Super-Resolution for Traditional and Omnidirectional Image Sequences

Attila Nagy, Zoltán Vámossy

Institute of Software Technology, John von Neumann Faculty of Informatics, Budapest Tech, Bécsi út 96/B, H-1034 Budapest, Hungary
attila.nagy.hu@gmail.com, zoltan.vamossy@nik.bmf.hu

Abstract: This article presents a simple method on how to implement a super-resolution based video enhancement technique in .NET using the functions of the OpenCV library. First, we outline the goal of this project and after that, a short review of the steps of super-resolution technique is given. As a part of the discussion about the implementation itself, the general design aspects are detailed in short. Then, the different optical flow algorithms are analyzed and the super-resolution calculation of omnidirectional image sequences is discussed. After all that, the achieved results can be seen and finally, a short general conclusion can be read. This paper is a revision of our previous work [1]. In this edition, we focus on the super-resolution of omnidirectional image sequences rather than the technological issues that were discussed in our previous article. Further information about the implementation and wrapper development can be found in [1 and 12].

Keywords: super-resolution, optical flow, omnidirectional vision, OpenCV wrapper

1 Introduction

Our purpose is to develop an application which is able to process an input video file and create a new stream from it that has higher resolution than the original one. The goal is that more detail should be seen on the processed video than on the original. A program like this would be useful for many tasks. For example, it could improve the poor image quality and resolution of videos created by security cameras or mobile phones. Generally, the program is capable of the quality enhancement in case of low resolution and/or if the input has noticeable noise on the frames. These capabilities exist because of the features of the super-resolution technique which is presented in section II. To get an impression about the quality enhancement that can be achieved using our system see Fig. 1. During the development of the system, we lean upon the functions of the quite effective OpenCV image processing library [2]. Hereby, the comparability of the implemented methods is ensured as the next paragraph outlines it.

Reconstruction based super-resolution technique consist of several steps. One of these is the optical flow calculation. The purpose of this step is to find out how did the pixels of the frames move compared to the previous frame of the video. In the OpenCV library, numerous optical flow methods are implemented. Our program makes it possible to use any of them for motion estimation. This way it is possible to compare which optical flow algorithm produces the best result for a given situation.

Figure 1
Enlarged frame of the original video – as later, always to the left – using bicubic interpolation. The result image – as later, always to the right – generated by our program. The quality enhances mainly at the engine hood and at the roof

2 Super-Resolution

Two main approaches exist for super-resolution. One is the reconstruction-based approach [3] and the second is the so-called learning algorithm based super-resolution [4]. These approaches are substantially different. In case of the learning algorithm based technique, the problem is generally solved by neural networks. Learning based super-resolution is able to work even if only one static image is given but a database of sample images is also necessary. From the database, the algorithm can learn how to enlarge certain parts of the input image. In nutshell, this method works like this: a database of images is used to find out how a certain part looks like in low and high resolution. After the learning step, the algorithm processes the image that needs to be enhanced and every portion of the input image is replaced with its high-resolution version. Hereinafter the reconstruction-based approach is discussed and the term ‘super-resolution’ refers only to that.

The point of reconstruction-based super-resolution is that an input video stream is given and the algorithm processes each frame sequentially. For every frame, an optical flow vector field has to be calculated. This vector field describes the displacement of the pixels of the frames. Since each frame is known and information about the displacement of pixels can be calculated, it is possible to use this information to approximate a frame from its neighbours. The pixels of any frame i can be transformed using their offset vectors to approximate an arbitrary

frame *j*. For example, if the pixels of the frame *n* are translated using the optical flow vectors, a new image can be got, which approximates the frame *n*+1. Using these new images, extra information can be gained which makes it possible to enhance the resolution and content wealthiness of the frames. Of course, all this is only theoretical because it is possible, for example, that a car appears on the frame 20, but not on the frame 500 by the reason of movement of the camera. This is one of the reasons, why this super-resolution method can work only between certain limits. However, we assume that the consecutive frames are almost the same, namely the movement speed of the camera is less than a particular limit.

Optical flow calculation has several other pitfalls such as the appearance of transparent and reflective surfaces, problems related to the characteristics of the surface sample (like repetition, homogeneous color fragments, etc.). These pitfalls limit the effectiveness of the video enhancement but under certain conditions quite good result can be achieved. In the next subsections, the main steps of the super-resolution are reviewed to make the later demonstration of our system easier. These main steps are considered as in [3]. A schematic figure of the steps can be seen on Fig. 2.

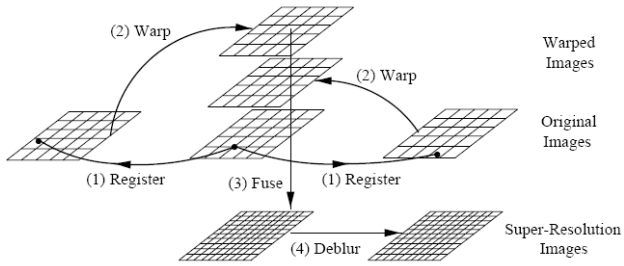

Figure 2
Schematic figure about the main steps of super-resolution [3]

2.1 Registration

The purpose of this first step is to enlarge the resolution of the input video and execute optical flow calculation on it. For the enlarging, conventional subpixeling techniques can be used such as nearest neighbour algorithm, bilinear or bicubic interpolation. In practice, only bicubic interpolation is appropriate since the other two algorithms are not producing such fine result and therefore they are suitable only for testing and comparing. Moreover, several special subpixeling algorithms were designed specifically for super-resolution: BLAZE, SIAD [5] and LAZA [6].

When the source frame is enlarged, optical flow calculation has to be performed on it. This is necessary because using the calculated offset vectors of each pixel; the given frame can be transformed to any other. For instance, to retrieve extra

information for the frame *n*, several other frames are needed to be transformed to approximate it. OpenCV implements Lucas-Kanade, pyramid based Lucas-Kanade, Horn-Schunck, and block matching optical flow algorithms. In our system, any of these can be chosen for optical flow calculation. The quality of the outcome is highly affected by the selected method and its parameters.

2.2 Warping

When the enlarged version of the input frame is created and the offset vector for each pixel is calculated, these can be used to perform a per-pixel geometrical transformation (warping) on the subpixeled image. To do this, the positions of each pixel have to be translated with the appropriate optical flow vector or with its inverse. Thus, a new image *m* is generated which is almost the same as the frame *n*+1, but small differences can occur. The extra information is provided by these differences, and eventually they ensure that new details became visible.

Naturally, there are no constraints on how many consecutive frames can be used for warping. For example, using the frames *n*-2, *n*-1, *n*+1, *n*+2 and perform the transformation on them, multiple images can be mapped to approximate the frame *n*. The number of frames used by this transformation also highly affects the quality of the result image created by this super-resolution algorithm. If the camera or the objects in the video move slowly, more frames can be used since this can improve the overall quality of the generated video. Otherwise, it is much better to use small number of frames for this operation to avoid blurring and distortion. Those side effects come from the imprecision and the errors of optical flow estimation. The farther the frame from the frame *n*, the errors became larger because of the additional errors of the optical flow calculation. Any number of input frames we use for this step, the output will be *k* pieces of images which illustrate roughly the same instant with small differences.

2.3 Fusion

The last major step of super-resolution is the fusion. The goal of this is to produce a single output frame from the warped *k* input images. Numerous methods exist to perform this kind of merging on a collection of input images. The most trivial method is the simple averaging. In that method, every pixel's intensity at a given coordinate of the warped frames is averaged and the result is written into the output image at the exact same coordinate. Besides, many other techniques exist for fusion such as median and other more sophisticated algorithm as they are summarized in [3]. Since averaging produces relatively good result according to its simplicity, we use that in our implementation.

2.4 Deblurring

In the last stage of the algorithm, post-processing methods can be performed on the result of the fusion. This will generate the output frame with the enhanced quality. In [3], post-processing/deblurring step is defined as an option. The aim is to achieve additional quality enhancement on the output image of fusion. To obtain good enough result with deblurring, the authors propose a deconvolution technique. That is based on the idea of reversing the image distortion which leaded to the blurry image. For deconvolution, every parameter of the camera is needed to be modeled including: the aperture, the focus distance, the lenses, other geometrical features, and the refraction characteristics. These parameters are not available when the task is to create a general-purpose solution and not a solution that works only with the images of a specific camera. Therefore, we implemented other more common techniques like unsharp masking and Laplacian sharpening. However, it is much easier to avoid the formation of blur in the subpixeling or fusion stage.

3 Theoretical Background of Implementation

The basic concept of the design and implementation was to create a relatively efficient program with an easy to read and clear source structure.

For implementation, Microsoft Visual C# 2005 and .NET 2.0 are been used, but the using of these inhibit the utilization of the Intel OpenCV library, because in .NET environment it is not possible to use directly an unmanaged library like OpenCV. But, if an intermediate layer (a wrapper) is placed between the .NET application and the unmanaged library, it is possible to use OpenCV indirectly. Because of this, a wrapper is necessary to bridge the differences between the working of managed and unmanaged code. Some wrappers already exist for OpenCV ie. [6, 7] but these were not good for us. Many of them are obsolete or buggy, poorly documented, examples are not available for them, or simply they do not make possible the usage of certain functions what we needed. Because of these, we decided to develop a new .NET wrapper for OpenCV. Further information about wrapper development can be read in [8, 9].

During the designing of our program, object oriented paradigm was fallowed and we aspired to create a program that can be easily extended with new algorithms. Extensibility is present at each step of super-resolution; new techniques can be implemented and tested easily.

To achieve the requested extensibility, abstract base classes were defined for each step of super-resolution. Specific algorithms of subpixeling, optical flow calculation, warping, and fusion can be chosen arbitrarily. Each base class

determinates the inputs and outputs for a given step and this way the implementation is completely irrelevant outside of a specific class. Beside the base classes, there is a main control class too. It coordinates the steps of super-resolution and the working of the entire system. Specific classes that implement a given step of super-resolution can get the values of general-purpose parameters from this control class to do their job.

First, the class of subpixeling does its task and enlarges the input frame. After this, the optical flow calculator class does its job. Optical flow is estimated between the current and the previous frame, and the result is stored in a queue. This storage is useful otherwise the algorithm should have to calculate the vectors for the same frames more times that is obviously unnecessary. If in iteration *i*, the flow vectors are calculated between frames *n* and *n*+1 then in the next iteration the same optical flow vectors would be calculated. This come from sequential processing. In iteration *i*+1 the frame *n* is replaced with *n*+1, and *n*-1 frame replaced with *n*, so eventually vectors are needed to be calculated between *n* and *n*+1 again since the algorithm uses several frame around *n* (i.e. *n*-2..*n*+2). Similar optimization can be performed by the storing of warped frames but in that case, only the frames before *n* are needed to be stored. After the calculation of flow vectors, warping is executed on the subpixeled frames to approximate frame *n*. In the next stage, the class of fusion merges the previously warped frames. Our implementation uses per-pixel averaging to achieve this. Finally, the output of fusion can be post-processed to increase the sharpness of the result. The main steps of this super-resolution algorithm are illustrated on Fig. 3.

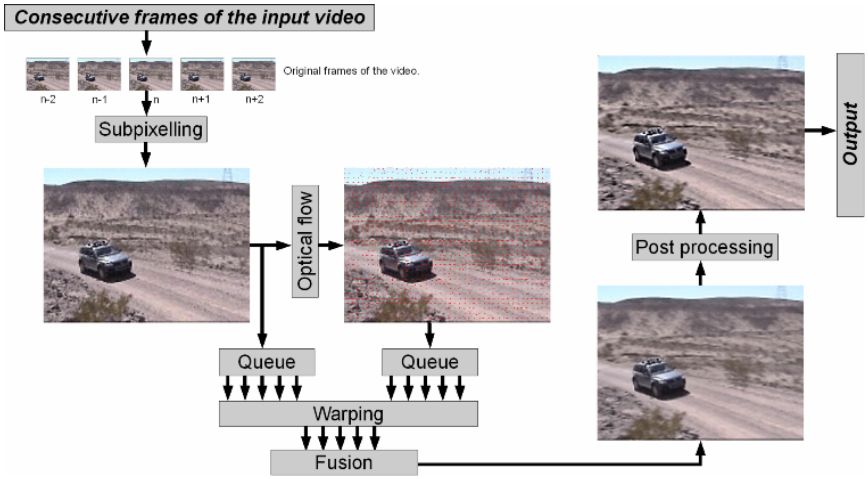

Figure 3
Main steps of the implemented super-resolution algorithm

4 Review of the Optical Flow Methods

In OpenCV, four optical flow algorithms are implemented. This section gives a short review about the general features of these and how they affect the output of the super-resolution algorithm.

Each method is differential optical flow estimation which takes two sequential frames as input. *X* and *Y* components of the estimated offset vectors are calculated into separate floating-point images[1]. Unfortunately, optical flow calculation is a problem that cannot be solved squarely since equations contain too many unknowns. Anyway, to solve the calculation problem, some constraints and conditions are needed to be defined to have more equations. The various optical flow algorithms use different constraints and conditions. Because of these, there is no perfect algorithm exists. The task and the input determinate which technique gives the best result in a certain situation.

4.1 Lucas-Kanade Optical Flow Method

The initial constraint of this method is that the optical flow has to be constant in a small window. First, both *X* and *Y* direction of partial derivative needed to be calculated and then a Gaussian filter is performed on the input images to reduce the noise and get finer vectors.

One of the features of this method is that the offset vectors are accurate mainly at the borders of the moving regions. The vectors are calculated locally and because of this, they change quickly between consecutive frames. A big advantage is that noise does not affect the computed vectors so much, and the offset vectors are estimated for each pixel. In super-resolution, this is necessary, since later in the warping step; every pixel has to transform using its offset vector. As we shall see later, not all optical flow algorithms calculate the vectors for each pixel. For those algorithms, we have to ensure that the vectors are known for every pixel. The simplest way to do this is to enlarge the floating-point vector images to the same size as the subpixeled frames.

According to our tests, this function is not the best to calculate optical flow for super-resolution since the vectors are accurate only at the borders of homogeneous regions and only small offset vectors can be calculated. These features can lead to the blurring of the ‘enhanced’ image.

[1] An image with a single channel of 32-bit floating point values.

4.2 Lucas-Kanade Method with Gaussian Pyramids

In this case, optical flow is calculated on separate levels of Gaussian pyramids which are built from the input images. The advantage of this is that first, the function works only on the small version of the images, and estimates the offsets from those. If the error of the calculated vectors is too large, the algorithm processes finer levels of the Gaussian pyramid. When the result of processing of the smaller levels is quite accurate, it is not necessary to continue the algorithm on more detailed levels. This speeds up the calculation, since generally offsets can be estimated quite accurately at the small levels. Therefore, the technique needs to process much less pixels. Another feature is that larger offset vectors can appear. This eliminates one of the disadvantages of the basic Lucas-Kanade technique.

The larger offset vectors can have bigger errors. The big errors in the optical flow can cause raindrop-like distortions on the result of the super-resolution as Fig. 4 shows. The reason for this is that, a very distant random image part is mapped to a small portion of the image because of the large and bad offset vectors.

Figure 4

Raindrop-like effect on the output of super-resolution when using the pyramid based Lucas-Kanade technique. In this sample, bad parameters were passed for the function, so the effect is more noticeable

Opposite to the previously presented method, this does not estimate the vectors for each pixel but only for a given set of ‘feature points’. There are several algorithms for finding feature points, but as these points are placed randomly across the image it would be hard to find out the offsets of all pixels. To solve this, coordinates of a regular grid are used to estimate the offsets. For example, every fifth or tenth pixel coordinates are appropriate to be ‘feature points’. This way, only the resizing of X and Y offset images is needed to be performed later.

This optical flow technique is useful where the following of certain feature points is sufficient and large pixel offsets can occur between sequential frames. For super-resolution, this method does not provide good enough result because of the possible large errors of the offset vectors and hence because of the raindrop-like effect.

4.3 Horn-Schunck Optical Flow Method

Generally, our super-resolution system provides the best quality when Horn-Schunck optical flow method was used. It works similarly like the basic Lucas-Kanade algorithm but defines a global constraint for smoothness to solve the optical flow calculation. This means that it is necessary for the adjacent offset vectors to be similar with each other. By this, the technique is able to estimate the inside of homogeneous regions accurately. Therefore, while Lucas-Kanade algorithm can only estimate accurately the vectors at the borders of moving regions, Horn-Schunck algorithm generates accurate vectors inside those homogeneous areas too. To achieve this, it uses the calculated vectors at the borders of moving areas for the rest of the homogeneous region. Since the vectors at these borders influence the offset values of a relatively big region, it is crucial for the border vectors to be accurate. As this cannot be guaranteed, the technique is more sensitive to noise. Even so, this algorithm generates the best result for super-resolution, because of the handling of homogeneous regions. As result, the vector field is more accurate, less blurring is evolved at the homogeneous regions as warping can use more correct optical flow vectors.

This technique is well-scalable through a termination criteria parameter that can be passed to describe how accurate estimation should be done for the vectors.

4.4 Block Matching Optical Flow Method

The only technique that has not been discussed yet is the block matching optical flow algorithm. It works by finding similar blocks on the consecutive frames where the intensities of the pixels are almost identical. The shifts of these blocks give the offset vectors of optical flow. Because of this, motion vectors are not calculated for every pixel but only for the blocks. This kind of block-matching technique is originally used for video compression to minimize the redundancy of the video frames and hence make the size of the video smaller.

The size of the vector field is determined by the block size, and after performing the optical flow estimation, resizing of the vector field is necessary. If the size of the original input image is O and the block size is B, then the resolution of the vector field is O / B.

Block matching optical flow is the second best algorithm for our super-resolution program. Offset vectors change smoothly amongst the frames, and the vectors are quite accurate at homogeneous regions too. So blurring is not significant, however sometimes false offset vectors are estimated at the edges of the images when wrong block size was used as Fig. 5 shows.

Figure 5
Wrong block sizes can cause the false estimation of flow vectors. This can lead to blurry and distorted output image

5 Super-Resolution of Omnidirectional Image Sequences

Images taken by traditional imaging optics show only a narrow subregion of the environment. There exist, however, techniques that provide full image of the 360-degree-space using special optical methods. One such method is the omnidirectional vision sensor where a spherical mirror is put in front of the CCD camera. PAL optics [11] is another device that transforms the surrounding environment to an annulus. The main advantage of omnidirectional imaging is that from the image provided by this optics the direction and angle of the object can be directly computed. That is why these sensors are often used in mobile robotics. However, one drawback of these image sequences that the information about the whole surroundings of the robot is transformed to a ring shaped image and consequently the resolution is small. The goal of this project's part to examine the developed super-resolution system using omnidirectional image sequences.

The robot mounting the omnidirectional imaging system moves on a planar ground floor as Fig. 6 shows. The optical axis of the camera and the mirror is perpendicular to the ground floor, lies in the direction of the robot gravity axis and the optical system is downward-looking. The camera on the robot captures a sequence of images. During the tests, the robot moves as a result of the rotation around its gravity axis and the translation.

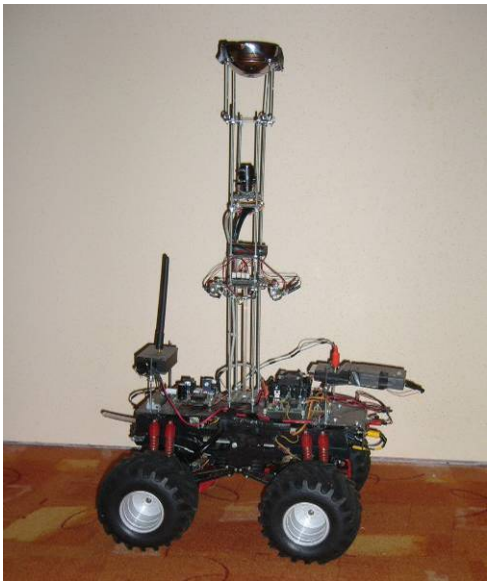

Figure 6
The mobile robot with the mounted omnidirectional imaging tool

For small motions, the dense tracking techniques, which associate a velocity with every pixel in the image (Horn-Schunck method and block matching), gives smoothed optical flow (Fig. 7) and better enhancement (Fig. 8) than the Lucas-Kanade method. The latter supposes the flow locally constant and even if it works well for traditional images, this assumption is less appropriate in case of omnidirectional image sequences because it can cause the output image to be distorted.

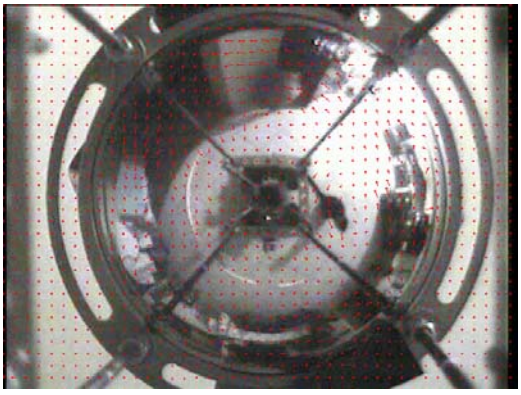

Figure 7
Optical flow with rotation around the optical axis calculated with Horn-Schunck method (The cat, to the right from the robot, near the center of the image moves in the opposite direction.)

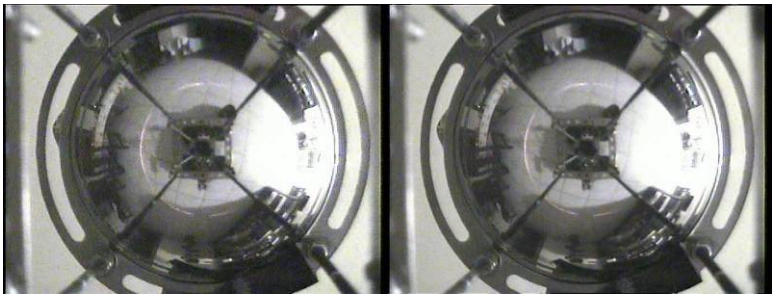

Figure 8
Bicubic interpolation of an omnidirectional video frame (left) and the enhanced frame (right) using Horn-Schunck optical flow algorithm

6 Achieved Results

As in the previous section can be read, Horn-Schunck optical flow method produces the best result for our super-resolution implementation and for the test videos. Besides, bicubic interpolation is used for subpixeling, and the *cvRemap* function of the OpenCV for warping. *cvRemap* maps each pixel to a given coordinate that is exactly the thing what is needed. For fusion only the simple average of the warped images are used. On the fusioned image other post-processing operations can be executed. Our system implements unsharp masking and Laplacian sharpening. The result of post-processing is the output of the system. Figs. 9 and 10 shows sample images which were generated by our program. Generally, the outputs of the program have much less noise than the inputs, because many frames are averaged during the fusion step. Furthermore, the result is much smoother and aliasing is less noticeable as in Fig. 11 can be seen. Motion blur and other distortions can occur when there is a lot of changing on the frames like in case of Fig. 11.

Figure 9
Bicubic interpolation of a video frame (left) and the enhanced frame (right) created by the program. For the enhancement Horn-Schunck optical flow algorithm was used

Figure 10
Bicubic interpolation (left) and the enhanced frame (right). Compare the images at the shoulder of the man. The right image is much more realistic since it is smoother and there is less noise

Figure 11
Motion blur is generated when fast moving happens. This is caused by the warping step if the lengths of offset vectors are too small. Motion blur is present at each technique more or less

Conclusions

In summary, we developed a well-extensible, simple handable system, in which different reconstruction based super-resolution methods can be tested and compared with each other. In contempt of simplicity, sometimes quite good results can be achieved either in the case of traditional image sequences or captured video using omnidirectional imaging. During the comparison of the different optical flow methods for super resolution, sometimes the generated output image has intensive blur, mainly because of the imprecision of the given optical flow method.

Furthermore, a .NET wrapper named DirectCV [12] was developed for OpenCV, which is available freely for anyone, easy-to-use, well-documented, and has extensive IntelliSense support.

References

[1] Nagy, A., Vámossy, Z., „OpenCV C# Wrapper-based Video Enhancement Using Different Optical Flow Methods in the Super-resolution", in proc. of 6th International Symposium on Intelligent Systems and Informatics, Subotica, Serbia, 2008, CD issue, IEEE C. N. CFP0884C-CDR

[2] Intel, Open Source Computer Vision Library, http://www.intel.com/technology/computing/opencv, visited on 2008-08-12

[3] S. Baker, T. Kanade "Super-Resolution Optical Flow", CMU-RI-TR-99-36, 1999

[4] L. Zhouchen, H. Junfeng, T. Xiaoou, T. Chi-Keung, "Limits of Learning-Based Superresolution Algorithms", Technical Report MSR-TR-2007-92

[5] S. Battiato, G. Gallo, F. Stanco, "Smart Interpolation by Anisotropic Diffusion", IEEE 12th Int. Conf. on Image Analysis and Processing, 2003, pp. 572-577

[6] S. Battiato, G. Gallo, F. Stanco, "A Locally Adaptive Zooming Algorithm for Digital Images" – Elsevier Image and Vision Computing, 20/11, 2002, pp. 805-812

[7] OpenCVDotNet, .NET Framework Wrapper for Intel's OpenCV Package, http://code.google.com/p/opencvdotnet/, visited on 2008-08-12

[8] Rhodes University, SharperCV project, http://www.cs.ru.ac.za/research/groups/SharperCV/, visited on 2008-08-12

[9] J. Richter, Applied Microsoft .NET Framework Programming, Microsoft Press, 2002

[10] C. Nagel, B. Evjen, J. Glynn, M. Skinner, K. Watson, A. Jones: Professional C# 2005, Wiley Publishing, Inc., 2006

[11] P. Greguss, F. Alpek, M. Patko, "Changing from Guided Robocars to Autonomous Robocars by Using Humanoid Machine Vision System", in. *Proc. of 7th International Workshop on RAAD'98*, Bratislava, Slovakia, 1998, pp. 127-131

[12] A. Nagy, DirectCV, "A lightweight .NET wrapper for Intel's OpenCV library.", http://code.google.com/p/directcv/ visited on 2008-08-13