

DESY

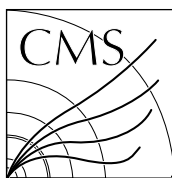
DESY SUMMER SCHOOL 2012

An optical deformation measurement setup

Author:
Sander VANHEULE
Ghent University

Supervisor:
Andreas MUSSGILLER
DESY
CMS Group

6th November 2012



Abstract

This document reports on the work done during the DESY summer school 2012 on the optical deformation measurement setup for the Atlas and CMS groups. The composition of the setup is discussed, as well as the redesigning of the software, and a more extended discussion on the use of colour filters.

Contents

1	Introduction	3
2	Setup	4
2.1	Basic geometry	4
2.2	Composition	4
2.2.1	Illuminated grid panel	5
2.2.2	Image acquisition	6
2.2.3	Temperature control	7
2.3	Calibration	7
3	Software refactoring	8
3.1	Model-View-Controller	8
3.2	Peripheral communication	9
3.3	Dot finding algorithm	10
3.4	Surface reconstruction	10
4	Colour filters	12
4.1	Colour space representations	12
4.2	Colour recognition	13
4.2.1	Filter properties	13
4.2.2	Hue drift	13
4.2.3	Brightness considerations	17
5	Measurement	18
6	Conclusion and outlook	19

Chapter 1

Introduction

When a device consisting of several different materials is cooled down or heated, mechanical stress might occur. This stress is caused by the different thermal expansion coefficients of the materials, causing parts that are fixed together to expand or contract by unequal amounts. This may then result in a deformation of the device. Providing the surface of the deformed device under test (DUT) is sufficiently reflective however, one may use the reflection of a grid of illuminated dots to track deformations. As the surface deforms, the reflected image will deform accordingly.

For the high luminosity upgrade of LHC after 2020, this kind of complex devices are going to be installed in the CMS and ATLAS detectors [7, 2]. An example is shown in figure 1.1. Since the constituent material properties are not always very well known, simulations provide an initial idea of the deformation when cooling down to -20°C , but these need to be checked. As these deformations might be of the order of only tens of microns, measuring these for large surfaces is no easy feat. A setup providing the necessary means to check these simulations, from cooling down the device to measuring the deformation, will be introduced in this report. The device and its peripherals will be introduced in chapter 2. The work performed for this setup will then be discussed more extensively. Finally, an initial deformation measurement will be discussed and an outlook will be provided.

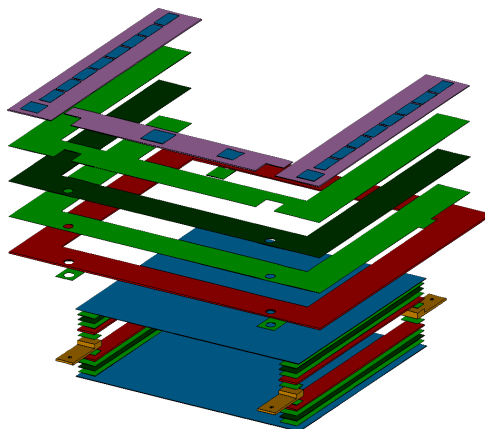


Figure 1.1: A blow up of one of the proposed modules for CMS with the high luminosity upgrade. Different colours represent different materials.

Chapter 2

Setup

The experiment consists of a back illuminated dot grid, any device that has a reflective surface, and a camera to capture the reflection of the dot grid off the device. When the surface is deformed, the slope locally changes, making the reflection point to a different location on the dot grid. Thus the reflected image of the dot grid is distorted. Given a certain reference, converging points indicate an upward or convex deformation, diverging points a downwards or concave deformation with respect to the observer — completely analogous to the funny images one may see of him- or herself when standing in front of a wobbly mirror at a fair. One may compare two images to determine the deformation of the one, relative to the other. The pictures in figure 2.1 can be used as indicated, with the first as the reference, or the other way around, where algorithm will then find the opposite deformation. This also means that no absolute measurements as tilts or surface shapes can be obtained from this information.

2.1 Basic geometry

A basic schematic of the setup is shown in figure 2.2. Both the camera frame and dot grid assembly are connected to a common hinge. Furthermore, the camera has some extra freedom as it can slide from the outer to the inner side of the supporting frame, where the hinge is located. This allows some flexibility of the experiment, depending on the requirements of the measurement that is to be performed. Currently, however, both frames are still positioned at a fixed angle and the camera can only be placed in the outermost position.

2.2 Composition

The experiment consists of several parts that are individually controllable via the workstation at the experiment. These parts are controlled via serial or USB, either directly or (partly) via an intermediate

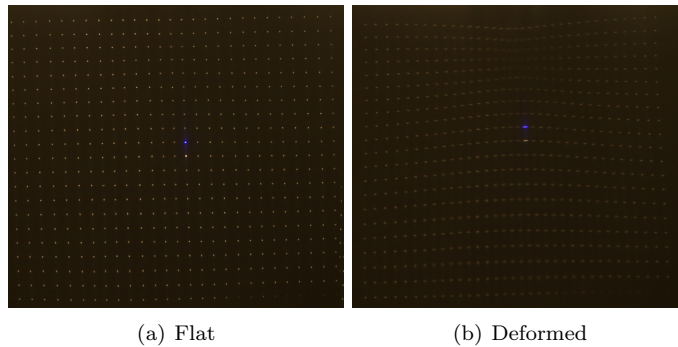


Figure 2.1: An example of two pictures that could be used for a deformation measurement.

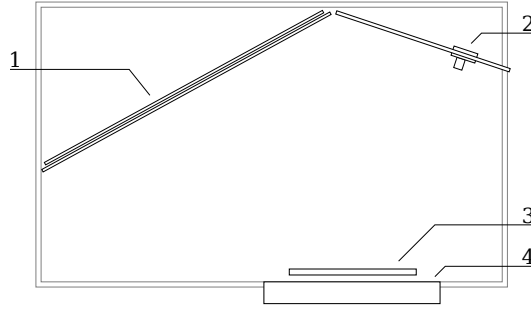


Figure 2.2: Basic layout of the optical deformation measurement setup. Indicated are (1) the dot grid assembly, (2) the camera frame with camera, (3) the DUT, and (4) the granite table.

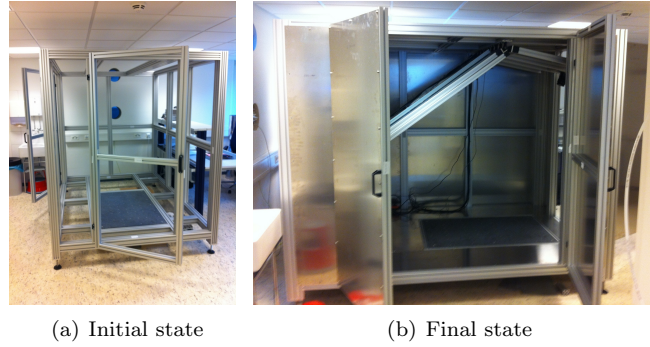


Figure 2.3: The progress made with the assembly of the setup during the summer school programme. The initial and final state are shown in (a) and (b)

relay switch card. This relay switch is controlled via USB and allows to remotely power up to eight independent power sockets. Currently, five are used for the LED-panels (see section 2.2.1), one for the camera (see section 2.2.2), and one for the reference LEDs (see section 2.2.2). Stainless steel sheets are used for shielding the cage. This ensures that the main light source inside the setup is the dot grid and convection of air is limited to the inside of the experiment.

2.2.1 Illuminated grid panel

The illuminated grid panel, whose reflection is used for deformation tracking, consist of a back-illuminated sheet of transparent plastic where a dot grid (negative) was printed on, i.e. only the dots are transparent. The illumination consists of five commercially available LED-panels, arranged as show in figure 2.4. LED-panels were chosen to achieve uniform illumination, while limiting the thickness of the full grid panel assembly, thus allowing for the grid to be placed as far from the DUT as possible. The supporting frame consists of two subframes, one for the grid sheet, and one for the LED panels. The latter is slightly smaller and both are fixed together with right angle pieces. Furthermore, the different LED panels are supported by transparent plastic bars fixed perpendicularly to the grid sheet (origin of the spacing between the panels in figure 2.4). These bars provide the extra support ledges for the light panels, and also ensure that the grid sheet does not sag down.

The metal light panel edges and panel spacing create areas in the illumination plane that do not output any light. Initially, the grid sheet was transparent and nothing was in between the sheet and lighting, thus the reflected image was that of the LED panels, as seen *through* the dots in the print. As can be seen in figure 2.5(a), this produced a dark band in the image. A very dim illumination of the dots was still achieved due to multiple reflections inside the panel frame. Because of the shadow band, it was impossible to achieve a fully illuminated, continuous dot grid. For larger surfaces, such as the ALTAS detector parts [2], this would render it impossible to correctly reconstruct the full surface deformation. After some preliminary tests, it was decided to both sand the back of the grid sheet, and put a layer of

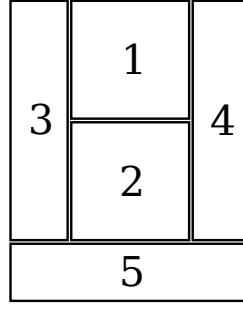


Figure 2.4: The layout of the panels with the numbering as used in the software (bottom view).

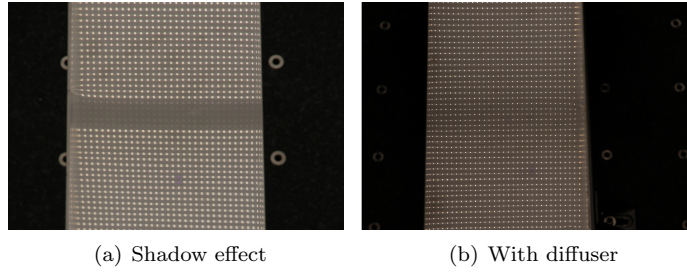


Figure 2.5: Shadow effect before (a) and after (b) adding the diffusive layer.

translucent paper on the back of the grid sheet. As a result, the light coming from the light panels is scattered much more efficiently due to this added diffusive layer. This can be seen in figure 2.5(b), where the shadow region has almost fully disappeared compared to figure 2.5(a).

Employing the relay switch to remotely control the individual panels, this allows the user to determine the size of the illuminated plane. When using a small device, for example only panels 1 and 2 could be powered. Furthermore, these can be powered only when required for picture taking, thus reducing the amount of heat dissipated into the setup.

2.2.2 Image acquisition

To acquire pictures of the reflected dot grid, a Canon EOS 550D DSLR camera is installed in the setup on the location indicated in figure 2.2. It is connected via USB, but due to technical reasons, the maximum USB cable length is limited to approximately 5 m, provided an electrically suitable cable is used [1]. During tests using a maximum length cable however, it was found that the connection to the camera was unreliable and dropped at random times. This is not allowable during operation, where the camera connection is to be reliable for an extended period of time. As such, the camera is currently connected to the workstation via a USB-over-Cat 5e extension cable to allow for enough cable length, while maintaining a reliable connection.

Calibration of the image is achieved by placing four reference LEDs near the corners of the granite table. These will serve as absolute reference points in the image. Tracking their position will provide information on whether the image has drifted or rotated during the course of the measurement. Using this information, the position of the reflected dots can be corrected to reduce systematic errors. Electronic static in the camera sensor could also be a source of errors, something that could be reduced by choosing a low ISO value or maybe even averaging out multiple pictures. These possible sources of errors still have to be checked.

The communication with the camera was established via libgphoto2, which is discussed more extensively in chapter 3. A discussion of some of the camera settings may be found in chapter 4.



Figure 2.6: The device that was used for the testing of the setup. The micrometer screw is visible in the centre of the image.

2.2.3 Temperature control

In order to cool down the detector parts examined by the setup, a Julabo FP50 chiller will be connected in the future. The chiller is connected to the workstation via a serial port and the communication is implemented and tested. When the air present in the setup is cooled down to -20°C , the water vapour present will most likely condense onto the DUT. This might not only damage the device and setup, but will also render the acquired pictures useless due to refraction by the water droplets or ice present on the reflecting surface. For this reason, the chamber will be flooded with dry air so that no condensation of water vapour can occur.

Temperature readout is arranged by means of a Keithley 2700 multimeter that can read out up to ten temperature sensors. Different types are supported and currently PT100 (four-wire) sensors are in use. These have been provided with sufficient cable length to allow for enough freedom in positioning them on or around the DUT.

2.3 Calibration

Before the setup can be used to correctly measure deformations, it has to be calibrated. For this, a calibration device consisting of a stainless steel sheet deformed by a micrometer screw will be used, similar to the one shown in figure 2.6. The final calibration device will also be able to rotate in the horizontal plane. This calibration device will be calibrated itself by means of imaging with a microscope. This will provide an (absolute) height measurement from which the deformations can be calculated and compared to the values found with the optical deformation measurement setup.

Chapter 3

Software refactoring

The original *defo* software — as in ‘deformation’ — already implemented a large part of the functionality and came with an extensive documentation file that described how the underlying algorithms worked and what work was still to be done. It was written using the Qt library with the Qt Creator IDE. The documentation allowed me to quickly get started with using the software and code, without knowing all the intricate details and structure of the source files.

There were however, some problems. The most visible ones were that the graphical user interface (GUI) had a fixed size and that disabled elements were barely discernible from enabled ones due to a customised colour palette. The former results in an interface that does not fit on small displays, and wastes space on large displays. The latter made the user think a button is active, while in reality it was not. This lack of a clear visual cue in the GUI severely reduces the user friendliness. As the GUI had become rather complex, designing it with the Qt IDE had made the main file, containing the code for the user interface, very large. The main file implemented all the callbacks from the user interface, rendering its functionality very diverse, and making it hard to find anything back. Furthermore, device control was handled directly by the GUI callbacks. This meant that these had to update themselves *and* any other elements that depended on the device status. Such a design is prone to errors as one can easily forget to update some elements when adding new functionality.

3.1 Model-View-Controller

In an advanced user interface, as the setup requires, information can be displayed in different ways. Control over this information might be implemented in diverse ways, as to make the GUI as intuitive, and thus user friendly, as possible at all times. In such a scenario it is desirable to separate storage of information with the graphical representation. This can be achieved by using a coding scheme that is based on the *model-view-controller* design pattern, where visualisation (view) and control (controller) of the data are separated from the representation and storage (model). As shown in figure 3.1, the user now only interacts with the controllers, which in turn update the models. This signals the views that changes have been made to the data, whereupon they respond by updating the displayed information. This single point of information — the model — allows for a cleaner structuring of the code.

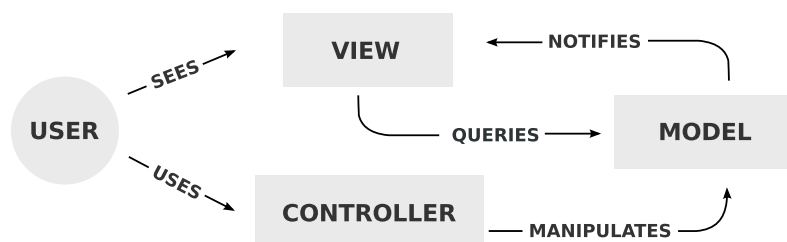


Figure 3.1: A schematic representation of the interactions in the MVC design pattern.

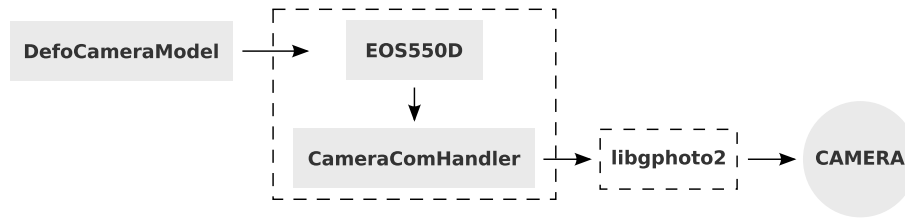


Figure 3.2: An example of the device communication path in the defo software. Rectangles denote specific classes, dashed lines represent libraries.

3.2 Peripheral communication

An important part of the software consists of controlling setup components as the LED-panels, the camera, etc. In order to be able to provide a consistent interface for this in the code, several layers of abstraction exist. For each device a separate library is compiled that combines two of these layers, the (low-level) hardware communication and setup, and protocol implementation using the low-level class. The libraries provide a real device class, and a fake device class for testing purposes¹. The advantage of this approach is that e.g. serial port communication only has to be implemented once, and can then be reused for the device specific protocol implementations.

Most of the devices were already implemented using this scheme, except for the camera. The old implementation relied on calling the gphoto2 executable from the defo software, instead of calling the libgphoto2 library [3]. Camera options were also hard-coded into the software, which is unwanted if the camera is ever to be upgraded. The current implementation, as shown in figure 3.2, provides a simple wrapper (`CameraComHandler`) around `libgphoto2` that simplifies the interactions with the camera. Then a class exists (`EOS550D`) that implements the specifics of communication with a Canon EOS 550D. These two classes compose the device communication library. In the defo software itself, the `EOS550D` class is called from the communication library by the camera model class (`DefoCameraModel`).

The single point of information in the defo software concerning the camera status, is now the camera model. Several GUI elements can connect to the notifications this model sends out, and these elements are shown in figure 3.3. Elements that can be manipulated also serve as controller for the model. This merger of view and controller is sometimes called an ‘interface’ (for the model). For example, if one tries to enable the camera, one simply ticks off ‘Enable camera’, an action which tells the model to enable the camera communication. When the model then tries to enable the camera, but fails, the views are notified of the new status, being ‘off’. When this signal reaches the check box, it unchecks itself to match the current model information. Note that at no point the check box, or any other view, checks if an action succeeded, but that the model will always provide this information. The combo boxes also connect to the current camera status, so that they will enable or disable themselves to reflect the current status. This is an improvement over the old software, where the callback handler of the check box would also have controlled the combo boxes.

The `DefoCameraModel` class is a subclass of an abstract device class that provides a common interface for all the device models. A specific model was developed and tested for each device, as well as widgets for testing these models. The test widgets for the camera and Conrad switch can be seen in figure 3.6. The widgets for the chiller and multimeter are not shown.



Figure 3.3: Several widgets connecting to the camera model are shown, as well as a button to manually load a picture (‘Load file’) that was acquired earlier.

¹Except for the Canon EOS 550D, for which currently only a real device class exists.

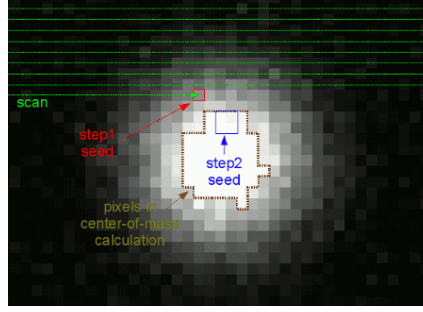


Figure 3.4: A graphical representation of the three-step point finding algorithm.

3.3 Dot finding algorithm

To find the reflected dots in the pictures, a three step image processing algorithm is employed, as illustrated by figure 3.4. Shown in green is the path followed by the scanner. Line per line is checked for pixels whose brightness Y' (determined by equation (4.4)) exceeds a first threshold value Y_1 . When this occurs at the point (x, y) (red square), the scanning halts and the average brightness of a region of (2×2) pixels slightly further down the picture, at $(x + 2, y + 2)$ and shown in blue, is calculated. If this average brightness exceeds a second threshold Y_2 , a square region, about the size of figure 3.4 is centred around (x, y) . Then the pixels, with position \mathbf{r} , within this region whose brightness values exceeds a third threshold Y_3 , indicated in brown, are used to calculate the centre of gravity \mathbf{R} of this dot, according to equation (3.1). The square is then centred at \mathbf{R} and the calculation is repeated three times to allow for potentially clipped pixels to enter the region as \mathbf{R} relaxes to its true position.

$$\mathbf{R} = \frac{\sum_{Y' > Y_3} Y' \times \mathbf{r}}{\sum_{Y' > Y_3} Y'} \quad (3.1)$$

Originally, the software allowed to select a certain region of the picture for dot searching. This functionality was maintained in the code, but is as of yet not exposed in the new GUI. It was used however, to implement parallel processing of the image. The workstation used with the setup has eight hardware threads available. Where an image containing $\mathcal{O}(2000)$ points originally took $\mathcal{O}(1 \text{ min})$ to process, it was speeded up by a factor of 5 – 6 by splitting the image up in 6 pieces and handing each to a different processing thread. Care has to be taken to avoid double counting of dots, but this can be achieved by requiring the found centre-of-gravity to be inside the search area given to the thread.

To allow for user friendly configuration of the algorithm's parameters, a preprocessed version of the image is shown in the GUI. This can be seen in figure 3.6 and figure 4.4(c). The false colours in the images indicate the range the pixel's brightness is in. They are coloured black, red, green or blue for the respective ranges $[0, Y_1]$, $[Y_1, Y_2]$, $[Y_2, Y_3]$, and $[Y_3, 255]$.

3.4 Surface reconstruction

With the necessary dot finding algorithm in place, the software is able to locate the reflected dots in the acquired images. In order to reconstruct the deformation, the dots of two images need to be compared, and these dots have to be the reflection of the same dot on the grid. This requires a certain reference dot from which the dot grid can be reconstructed. The chosen approach is to colour one dot and take this as the reference point. The indexing is then performed by calculating the average spacing of the dots and using this average distance to leap from dot to dot. Scanning the list of found dots in the horizontal and vertical direction, the dots are indexed in an (integer) (x, y) fashion. By then comparing the position of a dot with the same indices in two pictures, a shift can be determined that, given the setup parameters, can be translated into a slope change. Once the slope change for each point has been calculated, a square spline is fitted through these points. This is a piece-wise smooth and continuous curve built up of parts of the form given by equation (3.2), fitted per two adjacent points, independently for the x- and y-direction. Using the two slopes, the parameters A and B can be determined. C is then set by requiring the spline

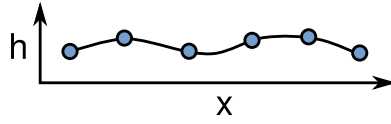


Figure 3.5: A depiction of a spline represented (piece-wise) by equation (3.2).

to be continuous. By requiring the reference point — with index $(0,0)$ — to be at $h = 0$, the height difference for each dot is determined. Note that this ‘height’ is actually the deformation, it will only physically be a height if the reference surface was indeed flat.

$$h(x) = Ax^2 + Bx + C \quad (3.2)$$

The surface reconstruction was already implemented in the original software. This part underwent code deduplication and only some slight refactoring, and was not yet incorporated into the test program that was written. The measurement performed at the end of the summer school thus combined both the new and old software (see chapter 5).

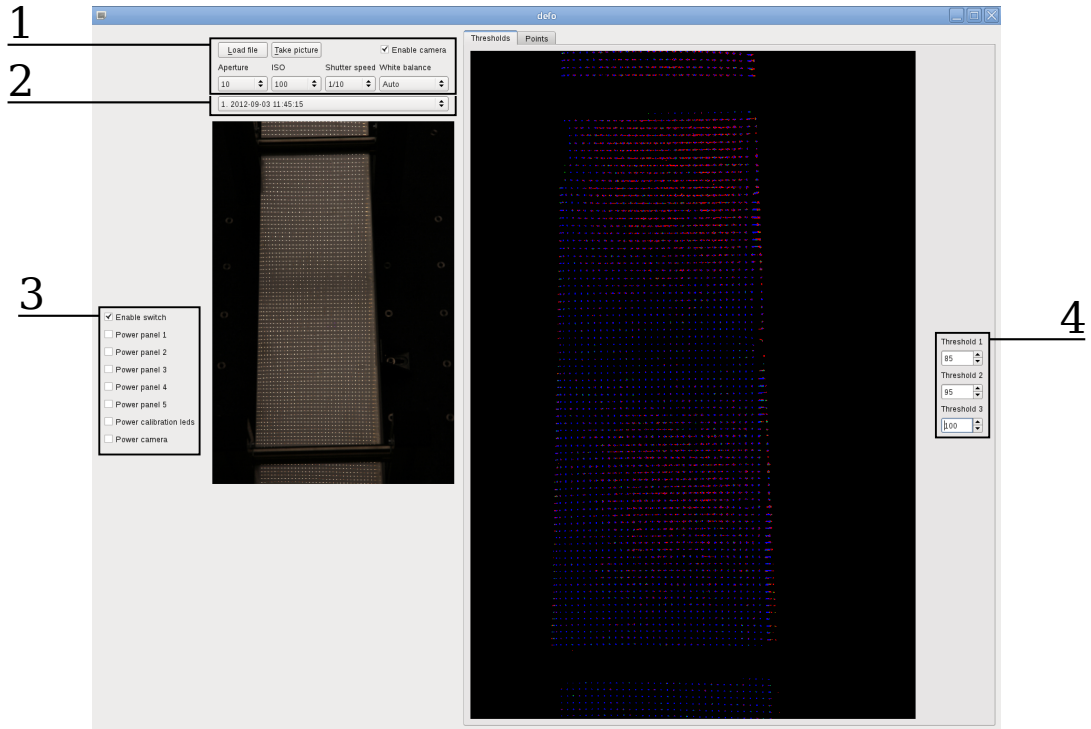


Figure 3.6: A screenshot of the (testing) interface developed during the course of the summer school. Shown are the widgets used for testing the (1) image acquisition, (2) measurement selection, (3) switch communication, and (4) dot finding configuration.

Chapter 4

Colour filters

As the optical deformation setup requires a reference point in the reflected dot grid, one has to differentiate one dot from the others. The chosen approach is to require the reference dot to have a different colour, currently achieved by placing a coloured light filter in front of the dot on the panel. This colour needs to be sufficiently saturated to provide a distinct hue, but also not too dark, so that it can still be recognised without overexposing the picture.

4.1 Colour space representations

A very common representation of the colour space is the red-green-blue (RGB) space. As it resembles the way the human eye detects different colours, technology to record and display realistic colour images has been developed using this scheme in the course of history. The separate red, green and blue components each indicate how bright this component is and in this way e.g. a computer monitor renders different colours. The RGB representation however, does not resemble the way we *perceive* colours. In this light, more intuitive colour schemes have been developed. One of these alternatives is the hue-saturation-value (HSV) representation. Representations of the RGB and HSV schemes can be seen in figure 4.1. In the HSV scheme all colours lie on a circular coordinate in the hue-saturation plane. For example, one usually defines the hue of red as $H = 0^\circ$. The complementary colour, cyan, then has $H = 180^\circ$. In this definition (pure) green and blue lie at respectively $H = 120^\circ$ and $H = 240^\circ$, which allows for a conversion of the RGB coordinate to the hue-saturation plane according to equation (4.1) and equation (4.2), analogous to how one converts Cartesian coordinates into polar coordinates. It uses the two value function `atan2(y, x)` which returns an angle in the range $[0^\circ, 360^\circ[$. The value coordinate is given by equation (4.3)¹.

$$H = \text{atan2}\left(\sqrt{3}(G - B), 2R - G - B\right) \quad (4.1)$$

$$S = \sqrt{R^2 + G^2 + B^2 - RG - GB - BR} \quad (4.2)$$

$$V = \max(R, G, B) \quad (4.3)$$

Although the RGB space resembles the human eye response, it does not take into account the fact that the response to e.g. green light is different than the response to the other components. The HSV scheme, while being more ‘natural’, also lacks this property. This means that a shade of grey defined as the value V of the HSV representation, given by equation (4.3), will not necessarily seem as bright as the colour the RGB coordinates represent. As an example, one can compare a pure blue ($RGB = (0, 0, 1)$) and white ($RGB = (1, 1, 1)$). When displayed on a computer monitor, the white will emit more light and appear brighter, because all three subpixels are active instead of just one, while both the blue and white have $V = 1$. Because of this, usually the luma Y' is used for determining the desaturated greyscale value, originating from its use in image compression and transmission [5]. Different definitions exist, but the one that will be used here is given by equation (4.4)². In this report, the Y' of a pixel will usually

¹These definitions differ from the definitions more commonly used in software, as described by Smith [9]. The differences between both models make no difference for discussion in this document.

²This is the luma definition provided by the Qt function `qGray()`.

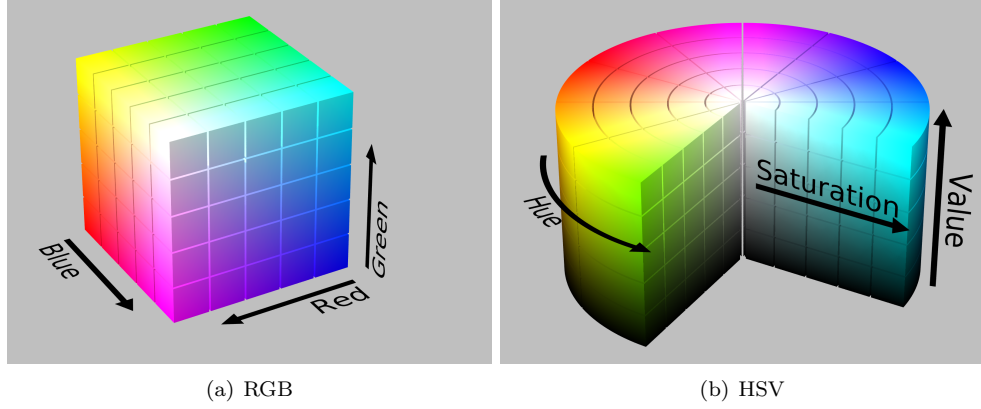


Figure 4.1: Two different representations of the colour space: Cartesian (RGB) and cylindrical (HSV). By Horvath [4].

be referred to as its brightness.

$$Y' = \frac{11R + 16G + 5B}{32} \quad (4.4)$$

4.2 Colour recognition

Using equation (4.4) a greyscale version of the acquired picture is created. After determining the dot positions \mathbf{R} , given by equation (3.1), a small region is defined around the dot to calculate the average colour. In this region the pixels that exceed a certain brightness threshold Y_c are used to calculate a brightness weighted average RGB value, shown in equation (4.5), that is subsequently converted into an HSV coordinate. For colour identification, currently only the hue is needed, given by equation (4.1).

$$\overline{(RGB)} = \frac{\sum_{Y' > Y_c} Y' \times (RGB)}{\sum_{Y' > Y_c} Y'} \quad (4.5)$$

4.2.1 Filter properties

Several filters from LEE Filters [6] were used for testing the software and determining a good transmission value. The list of investigated filters is provided in table 4.1. A good approximation of the total transmission can be made by weighting the spectral transmission with the luminous efficiency function $V^*(\lambda)$ as provided by Sharpe et al. [8] and shown in equation (4.6). These approximations are provided in parentheses in table 4.1. Most likely, the total transmissions provided by LEE Filters have been calculated by weighting the spectral transmission with three spectral sensitivity functions of the human eye. More information might be retrieved from LEE filters, but the exact values are of little importance for this discussion.

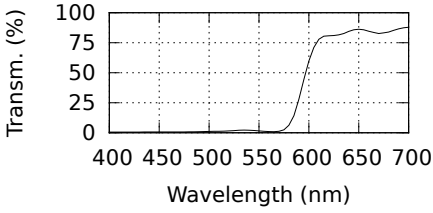
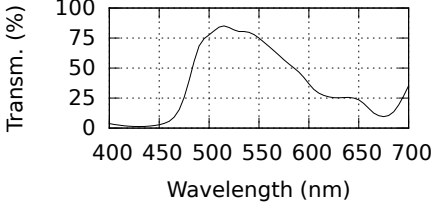
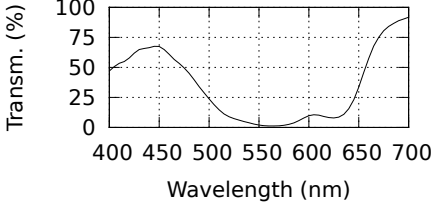
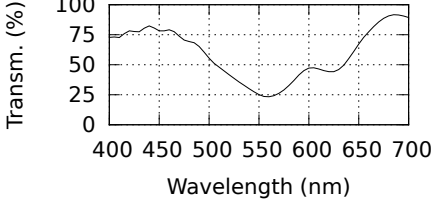
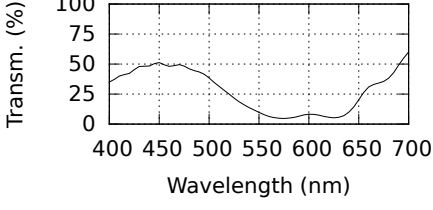
$$Y = \frac{\int Y(\lambda) V^*(\lambda) d\lambda}{\int V^*(\lambda) d\lambda} \quad (4.6)$$

4.2.2 Hue drift

To test the hue determination, pictures were taken using three different exposure times: 1 s, 0.5 s, and 1/5 s. In each of these pictures an identical area containing most of the points was processed for dots. The hue of each dot was determined and the resulting histograms are shown in figure 4.2. To determine the mean hue, Gaussian peaks were fitted to the ‘white’ dots in each picture. The average peak position m is given by 4.7.

$$m = 38.5^\circ \quad (4.7)$$

Table 4.1: Listing of the different colour filters by LEE Filters used for testing in the setup. The first two columns provide the identification number and a description of the colour. The last columns give a summary their transmission, with the total value weighted according to the luminous efficiency. Information was retrieved from LEE Filters website [6]. The values in parentheses are calculated using the spectral transmission and [8] (see equation (4.6)).

Number	Approximate hue	Total transmission	Transmission spectrum
019	Red	$Y = 18.3\%$ (18.8%)	
121	Green	$Y = 64.0\%$ (58.7%)	
701	Dark purple	$Y = 9.4\%$ (10.4%)	
704	Pink	$Y = 40.0\%$ (39.7%)	
719	Blue	$Y = 19.3\%$ (15.0%)	

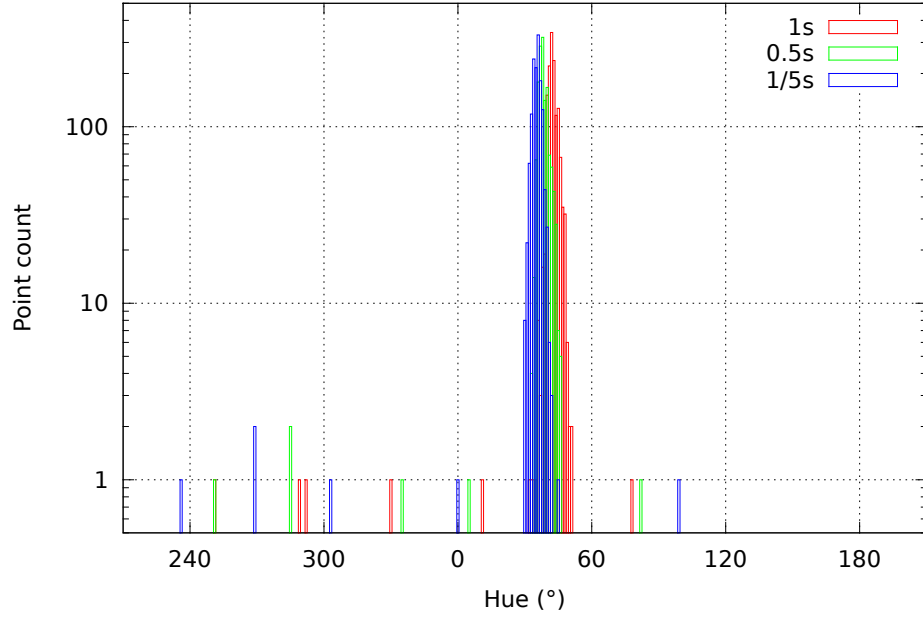


Figure 4.2: Dot hue counts for different exposure times. The central peak comes from the white dots. The coloured dots show up in the following order (left to right): blue, purple (two dots), pink, red, (white,) green. Note that for 1 s and 0.5 s the blue hues coincide.

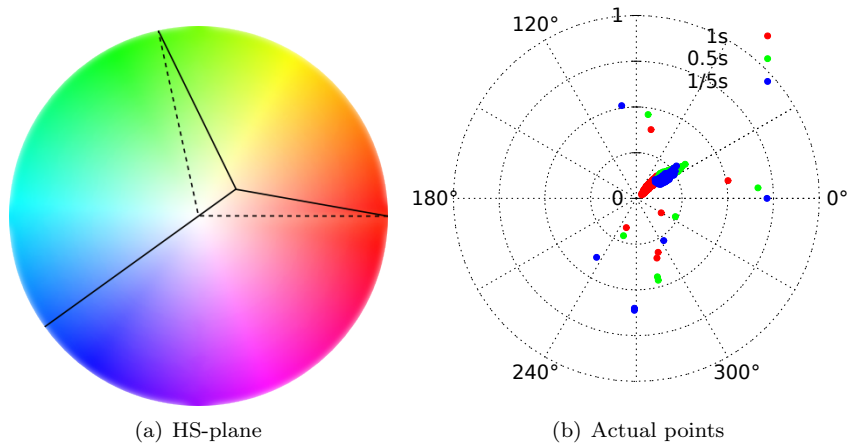


Figure 4.3: White point shift in the hue-saturation plane for $V = 1$. (a) Dashed lines intersect at the white point of the image. Solid lines intersect at the white point of the illuminated dot grid. (b) The data displaying this behaviour.

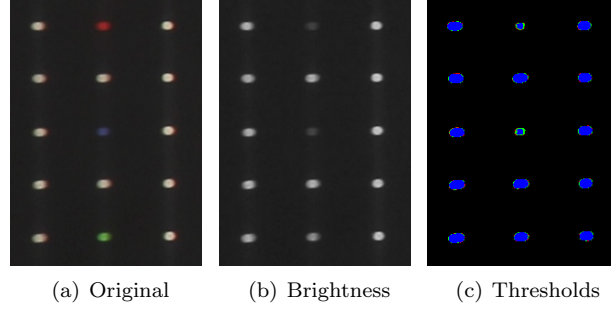


Figure 4.4: Comparison of the green (121), blue (719), and red (019) filters for an exposure time of 1/5 s. Shown are the original picture (a), the greyscale image (b) and the threshold processed image (c). The false colours in the last image indicate the range the pixel’s brightness is in, thresholds are 50, 52 and 60.

A first notable feature of figure 4.2 is the existence of the ‘white dot peak’. If white balance of the pictures would be so that the (non-filtered) illuminated dots produce a neutral, white tint, these dots would have a distribution around $S = 0$ in figure 4.3(a), in the centre of the disc. This is the white point indicated by the intersection of the dashed lines. Projection on H should then produce a uniform background for the white dots, that could be filtered out by putting a lower limit on S to find the coloured dots. When shifting the white point to the location indicated by the intersection of the solid lines, the projection on H now results in a single peak for the white dots. This can be seen in figure 4.3(b), where one may also see that the white points have an decreasing saturation for a 1 s exposure time, nearing overexposure. The peak position around the value indicated by equation (4.7) is an orange-yellow hue.

The second feature is that the coloured dots display a drift in their hue for longer exposure times. A green and red hue are chosen to show the hue shift as the colours approach the white point. The complementary hue of the white point is also chosen to show that it does not display a hue shift. The hue space in the figure 4.2 is rotated to make it more clear that this drift is towards the peak, independent of the dot colour. This can also be explained by the shifted white point. When using a longer exposure time, the colours become less saturated, i.e. they drift towards the white point. However, the white point in this case is the light emitted by the white dots, which does not coincide with the picture’s white point, as noted before. This results in a trend of the colours to shift towards the more yellow hues on longer exposure times, a behaviour that is partly compensated by using a brightness threshold, as is done in equation (4.5). The hues that display the least drift, are the ones that lie on the line connecting the real white point and the picture’s white point. Since one of these is the mean hue of the white points, the most stable hue for the reference dot would thus be a light blue ($H \approx 220^\circ$). Note that if this were actually present in the picture and the exposure time would be increased, the colour would never reach the actual white point, but only the image’s white point, i.e. where the hardware limits are reached (overexposure). To allow for fluctuations, either random or due to hue shift with longer exposure, one thus has to define a window size around the reference colour. If one were to align the two white points, this would allow for a wider range of stable hues that can be used for referencing. This then creates the possibility to have several colours present at any given time, but selecting only one for referencing, by defining a window in the hue coordinate that only contains the reference dot colour.

Note that this would be far harder to achieve in the RGB space. Originally, the reference colour was defined as given in equation (4.8). This is a definition that displays no problems when using a very yellow light from incandescent lighting as was used in the original dummy setup. With the much whiter/bluer light from the LED panels that are now used for illumination of the dots however, this would tag almost every point as ‘blue’. Given a perfect white balance for any light source, this would give the same issue, independent of threshold T , as the ratio would always be very close to 1.

$$\frac{2B}{R+G} > T \quad (4.8)$$

4.2.3 Brightness considerations

Given that the dot finding algorithm currently depends on absolute brightness thresholds to find the dots, the filters have to allow enough light to pass through. Otherwise they will become too dim and there is little room left between the brightest background pixels and the darkest pixels of the coloured dot. It can be seen in figure 4.4(b) that the green filter, having a high (64 %) total transmission, appears only slightly dimmer than the white dots. The red and blue dots, however, having a lower total transmission ($\sim 19\%$), are visibly darker. When applying the three threshold values (50, 52, 60) by colouring the pixel according to the range they fall in, it becomes clear that the green dot has a very similar surface used for position and colour calculation. The red and blue dots already have a smaller amount of pixels, and will disappear before the other dots when increasing the threshold values. Having to fine tune these thresholds lowers user friendliness and increases the time spent on configuring the setup and is thus to be avoided as much as possible.

The green dots in figure 4.3(b) are of similar saturation as the red dots, while having a total transmission value that is ~ 3.5 times bigger, as shown in table 4.1. From figure 4.3(b) one can also see that the pink filter, which has a relatively high total transmission of 40.0 %, could also be used as a reference, since it is still clearly separable from the white points. This requires however that the white balance is set correctly so that less hue drift occurs and a narrower hue window may be used.

Chapter 5

Measurement

To perform an initial measurement, the calibration device was fixed to the granite table and deformed using the micrometer screw. Two pictures were then taken, one reference and one with an extra millimetre of deformation. Dot finding parameters were fine tuned using the new software and then used as input for the old software, which can be done since the dot finding algorithm hasn't changed. To incorporate a solution for the problem with equation (4.8), the old software was slightly adapted to correctly find the reference point, as described in section 4.2.2. The setup parameters (distances and angles with respect to the DUT) were very crudely determined and also used as input for the old software. To speed up the analysis a central region of (15×18) dots was selected, limiting the number of dots to 270 instead of the initial $\mathcal{O}(2000)$. The result is show in figure 5.1, were it can be seen that the maximum deformation is $\sim 80 \mu\text{m}$. From the grid in the image, it can also be deduced that the dot pitch is calculated to be $\sim 0.5 \text{ cm}$, which is a factor of 2 different from the real value of 1 cm . The retrieved values are of the correct order, but the setup parameters need to be determined more precisely if the measurements are to be of any real use.

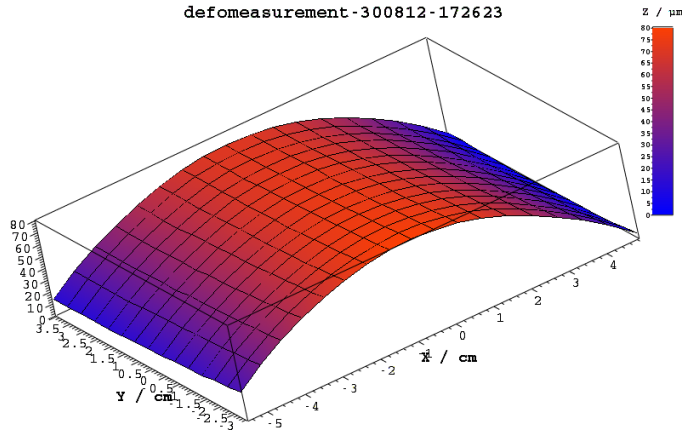


Figure 5.1: A plot of a first deformation determination using the new setup, given a crude determination of the setup parameters.

Chapter 6

Conclusion and outlook

During the course of the summer school program the assembly of the optical deformation measurement setup has made good progress. Currently the LED light panel and dot grid assembly are in place, as well as the (light) shielding and camera frame. Both the dot grid and camera frame still need a permanent support and position adjustment mechanism, probably in the form of a winch. The camera also needs a new pedestal in the sliding frame, so that it can be slid across the full frame, and can be rotated to a few fixed positions. The other peripherals, as discussed in chapter 2, still need to be connected. A permanent feedthrough for all the wiring to the exterior of the setup also needs to be made to replace the current feedthrough: a loose floor panel.

For surface reconstruction and deformation determination, the software needs to be further developed, using of the original defo software as a starting point where desired, and continuing to use the MVC concept for maintaining an extensible codebase. This also includes designing the GUI such that all the necessary options are available to the user, taking into account user friendliness, and possibly even providing two programs: one for online monitoring and configuration, and another for offline analysis. To allow the user to use advanced data taking routines, a scripting engine provided by QtScript will be used. Hooking into the MVC models — either directly or indirectly — already developed, this could provide a clean way of automating the experiment. Perhaps most importantly, once the software has been sufficiently consolidated, its documentation needs to be updated.

As discussed in chapter 4, it has been found that the green filter (LEE Filter 121) is a good choice for the setup, as it gives a saturated colour and sufficiently bright dot. Ideally one would want the colour of the white dots to coincide with the white point of the image, to eliminate the hue drift with different exposure times, as the hue is used to tag the reference dot. This could be achieved by selecting the best fitting pre-set on the camera, or acquire images in a raw format and apply post-processing in the software. Once the white balance is under control, less saturated filters (e.g. LEE Filter 704) may be used. A short exposure time would have to be used, but this is preferable, independent of the filter under consideration.

An initial deformation measurement was performed, but at the same time it was shown that the setup parameters need to be determined, and the setup needs to be calibrated.

I would like to thank my supervisor Andreas Mussgiller, the technical personnel and my fellow summer student Sajedeh Manzelli for providing the parts, instructions and help for building the setup and software. Additionally, I would like to thank the CMS group for the great working environment, and the summer school students and organisation for providing me with a wonderful summer.

Bibliography

- [1] Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies In, Microsoft Corporation, NEC Corporation, and Koninklijke Philips Electronics N.V. *Universal Serial Bus Specification*, 2.0 edition, April 2000.
- [2] Sergio Diez. Silicon strip staves and petals for the ATLAS upgrade tracker of the HL-LHC. Technical Report ATL-UPGRADE-PROC-2012-002, CERN, Geneva, January 2012.
- [3] *gPhoto2 documentation*. gPhoto, August 2009. URL <http://www.gphoto.org/doc/>.
- [4] Michael Horvath. Colour space drawings, 2010. URL <http://commons.wikimedia.org/wiki/User:SharkD>. Creative Commons Attribution-Share Alike 3.0.
- [5] Recommendation ITU-R BT.601. *Studio encoding parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*. ITU-R, 7 edition, March 2011.
- [6] *Spectral charts*. LEE Filters, Hampshire, UK, August 2009. URL http://www.leefilters.com/images/pdf/Art_of_light_spectral_curves.pdf.
- [7] Stefano Mersi. CMS silicon tracker upgrade for HL-LHC. Technical Report CERN-CMS-CR-2012-137, CERN, Geneva, June 2012.
- [8] Lindsay T. Sharpe, Andrew Stockman, Wolfgang Jagla, and Herbert Jägle. A luminous efficiency function, $V^*(\lambda)$, for daylight adaptation. *Journal of Vision*, 5(11):948–968, 2005. doi: 10.1167/5.11.3.
- [9] Alvy Ray Smith. Color gamut transform pairs. *Computer Graphics (SIGGRAPH 78 Proceedings)*, 12(3):12–19, August 1978.