time we have to save only the index in vector and the value of the least value this procedure is not really longer than the calculation of D,I,O matrixes.

Conclusion

In this paper an FPGA architecture was presented for pipelined parallel calculation which takes advantage of the data and logical parallel opportunities offered by a field programmable gate array to perform the correspondence of moving objects in video frames in real time. Because of frame size the transmission rate between

camera and FPGA was about 20 MHz, the system is able to run at 50 MHz and process a frame size of 320 x 240 safely. However, frequency of the transmission is only 20 MHz, the system is able to process frames of 320 x 240 pixels per frame in real time, that is, 25 frames per second. Limitation of the calculation method gains by size of the frames. The motor of the camera is the slowest part in our process. Camera can be moved 4-6 times in a second. If the frame size is increased we may eliminate every second or second and third frames to be enough time to fix new position of motor of the camera.

References

[1] Gyula Max: *When a Slow Device is Faster than a High Speed PC in the Intelligence Space* Proceedings of the 6th International Symposium of Hungarian Researchers on Computational Intelligence, pp. 625-638, Budapest, 2005

[2] Gyula Max, Péter Szemes: *Limits of a Distributed Intelligent Networked Device in the Intelligence Space* Proceedings of the 5th International Symposium of Hungarian Researchers on Computational Intelligence, pp. 173-182, Budapest, 2004

[3] B. Reskó, P. Baranyi, P. Korondi, P. Szemes, H. Hashimoto: *Stereo Matching in Robot Vision by Artificial Neural Networks* In International Conference on Industrial Technologies, Maribor, Slovenia, 2003

[4] Barna Reskó, Péter Baranyi: *Stereo Camera Alignment based on Disparity Selective Cells in the Visual Cortex*. I In Proceedings of IEEE 3rd International Conference on Computational Cybernetics (ICCC 2005), pp. 285-290, Mauritius, April 2005

[5] Annika Kuhl: *Comparison of Stereo Matching Algorithms for Mobile Robots* Ilmenau, Germany, 2005

[6] Xilinx. *Spartan-3 Starter Kit Board User Guide UG130 (v1.1)*, Xilinx, 2005