Technology
Arts Sciences
TH Köln

Project Documentation

Interactive Lighting Detector

written by

Vera Brockmeyer (Matrikelnr. 11077082)
Laura Anger (Matrikelnr. 11086356)

Image Processing in SS 2017

Supervisor:
Prof. Dr. Dietmar Kunz
Institute for Media- and Phototechnology

Contents

1 Introduction

The detection of digital image forgery is an actual research topic, which analyses various image properties to get a conclusion about the authenticity of a digital image. It can be divided into two sections [9]. The first section is the source digital camera identification. It considers image properties like lens aberrations, sensor imperfections or use a colour filter array (CFA) interpolation to analyse the suspicious items. Those mentioned properties can give an assumption on the camera or even the camera model, which was used to capture the picture.

The other section is the actual image forgery analysis. A common method to detect the application or device that was used to save the image file is the analysis of JPEG patterns. Each has an individual JPEG pattern which can be extracted and compared to a database of acknowledged patterns. Another approach for image forgery analysis is the detailed observation of the chromatic aberration camera response function (CRF) that can give conclusions about manipulations in form of retouching or composing of new image content.

Due to the assumption that different images are taken under different light conditions, the manipulative composition of two images can be often detected by analysing the light vectors of various objects depicted in the image. But this method is also very difficult because it needs an image sequence of the scene to estimate the 3-dimensional light direction successfully with established algorithms. This is not very useful in reality because in most cases only the suspect image is available without any information about its real environment. Nevertheless, there exists an approach by Johnston and Farid [7] that can rather estimate the object's light direction under some pre defined assumptions even from single images with a common least square approach. The assumptions are clearly defined and result in a simplified light model that is related to an infinite light source and a constant reflection values of the object's surface Stimmt das so? Der Refelektionswert wird ja für jeden patch als konstant angenommen und nicht für das ganze Objekt. Würde da nur das mit der unendlichen Lichtquelle schreiben, weil sich der rest pro approach unterscheidet :).

1.1 Motivation

In the last decade, it became more easily to create an image forgery for everyone. Software applications like *Adobe Photoshop* or *Gimp* and almost every mobile phone or social media platform offers possibilities to sophisticate digital images. Every user can manipulate images or compose a new image of various other images without any significant knowledge. For example, two portraits of prominent people can be combined in a way that the impression of a relationship accrues which does not really exist. If this is made in an imperfect way the forgeries can be recognised very quickly by the human visual system but if they are made in a professional way it is quite challenging to detect the manipulation.

In this case, a professional analysis tool is required to give a reliable rating which can stand in court as evidence. As mentioned, a main problem of the analysis is

that many approaches ask for a sequence of images from the scene for trustworthy and validated results but in most cases only the suspect image is available. Thus, an approach which estimates the light vector under some refused assumption has to be used and eventually confirmed by other image forgery detection methods.

1.2 Project Goal

According to the previous sections, the project goal is to implement and evaluate the approach of Johnston [7] in *C++* with an adequate *Qt* Graphic User Interface (GUI). This GUI offers a segmentation of the object of interest, as well as an interactive selection of the most suitable contour part, to estimate the light direction. The latter is an important requirement of the Johnston approach. To validate the results, several test images have to be captured with a simulated infinite light source. The infinite light source can be the sun on a cloudless day, for example. A sun clock needs to be added to the test scene to verify the current light direction and various objects with a simple and even shape complete it. All resulting vectors and contours are printed in the actual image or respectively saved in a text file for the following evaluation.

2 State of the Art

In the following Sections the basic scientific knowledge to understand the *Interactive Lighting Detector*, whose functionality is explained in Section 4, is presented.
A general introduction to image forensic is given in SectionFir2.1. Furthermore, other approaches using light vectors, which are detected using similar assumptions to [7], are shortly presented in Section 2.2.

2.1 Image Forensic

In [9] the authors claim image forensic to become more important over the years. Furthermore, they divide the field in two approaches. First of all, image forensic can be used to identify the recording device of an image. This can, inter alia, be done by taking sensor imperfections like pixel defects or the lens aberration into consideration.
The second field of interest is the detection of image forgery [5]. Beside using the camera response function, there can be other details in the image which are informative to differ between an original or forgery. For example, the light situation in an image must be consistent. This can be proofed by calculating light vectors at various points in the image. The *Interactive Light Detector* is using exactly this method (compare Section 4). Related approaches are described briefly in Section 2.2.

2.2 Related Approaches

The *Interactive Lighting Detector* was implemented according to the paper by Johnson and Farid [7]. The foundation of their assumptions were set in 2001 by the publication of Nillius and Eklundh on an "*automatic estimation of the projected light source direction*" [10]. Whereas the earlier theory is taking three dimensional surface normals to determine the light vectors pointing into the direction of the light source, the newer approach by Johnson and Farid uses only one image. Therefore, two dimensional surface normals are used to achieve the same goal.
An other related approach, which is also presented by Johnson and Farid, estimates the three dimensional light direction from the light's reflections in the eyes of human. Therefore they determine the light vector by using the surface normal and the view direction of the person [8].
Further research did not offer other approaches that uses single images and a two dimensional vector space. Most of them ask for several images from different view points as well and thus can not be compared with the approach of Johnston.

3 Materials

The following sections describe the resources and tools required for the completion of the project. Furthermore, the required test images are presented in Section 3.3.

3.1 Hardware

During the implementation phase, the application was run on two computers which are described in the following two sections. Both computers needed to be able to deal with the software components described in section 3.2. An extract from their data sheet is shown in Table 1 respectively Table 2.

Acer E5-571G	**Description**
Processor	Intel Core i7 CPU @ 2.40 GHz
RAM	8 GB
Graphic Card	NVIDIA GeForce 840M
Operating System	Windows 10 Education 64 bit

Table 1: Extract from the Data Sheet of the Acer E5-571G

.

Acer Aspire 5820TG	**Description**
Processor	Intel Core i3 CPU @ 2.40 GHz
RAM	4 GB
Graphic Card 1	AMD Mobilty Radeon HD 5000 Series
Graphic Card 2	Intel(R) HD Graphics
Operating System	Windows 10 Education 64 bit

Table 2: Extract from the Data Sheet of the Acer Aspire 5820TG Notebook.

3.2 Software

In order to develop the *Interactive Lighting Detector Qt* was used (compare section 3.2.1). To take advantage of already existing functionalities the *OpenCV*-library, which is described in section 3.2.2, was included in the project.

3.2.1 QT

The *QT Creator* was invented by *The Qt Company*. It is an integrated software development environment in the programming language C++. More functionality can be added by using the Qt project's library, which is called *Qt*. As a cross-platform tool, the *QT Creator* can be used on all common operating systems [12]. Besides extensive database functions and XML-support the software can build GUI.

For this project the algorithm was transcribed in source code using the *Qt Creator* and the GUI was designed in the *Qt Designer* [11].

3.2.2 OpenCV

The *Open Source Computer Vision* (OpenCV) is an open source library for image- and video processing, which is among others available in the programming language *C++*. It has been introduced ten years ago and is developed by various programmers since then. This library offers the most common algorithms, as well as current developments in image processing [4].
In the case of the implementation of the *Interactive Lighting Detector* the library was mainly used for the detecting of the contours (compare Section 4.1) and solving the minimization problem (compare Section 4.4) introduced by Johnson and Farid [7].

3.3 Test Images

Due to the assumption that the objects shown on the test images described in Section 3.3.1 have a too complicated shape, a second batch of images was made (compare Section 3.3.2). Images of both batches were used to test the functionality of the algorithms used for the light detection. All images have in common that besides the actual object, they show a sundial to simplify the determination of the light direction for the user. Consciously, a natural setting and reflective surfaces were selected.

3.3.1 First Batch

Four examples of the first batch of test images are shown in Figure 1. There are different objects depicted next to the mandatory sundial, like a helmet, a handbag, a bucket or a hot-water bottle. Those objects differ in their surface texture, as well as their size. They are shot from different angles to produce different light directions. It is necessary that the objects are not trimmed at the boarders of the image, because the algorithm requires a full contour of the selected object.

Figure 1: Examples of the test images of the first Batch.

3.3.2 Second Batch

In contrast to the first batch, the test images described in this section show easier objects with a round surface. As depicted in Figure 2, all images show the mandatory sundial and one or two table tennis balls in yellow and pink, which have a matt texture. For the actual algorithm of the *Interactive Lighting Detector* only one of this balls is taken into consideration (compare Section 4). Properties like size of the object, the camera angle and the light direction differs in each image.

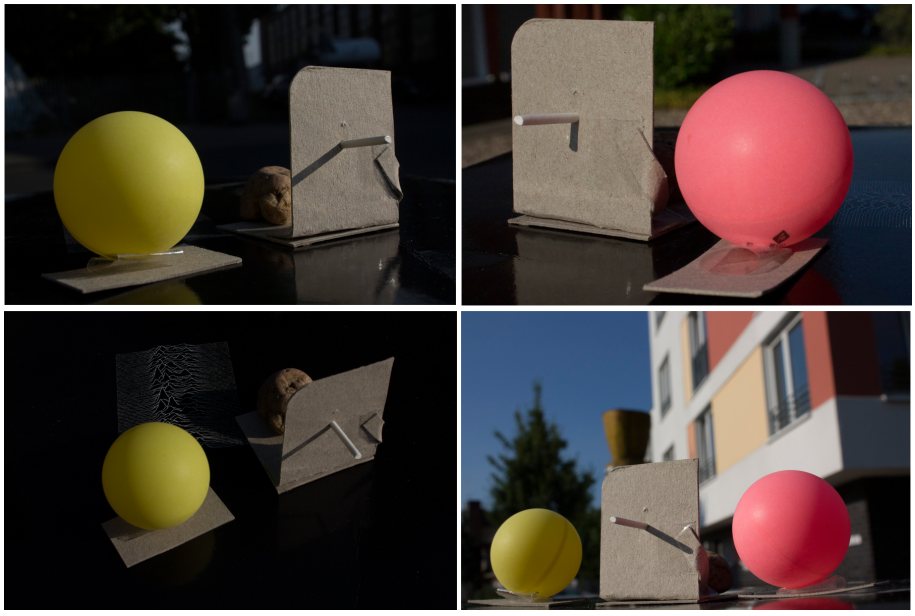


Figure 2: Examples of the test images of the second batch.

4 System

This chapter describes the contour searching method and different approaches to estimate the light direction of an object in detail. In the beginning, the contours of an object of interest are generated in the application to get all pixel coordinates along an adequate contour patch. Followed by the actual computations to ascertain the light direction of an infinitive light source that shines on the object belonging to the contour patch. All necessary steps to calculate the light direction can be done and understood in the GUI described in section 4.5.

4.1 Contours

To estimate the light direction of an object in a digital image, it is firstly necessary to locate the object of interest and get its shape information. All information are described by the contour, which is forced to be perfectly even to guarantee a successful light direction estimation. The most important required information are all pixel coordinates along the outer bound of the object in an ordered way.

Beside a correctly localisation, an application with a good usability is preferred as well. An pre implemented interactive segmentation [**?**] was integrated into the application to generate the affordable contours with manual constraints that are defined by the user. This method is called *Live-Wire* [1] and based on a maximum flow problem. However, several test showed that this extension calculates noisy and fault contours as well as it lacks of robust control options for the user. The main problem was that the generated contour seems to follow the valley between two edges of an RGB image and snap to the most significant edge in a wide neighbourhood. Even more, constraints set by the user did not result in adequate and useful contours.

Due to the fact that those results were unusable, the team decided to extract the contours from perfectly even mask images of the objects of interest like shown in Figure 3. These images are prepared manually in *Adobe Photoshop CS 6* with the pen tool that offers perfectly round and even Bezier curves. All created masks are shrink about almost two pixels to ensure that only pixel information of the actual object are used for further proceedings and none from the background.

If the file name of a mask image is the same as the one belonging to the selected RGB image with the extension `_mask.jpg`, it will be loaded automatically into the application at the same time when the belonging RGB image is opened (compare Section 4.5). This mask data will be used for all further contour calculations.

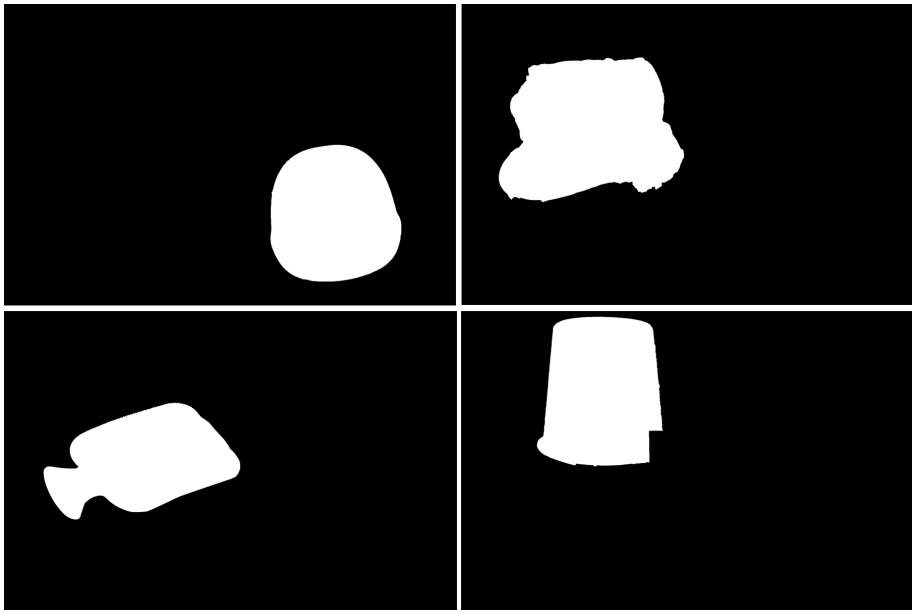


Figure 3: Manual created mask images of the first batch.

4.1.1 Calculate Contours

Several *OpenCV* (compare Section 3.2.2) functions are used to compute all required pixel coordinates along the outer object boundary. In the beginning a common erosion filter with a filter size of three is applied to the mask image to eliminate small outliers and in order to shrink the mask to ensure that it is not located directly or outside the object boundary. In the next step, a Canny filter with a filter size of five is used to extract the edges. Then the *OpenCV* function `findContours()` calculates the vector representations of the binary canny image with the border following algorithm of Suzuki et Al. [14]. All results are not optimized by any approximations because otherwise important coordinate data will be lost. The resulting contours will be printed into the preview of the GUI (compare Setcion 4.5) to allow the user a selection of a suitable sub-contour to feed the light direction estimation.

4.2 Generate Subcontours

As mentioned, the user is forced to select an adequate part of the contour to contain the further calculations. He can do so in the GUI with a simple rectangle selection in the preview. This selection can be renewed, if affordable, and saved. After the storage of all contour points within the rectangle, they are stored and sorted in a so called sub-contour. At this point, the sort is needed because all contour points are stored like a ring buffer that starts at the highest point and then follows counter clock wise the boundary. Thus, it has to be reordered to ensure an open and gapless sub-contour, if the selection includes the top of the complete contour.

After this process, we have a sub-contour that includes all important edge points along the boundary but it still lacks of all pixel coordinates along this part of the

boundary. This circumstance can be solved by making a binary image and print the required sub-contour into it with a line thickness of one pixel. Then a *OpenCV* `LineIterator` is applied to this image. It is treated as versatile implementation of the Bresenham algorithm [3] and allows the notice and storage of every required coordinate. Now, these coordinates can be used for the following approaches of the light direction estimation in a single digital image.

4.3 Illumination Model

To understand the following approaches (compare Section 4.4), it is useful to give a short introduction to the definition of an illumination model. One of the most acknowledged illumination model is the Blinn model [2](compare Equation1) which defines a basic illumination in a three dimensional space. In this model is $\vec{n}$ the surface normal, $\vec{l}$ the light direction vector and $\vec{h}$ is the direction of the maximum highlights. All p variables that are related to the proportions of each reflection type which are part of the sum and s is the amount of the specular reflection. This illumination is finally a simple sum of the ambient a, diffuse d and specular sp light parts.

$$I(x, y, z) = (\vec{n} \cdot \vec{l}) * p_d + (\vec{n} \cdot \vec{h})^s * p_s + p_a = d + sp + a \tag{1}$$

Due to the high number of unknown variables and vectors, the Blinn model is too complex to calculate the 3-dimensional vector $\vec{l}$ with standard approaches like the least square method. Hence, some assumptions to simplify the mode had to be made [7]. The first assumption is to state the current surface reflects light isotropically. This surface has a constant reflectance value r and is illuminated by a faraway infinitive light source. Finally, the angle between $\vec{n}$ and $\vec{l}$ is in the range of 0° and 90°. All previous assumptions lead to the simplified light model of equation 2.

$$I(x, y, z) = r(\vec{n} \cdot \vec{l}) + a \tag{2}$$

4.4 Different Approaches

Three different approaches were tried out because a reliable solution could not be found. All approaches are based on the paper of Johnson and Farid [7]. Two of this approaches remained unchanged (compare Section 4.4.1 and 4.4.2) but the last approach takes the assumptions of Johnson and Farid and expand them (compare Section 4.4.3).
In the following sections, it has to be noted, that we do not have a three dimensional space like almost all other light direction estimations, which are based on images. All realised approaches of Johnson are based on a infinite light source in a two dimensional image space.
The basic idea of the approach is a simplified light model that is described in Section 1. To differentiate the 2D case all parameters are written in upper letters. In 2D Equation 2 is referred to as follows:

$$I(x,y) = R(\vec{N}(x,y) \cdot \vec{L}) + A \tag{3}$$

To standardise the explanations in the following sections a list of necessary variables is introduced:

- $\vec{L} \in \mathbb{R}^2$: vector, which points in the direction of the light source
- $\vec{N}(x,y) \in \mathbb{R}^2$: surface normal at the point (x,y)
- $I(x,y)$: intensity at the point (x,y)
- A: constant ambient light term
- R: constant reflectance value

All equations in this section, as well as its subsections, are taken from [7].
For the second and the third approach, the light vectors are summed up in patches of a size of four pixel.

4.4.1 1. Approach: One Lightvector

The first approach tries to simplify the two dimensional case by solving Equation 4 and getting one final light vector per sub-contour plus the ambient light term. The Matrix M, which contains the surface normals, can be found in Johnsons and Farids paper [7] in Equation (6) and p denotes the number of points on a contour with the same assumed reflectance. All other variables are explained in Section 4.4. Thereby, the reflectance along the entire contour is thought as constantly.

$$E(\vec{L}, A) = \left|\left| M \begin{pmatrix} L_x \\ L_y \\ A \end{pmatrix} - \begin{pmatrix} I(x_1,y_1) \\ I(x_2,y_2) \\ \vdots \\ I(x_p,y_p) \end{pmatrix} \right|\right|^2 = \left|\left| M\vec{v} - \vec{b} \right|\right|^2 \tag{4}$$

This options can be activated in our script `mainwindow.cpp` by setting the variable `usePatches` to `false`. Equation 4 is then solved by using the *Singular Value Decomposition*. The resulting light vector $\vec{L}$ is drawn into the test image to visualize the result.

4.4.2 2. Approach: Averaging Lightvectors

For the second approach, it is not only assumed that the light source is infinite, but also that the reflection, created by it, is constant within each surface patch. A more detailed description of this approach can be found in section 2.2.1 in the paper of Johnson and Farid [7].
It is assumed, that a minimization problem according to Equation 5 needs to be

solved. The related Matrix M can be found in [7] in Equation (8).

$$E_1(\vec{L}^1, ..., \vec{L}^n, A) = \left\| M \begin{pmatrix} L_x^1 \\ L_y^1 \\ \vdots \\ L_x^n \\ L_y^n \\ A \end{pmatrix} - \begin{pmatrix} I(x_1^1, y_1^1) \\ \vdots \\ I(x_p^1, y_p^1) \\ \vdots \\ I(x_1^n, y_1^n) \\ \vdots \\ I(x_p^n, y_p^n) \end{pmatrix} \right\|^2 = \left\| M\vec{v} - \vec{b} \right\|^2 \tag{5}$$

This options can be activated in our script `mainwindow.cpp` by setting `usePatches` to `true` and `useHighestIntensity` to `false`. As stated in Section 4.4.1, Equation 5 is solved using *Singular Value Decomposition* as well. This leads to calculating one light vector per patch. To get the final vector $\vec{L}$ all these light vectors are averaged.

4.4.3 3. Approach: Lightvector with maximum Intensity

This last approach uses the same basic idea as the second one described in section 5. It is also based on dividing the surface normals $\vec{N}$ into patches and solving Equation 5 using *Singular Value Decomposition*. Afterwards, there is no averaging done but the vector $\vec{L}$ belonging to the patch with the maximum intensity is taken as the final light vector.
This idea is not described in the paper of Johnson and Farid [7]. It was a try to make the results of the *Interactive Lighting Detector* more trustworthy and reproducible. This options can be activated in our script `mainwindow.cpp` by setting `usePatches` to `true` and `useHighestIntensity` to `true`.

4.5 Graphic User Interface

As the *Interactive Lighting Detector* is implemented as an half-automatic application, a GUI is needed to give the user the opportunity to make decisions. The GUI leads the user from loading an image to the final results. Here, an expiry is given. For reasons of space all corresponding screenshots can be found in the attachment (compare Section 9).

- **loading an image:** If the user presses the button *Load Image*, he can chose an image from his hard disk drive, which has the file format JPEG or Tiff (compare figure 10). In addition, the mask image is then loaded in the background as described in Section 4.1.
- **contour detection:** Once an image is loaded, the *Run Contour Detection*-button gets visible. After clicking on it the contour is drawn onto the displayed image.

- **sub contour:** As the algorithm does not need a closed contour, the user can select a subcontour by pressing the *Select Part of Contour*-button and drawing a rectangle, including the part of the contour (marked in green) he wants to select (compare figure 11). If he is dissatisfied with his selection he can renew it by pressing the *Delete Selection*. Otherwise he can apply the selection by pressing the *Save Selection*.

- **calculating normals:** By clicking the *Show Normals*-button the surface normals for the subcontour are determined and drawn in red (compare figure 12).

- **calculating lightvectors:** The last step is to calculate the lightvectors by pressing the *Show Lightvectors*-button. The final lightvector is then drawn in blue. If it is chosen to calculate the light vectors for every single patch (compare section 4.4), than all other light vectors are drawn in white (compare figure 13).

To simplify the expiry the GUI can be refreshed by pressing the *Restart*-button. In addition to that the application can be closed at any time Wo?.

5 Evaluation

This chapter includes the detailed evaluation of the final application to get a conclusion about the applicability of the approach in reality and the trustworthiness of the light direction estimation. Thus, all test image of the second batch and some of the first batch were used three times to estimate a light vector with the application. Then all resulting vectors are printed into the images and compared. Every surface normal $\vec{N}$ of a path is printed in red along the green sub-contour while the patch light vectors $\vec{L}^n$ are visualised by white lines. The final estimated light vector $\vec{v}$ is displayed as a blue line.

As mentioned in Section 4.4, three approaches were implemented and two of them were evaluated. The first approach was discarded because it generates very imprecise results due to the assumption of a overall constant reflection on the object's surface. However, it turned out that the results of both evaluated approaches seem to be almost similar and therefore this evaluation focus on the first approach which appear to be more suitable. The same similarity could be observed between the results of the first and second batch of test images. Hence, the second batch is preferred because it is simpler and thus a better reproducibility. All visual observations of the evaluation are listed, discussed and rated in the following sections.

5.1 Evaluation 2. Approach

During the generation of resulting $\vec{v}$ from the test images, it became obvious that almost correctly light vectors could be calculated with the approach. As shown in Figure 4, the final light vector $\vec{v}$ almost points into the direction of the sun clock shadow. Little variations can be related to image noise and numeric rounding. Some results show vectors that are flipped exactly horizontally into the opposite direction.

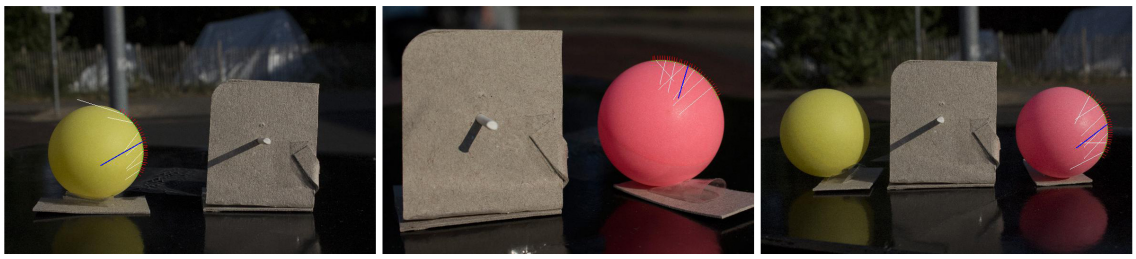


Figure 4: Correct estimated light vectors.

However, most of the approximated light vectors $\vec{v}$ despite the few mentioned correct ones, are significantly wrong. Figures 5 and 6 show that the estimated patch light vectors $\vec{L}^n$ are not coherent and thus the final $\vec{v}$ has huge differences to the validated light direction of the sun clock as well. It is obvious that the norm of $\vec{v}$ does not correlate to the length of the sun clock shadow. All these mentioned estimation errors appear to be randomly and without any correlations that can be determined.

To consider those correlations, the second batch of test images is split into two

groups. The first part contains images that are captured with increasing distances to the object of interest (see Figure 5). In comparison to the general observations, the quality of these results turned out to be almost similar and the lack of coherence is still given. Thus, it can be assumed that the size of image section and the related number of patches have no big influence to the estimation error and proofs a low robustness of the approach.

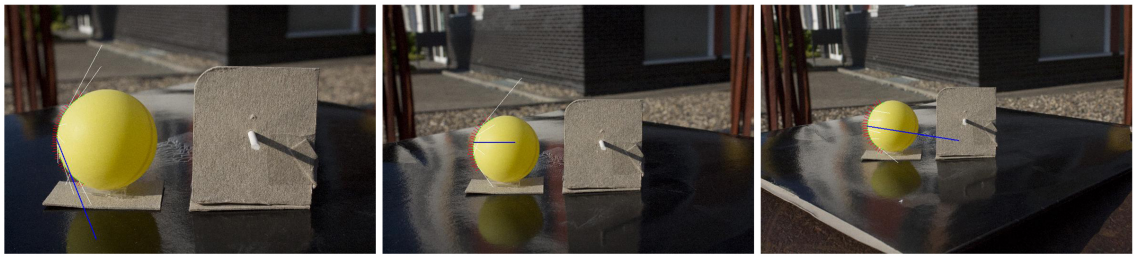


Figure 5: Results of images with increasing distances.

The second part includes images with various light direction to consider if the Johnson approach has problems with special circumstances like the angle between $\vec{N}$ and $\vec{L}$ or a certain light directions. In Figure 6 is a collection of some examples of this group shown. It gives the assumption that the algorithm is correctly implemented because the coherence is in the same range than with the general results. No significantly differences can be figured out related to certain light directions or angles and thus it can be assumed that the least square problem is correctly implemented and the high error rate is related to external influences like image noise, surface reflections and a huge ambient light proportion. ergibt sich die hohe error rate nicht eher weil die Annahmen schrott sind? Oder verstehe ich da was falsch? JA, das sind ja genau die Annahmen, dass diese Sachen nicht gibt.

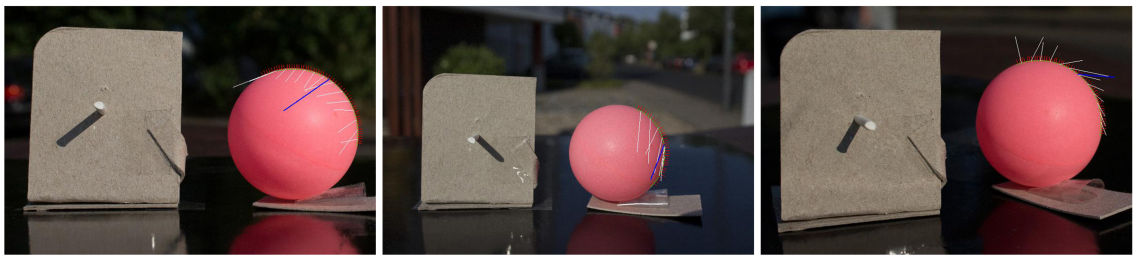


Figure 6: Results of images with various light directions.

This assumption is affirmed by the observation of the sub-contour position influence. The best results can be generated if the sub-contour fits closely and symmetrically around patch with highest intensity as shown in Figure 4. All successful estimations are generated from sub-contours whose selections have the patch of maximum mean intensity almost in the middle and includes a selection of patches that are related to the light direction of the infinitive light source without any shadow parts. The length of these sub-contours needs to be longer if the light incidence is less planar.

Nonetheless, the success is quite sensible as shown in Figure 7. Slightly variances in the length and position of the sub-contour results in significant changes of the light vector and its length. This proofs also that correct results seems to be occasional and can not much forced by user intervention.

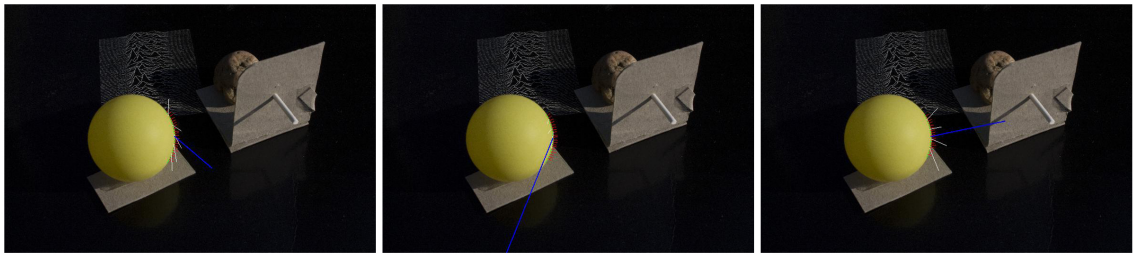


Figure 7: Results of slightly different sub-contours.

The more complex objects of the second batch are evaluate as well and displayed in Figure 8. Here, the image show obviously the randomness of the estimated vectors as well. Especially at the points where the contour changes significantly the direction, the $\vec{L}^n$ seems to have extremely wrong estimations but those outliers can be recognised in the other images like the left one in Figure 7 or the middle one in Figure 5 as well. Hence, the shape of the sub-contour seems to have no influence to the quality and robustness of the Johnston approach.

Figure 8: Results of more complex objects.

5.2 Evaluation 3. Approach

Figure 9 displays the results with the third approach which sets the light vector estimation $\vec{v}$ as $\vec{L}^{max}$ of the patch with the maximum intensity. The directions seem to be as random as in the second approach and the algorithm lacks of robustness and continuity. Even the $\vec{L}^n$ of the images of the first batch point in various direction with no coherence and the significant outliers at contour points where the direction changes can be recognised. In case of the lower left image an almost correct light vector could be approximated as well. Another advantage is the more reproducibility of the final light vector when the sub-contour only differs slightly. These observations yields to the conclusion that there are no significant differences in quality and trustworthiness between both approaches.

Figure 9: Results of third approach with the light vector of the patch of highest intensity.

5.3 Discussion of Evaluation Results

The promised robust and correct results of Johnson [7] can not be reproduced. It is obvious that the authors used very simple and homogeneous objects as well and did not display the patch related light vectors in the complex scenes like the image of prominent people. This leads to the suspicion that only the best results were publicized which could be occasionally estimated with the reproduced implementation, too.

It has to be mentioned, that this evaluation is based on visual impressions, but the results differ very significantly from the validated sun clock light direction, and can be easily recognised. There were no evidences of correlations in the considered images and the final light vectors seems to point into none comprehensible directions. This randomness is proofed by the sensibility of slightly different sub-contours which influence the results vastly. Other assumptions like the influence of the image section size or problems with special angles between $\vec{N}$ and $\vec{L}$ can not be confirmed, beside the horizontally flipped vectors if the angle is in the range of $0°$ and $-90°$.

The comparison of the second and third approach proofs enhance the assumption that the lack of robustness is related to external influences. Those can be additional reflections at surfaces in the direct neighbourhood of the object of interest or a specular highlight in the proximity of the outer object boundary that forge the constant reflection assumption. These assumptions were made in Section 4.3 to simplify the approximation problem in Equation 1. It discards completely the existence of specular reflection parts and the fact that surfaces reflects not ideal isotropic in reality. Another failure is the ignition of surface reflections in the object's environment that can be interpreted as additional light sources which forge the pixel intensities as well.

In addition, it can not be figured out if the complexness of the object's shape has any influence of the quality, because of the lacking correlation. The recognised outliers

can be observed in the results with the perfect round and even contours of the second bath=batch?? as well. Therefore, it is challenging to get a conclusion about the influence of patches at direction changing contour points because they can be extremely outliers and no regularity.

A totally different hint to the none usefulness of this two dimensional approach can be the lack of publications from other authors that work with this method. Most of all other published methods work with multiple views and solve three dimensional problems to estimate the light direction. Thus, it can be assumed that maybe the conversion in a two dimensional space leads to the high lost of precision.

5.4 Conclusion of Evaluation

Hence, it can be considered that the previously made assumptions to simplify the approximation problem do not stand in reality. This model is related to perfect laboratory conditions and objects with an even surface that reflects nearly isotropically. It is obvious the reflections and other external influences distort the results in reality and thus the measuring of the intensity values along the sub-contour is not sufficient. No correlations could be made between the fault direction estimations that consider the lost of important information during the conversion into the two dimensional space. Therefore, it can be assumed that the results are not reliable and completely random.

Thus, the approach of Johnson [7] is definitely not useful for professional validated image forgery. The promised results are not reproducible and this fact leads to the suspicion that only the best results were publicized.

6 Project Management

In this section, the planning and the management of the *Interactive Lighting Lab* is described. As there were only two persons involved in this project the following part is going to be a bit more compressed than the reader might expected it.
To give the customer an all-encompassing idea about the *Interactive Lighting Lab* a project proposal was handed in, which is written in German. It is divided in two sections. First of all, a motivation is given and afterwards the exercise, which should be implemented is determined. It is translated into English and can be found in Section 6.1.
Before starting the actual project, it is wise to make some mindful project plans in order to organize the project. First of all, a catalogue of requirements and specifications (compare Section 6.2) was written to formulate all important basic information of the project.
The project structure plan, which is presented in Section 6.3, serves to arrange the work packages of the project in a sorted way. The relating milestone description can be found in Section 6.4.
For staying in time during the implementation phase, a time exposure is required (compare Section 6.5) as well.
A list with all early detectable risks can be found in the table in Section 6.6.
In Section 6.7 a final resume about the changes during the actual realisation of the project is given.

6.1 Project Proposal

Motivation
Due to progressive changes in image processing programs and high-resolution images, the manipulation of digital images has became easier. A common form of manipulation is the so-called "*image splicing*". Therefore, image regions of at least two images where combined to a new one. The transitions between the individual image parts can get invisible for the user by using accurate and user-friendly image processing tools. According to [6], there is still no algorithm, which makes images completely forgery-proof by adding watermarks to it. There is put much effort in the integrity of images and the recognition of manipulated image parts, despite the lack of prior knowledge of the image content. This should be improved by the *Interactive Lighting Lab*. An algorithm should be established, which proofs the consistence in light directions on various surfaces presented on an image. The light vectors will be conveniently estimated to give an assertion whether the image is manipulated or not.

Work Steps
Until now there is no implementation given by *OpenCV*, which estimates the light vector of an infinite light source on one surface and compares it with the according vector of another surface in the same image. An approach in the programming language *C++* should be implemented using the assumptions of Johnson and Farid [7] to compute, analyse and visualize the light vectors.

The following work stages are necessary:

- Creation of test images with an infinite light source, e.g. in nature on a sunny day
- Implementation of a GUI for the visualisation of the approach
- Implementation of the algorithm of Johnson and Farid [7]
- Optional: The approach might be amended by also taking a local (not infinite) light source into consideration [7]

6.2 Catalogue of Requirements and Specification

Project Manager: Laura Anger
Team Member: Vera Brockmeyer
Supervisor: Prof. Dr. Dietmar Kunz

Project Goals

This project aims to implement an interactive image forensic tool to detect light detections of various objects in an image. The tool is limited to images with infinitive light sources. All resulting direction vectors are compared with each other to give a statement if the image is partly digital modified.

The development of the tool requires the following parts:

- a set of test images with a simulated infinitive light source
- an interactive partial contour detection tool
- a calculation and validation of lighting detection vectors
- a graphic user interface which includes Live-Wire and visualisation tools

The project will be realised in the programming languages C++ with the OpenCV, the Dlib C++ Library and the Qt Library. The release of the project is planned on 4th August 2017.

List of Requirements

Set of Test Images

A set of ten test images is required to validate the image forensic tool. These images have to be captured with an simulated infinitive light source. Thus, all image are taken at a sunny day with no cloudy heaven. Each image shows either different types of objects with surfaces and contours like reflecting metal and glass, diffuse natural

tissues as well as different persons. Another interesting option is the validation of concave and convex objects. Furthermore, a sun clock is placed in each image to display the current light direction precisely. Finally, all collected test images are partly composed together with images of other light.

Calculation and Validation of light detection vectors

The Method of the image forensic tool to calculate and validate light detection vectors of objects in an image is given by the publication of Johnson [7]. Johnson offers an algorithm that calculates the light direction of infinite light sources by minimizing several samples of direction vectors along a contour segment of an object. Each sample is computed by extracting and minimizing features from the inner and nearby neighbourhood of a contour patch. Every required minimization is computed by the least square method of the Dlib C++ library. After a successful computation of the vectors the method computes angles of all calculated vectors and compare them with each other. If the difference between them is off a predefined threshold, the object has probably been retouched into the image.

Interactive contour detection tool

The interactive Live-Wire tool [1] is going to be realized to compute the required surface contour segments of the objects. This tool serves a segmentation algorithm which is initialized and controlled by seed points. Every seed point is set by the user with the mouse cursor and a cost minimisation of local features computes the optimal path between the last two seed points. These required features are extracted with OpenCV functions and minimised with the Dlib C++ library. Furthermore, all specified seed-points and resulting contour segments between them are previewed in the GUI.

Graphic User Interface (GUI)

The GUI includes two main parts. First, the interface of the Live-Wire tool, which provide the interactive generation of the contour segment in the current image. This image is opened and displayed in a preview panel. The second part serves visualisations of the computed final direction vectors, all vectors of each contour patch as well as an indication of the digitally modified image sections. For the implementation is the QT Library used.

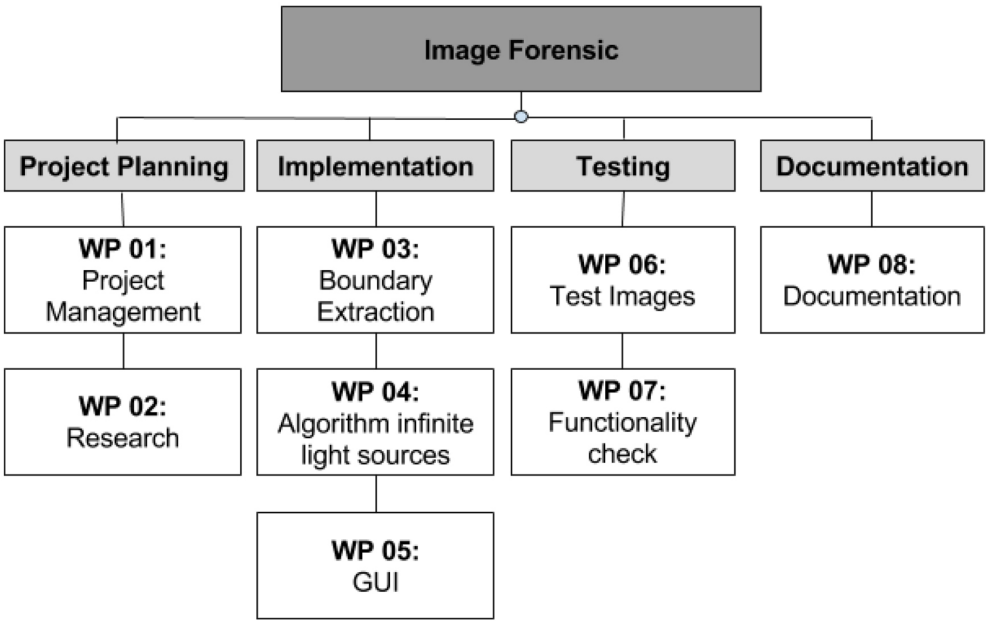
Functionally test of the prototypes

All prototypes are tested with the predefined set of images with a simulated infinitive light source. For the second prototype, the accuracy of the Live-Wire is going to be tested visually by the developers. On the other hand, the angles of each computed direction vector are going to be compared with the angle of the sun clock shadows to validate the functionality of third prototype.

Dates

There will be a first prototype in form of a Paper Mock Up on the 20th of April 2017 and a second prototype on the 9th June 2017. The final version of the Interactive Lightning Detector will be realised until the 17th of July 2017. The deadline for this project is the 4th of August 2017.

6.3 Project Structure Plan



6.4 Milestones

Number	Milestone	Description	How to get there	Planned Date
1	Start Implementation		Tasks 1.1, 1.2 and 1.3 needs to be done before, as well as WP 2.	17.04.2017
2	First Prototype (GUI)	Paper Mockup of the GUI should show the possibilties for the future user.	WP 2 should be concluded.	20.04.2017
3	Second Prototype	GUI is working, as well as the edgedetection	Mostly MS 1 and MS 2 must finished.	09.06.2017
4	Third Prototype	Algorithm by Johnson is implemented and working.	Mostly WP 2, WP 3 must finished.	10.07.2017
5	Final Version	Implementation is ready.	All bugs found while testing, must be fixed.	17.07.2017
6	Submission	This includes source code and the documentation	The final version of the Image Forensic Tool is ready, as well as the documentation.	04.08.2017

6.5 Time Exposure

Nr	Workpackage	Subpackage	Planned Time Period	Time in Days			Description
				min	avg	max	
1	WP 01: Project Management			3	4	5	
1.1		Create Projectdescription	27.03.2017 - 01.04.2017	1	1	1	Brief description of the project. Can be found in our Ilias folder.
1.2		Define Workpackages	01.04.2017 - 10.04.2017	1	2	3	Includes the definition of the Work packages (project-structure- plan.pdf in our Ilias Folder), as well as this time exposure.
1.3		Calculate Risks	01.04.2017 - 10.04.2017	1	1	1	This plan describes the risks, which may arise. Will be found in our Ilias Folder.
2	WP 02: Research			8	12	16	
2.1		General Topic Search	13.03.2017 - 27.03.2017	3	4	5	
2.2		Boundary Extraction	03.04.2017 - 17.04.2017	2	3	4	Search a semi-automatic algorithm and become familiar with its possible implementation
2.3		Algorithm of Johnson	03.04.2017 - 17.04.2017	2	3	4	Understand the algorithm of Johnson and find its mathematical basic functions in a library
2.4		GUI Implementation	03.04.2017 - 17.04.2017	1	2	3	Find out about the possible ways to build a GUI in c++
3	WP 03: Boundary Extraction			8	12	14	
		Implementation of Live-Wire	24.04.2017 - 09.06.2017	8	12	14	Implementation of the interactive Live-Wire tool; interagtion of the tool in the GUI
4	WP 04: Algorithm of Johnson			6	11	18	
4.1		Compute Lighting Vectors as Johnson	10.06.2017 - 10.07.2017	4	7	12	Exctract Features for Lighting Vector Computation, compute Minimizations with Math Lib
4.2		Validate Lighting Vectors	10.06.2017 - 10.07.2017	2	4	6	Measure Angles of each surface and validate them
5	WP 05: GUI			3	5	7	
5.1		Basic Surface	24.04.2017 - 09.06.2017	1	1	1	The GUI frame can be seen on the screen and basic functions, like loading a new image are included
5.2		Boundary Extraction	24.04.2017 - 09.06.2017	1	2	3	The GUI gives the user the possibilty to mark areas in the image to allow the algorithm described under 3 to run the semi-automatic boundary extraction
5.3		Visualisation	24.04.2017 - 09.06.2017	1	2	3	Feautures like object boundaries and the light vectors can be drawn into the images.
6	WP 06: Test Images			2	2	2	
6.1		make images	15.05.2017-21.05.2017	1	1	1	Pictures with an infinite light source are needed. The presented objects should differ in number and form as well as viewing angle.
6.2		select images		1	1	1	An adequate number of useful images must be selected
7	WP 07: Functionality check			3	5	7	
7.1		Testing	10.07.2017-14.07.2017	2	2	2	External Parties should be invited to test the application. This includes NO usability study.
7.2		Correction	14.07.2017-17.07.2017	1	3	5	The most pressing issues must be fixed.
8	WP 08: Documentation			12	18	24	
8.1		write documentation	17.07.2017 - 03.08.2017	10	15	20	Description of the actual system as well as its functionality.
8.2		add comments to source code	17.07.2017 - 03.08.2017	1	2	3	Clean up the soucre code and add descriptions.
8.3		print documentation	04.08.2017	1	1	1	Print documentation and submission.
			Estimated Duration	**45**	**69**	**93**	

6.6 Risks

Description and Source of the Risk	Possible Area of Appereance	Effects	Preventive Measures
1. Technical Problems			
1.1 Implementation Problems	mostly Implementation	Some parts of the implementation might take longer than planned. Maybe other solutions need to be taken into consideration	Try to get to know how complicated the implementation is during the research period
1.2 Algorithm	mostly Implementation	Sometimes the described algorithm is limited. For instace the described results can only be achieved using a specific type of input data.	One should read the according paper carefully. Sometimes more or less hidden hints are given on how good the algorithm works.
1.3 GUI latency	Preview of image	preview of image has a latency which is reconsiable for the users	regard an low costly implementation and prevent from unnecessary memory and code implementations
1.4 wrongness of Lighting Vectors	concave and convex objects	lighting direction is misinterpret because for concave objects the lighting seems to come from the opposite direction	observe problem and figure out characteristics of the problem
2. Time Problems			
2.1 time-collision with other projects	all Workpackages	On Task or a whole Workpackage can't be finished in time.	Try to plan all upcoming projects as detailed as possible to avoid time overlap
2.2 false planning	all Workpackages	It is not possible to stick to the Sheduling	Try to make a thoughtful project planning and do not forget buffer time
2.3 illness	all Workpackages	Restriction of workability and thus problems with finishing a task or a even a whole workpackage in time.	

6.7 Conclusion on the Project Progress

In general, the project progress went well and most of the project goals were achieved. Therefore the *Interactive Lighting Detector* is a complete working application. The final system contains a GUI, in which the user can select parts of the contour (compare Section 4.1).
However, some of the project goals were too ambitious for the resources capacity in the scheduled time window. The planned *Live Wire* algorithm was replaced by simply taking a binary mask image of the actual image and detecting the contour as described in Section 4.1.1. In addition, it is only possible to calculate the light vector for one object in a scene. It was planned to detect light vectors for several objects to give the opportunity of comparing the directions to each other. But nevertheless, the calculated light vector can be compared to the shadow created by the sundial.
The determination of the light vectors is not working trustworthy. Reasons for that are discussed in evaluation (compare Section 5) and summed up in the conclusion (compare Section 7).

7 Conclusion and Prospect

The project team developed a working light detection application and most of the project goals were achieved. Unfortunately, the integration of the interactive segmentation tool *LiveWire* failed, because of its lack of precision and controllability. Due to the challenging project schedule, the team decided to realize a segmentation, which is based on manually created mask images. Maybe, there exists a better solution like the *GraphCut* method [13] that can be integrated into the framework and offer similar results with less user effort.

It was planned to detect light vectors for several objects to give the opportunity of comparing the directions to each other. This goal needs to be implemented in future, as well, but it is dispensable because the approach appears to be not trustworthy. Several assumptions were made in the beginning to simplify the approximation problem and cause big trouble in reality. External influences, like ignored reflections in the object's neighbourhood and the disregard of specular reflection on its surface, forge the results extremely.

8 Division of Labour

All parts of the *Interactive Lighting Detector* were implemented as a team, which supported each other with their responsibilities. Laura Anger mainly focused on the development of the GUI and the realisation of all necessary systems of equations for the various approaches and their solution, using least square method. Vera Brockmeyer did the capturing of both test image batches, build the framework and implemented the contour extraction and the output of the resulting data.

The writing of documentation is divided as follows:

Laura Anger wrote the sections 2, 3, 4.4, 4.5 and 6 including all subsections. The other parts 1, 4.1-4.3, 5 as well as 7 including all subsections were written by Vera Brockmeyer.

9 Attachment

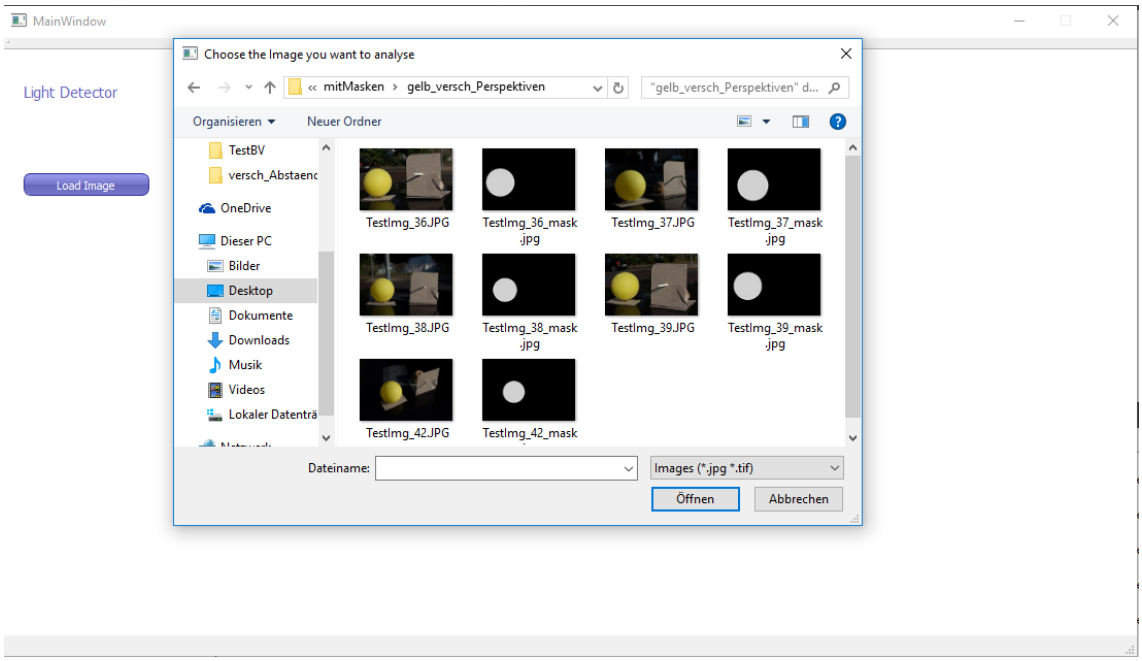


Figure 10: GUI: Loading an Image.

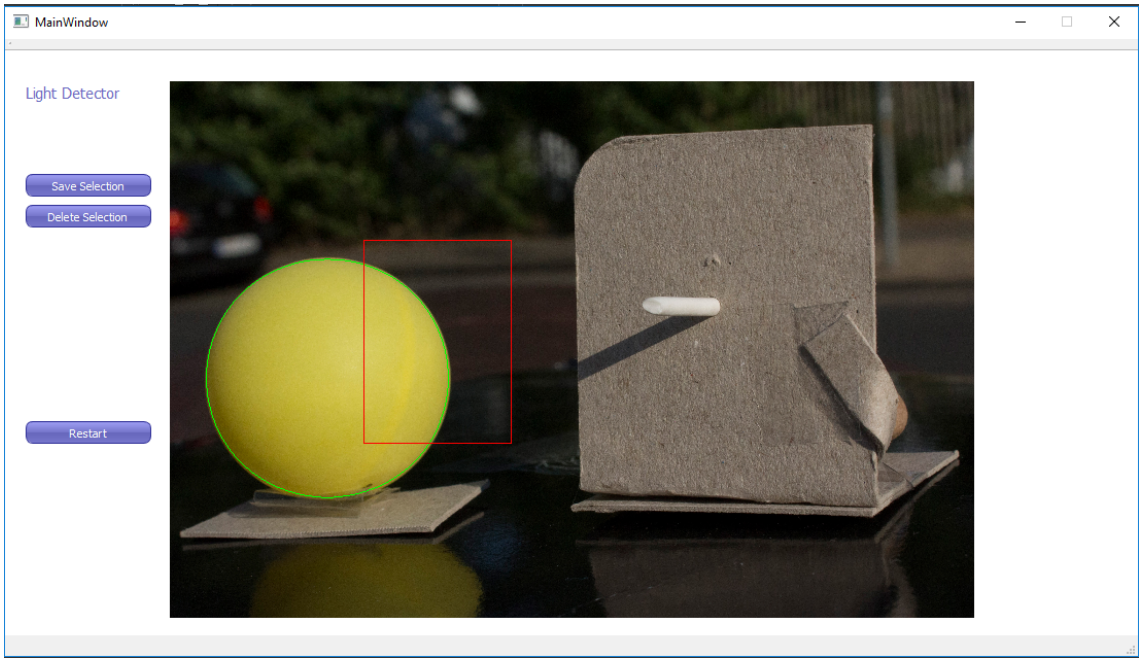


Figure 11: GUI: Loading an Image.

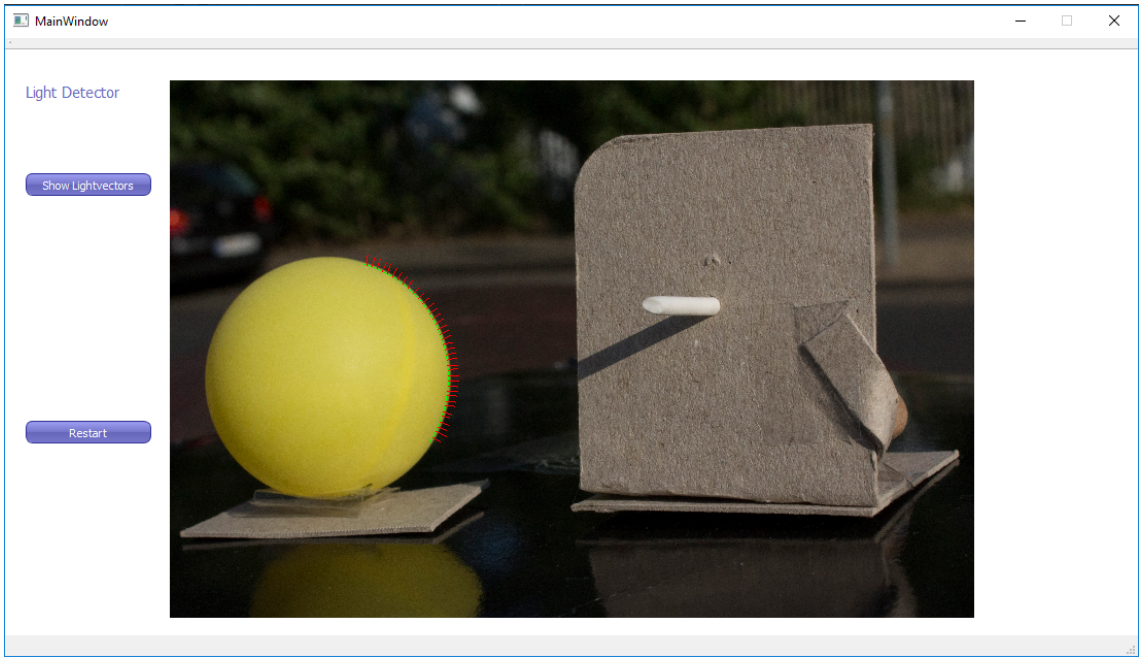


Figure 12: GUI: Calculating Normals.

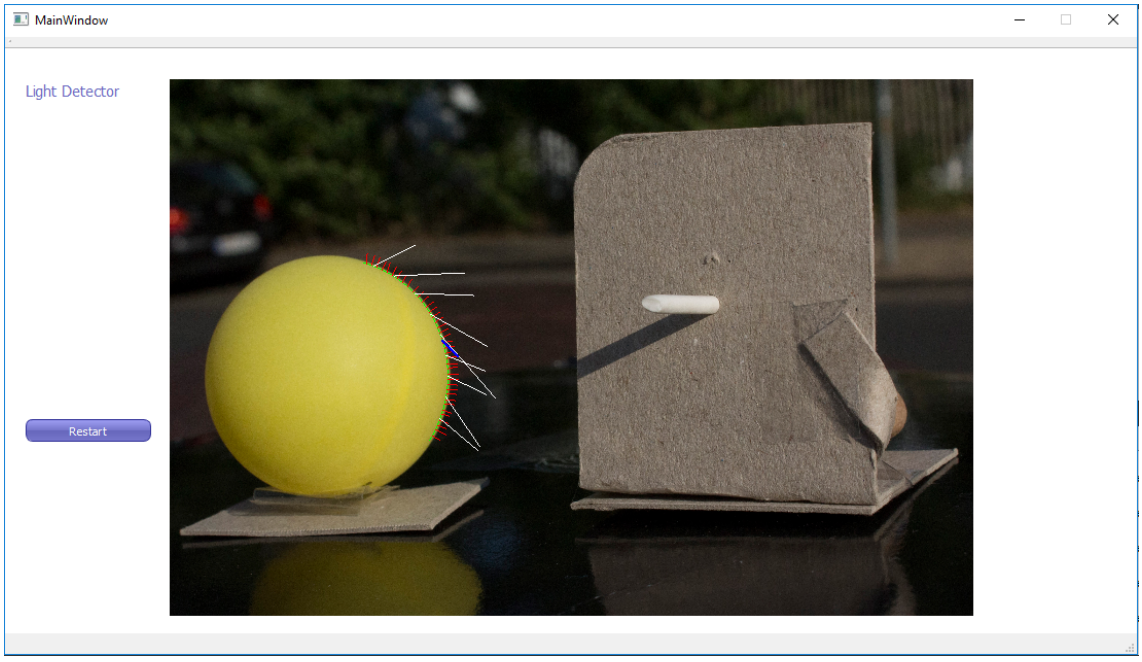


Figure 13: GUI: Calculating Lightvectors.

References

[1] William A. Barrett and Eric N. Mortensen. Interactive live-wire boundary extraction. *Medical Image Analysis*, 1(4):331 – 341, 1997.

[2] James F. Blinn. Models of light reflection for computer synthesized pictures. *SIGGRAPH Comput. Graph.*, 11(2):192–198, July 1977.

[3] J. E. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[4] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek. A brief introduction to opencv. pages 1725–1730, May 2012.

[5] H. Farid. Image forgery detection. *IEEE Signal Processing Magazine*, 26(2):16–25, March 2009.

[6] Yu-Feng Hsu and Shih-Fu Chang. Detecting image splicing using geometry invariants and camera characteristics consistency. In *ICME*, 2006.

[7] Micah K. Johnson and Hany Farid. Exposing digital forgeries by detecting inconsistencies in lighting. In *Proceedings of the 7th Workshop on Multimedia and Security*, pages 1–10, New York, NY, USA, 2005. ACM.

[8] Micah K. Johnson and Hany Farid. Exposing digital forgeries through specular highlights on the eye. In Teddy Furon, François Cayre, Gwenaël J. Doërr, and Patrick Bas, editors, *Information Hiding*, volume 4567 of *Lecture Notes in Computer Science*, pages 311–325, 2008.

[9] T. Van Lanh, K. S. Chong, S. Emmanuel, and M. S. Kankanhalli. A survey on digital camera image forensic methods. In *2007 IEEE International Conference on Multimedia and Expo*, pages 16–19, July 2007.

[10] P. Nillius and J. O. Eklundh. Automatic estimation of the projected light source direction. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–1076–I–1083 vol.1, 2001.

[11] Qt. Qt Designer Manual. http://doc.qt.io/qt-5/qtdesigner-manual.html. retrieved: 13. July 2017.

[12] Ray Rischpater. *Application Development with Qt Creator - Second Edition*, volume 2. Packt Publishing, 2014.

[13] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.

[14] Satoshi Suzuki and KeiichiA be. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*, 30(1):32 – 46, 1985.