# Real-Time Simulation of Camera Errors and Their Effect on Some Basic Robotic Vision Algorithms

André Hinkenjann, Thorsten Roth, Jessica Millberg, Hojun Yun
*Institute of Visual Computing*
*Bonn-Rhein-Sieg University of Applied Sciences*
*Sankt Augustin, Germany*
*Email:* {*Andre.Hinkenjann*|*Thorsten.Roth*|*Jessica.Millberg*}*@h-brs.de*
*Hojun.Yun@smail.inf.h-brs.de*

Yongmin Li
*Information Systems and Computing*
*Brunel University*
*Uxbridge, Middlesex, U.K.*
*Email: yongmin.li@brunel.ac.uk*

***Abstract*—We present a real-time approximate simulation of some camera errors and the effects these errors have on some common computer vision algorithms for robots. The simulation uses a software framework for real-time post processing of image data. We analyse the performance of some basic algorithms for robotic vision when adding modifications to images due to camera errors. The result of each algorithm / error combination is presented. This simulation is useful to tune robotic algorithms to make them more robust to imperfections of real cameras.**

***Keywords*-Robot vision systems, computer simulation, image processing**

## I. Introduction

Modern robots, especially service robots, do no longer operate in well defined environments like factories or high-rack warehouses. They rather work under every day conditions, where they need to build maps, navigate and, e.g., identify objects. For these tasks optical cameras play an important role. They deliver video data streams, which are pre-processed in real-time, before certain regions of interest are stored in bitmaps in memory in order to enable the functions of the robot. Traditionally, these algorithms contain typical standard computer vision algorithm blocks, like edge detection, Hough transform, disparity maps, feature detection etc.

Under different technical and environmental influences to the video images, they achieve different results. This means that the ultimate task of the robot might not be carried out correctly. We present an examination of technical influences, i.e., camera errors, to the outcome of basic robotic algorithms building blocks. Camera errors we examined are vignetting, chromatic aberration and thermal ccd noise. These errors are realized as real-time post-processing filters of color and depth information, approximating the physical phenomena. This has the advantage of being able to interactively simulate the influence of an imperfect camera system in real or synthetic environments.

Every robotic system that needs to be simulated (because it will be used in a difficult to reach environment, like space or underwater, e.g .) can profit from a more realistic simulation of all parts, including the optical camera system. Obviously, this is only a subset of the most important phenomena which can have an influence on vision algorithms. The feature of an interactive simulation makes it possible to run through different simulations quickly and see how the robotic systems reacts to changes of the environment or the camera parameters. We hope that our examination can be useful to make vision based robot algorithms more robust and to prepare robotic missions better.

Note that although our simulation as presented in this paper is limited to camera errors, it could also be extended to cover further phenomena, especially including environmental changes. To achieve this, it would be possible to incorporate our own global illumination renderer to generate images on-line [1].

## II. Related Work

Related work has been done in the field of robot video camera simulation. Typical optical camera effects and imperfections, like distortion, chromatic aberration, depth of field, sensor saturation and noise were simulated in real-time in [2] in order to validate the simulation of a virtual space robotics test bed. No evaluation was given on the influence of single effects on basic vision algorithms.

In another paper Rossmann et al. [3] discuss their optical sensor simulation framework for robot applications. They especially focus on the simulation of depth in field sensors, like stereo cameras or 3D laser range finders.

In [4], Asanuma et al. present a simulator for autonomous mobile robots that considers camera characteristics. Images are generated using an OpenGL renderer and after that the images are modified according to camera characteristics. The authors show that high-level tasks of the simulated robot, like estimating a ball's distance or self-localisation, are simulated more realistically than without considering camera effects. Also, no attribution of the result to the basic vision algorithms is given.

An image based lens model is described by Heidrich et al. [5]. They use a similar approach to the one presented here. The very time consuming exact realization of the lens (and limitations, like vignetting) is approximated in image space. The paper does not evaluate the simulated lens in the area of robotics and vision. Similar work is done by Kucis [6].

## III. Simulation of Camera Errors

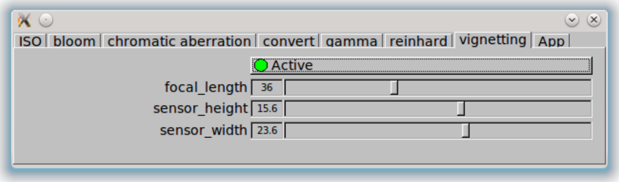
To achieve a real-time approximation of optical errors, we employ our own graph-based post-processing framework called *GrIP*. We concentrate on three common errors in images generated by optical systems: Thermal (amplifier) noise, chromatic aberration and vignetting. Each of these effects is implemented as a plugin for GrIP, allowing for an easy composition of the effects in a simple filter chain. In the following subsections, we will give an overview of GrIP (for detailed information, please see [7]) and some details on the implementation of each of the plugins for the simulated optical errors. All mentioned effects have been implemented using NVIDIA's CUDA technology. It is also possible to use other platforms for plugin implementation, but for now only CUDA and host-based nodes are directly supported by GrIP.

### A. Real-Time Post-Processing

GrIP is an extensible post-processing framework, which can be used for experiments with algorithms based on a screen-space approach, i.e., the processing of information available on a per-pixel basis. It provides a basic plugin system, allowing for easy implementation of new effects. This system is based on shared objects loaded at run-time and connected via a directed, acyclic post-processing graph (PPG). The plugins are represented by nodes in this graph and can be arbitrarily connected to each other to define their dependencies, thus yielding a specific execution order. Defining the PPG is done using an XML description, making it easily editable. More information on the graph description format can be found in [7].

Through GrIP, it is easy to create the desired composition of effects and filters and parameterize them arbitrarily. The XML description allows for the definition of predefined, fixed parameters as well as dynamic parameters which can be changed by the user at runtime. The main design goal during the development of GrIP was to provide an easily usable framework for post-processing, where the user can arbitrarily arrange the building blocks of the algorithms, allowing for a great amount of reusability.

Input and output data for specific nodes are not directly dependent on the defined edges, but are defined separately by assigning simple names which are then used for storing the information in an instance of a universal container class. This class is responsible for storing all data which has to be provided for and exchanged between the different nodes in the graph. Via the graph runner class, a subset of methods for setting objects in and getting objects from the container is also exposed to the user. In general, this class is the central entity for all interactions of an application with GrIP.



focal_length 36
sensor_height 15.6
sensor_width 23.6

Figure 1. An example for the automatically generated user interface with the vignetting node currently being selected. The tabs represent the nodes in the graph, the sliders in each tab are used for parameterizing this specific node, so that the effects of changing parameters are directly visible. The "App"-Tab provides node-independent parameters which can be used for the application employing GrIP.

Loading a graph is done here, as well as specifying whether a graph visualization and a parameterization interface should be created. The latter is a central component of GrIP: Through the parameterization GUI, all node parameters which have been marked with the mutability attribute as `dynamic` are presented via sliders or similar elements, depending on the underlying object type. These elements are grouped by their respective nodes and an extra group for application parameters is also present, which means that GrIP can also be used for quickly providing a simple graphical user interface to your application.

Additionaly, some nodes can be toggled on and off via a simple button, so a direct comparison between before and after processing can be done. This can be useful especially for otherwise somewhat subtle effects like a slight amount of noise or vignetting. Whether the toggling is possible depends on the implementation of the plugin, as the conversion of input to output data still has to be handled somehow for an inactive node. The default state for a node can be set with the `defaultstate` attribute. An example for the GUI is shown in figure 1.

In addition to the central container for all exchanged data, GrIP also provides a "repository" for temporary buffers. This means that a node which requires several temporary buffers does not have to allocate the necessary memory each time the PPG is executed, but can rather allocate this memory once and then pass control over the allocated memory to the `TempBufferRepository`. The repository will return the closest matching buffer upon request of a specific resolution. Also it is possible to create several "sub-repositories" by just passing a name. This can make sense, e.g., if some buffers should only be used for one specific node A and the results must not be overwritten by other nodes before the next call of A.

### B. Camera Errors

*1) (Thermal) Noise:* Even when an optical sensor is not exposed to light, dark current causes noise to appear in the image. With higher ISO values and depending on environmental conditions (e.g., temperature) this noise tends to increase. This is a problem especially in sensors with a small area per pixel. As the main influencing factors for sensor noise are the area/resolution ratio and temperature, we implemented a plugin for GrIP which takes the sensor size and a temperature as its parameters (an additional ISO value could be easily added). Based on this, gaussian noise is computed for each pixel and color channel which is then added to the input image. An example for an artificial image processed with the noise node is given in figure 2 (the noise-free version is also shown for comparison).

In the evaluation section, we look at four different noise levels (low, medium, high and extreme), corresponding to higher temperatures and ISO values in real cameras. Although these noise levels are not necessarily phyiscally correct for a specific sensor, they closely match what you will see in images captured with a real-world camera and thus provide a good hint on how noise will influence your algorithms.

*2) Vignetting:* Vignetting is an effect apparent in all images created with optical systems and denotes the falloff of perceived image brightness in peripheral image regions. Vignetting is based upon the $\cos^4$-law (equation 1), which means that the falloff depends directly on the fourth power of the cosine of the incident light's angle $\alpha$:

$$E_\alpha = E_0 \cos^4 \alpha, \tag{1}$$

with $E_0$ being the brightness in the image center. Note that, e.g., with low aperture values there can also be mechanical vignetting, i.e., light being blocked by the objective itself, while our implementation only accounts for the natural vignetting effect.

To achieve the desired effect, our implementation of natural vignetting is parameterized with sensor size as well as focal length. From these values, the maximum incident light angle (apparent in the image's corners) can be easily calculated. Based upon this, the correct vignetting values can be computed for each pixel with respect to the $\cos^4$-law. Examples of several settings for the vignetting node are shown in figure 2.

*3) Chromatic Aberration:* Chromatic aberration results from the different refraction of the light's various wave lengths when passing through the lens system. This leads to the varying focal planes, shifted along the optical axis with respect to the specific wavelength. The intensity of this effect can be described by the Abbe Number $v_e$ [8]. $v_e$ can be calculated from the main refractive indices for wave lengths 486.1nm ($n_F$), 587.6nm ($n_e$) and 656.3nm ($n_C$) as follows:

$$v_e = \frac{n_e - 1}{n_F - n_C}. \tag{2}$$

Due to the shift along the optical axis, focal planes for different wave lengths do not line up and thus a blurry image results with the intensity of the blur being dependent on the according wave length. To model this effect via our plugin, we define a focal difference in percent of the focal distance. Based on this and the also definable sensor size, the appropriate size of the computed gaussian blur can be determined. Note that this approach is not physically correct, but yields convincing results in our tests when compared to real chromatic aberration. Thus, we also used this effect for our evaluation. An example of an image without and with our chromatic aberration filter applied is shown in figure 2.

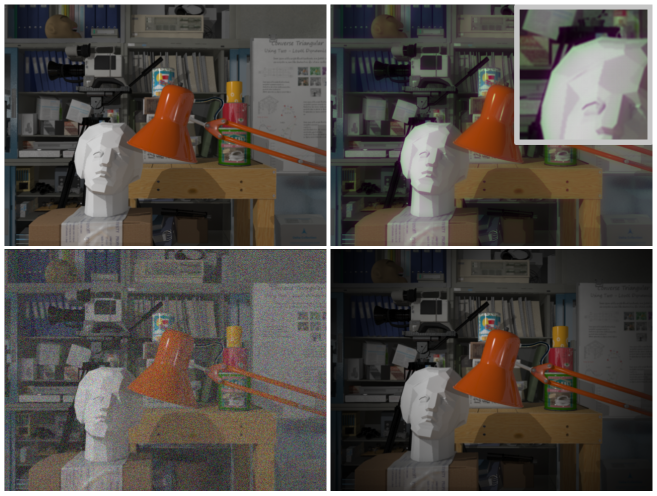


Figure 2. An image from the Tsukuba sample dataset, without post-processing (top left), with chromatic aberration (top right), noise (bottom left) and vignetting (bottom right)

## IV. Basic Computer Vision Algorithms for Robotics

In our evaluation we determine how much the integration of optical errors influenced the outcome of various image processing algorithms. We integrate some standard computer vision algorithms into our system for evaluation purposes. For this we employ the open source computer vision library OpenCV, which can also be used for the Robot Operating System (ROS). We choose some vision topics that are frequently used in robotic applications and select specific algorithms that are common for the corresponding topic. We take into account the following topics and algorithms:

### A. Stereo Vision

Sometimes there is the need to obtain depth information for the environment. The depth can be used for example for obstacle avoidance [9] or to segment specific areas [10]. One way to obtain the depth is using stereo correspondences and creating a disparity map from it. With a triangulation step the depth can be calculated from the disparity map. We create disparity maps from stereo images using block matching. An example for disparity maps generated for the original and processed images is shown in figure 12.

### B. *Feature Matching*

Tasks such as object detection or categorization often require feature matching. In [11] and [10] such an approach is used to enable a service robot to detect trained objects. For the detection and description of blob-like features we use the Speeded Up Robust Features (SURF) [12] method. As a matching strategy we apply a simple brute force algorithm, but filter the matching results using RANSAC [13] during homography computation in order to receive a reliable set of inliers.

An example for feature matching performed on the original and processed images is shown in figure 13.

### C. *Edge detection*

Sometimes robotic navigation applications require knowledge about lines in the image. These could be either artificial lines such as lane markings [14] or edges that result from the structure of an object like a door [15]. For simple edge detection we use the Canny algorithm [16].

First we filter the image noise with a 5×5 Gaussian kernel and then apply a Canny edge detector. The 5×5 kernel is used for all images, as our aim is to analyse the influence of optical errors on fixed effects. Thus, we tried to find the best parameterization for edge detection for the worst image quality, which was present with high noise. After the pre-filtering we use the probabilistic Hough line transform for the recognition of straight line segments in the resulting Canny image.

An example for Hough line transform performed on the original and processed images is shown in figure 14.

As a complex 3D model scene, we use the new Tsukuba CG stereo image dataset [17], [18] as our testing environment. This image sequence has the advantage that it is completely synthetic just as rendered images from a simulation environment. It also provides stereo images. Furthermore, the synthetic images guarantee that there are absolutely no optical errors involved that would be otherwise produced by a sensor and might add up to the errors we simulate.

## V. Evaluation and Results

For the evaluation we applied the implemented camera errors (thermal noise, vignetting, chromatic aberration) to the given dataset. These manipulated images were tested with every integrated vision algorithm (see section IV). To generate ground truth data we also applied the vision algorithms to the pure dataset images. We collected for every image the number of valid pixels in the disparity images (stereo vision), the number of matched inliers (feature matching) and the number of found Hough lines (edge detection).

Videos of the outcome of applying different basic vision algorithms to images with different camera imperfections can be found on the project website for GrIP, accessible through http://www.ivc.h-brs.de.

### A. *Chromatic Aberration*

*1) Disparity Images:* The generation of disparity images did not suffer from any quality loss when processing the input data with the chromatic aberration filter (cf. figure 3). Contrariwise, the number of pixels for which a disparity value could be determined even increased with a greater focus difference and thus a stronger aberration effect. While this might seem surprising at first, it seems logical when taking a closer look. The first explanation we came up with was that slightly blurring the input data removes a fraction of high frequencies from the image, so that some of the originally higher-frequency areas could be matched better by the block matching algorithm. This theory is also supported by the fact that in frames with lower frequency content the difference is less significant (e.g., around frame 1550), while frames with higher frequencies (such as the window blinds, e.g., between frame 600 and 800) show a stronger effect of the blurring. We also verified this by applying just a simple Gaussian blur on all color channels, which yielded even better results.

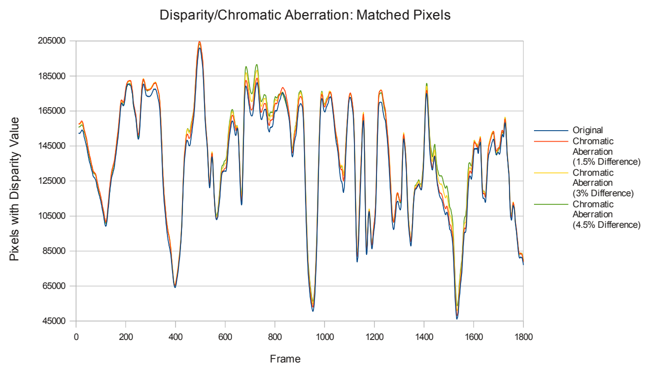


Figure 3. Impact of Chromatic Aberration on Disparity Images

*2) Feature Matching:* Feature matching did also not suffer from any significant quality losses (cf. figure 4). However, in contrast to disparity image generation, the number of matched features decreased by approximately 12 percent in higher-frequency frames, while it never increased. As chromatic aberration is in fact a slight color-dependent blur, less features can be detected and in turn also be matched. Accordingly, the number of matched features (inliers) decreased consistently with a stronger aberration effect. This is mostly evident in sub-sequences with higher frequency such as between frame 600 and 800.

*3) Hough Lines:* The explanation for the effect of chromatic aberration on Hough line transform (cf. figure 5) is similar to the one for its effect on feature matching. Edges in an image are slightly blurred by the chromatic aberration and thus their detection by the Canny edge detector becomes more difficult. Consequently, the Hough line transform degrades more for sub-sequences with greater

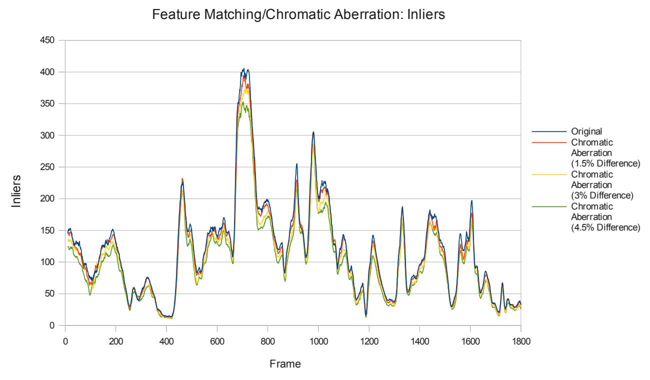


Figure 4. Impact of Chromatic Aberration on Feature Matching

areas of high-frequency content, like those at the beginning of the image sequence and, again, between frame 600 and 800. However, this effect is far less significant from frame 1400 onwards, where the same scene elements (especially the window blinds) are visible again, but closer and thus not causing as high frequencies as before.

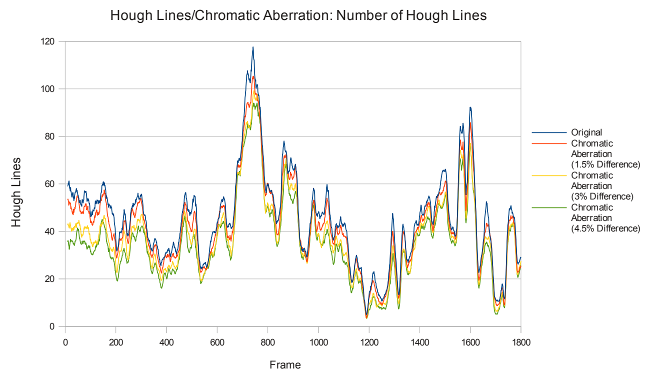


Figure 5. Impact of Chromatic Aberration on Hough Lines

## B. Noise

*1) Disparity Images:* The diagram in figure 6 shows clearly that higher levels of noise have a serious impact on the computation of disparity images. While the curve shape stays basically the same, the number of valid disparity values drops significantly with each increased noise level. Already at the lowest noise level the number of valid disparity pixels decreases by about 50 percent. At the highest level (extreme) only few pixels could still be assigned with a valid disparity value (about 2 percent compared to the results on the noise-free image). This is due to the fact that the stereo algorithm we integrated, uses SAD-based (sum of absolute differences) block matching. This method is quite sensitive to intensity changes which are for example caused by noise. Of course, applying a blurring operation

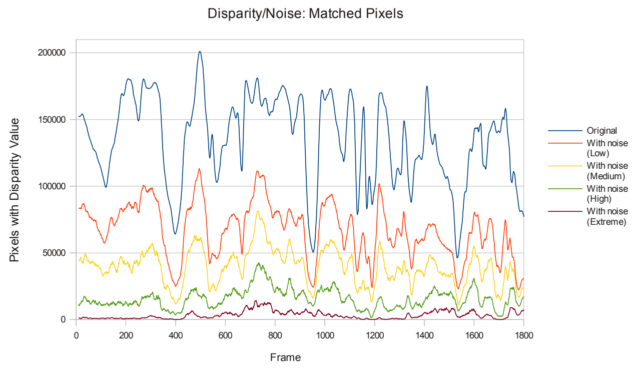


Figure 6. Impact of Noise on Disparity Images

on the noisy image beforehand, would improve the results. Certainly, block matching remains very sensitive to noise.

*2) Feature Matching:* Figure 7 shows the impacts of noise on feature matching. While lower noise levels still result in comparable numbers of inliers, the count of matched correspondences decreases with increased noise. In the low noise images the algorithm still found about 95 percent of the number of matches found in the original image, while medium noise results in about 85 percent and high noise in about 70 percent. However, in comparison with the block matching even the addition of extreme noise still results in a notable number of inliers (about 40 percent), especially on the images that have a lot of textured elements and therefore yield better matching results.

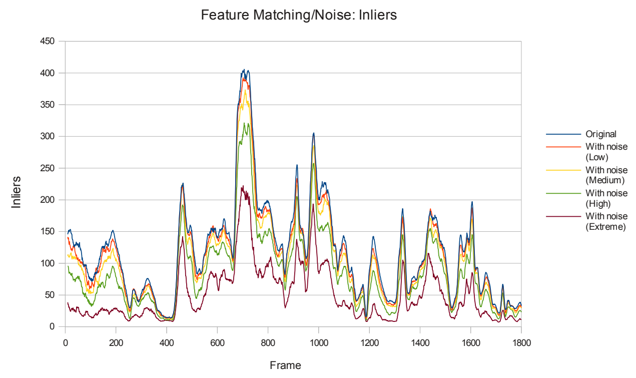


Figure 7. Impact of Noise on Feature Matching

*3) Hough Lines:* In figure 8 the influence of noise on Hough line transform is shown. With a low noise level the number of detected Hough lines is not significantly different from the original one. Even extreme noise still results in a useful number of lines (about 70 percent are found compared to the original noise-free results). This is probably due to the Gaussian blurring which is performed before the Canny edge detection.

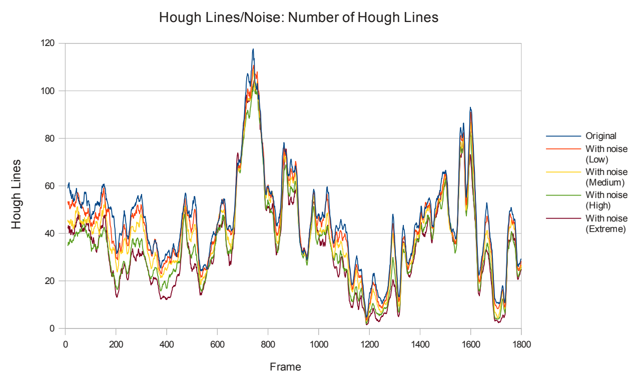


Figure 8. Impact of Noise on Hough Lines

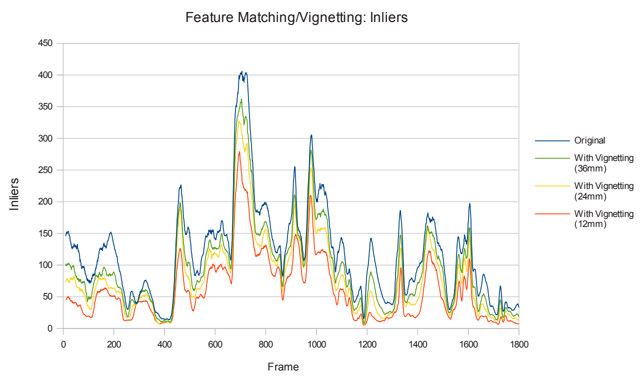


Figure 10. Impact of Vignetting on Feature Matching

### C. Vignetting

In this section we describe the impact of the camera errors on vignetting for a fixed sensor diagonal of 28.29mm and varying focal lengths.

*1) Disparity Images:* Figure 9 shows the influence vignetting has on disparity matching. The number of pixels with a disparity value dependent on the frame number is given for the original stereo images and different levels of vignetting. As expected, the more vignetting, the less pixels have disparity values determined. There is a noticeable drop of about 15-40 percent of the number of disparity pixels when comparing strong vignetting (orange curve) with the no vignetting (blue curve). The overall characteristic of the curve of disparity of the original images is still maintained for vignetted image sequences.

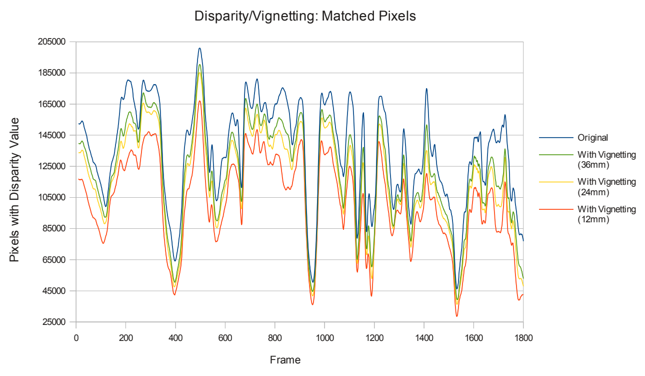


Figure 9. Impact of Vignetting on Disparity Images

*2) Feature Matching:* Figure 10 shows the number of inliers after feature matching for each frame, dependent on the strength of vignetting. It can be noticed that - while maintaining the curve's characteristics - the number of matched features decreases with the strength of vignetting. The number varies between about 20 percent and up to 100 percent for strong vignetting (orange curve, compared to blue curve). Depending on where the matched features are (towards the border or the middle of the images) the quality of feature matching for vignetted images varies significantly.

*3) Hough Lines:* Even more differences can be noticed when comparing the number of Hough lines found for the images of the sequence to the strength of vignetting. Strong vignetting makes a big difference and is very influential across the complete series of images. The number of Hough lines found compared to unvignetted images ranges from zero to mostly 10-20 percent up to about 55 percent. When line structures (like the window blinds in the images around number 950) are visible, many Hough lines are detected. When they move towards the image border, vignetting prevents them from being detected, so the difference of detection increases. Compared to disparity matching and feature detection, Hough line detection is affected most by strong vignetting in our experiments.

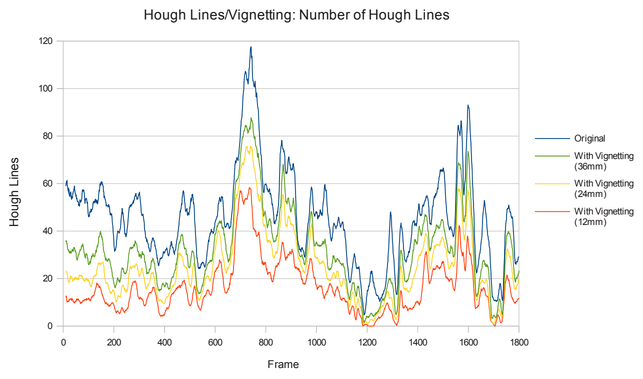


Figure 11. Impact of Vignetting on Hough Lines

## VI. Conclusion and Future Work

Our tests have shown that the examined vision algorithms' outcomes vary when tested under influence of the presented camera errors and their different parameterizations.

Chromatic aberration did not influence the outcome of the algorithms significantly in most cases and even improved the quality of stereo block matching.

The greatest impact by far could be observed when trying to apply stereo block matching to an extremely noisy image. The amount of pixels with a valid disparity value already decreased by almost 50 percent with low noise. The resulting disparity images were practically useless from medium noise onwards, where only a maximum of 7 percent of all pixels could be assigned a valid disparity value, compared to more than 80 percent in the corresponding noise free image.

The effects observed for vignetting correspond with the lower visibility of scene elements in the peripheral image regions and completely match the expectations.

None of the vision algorithms has been rendered completely useless by the simulated camera errors. When the errors are applied with moderate parameters resulting only in a small impact on the image, most of the tested scenarios had only a small loss of information. When the simulated errors got stronger the vision algorithms returned poorer results. Noise and vignetting were affected most by the reduced image quality of our simulated errors. These results show that a simulation of a robotic system should take camera errors into account, when they are expected to some extent in the real system.

While our observations already give some hints on how some of the basic building blocks of robotic vision are influenced by camera errors, this research could be continued in several ways. First, all of the presented algorithms take a set of parameters. e.g., the Canny edge detector uses a threshold and the input image is blurred before Canny is applied. The subsequently executed Hough line transform also takes several parameters, such as the minimum line gap and minimum line length.

For our experiments, we used a set of standard parameters which work well for mostly error-free images and observed how the algorithms behave when applied to erroneous images, but with the same parameterization. Future work could especially include an in-depth look into which parameters have to be adjusted in which way to work against the impact of specific camera errors and, going even further, combinations of these.

Furthermore, other robotic vision algorithms could be analysed with the same methods. Also, it would be possible to integrate our image filters into a testing framework for vision algorithms, which could in turn also be used for more automated testing and direct comparison of various algorithm parameterizations.

While we applied our developed filters to a pre-defined image sequence in this paper, our global illumination renderer *Spark*, which has already been applied successfully in the project IVAB [1], could be utilized to test the algorithms under changing environmental conditions in the future.

## References

[1] "IVAB: High-Quality Rendering in the Planning of Prefabricate Houses," http://vc.inf.h-bonn-rhein-sieg.de/ivc/?page_id=1554, 2012, online; accessed 2013-03-13.

[2] J. Rossmann, T. Steil, and M. Springer, "Validating the camera and light simulation of a virtual space robotics testbed by means of physical mockup data," in *11th International Symposium on Artificial Intelligence, Robotics and Automation in Space i-SAIRAS,*, 2012, pp. 1–6.

[3] J. Rossmann, N. Hempe, M. Emde, and T. Steil, "A real-time optical sensor simulation framework for development and testing of industrial and mobile robot applications," *Robotics; Proceedings of ROBOTIK 2012; 7th German Conference on*, pp. 1 –6, may 2012.

[4] K. Asanuma, K. Umeda, R. Ueda, and T. Arai, "Development of a simulator of environment and measurement for autonomous mobile robots considering camera characteristics," in *RoboCup 2003: Robot Soccer World Cup VII*, ser. Lecture Notes in Computer Science, D. Polani, B. Browning, A. Bonarini, and K. Yoshida, Eds. Springer Berlin Heidelberg, 2004, vol. 3020, pp. 446–457.

[5] W. Heidrich and H. peter Seidel, "An image-based model for realistic lens systems in interactive computer graphics," in *In Graphics Interface 97*, 1997, pp. 68–75.

[6] M. Kui and P. Zemk, "Simulation of camera features," in *Proceedings of the 16th Central European Seminar on Computer Graphics*. Technical University Wien, 2012, pp. 117–123.

[7] T. Roth and A. Hinkenjann, "Grip: A framework for experiments with screen space algorithms," in *8. Workshop Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR*, S. M. Christian-A. Bohn, Ed. Shaker Verlag, 2011, pp. 85–96.

[8] "Calculation of abbe's number for glasses," http://glassproperties.com/abbe_number/, online; accessed 2013-01-20.

[9] D. Burschka, S. Lee, and G. D. Hager, "Stereo-based obstacle avoidance in indoor environments with active sensor re-calibration," in *Proceedings of the 2002 IEEE International Conference on Robotics and Automation, ICRA 2002, May 11-15, 2002, Washington, DC, USA*. IEEE, 2002, pp. 2066–2072.

[10] D. Asanza and B. Wirnitzer, "Improving feature based object recognition in service robotics by disparity map based segmentation," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 18-22, 2010, Taipei, Taiwan*. IEEE, 2010, pp. 2716–2720.

[11] C. A. Mueller, N. Hochgeschwender, and P. G. Ploeger, "Towards robust object categorization for mobile robots with combination of classifiers," in *RoboCup*, 2011, pp. 137–148.

[12] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, 2006, pp. 404–417.

[13] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Commun. ACM*, vol. 24, no. 6, pp. 381–395, jun 1981.

[14] A. Assidiq, O. Khalifa, R. Islam, and S. Khan, "Real time lane detection for autonomous vehicles," in *Computer and Communication Engineering, 2008. ICCCE 2008. International Conference on*, may 2008, pp. 82 –88.

[15] X. Yang and Y. Tian, "Robust door detection in unfamiliar environments by combining edge and corner features," in *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, june 2010, pp. 57 –64.

[16] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, jun 1986.

[17] S. Martull, M. P. Martorell, and K. Fukui, "Realistic cg stereo image dataset with ground truth disparity maps," in *ICPR2012 workshop TrakMark2012*, 2012.

[18] M. P. Martorell, A. Maki, S. Martull, Y. Ohkawa, and K. Fukui, "Towards a simulation driven stereo vision system," in *ICPR2012*, 2012.

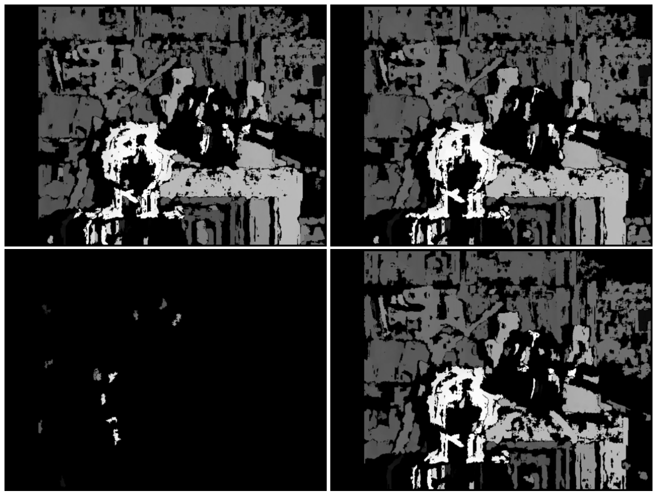


Figure 12. Disparity image generated from original images (top left), with chromatic aberration (top right), strong noise (bottom left) and vignetting (bottom right)

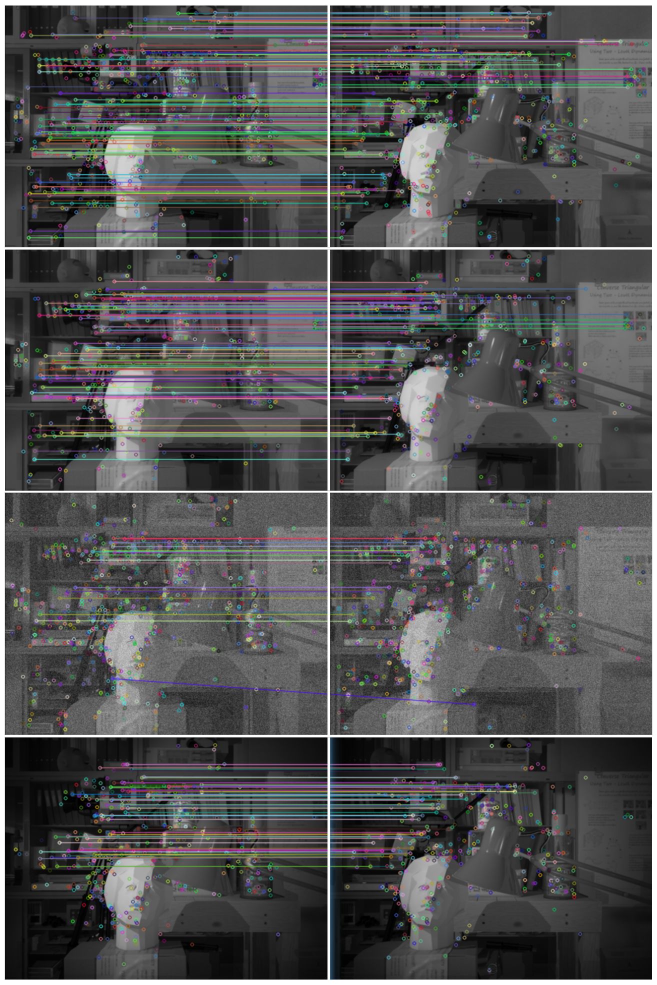


Figure 13. Feature matching performed on original images (first row), the images with chromatic aberration (second row), strong noise (third row) and vignetting (fourth row)

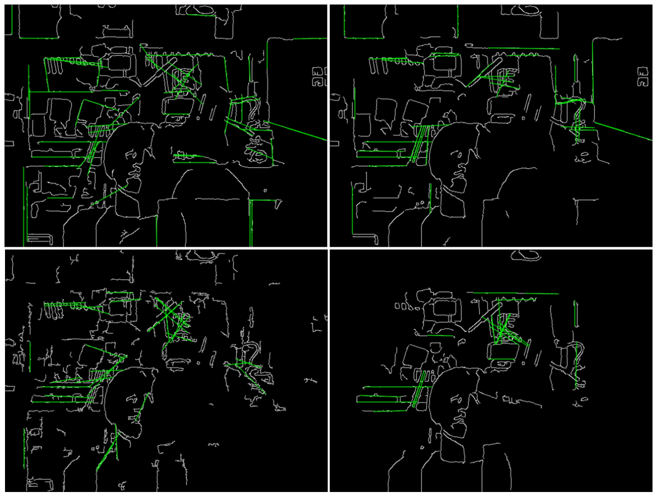


Figure 14. Hough Line transform performed on original image (top left), the images with chromatic aberration (top right), strong noise (bottom left) and vignetting (bottom right)