THE HUMAN REMOTE CONTROL

Automatic Perception

Medialogy

4th semester

Aalborg University, Esbjerg - Copenhagen

Date: June 4, 2003

Supervisors: Volker Krüger and Rune E. Andersen

Written by

Camilla Bannebjerre Nielsen, Malene Benkjær,

Maria Tønnesen, Mikkel Byrsø Dan,

Morten Wang and Simon Larsen.

1 Abstract

With the use of EyesWeb we have tried to create an alternative way of controlling a movie- and sound-clip. We have succeeded in developing a system where you can play, pause and fast-forward the clip, zoom in/out and adjust the volume by using your hand and a red glove.

We have used color tracking to track the red glove and afterwards performing blob tracking to output coordinates of the movements of the hand. These coordinates are used to perform decisions on which actions to do with the movie- and sound-clip.

We encountered different problems during the experimental work. Some of them were predicted already in the starting phase such as camera noise, resolution, change in illumination, incoming objects and “The Popcorn Problem”. The only problem still standing is the problem concerning incoming objects with the same color as the glove. The well-known illumination problem was partly solved by using the HSI color model instead of RGB. “The Popcorn Problem”, where the system doesn’t know if it is active or not, were solved by defining an open/close command that included clenching the fist.

All in all we have reached our goal, and we are satisfied with the outcome of the project and the final prototype of the Human Remote Control.

The report is divided into two parts. The first part summaries all the theory used in creating the experiments and the experiments themselves are documented in part two. The theory covers areas such as convolution, digital filters, morphological operators, blob tracking, color models and other relevant subjects. The documentation of the experiments shows the step-wise development used to finalize the EyesWeb patch. The EyesWeb experiments were conducted with success, and the final version of our EyesWeb patch works accordingly to the goals we wished to accomplish.

2 Table of contents

PART I:
The EyesWeb Experiments

3 Preface

This report and the accompanying EyesWeb experiments are made as part of the Automatic Perception course on 4th semester on Medialogy at Aalborg University, Esbjerg in the spring of 2003. The project is supposed to form the foundation for the examination.

It has been an exciting and challenging process to develop the project. We have achieved the result we wished for and have learnt a great deal about digital image processing as well.

We have made an effort to create a report that could be read by students commencing on this course in the future. We sincerely hope that this report would prove as a helping hand for future Medialogy students in their education.

Special thanks go to group 4 that proved to be an invaluable sparring partner. Also thanks to our supervisors for patiently answering all of our “stupid questions”.

Camilla Bannebjerre Nielsen

Malene Benkjær

Maria Tønnessen

Mikkel Byrsø Dan

Morten Wang

Simon Larsen

4 Introduction

In the movie “Minority Report” (Spielberg, 2002) the character played by Tom Cruise, police detective John Anderton, stands in front of a huge computer display and controls the flow of a movie clip with hand gestures. It’s the year 2054, and the future is highly advanced.
Technology that seems far fetched to today’s standards is common day. But what if controlling a movie clip with your hands isn’t so far fetched? What if it is possible to do it with present day household technology?

Technology is highly integrated and is used in many different ways in our society today. The technology is moving towards less distance between humans and computers and towards more transparency. Technology is all around us, but we are less aware of it.
Producers of all kinds of electronic gadgets are becoming better at disguising and perfecting the different technologies. But to control electronic objects you still have to use the technology directly, e.g. pressing a play button on a remote control or turning the light on/off with a switch on the wall. The fundamental way of controlling media hasn’t been changed much in the decades since remote controls first emerged. It would be a sad thing if we had to wait until 2054 before remote controls where replaced by more indirect ways of controlling your television.

Our idea is to make it possible to interact with a media without mouse, keyboard or other devices, but only using your body as a tool. Our project is to develop a human remote control, designed to interact with a video- and sound-player.

We imagine a person standing or sitting in front of a screen (computer, television, or other displays). The person would then be able to control a movie-clip by moving his or her hands vertically and horizontally and maybe using other gesticulations as well.
The hand movements of the person are registered by the computer and translated into commands that control the movie and sound file.

4.1 Problem area

4.1.1 Problem statement

In this project we will try to create a human remote control that will suit the purpose of a standard remote control, as we know them nowadays. However as we define the Human Remote Control (HRC) it is not going to fulfill all the needs of a common remote control, only some of the basic features.

The future goal of the Human Remote Control (HRC) will be to teach the computer common human gesticulation. A computer will at best be able to recognize some patterns and respond to them accordingly. Therefore the aim of the project is not to produce a fully developed consumer product that will replace the standard remote control, but rather to make a prototype and to learn from the process of experimenting during the project. The report will serve as documentation of our experiments, both successful and not, and as a summary of the theory used.

This project tries to answer the following question:

> ***What is required to develop a system where users are capable of controlling movie- and sound-clips using their hands as controller?***

We will examine the following:

- What theory ought to be included to support the experiments
- Which digital image processes techniques is necessary to create the Human Remote Control?
- Which experiments and blocks in EyesWeb are required to produce a prototype of the Human Remote Control?

The questions above will serve as the boundaries and as guidelines of our project. Furthermore the questions provide the reader with an overview of the project and of the main subjects the report contains. During researching, theory collection and experimenting with EyesWeb we will try to accomplish the task of creating the HRC.

4.1.2 Limitations

In order to make the best of the experiments and be able to reach our goal we have to constrain ourselves within certain boundaries.

- Keep the experiments within the color tracking domain
- Keep the remote control within the boundaries of EyesWeb (only controlling the player that is build-in in EyesWeb, hence not controlling Windows Media Player etc.)
- Closed environment in which we control the objects (none or few people within the room, and controlling the colors and illumination of the room).

The color-tracking domain

We have decided to use a red glove, as the remote control device because of our limited experience with image processing in general and because it is hard to do object recognition with the software that we use.

Keep the remote control within the boundaries of EyesWeb

Due to lack of programming experience and/or limitations in EyesWeb the prototype that we develop will merely function as a remote control device within EyesWeb. It cannot replace the standard mouse in windows, but only control video and sound clips inside EyesWeb.

Closed environment

It is imperative that we conduct the early experiments in a closed environment. By slowly adding more and more elements into the experiments we gain a better understanding of what works and not.

4.2 Research

To form a general view of the area, we have done some research on the topic. We discovered that tracking hand gestures was a major area. Many of the projects are built on the same idea as ours. Users interact with computerized equipment simply by using their body, hands, face, eyes or speech. It is supposed to make the systems transparent, easier and more natural to use.
Almost all projects that deal with this theme are built on the same structure.
First step is to choose an input type; hand signs/gesture, speech, eye movement or perhaps a combination. *Next* step is to track the object, often done by color (skin) tracking, diodes, electrodes or similar. *After this*, "laboratory" and real time experiments of shape or other recognizing are often made. The purpose of it is to make users capable of controlling computerized equipment using themselves as controllers.

These related projects have given us a good motivation for our project. Especially the project "*A prototype System for Computer Vision Based Human Computer Interaction*"[1] made by students from Kungl Tekniska Högskola/ Nada - Institutionen för Numerisk analys och datalogi in Stockholm, Sweden. They have used a complicated technique to develop a system that let the user control a television by a few simple hand gestures.

Several projects work with more specific and complex aspects of the area (e.g. hand posture recognition against complex backgrounds). These projects imply that you already have the basic knowledge on the area to understand these projects.

We have actually only collected a few usable publications, one is named "*Tracking Object by Color Alone*"[2], created by students at the Department of Computer Science at Yale University. As the title of the project states it deals with color tracking of objects. The project mentions various methods to achieve a result, some better than others. Three ways to get a solution are approached; they are all low-level image processing techniques. Edge extraction, variations on region-based correlation, or segmentation techniques, often referred to as "blob" tracking. They use blob tracking to track an object and recommend this technique.

[1] More information about this project can be found at www.nada.kth.se/cvap/gvmdi/
[2] The publication can be found at http://citeseer.nj.nec.com/rasmussen96tracking.html

PART I:
Theory

In this part of the project the theory necessary to create the Human Remote Control will be explained. We have made an effort trying to simplify the theory. Starting out with the basics of image processing and then moving on to the more complex issues. The theoretical level and writing style is adjusted so that coming Medialogy students hopefully will find it easy to understand in contrast to other materials on the subject.

The theory part mainly consists of the subjects: “History of image processing”, “Digital images and signals”, “Computing with images”, “Digital filters” and “Blob tracking”. These subjects lie within the basic image processing theories and the more specific ones used in creating the Human Remote Control.

5 History of image processing

Until 1964 there was little progress in digital imaging and digital image processing. NASA's Jet Propulsion Lab (JPL) in Pasadena, CA, was working with lunar images from satellites. The satellite, Ranger 7, was launched July 28, 1964, and sent more than 4,300 pictures on its entire mission[3]. But some problems occurred due to transmission errors. Therefore, they worked with image enhancement and correction to remove noise and other unwanted aspects in the digital images.

These techniques are often used in image processing today and more have been developed for other purposes. Basically, there are two different application areas, where digital image processing is interesting. One is the improvement of images for human perception or interpretation, e.g. enhancement of images such as deblurring, contrast normalization and color manipulation. The other is processing of scene data for autonomous machine perception, e.g. machine vision used for surveillance systems, industrial product assembly and inspection, medical image processing etc. An example of medical use could be the analysis of microscopic images. With an edge detection filter, it is easier for a computer to count a number of blood cells in an image, and for humans as well. Try to have a look at the images below in Figure 1.

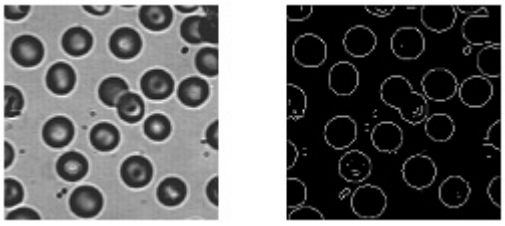


Figure 1: Microscopic image of blood cells (left) and the same image with edge detection applied (right).

Digital image processing has also found its way to the consumer market. There are a lot of image applications available for the consumers that make it possible to preprocess images. You can transform the images in many ways such as scaling and rotating, and enhance them by manipulating colors, contrast, brightness level and more. Common to these image applications is that they implement a series of standard algorithms for performing the tasks, and some more sophisticated than others.

[3] http://www.jpl.nasa.gov/missions/past/ranger.html

6 Digital images and signals

To get a fundamental understanding of how the computer represents and processes a digital image, we have to start from scratch and explain the basics of a digital signal. Digital technology breaks information down into discrete pieces and represents those pieces as numbers. E.g. the music on a compact disc is stored digitally as a sequence of numbers. Each number represents the voltage level of one short period of time (perhaps 40,000 measurements every second). The number of measurements per second is called the sampling rate. So the higher the sampling rate is, the more measurements will be taken each second, and the digitized signal will take up more memory in the computer. But because changes that occur in the signal between samples are lost, the sampling rate must be sufficiently high.

As mentioned before, computer store information as numbers, but those numbers are not stored as decimal values. All information in a computer is stored and managed as binary numbers. While the decimal system has 10 digits (0 to 9), the binary number system only has two digits (0 and 1). A single binary digit is called a bit. A single bit can represent two possible items or situations. The bit is either 1 or 0, like an electrical switch that is turned on or off. Two bits taken together can represent four possible items, because there are exactly four combinations of two bits (00, 01, 10, and 11). Similarly, three bits can represent eight unique items, because there are eight combinations of three bits. In general, n bits can represent 2^n unique items.

6.1 Digital image representation

An image, like all other information stored on a computer, must be digitized. This task is done by different sensor equipment such as digital scanners and the corresponding scanner software, the CCD chip in a digital camera etc. A CCD chip is a physical device that is sensitive to the energy radiated by the object we wish to represent. The sensors produce an electrical output proportional to the light intensity. Then there is a digitizer in the camera that converts the physical sensing into digital form.

An image is divided into picture elements, so called pixels. A pixel is a tiny dot that represents a very small part of the image. The more pixels used to represent an image, the more realistic it looks.

It will also take up more space on the hard drive or in the memory. The number of pixels in an image is called the spatial resolution, but is commonly referred to as resolution.

Figure 2: Reducing the spatial resolution leads to data reduction, but also quality reduction.

In more mathematical terms, an image is a two-dimensional function *f(x,y),* where x and y is the spatial (plane) coordinates and the amplitude (height) of f at any point (x,y). This is called the intensity or gray level of the image at that point. The image with the function *f(x,y)* has an origin in the top-left corner *f*(0,0).

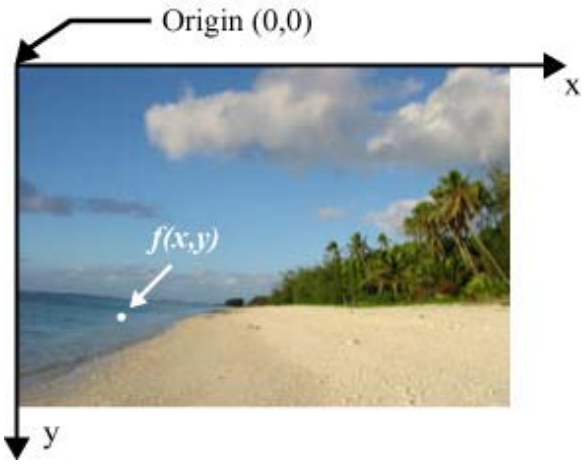


Figure 3: Illustrating the x and y-axis and the origin.

6.1.1 Sampling and quantization

To create and store a digital image in the computer, we need to convert the continuous sensed data (from different sensor equipment) into digital form. This digitizing is done by sampling and quantization. Digitizing the coordinate values is called sampling. Digitizing the amplitude values is called quantization. Figure 4 below shows an image before and after this process.

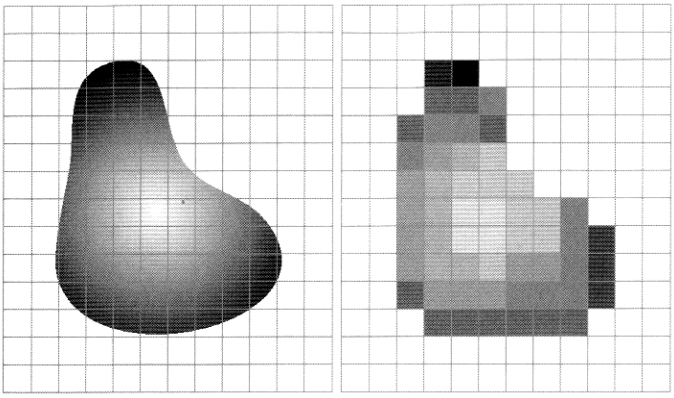


Figure 4: (left) Continous image projected on a sensor array. (right) Result of image sampling and quantization. (Free after "Digital Image Processing" page 54).

The result of sampling and quantization is a matrix or two-dimensional array of numbers. Assume that an image *f*(x,y) has been sampled so the resulting digital image has M rows and N columns. The rows relate to the height of the image, the columns to the width of the image and the value at a certain M and N element is the pixel value (the intensity). We now have a table of numbers. This way of representing the image in the computer, allows us to perform different actions on individual pixels, because you can address them directly by selecting a column and a row.

6.1.2 Grayscale images

As mentioned before, sampling is related to the spatial resolution. We gave an example of different resolutions and the visual effect on the image. Lets look at the quantization, which determines the intensity of a pixel, e.g. in a grayscale image. It is possible to have different gray-level resolutions, so the intensity can span over a number of gray-levels. For example, an image can have 256 or 128 shades of gray. This is often referred to as the bit depth, because we use a number of bits to represent a number of gray-levels. Often, the grayscale intensity is stored as an 8-bit integer giving 256 possible different shades of gray from black to white (2^8 gray-levels). If we only want to have two different shades of gray (black or white), we have what is called a binary image. Binary images are images whose pixels only have two possible intensity values. Numerically, the two values are often 0 for black and 1 for white. Remember that a single bit can represent 2 possible items or situations. In Figure 5 on next page we have illustrated some images with different gray-level resolutions to show the visual effect on the image.

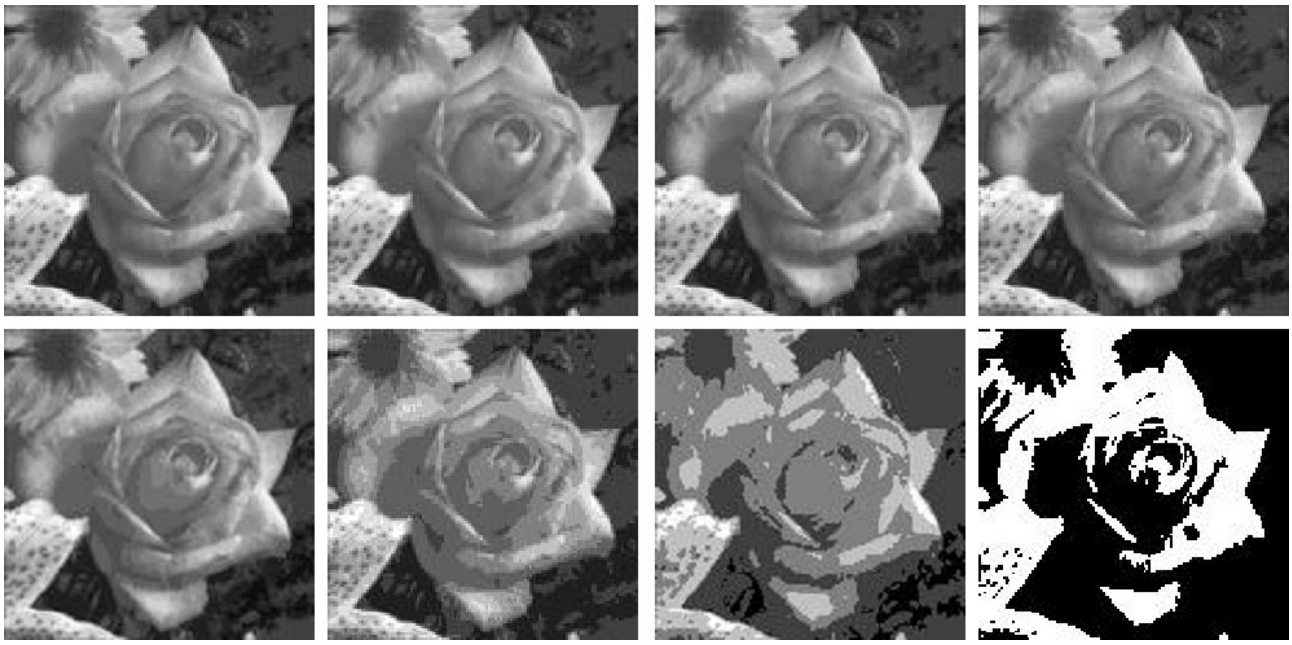


Figure 5: Different gray-level resolutions. From left to right: 256, 128, 64, 32, 16, 8, 4, 2 shades of gray.

As with the spatial resolution, a reduction in gray-level resolution leads to data reduction because we need fewer bits to represent each pixel. But the effect is hardly visible at gray-levels above 32 or 64 depending on the image. So it is better to reduce the number of gray-levels in an image rather than the number of pixels, if we need the image to fit a particular file size.

6.1.3 Color images

This is only a brief description of color images. We will use the RGB (Red, Green, and Blue) color space to explain the fundamentals. There are also other color spaces such as HSI and YUV that are used for different purposes[4].

A grayscale image has one intensity level for each pixel, which represents the gray-value from 0-255. A color image has three intensity levels for each pixel. One intensity level for red, one for green and one for blue (the three components). If we use 8 bit for each component, we can represent 256 values of red, 256 values of green and 256 values of blue. In this case it is a 24-bit true color image, which can represent about 16 millions different colors (2^{24} colors).
Therefore, color images take up more memory and require more computer power, when we want to perform some kind of operation on the image.

[4] For more about colors, see section 7 “Colors”.

6.1.4 Digital video

A video is just a sequence of images over time. The images can be of all the types described earlier, e.g. binary images, grayscale images or color images with different resolutions and bit depths. The more information you have in each picture, the larger the video file will be. But the framerate of the video also influences the size of the video. The more images the video has each second, the more information we will need to store.

As mentioned, an image is described with the function: $f(x,y)$. We can describe a video with the this function: $f_t(x,y)$. At each timestep t, we have an image $f(x,y)$. Figure 6 below illustrates this relationship.

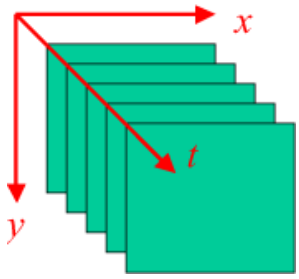


Figure 6: A number of images over time.

There are different video standards, e.g. the NTSC and PAL standard. In our project we will use a digital video camera with the PAL standard. The PAL standard has a resolution of 720x576 and a framerate of 25 images per second. This is how the camera records and stores the information on the tape, but when we transfer it to the computer we can easily change the resolution and framerate. We can also recompress the video into another format than the one of the camera (DV format), such as mpeg or an avi format using a codec (compressor – decompressor) like Cinepak or Intel's Indeo video codec.

7 Colors

Technically speaking colors are the way our brain, by use of our eyes, interpret electromagnetic radiation of a wavelength between 400 (seen as purple) and 700 (seen as red) nanometer. The spectrum spans the visual range for humans. An object looks as if they are colored, but they are actually just reflecting light for selected portions of the color spectrum[5]. These different wavelengths are seen as different colors; red, green, blue, yellow and orange are a few examples of different colors as shown in the spectrum below.

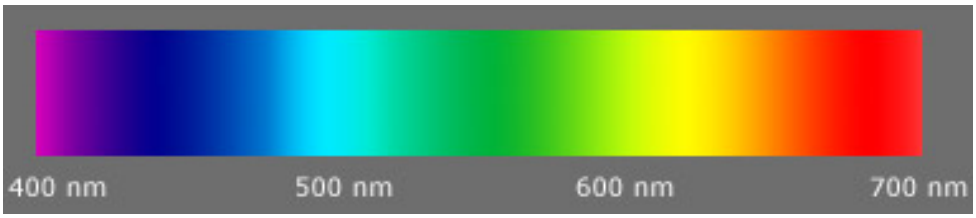


Figure 7: Wavelength, in nm with the electromagnetic energy spectrum of 400nm – 700nm.

As observed the pure green color covers a huge area of the spectrum; it sets off at about 500 nm to 600 nm. The pure red color on the other hand covers a smaller area of the spectrum and spans from about 650 nm to 700 nm. For color tracking red and blue are excellent because there are few variations in their pure colors. The green color could be hard to track because of the large color area, although our ability to observe the green color is excellent, because of the large color area it covers.

There are two different color techniques, the primary and the secondary colors.

7.1 Primary colors

A human can perceive a countless amount of colors. Due to the structure of the human eye, all colors perceived can be produced simply by adding different amounts of red (R), green (G) and blue (B) colors. This color model within the primary colors is usually referred to as RGB[6].
Three different color receptors exist within the human eye, each sensitive to red, green or blue. These receptors are the ones making it possible for a human to observe (almost) every color in the

[5] Matlin, Margaret W. and Foley, Hugh J.: "*Sensation and Perception*". Fourth Edition, page 215
[6] http://academic.mu.edu/phys/matthysd/web226/L0228.htm

combination of red, green and blue. RGB are mostly used color displays such as television and computer screens. On screen the colors are presented through lights. Meaning that lights are turned on and off in different strengths of red, green and blue. When all lights in the RGB-system are turned on in full strength at the same time, the result is white. If all the lights on the other hand are non-existent the outcome is black. The RGB model uses an *additive color technique*, because it adds or mixes the three primary colors of light together.[7]

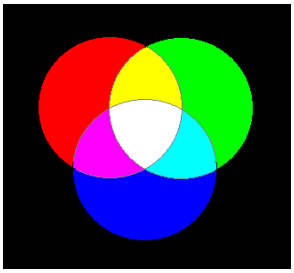


Figure 8: Additive colors. The outcome of adding two primary colors of light is illustrated in the color circle, which shows the different combinations of colors produced by the red, green and blue light.

7.2 Secondary colors

The second technique is the secondary colors also called CMYK[8], which represent cyan (C), magenta (M), yellow (Y) and black (K). Cyan, magenta and yellow are the secondary colors of light and the primary colors of pigment. Meaning that, CMYK is most important in association with the pigment colors in the printer and not important in connection with the display on screen.

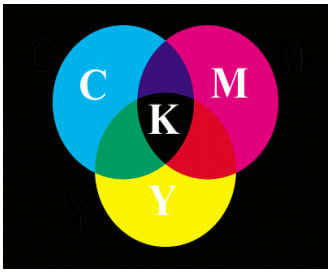


Figure 9: Subtractive colors. The CMYK color circle.

CMYK make use of *subtractive color technique*. White light is emitted by the sun, and the different colors filter/subtract other colors. E.g. red paint filters all light except red.

[7] Christensen and Fischer: ”*Udvikling af multimedier*”, page 159.
[8] http://academic.mu.edu/phys/matthysd/web226/L0228.htm

7.3 Characteristics of a color

There are three primary characteristics, which are used to define our perception of color and distinguish one color from another. This is hue, saturation and brightness [9].

Hue

Hue is associated with the dominant wavelength in a mixture of light waves. When we make use of the word "color" in a daily conversion, we typically mean hue instead. In other words when we call an object red, orange or yellow we are specifying its hue. Below is an example of the original picture with a changed hue.

Figure 10: Hue.

Saturation

Saturation refers to the strength or purity of a color; how pale or strong the color is. The saturation of a color is not constant, but it varies depending on the surroundings and which light the color is seen in. Pure colors are fully saturated. A color like pink, for instance, which is a mixture of red and white are not saturated.

Figure 11: Saturation.

[9] Matlin, Margaret W. and Foley, Hugh J.: "*Sensation and Perception*". page 216-218.

Brightness

Intensity of the color; brightness determines the total amount of light in the color. The brighter a color is the higher its value and the more light it produces. Zero brightness is black and one hundred percent is white.

Figure 12: Brightness.

7.4 Color models

Different color spaces or color models exist such as RGB (Red, Green, Blue). A color model is a way of describing or specifying a color in a standard way. Most color models today are oriented either toward hardware (such as monitors or printing devices) or toward applications where color manipulation is used for different purposes[10]. Examples of hardware-oriented models are RGB (monitors, color TV tubes, color), CMYK (the process colors used in printing etc) and YUV for commercial TV broadcasting. In the following sections we will shortly describe the RGB and the HSI color model, because we use them in our project related to the experimental work in EyesWeb.

7.4.1 RGB

RGB is the "classical" computer color model, and consists of the three color components red, green and blue. Images in the RGB model consist of three individual image planes, one for each of the primary colors. In every pixel in the image the three components can be stored as a vector / array in the computer memory. An example of this is shown in Figure 13 below.

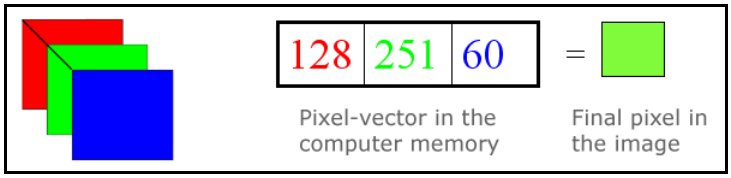


Figure 13: Demonstration of RGB pixel-vector.

[10] Gonzalez and Woods: "*Digital Image Processing*", 1st Edition, page 225

If the pixels are not stored as vectors, the complete red component is stored, the complete green and then the complete blue[11]. The RGB model is based on a Cartesian coordinate system that forms a cube as shown in Figure 14 with RGB at three corners, CMY at the three other corners, black at the origin, and white at the corner farthest from the origin. Grayscale extends along the line joining black and white, and colors are points on or inside the cube, defined by vectors extending from the origin. The assumption is that all color values have been normalized so that the cube is a unit cube. That is, all values of R, G, and B are assumed to be in the range [0,1][12].

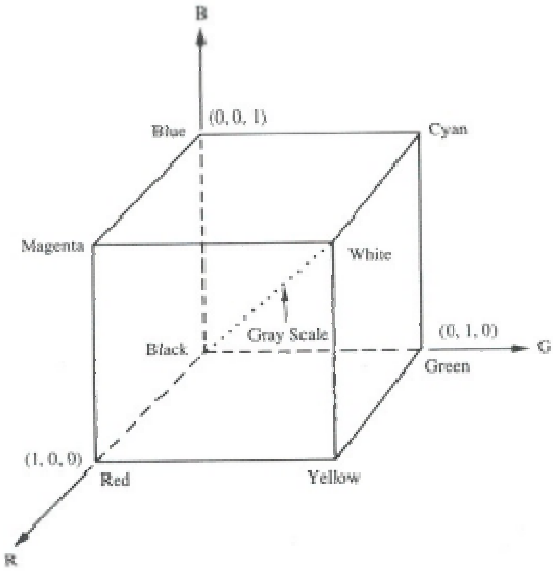


Figure 14: RGB color cube.

7.4.2 HSI

As mentioned, hue is an attribute that describes a pure color (i.e., it is associated with the dominant wavelength or the dominant color), whereas saturation gives a measure of the degree to which a pure color is diluted by white light; how pale or strong a color appears. The HSI color model has two advantages. First, the intensity component (I) can be decoupled from the color information in the image. Second, the hue and saturation components (chromaticity) are intimately related to the way in which human beings perceive color. A change in illumination will not affect the color attributes as much as in the RGB model, because the illumination or lightning goes more or less for the intensity in the model[13]. In the RGB a change in illumination will change the intensity of both the red, green and blue component and the final color will be changed severely.

---

[11] Slide ”2. About Colors” by Volker Krüger
[12] Gonzalez and Woods: “*Digital Image Processing*”, 1st Edition, page 226
[13] Because the intensity is decoupled, this color model is more robust when we are working with color tracking in different environments, e.g. changes in illumination.

The components in the HSI model can also be stored as a vector / array, as shown in Figure 15 below.

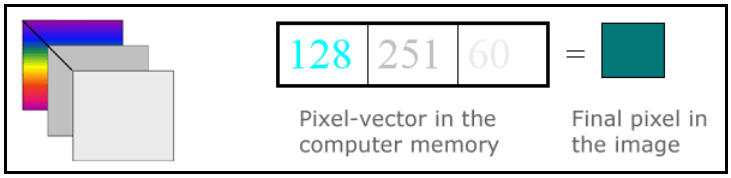


Figure 15: Demonstration of HSI pixel-vector.

The HSI model can be represented in a kind of coordinate system. In this case it is not a cube like the RGB. The color components (hue and saturation) are shown in the color triangle in Figure 16(a). The hue, H, of color point P is the angle of the vector shown with respect to the red axis. When H = 0 degrees, the color is red. When H = 60 degrees the color is yellow and so on. The saturation, S, of color point P is the degree to which the color is diluted by white and is proportional to the distance from P to the center of the triangle. The farther P is from the center, the more saturated the color is.

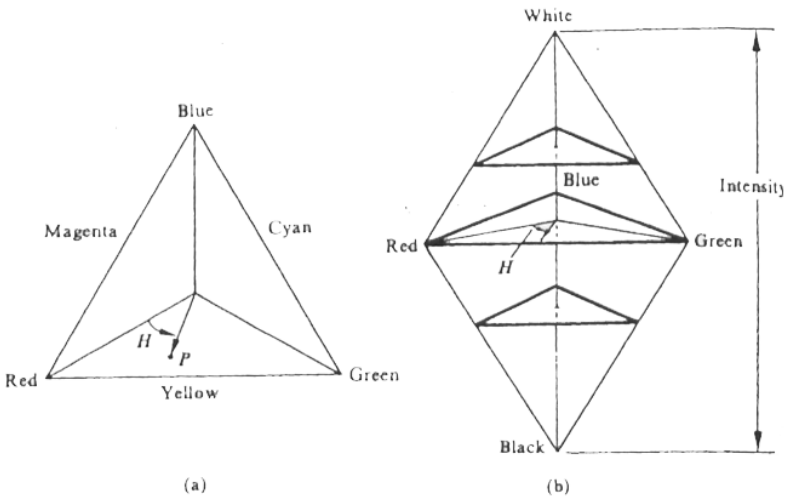


Figure 16: (a) HSI color triangle; (b) HSI color solid[14].

When the intensity is present, the model will become a three-sided pyramid-like structure as shown in Figure 16(b). Any point on the surface of this structure represents a fully saturated color. The closer the color is to the center line, the less saturated it is. The hue of the color is again determined by its angle with respect to the red axis. The intensity is determined by its vertical distance from the black point (the greater the distance from the black point, the greater is the intensity of the color)[14].

[14] Gonzalez and Woods: "*Digital Image Processing*", 1st Edition, page 229-230

8 Computing with images

8.1 General arithmetic operations

Arithmetic is used when two images are added, subtracted, multiplied or divided. As mentioned in the section “Digital image representation” on page 12, all images are represented as pixels and numbers. When a computer is performing the operation of adding two images, it computes the numbers within the image. In other words each number in the image is added pixel by pixel, as illustrated in the figure below:

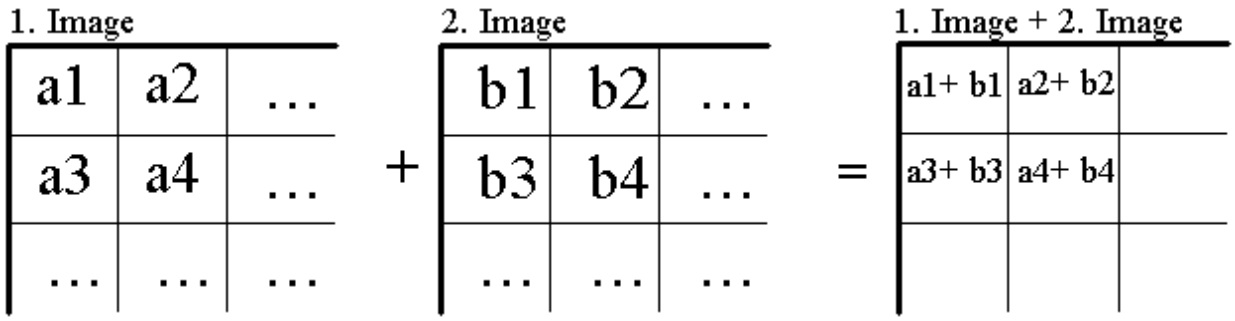


Figure 17: Pixel-wise adding.

The same method as shown above is used in all image arithmetic such as subtracting, multiplying and dividing. The figure below illustrates the result of adding of two grayscale images.

Figure 18: Example of adding of two grayscale images.

8.1.1 Overflow / Underflow

When arithmetic operations are performed on images overflow/underflow may occur. This undesired effect takes place if the sum is over or under the possible value range: 0...255. Overflow can occur if two images are multiplied or added and the sum of a pixel is larger than 255. Underflow on the other hand can take place if two images are subtracted and the result is smaller than 0. If the problem of overflow/underflow is not solved the computer may encounter problems when calculating or displaying the image.

Solution to the Overflow/Underflow Problem

Computing of 6 steps is necessary to solve the problem of overflow/underflow.

1. Write computation results into an intermediate image. Meaning that every value computed at first are saved in a "temporary" image, were overflow/underflow is not a problem. Pixel values are **float** values (32 bits/ 4 bytes). This is important because a float value allows decimal numbers; therefore almost any number can be stored.

50	1010	100	10

2. Find **maximum**

50	**1010**	100	10

3. Find **minimum**

50	1010	100	**10**

4. **Shift** the values so that minimum is 0, this is done by subtracting minimum from every pixel value. Shifting is completed so that no value is negative in the intermediate image.

40	1000	90	0

0,04	1	0,09	0

5. **Rescale** intermediate image to values 0…255. The input values, when scaled has to be between 0...1, because we multiply these values with 255.

10,2	225	20,25	0

6. Write results into final image, rounded to nearest integer.

10	225	20	0

8.2 Logical Operators

The logical operations are a simple form of image processing. Logical operators are often used to combine two images. The input values have to contain the same number of bits.

AND, NAND, OR, NOR, XOR, XNOR, NOT are all logical operators and work according to the principle of boolean algebra. It means that the output values only can have one of two possible values; true or false. We will describe the operators AND, OR and NOT separately, as these are the general operations that the others derive from.

8.2.1 AND

The AND operator can for instance be used when combining two images. The value of a pixel in the first image is ANDed with the value of corresponding pixel in the second image. The outcome of an AND operation can be presented in a truth-table as seen in Table 1.

If expression1 is	AND expression2 is	The result is
True	True	True
True	False	False
False	True	False
False	False	False

Table 1: Truth table for AND.

8.2.2 OR

The OR operator can also be used when combining two images. The value of a pixel in the first image is ORed with the value of corresponding pixel in the second image. The outcome of OR operation can also be presented in a truth-table as seen in Table 2.

If expression1 is	OR expression2 is	The result is
True	True	True
True	False	True
False	True	True
False	False	False

Table 2: Truth table for OR.

8.2.3 NOT

Logical NOT takes an image as input and produces its photographic negative. Light area becomes dark and the other way around. If you apply logical NOT to a binary image its polarity changes, as seen in Table 3.

If **expression** is	Then **result** is
True	False
False	True

Table 3: Truth-table for logical NOT.

8.3 Scaling / zooming of images

It is possible to perform different transformations on a digital image, e.g. scaling or zooming. When scaling an image you define a scaling factor in the x- and y-direction. This principle is expressed in the following formula.

$$I(x, y) \Rightarrow I(s_x x,\ s_y y)$$

Every pixel (x,y coordinate) in the input image are multiplied by the scaling factor, and will change position in the output or scaled image. So the scaled image gets s_x times bigger in x direction and sy times bigger in y direction. The operation can be performed with the following formula.

$$(x, y) \cdot \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} = (s_x \cdot x,\ s_y \cdot y)$$

Some pixels will be located outside the original image boundaries after the scaling, if the image size is not changed. These pixels won't be visible in the scaled image.

Example of scaling

We want to double the size of the balloon in both directions, so our $s_x = 2$ and $s_y = 2$. We choose to place the origin in the center of the object (balloon), which in our case also is the center of the image. The visual effect of the scaling with pixel coordinates can be observed in Figure 19 on next page.

We use the formula to illustrate the change for one pixel only (100; 0).

$$(100;0)\cdot\begin{pmatrix}2 & 0\\ 0 & 2\end{pmatrix}=(100\cdot 2)+(100\cdot 0);(0\cdot 0)+\ (0\cdot 2)=(200;0)$$

So the new coordinate for the pixel will be (200; 0).

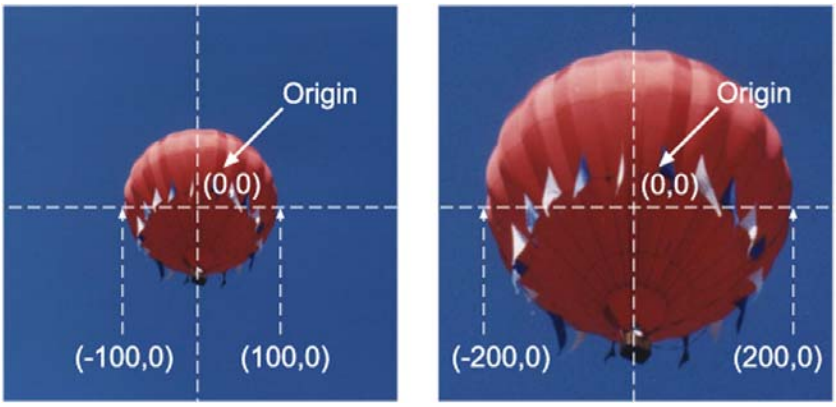


Figure 19: Scaling of an image.

8.4 Threshold

Thresholding is a point processing technique that only considers one pixel in the input image to produce a corresponding pixel in the output image. This is why it is referred to as point processing. The input to a threshold operation is often a grayscale or color image. In the simplest implementation, the output is a binary image. It is often useful to be able to see what areas of an image consist of pixels whose values lie within a specified range, or *band* of intensities. The process can be described with a *mapping function:*

$s = M(r)$

where r is the pixel value in the input image and s is the pixel value in the output image. The form of the mapping function *M* determines the effect of the operation. In the figure on the next page we have included some examples of mapping functions, and their corresponding visual output.

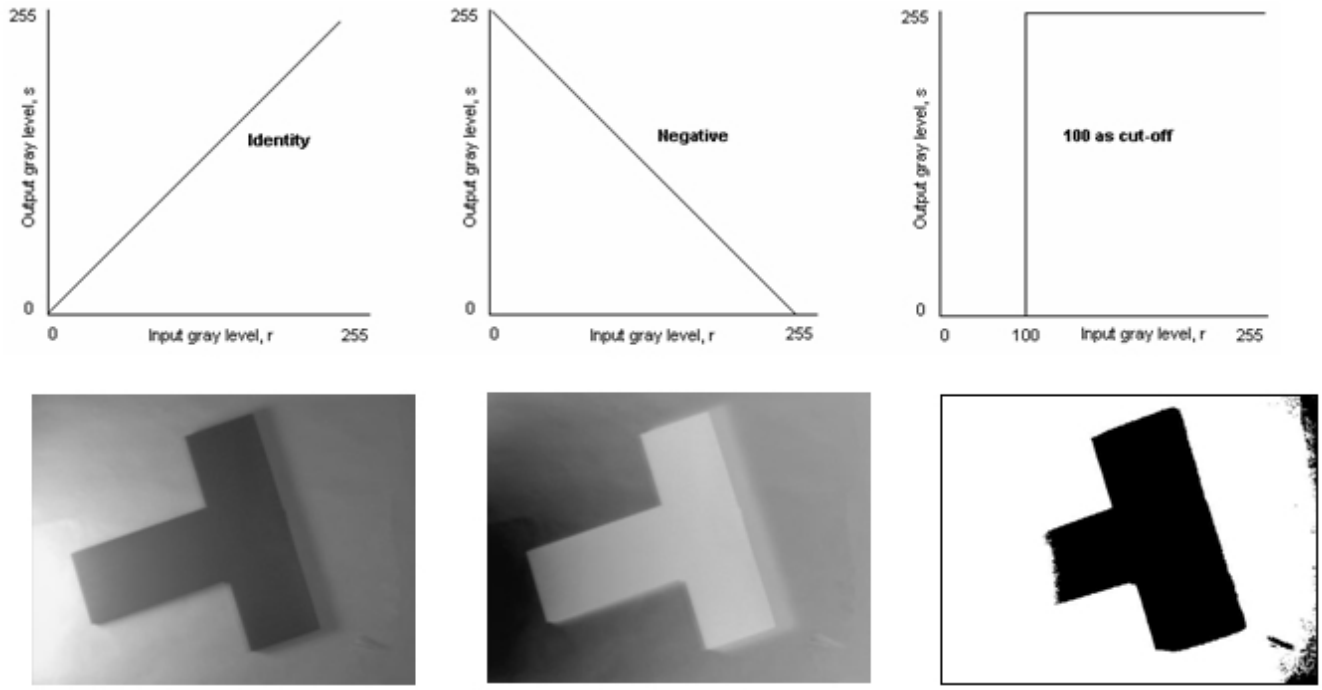


Figure 20: Different mapping functions and their visual output.

The first image is identical before and after the operation, because the output equals the input. This is illustrated in the mapping function for the first picture. The mapping can also be represented in a lookup-table like this:

Input (x)	0	1	2	….	127	….	255
Output (y)	0	1	2	….	127	….	255

The second image is an inverted version of the first image. The black will become white and vice versa. The mapping illustrates this. For instance, if the input pixel for is 0, the output pixel will be 255 etc. The lookup-table looks like this:

Input (x)	0	1	2	….	127	….	255
Output (y)	255	254	253	….	128	….	0

In the last image we have defined a cut-off value or threshold at 100. This means that all values below 100 in the input image will become black (0) in the output image, and all values with 100 or above will become white (255). This mapping therefore produces a binary image.

8.5 Convolution

Convolution is a mathematical operation, which is fundamental to many common image-processing operators. Convolution provides a way of multiplying together two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality[15]. The convolution operation is for example used when applying a Gaussian smoothing filter to an image, which will be described later.

8.5.1 Kernel

The convolution process needs a signal image and a kernel[16]. The kernel contains different numbers or coefficients depending on which filter mask it is. In this section we use the Gaussian filter and also fictive filters as examples. You can multiply two 1D filters to get one 2D filter, which is shown in Figure 21 below.

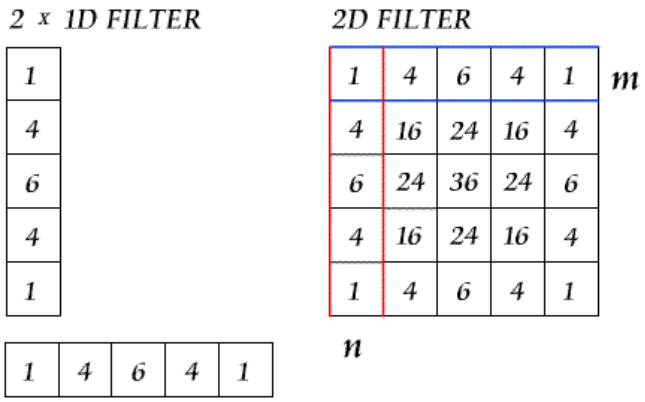


Figure 21: Multiplication of two 1D Gaussian filters.

The image has *M* rows and *N* columns, and the kernel has *m* rows and *n* columns. The filter can have all kinds of different sizes but the smallest meaningful size is a 3x3 filter[17].

Each filter has an origin - usually the center, and before applying the filter to the input image, the filter is mirrored as seen in Figure 22 below. This mirroring procedure is standard and is applied on all convolution filters.

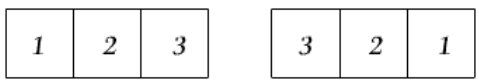


Figure 22: Mirroring.

[15] http://www.dai.ed.ac.uk/HIPR2/convolve.htm
[16] Convolution kernel is also referred to as filter, mask or window
[17] Gonzalez & Woods: "*Digital Image Processing*", 2nd Edition, page 116

8.5.2 Edge problem

When applying a convolution filter on an image a problem along the edges is encountered. Meaning the computer doesn't know the values outside the edge and therefore can't see if there is a null, zero or another value beyond the edge. This occur when the filter's origin (center) reaches the edge of the image, and one or more rows or columns of the filter will be located outside the image. This can however be solved by presuming that the values beyond the edge is zeros. But this sometimes resolves in distorted areas around the edges. Another way is to restrict the filter not to go beyond the edge. The resulting image will be smaller than the original, but in this way we ensure that all the pixels in the filtered image will be processed with the full mask.

8.5.3 Convolution operation

Shifting the kernel across the entire image where the kernel fits the image, starting from the top left corner, performs the convolution[18]. Then you multiply the each number of the filter with the corresponding pixel of the signal image. Then add the results from the multiplication and divide them by the sum of the filter. In this case it is: 1+2+1 = 4, which gives you the number below the fraction line - thus the normalization factor which normally is given as part of the filter. The output of the result is $\frac{5}{4}$. Afterwards the kernel is shifted for every pixel in the signal image and the operation is repeated. As seen in Figure 23 the operation doesn't prints the values at the edges.

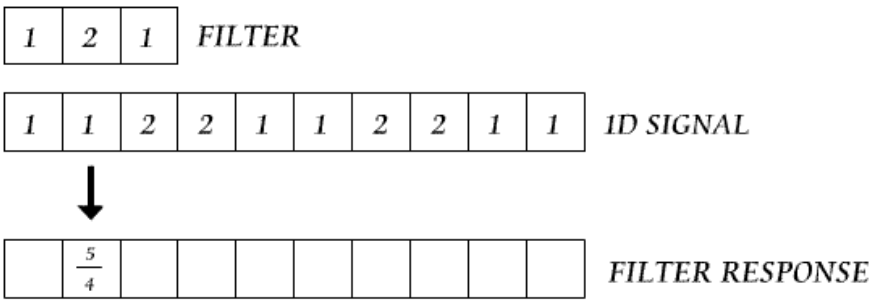


Figure 23: The basic steps of convolution with a 1d Gaussian filter.

[18] Or where the origin of the image is set.

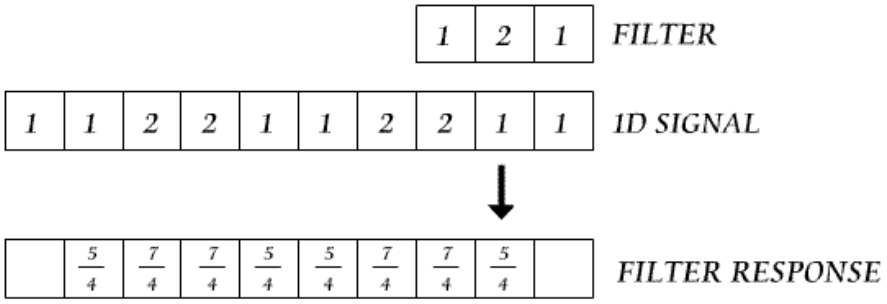


Figure 24: The final filter response.

The following figure shows a convolution operation applied to a 2-dimensional image.

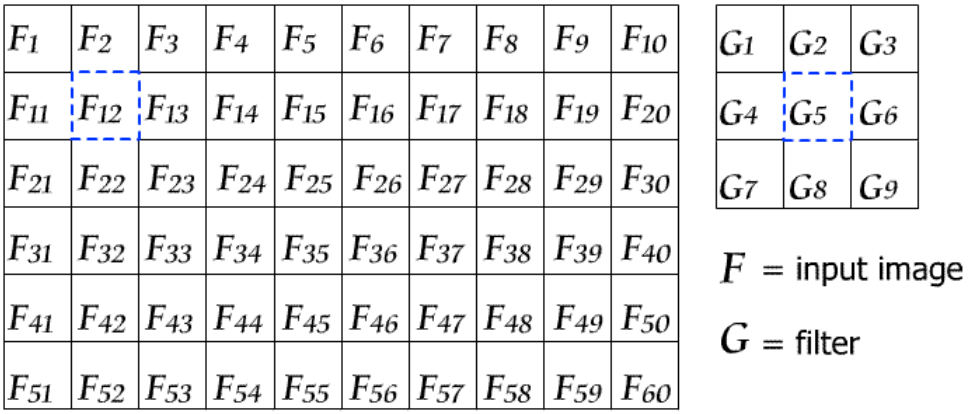


Figure 25: Convolution on a 2d image.

In Figure 25 above we have a 3x3 filter and the following equation shows the example of calculating the output value (O_{12}), when the filter's origin is placed above F_{12}.

$$O_{12} = \frac{F_1 \times G_1 + F_2 \times G_2 + F_3 \times G_3 + F_{11} \times G_4 + F_{12} \times G_5 + F_{13} \times G_6 + F_{21} \times G_7 + F_{22} \times G_8 + F_{23} \times G_9}{G_1 + G_2 + G_3 + G_4 + G_5 + G_6 + G_7 + G_8 + G_9}$$

8.6 Digital filters

A filter is a function that removes unwanted parts of a signal or extracts useful parts of the signal. The figure below shows the main principal behind a digital filter[19].

Figure 26: Digital filter process

Digital filters are widely used in applications such as noise reduction, video signal enhancement, and many other areas.

8.6.1 Smoothing

As we mainly operate with the smoothing filters in image processing, these are the only filters that will be described in detail. The smoothing filters contain - without excluding others - the median, mean and gaussian filter. Common to these filters is that they all blur or smooth images by removing image noise and/or they make hard edges less apparent.

8.6.2 Gaussian Filter

The gaussian filter, or gaussian smoothing as it is also called, is what is generally referred to as a spatial filter, because of its origin in the spatial domain. The gaussian filter outputs a weighted average, meaning the pixel in the center will have higher influence on the outputted value.

The operation of the gaussian filter can be described in the following way with a 1D 5 pixel gaussian filter.

The pixel values of the original image:

200	150	10	150	225	55	75	250	75	110

The gaussian filter:

1	4	6	4	1

[19] Lecture: "*Signal and sampling*", Automatic Perception, March 2003, Bo Rohde Pedersen

The sum of the gaussian filter is 16 and therefore the convoluted value must be divided by 16.
To calculate the third pixel[20] you multiply the values in the original image with the values in the gaussian filter:

200 x 1 = 200
150 x 4 = 600
10 x 6 = 60
150 x 4 = 600
225 x 1 = 225

Next, sum the values and divide by 16:

$$\frac{200 + 600 + 60 + 600 + 225}{16} \approx 105$$

The value in the outputted image:

		105							

The value becomes 105 because of the high values "surrounding" the original value 10.
The following values are then calculated by shifting the filter by one and convoluting again:

$$\frac{(150*1) + (10*4) + (150*6) + (225*4) + (55*1)}{16} \approx 128$$

And so forth, giving the final outputted image (rounded values):

		105	128	141	121	123	142		

Comparing this with the values in the original image it is clear that the values are now closer in range, resulting in a smoother image. The exactly same procedure can be applied if working with two-dimensional filters, i.e. a 5x5 gaussian. To calculate the values in a 5x5 gaussian filter you take the 5th step presented in the common Pascal triangle. Then you multiply these two 1D filters as described on page 29 in the convolution section to get a 5x5 2D gaussian filter.

[20] We avoid the first two pixels, because of the edge problem described on page 30.

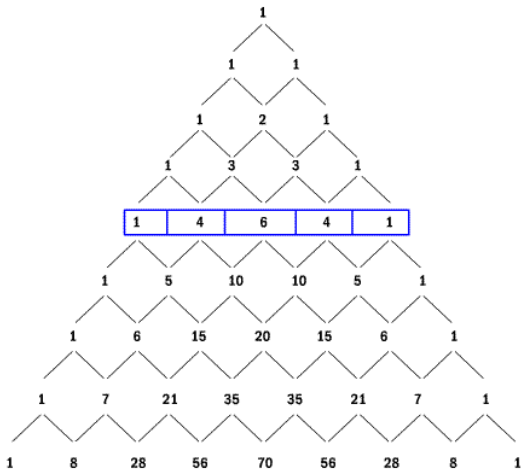


Figure 27: A 9th level Pascal Triangle.

This gaussian filter can also be represented as Figure 28 below. The bigger the gaussian filter the higher the filter also is.

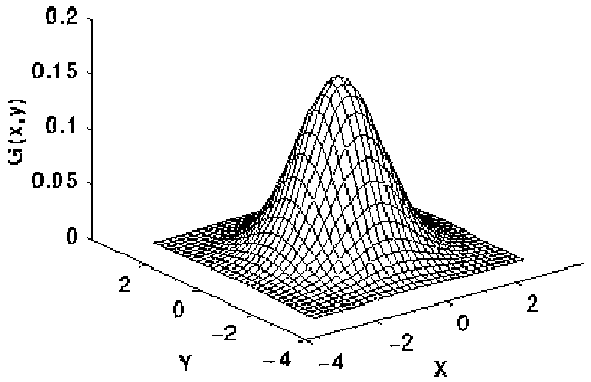


Figure 28: A gaussian filter show in 3D (image source: Hypermedia Image Processing Reference).

Because of the smoothing/blurring operation of the gaussian filter, more detail in the image is lost the large the filter is.

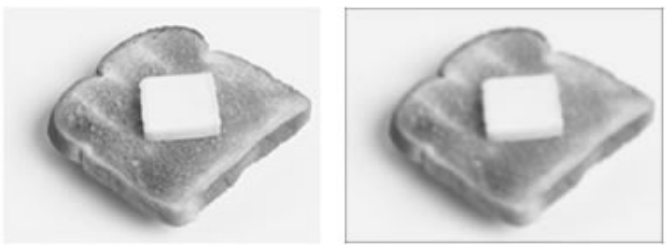


Figure 29: The left image is the original and the right image is after a 5x5 gaussian filter is applied.

8.6.3 Mean Filter

The mean filter works similar to the median and the gaussian filter by smoothing the image and removing image and/or camera noise.
But where as the median filter works as a sorting filter the mean have predefined values in the kernel and calculates the output image with the convolution operation (as described earlier).
The kernel values are defined by the size of the filter (pixel wise). If the mean filter is 3x3 the kernel values are $\frac{1}{9}$ as show in Figure 30 below.

$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$
$\frac{1}{9}$	$\frac{1}{9}$	$\frac{1}{9}$

Figure 30: 3x3 mean kernel.

The same rule is applied if the kernel is 5x5 or 7x7, which would give the kernel values of $\frac{1}{25}$ and $\frac{1}{49}$ respectively. The mean filter calculates the absolute average of the pixel in question and its surrounding neighbours. “This has the effect of eliminating pixel values which are unrepresentative of their surroundings”[21].

8.6.4 Median Filter

The Median filter smoothes the image data and removes noise without significantly blurring the edges. But the Median filter will remove very fine detail, such as sharp corners. The noise you can remove with this filter is often called salt and pepper noise or shot noise. The noise can be less noticeable than Figure 31 on next page, which is an extreme example. But it can be a problem when working with for instance color tracking, because it can make a color appear less homogeneous, and therefore more difficult to track.

[21] Quote taken from Hypermedia Image Processing Reference (http://www.dai.ed.ac.uk/HIPR2/mean.htm).

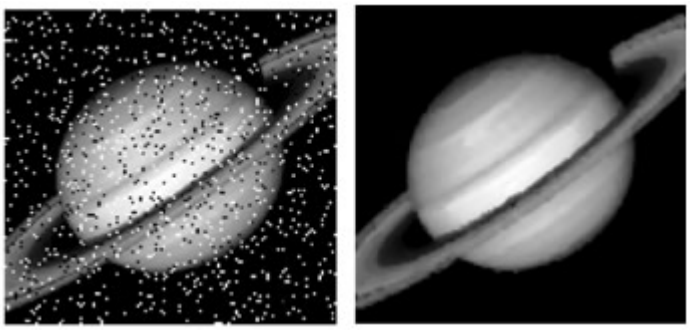


Figure 31: Image before and after filtering with the Median filter.

The Median filter is not a convolution filter, because the filter contains no values. The Median filter reads the values from the input image, sorts them from low to high, picks the middle value as the median and writes this pixel value to the output image. In this way the Median filter considers each pixel in the input image, and looks at its neighbouring pixels to decide whether or not it is representative of its surroundings. If the pixel value is much higher or lower than the neighbouring pixels it is most likely a dot in the image as in the figure, and it is removed. Depending on the filter size, bigger dots can be erased, but the image will be more smooth or blurry.

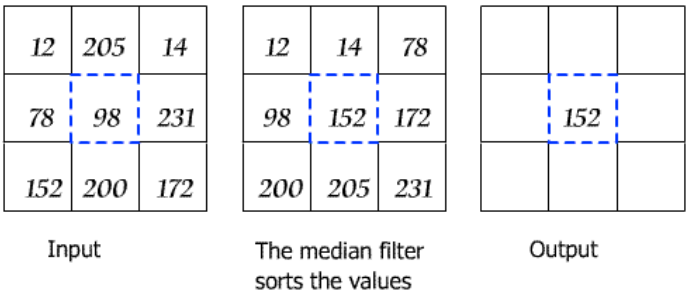


Figure 32: Example of a 3x3 Median filter that sorts the values and writes the median to the output.

8.7 Morphological operators

Morphological operators are generally used on binary images and their purpose is for instance noise removal or removing holes in the foreground or background. The white color is usually referred to as the foreground color and the black as the background.

The morphological operators have a structuring element (SE), similar to the kernel used in the convolution process. The SE can have varying values e.g. 1´s 0´s and none's. Similar to the kernel in the convolution process the SE has an origin. Where the origin is placed has a lot to say about the outcome. Figure 33 shows examples of the SE in different sizes.

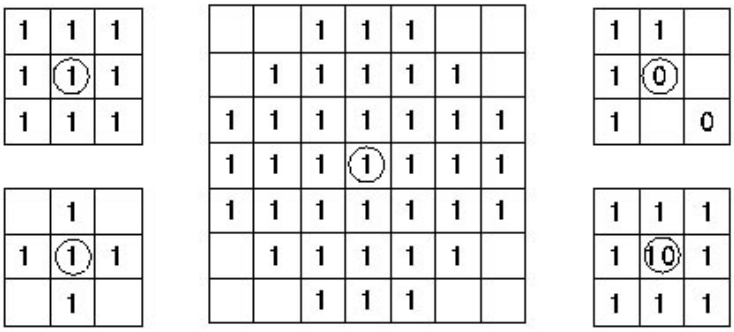


Figure 33: Different structuring elements.

Basically the SE works as a filter that slides pass all the coordinates of the image and checks for a match.

8.7.1 Dilation

Erosion and dilation are the fundamental morphological operations and are dual to each other. Meaning that erosion enlarges the background and shrinks the foreground while dilation enlarges the foreground and shrinks the background. The SE of the dilation filter contains only 1´s.
A small SE is used most frequently; larger SE tends to have extreme effects. Often you are able to get the same effect by using the SE repetitively instead of enlarging the SE. In Figure 34 and Figure 35 you see the basic function of the SE.

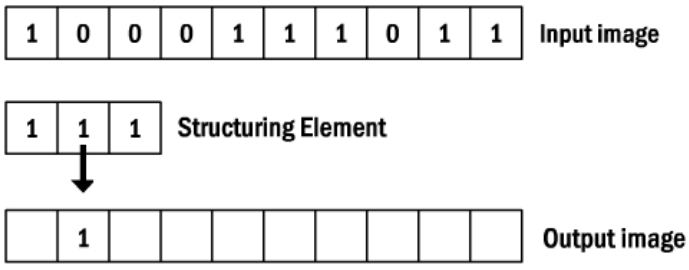


Figure 34: The output of a 1D structuring element.

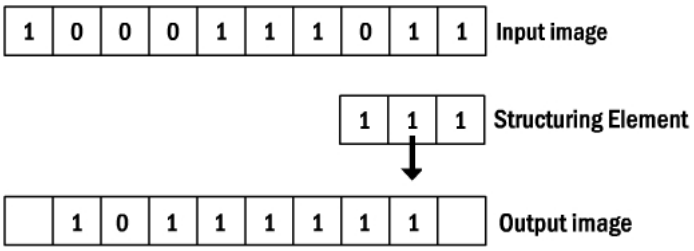


Figure 35: The entire output.

If the SE touches the foreground image it writes a “1” at the origin of the SE, meaning that one of the tree numbers in the filter has to match the signal to pass throw to the output image.

In Figure 36 below you see the effect on a binary image. The foreground is filled because holes in the background shrink.

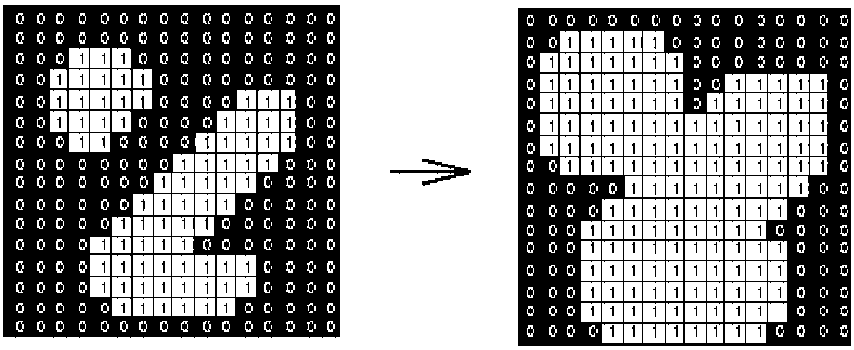


Figure 36: This is the effect of a 3x3 structuring element.

With larger structuring elements, you often use a disk shaped SE with the “right” measures to achieve the wanted effect, instead of working with a square SE[22].

In general when applying the disk shaped SE (dilation) on an image, convex boundaries will become rounded and concave boundaries will be preserved as they are.

Figure 37: The effect that is achieved by using an 11 pixels flat disk structuring element.

In order to get a horizontal dilation, the SE has to be for instance 1 pixel high and 10 pixels wide and by swapping the numbers, you should get a vertical dilation. All in all by changing the origin and the order of the 1´s and 0´s in the SE, you can change the output image in the way you desire - almost.

[22] http://www.dai.ed.ac.uk/HIPR2/dilate.htm

8.7.2 Erosion

Erosion is the dual of dilation, meaning that it has the opposite effect of dilation. The SE of the erosion checks for a perfect match in the image and if so the pixels are set as 1´s otherwise they are set as 0´s. Therefore the background of the image is most likely to grow.

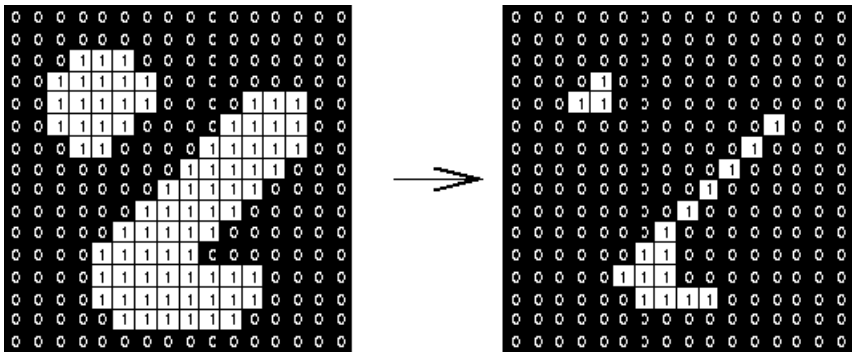


Figure 38: Erosion with a 3x3 structuring element.

8.7.3 Opening

The opening is similar to Erosion, used for spot and noise removal, only less destructive, and enlarges holes in the foreground. Opening is actually the same as erosion followed up by dilation, and using the same SE for both operations. Only the areas where the SE fits into, pass. It depends on the size and form of SE which result the output image gets. The SE only contains only 1´s.

8.7.4 Closing

Closing is similar to dilation, and removes holes in the foreground, and tends to enlarge foreground regions, thereby shrinking the background. The effect of closing is not as extreme as dilation. The SE contains only 1´s. Closing is the dual of opening.
Both opening and closing are idem potent, meaning that repeated application will have no further effect. The damage is done so to speak. They both derive from the fundamental operators, erosion and dilation.

9 Blob tracking

This is a method useful for tracking isolated objects within an image and focusing merely on a blob. Blob is short for “binary large object”. There are different methods to track a blob, both in binary images and in color images.

9.1 Binary image blob tracking

When applying blob tracking to a binary image there are four parameters that needs to be computed in a special order. First the center of the blob, followed by the variance, then the size of the surrounding box and last the corner positions of the box. In order to simplify the formulas used to calculate the parameters we have made an example with a fictive binary blob in a 6x6 image (see below).

9.1.1 Center of gravity

The center of the blob (also called the center of gravity) is found by computing the mean x and y position of all white pixels in the image. This is computed with the following formula:

$$\text{Mean\_x} = \frac{1}{N}\sum_{x,y} x \times B(x,y)$$

The mean_x equals 1 over the total number of white pixels (N) multiplied with the sum of x-pixels and y-pixels in the image and B (x, y) is the given binary image. The same procedure goes for the mean_y, just by swapping the x with the y.
We have tried to demonstrate center of gravity with the blob above. The blob digits below give you an idea of the blob coordinates.

	1	**2**	**3**	**4**	**5**	**6**
1	0	0	0	0	0	0
2	0	1	1	1	0	0
3	0	0	0	1	1	0
4	0	0	0	1	0	0
5	0	0	0	0	0	0
6	0	0	0	0	0	0

Figure 39: A blob example.

Our calculation starts from the top left corner as seen below in the first equation for mean_x.

Mean_x = (1 x 0 + 2 x 0 + 3 x 0 + 4 x 0+ 5 x 0 + 6 x 0

1 x 0 + 2 x 1 + 3 x 1 + 4 x 1 + 5 x 0 + 6 x 0

1 x 0 + 2 x 0 + 3 x 0 + 4 x 1 + 5 x 1 + 6 x 0

1 x 0 + 2 x 0 + 3 x 0 + 4 x 1 + 5 x 0 + 6 x 0

1 x 0 + 2 x 0 + 3 x 0 + 4 x 0+ 5 x 0 + 6 x 0

1 x 0 + 2 x 0 + 3 x 0 + 4 x 0+ 5 x 0 + 6 x 0) = 22 / 6 = 3,666667 ≈ **3,7**

Mean_y = (1 x 0 + 2 x 0 + 3 x 0 + 4 x 0+ 5 x 0 + 6 x 0

1 x 0 + 2 x 1 + 3 x 0 + 4 x 0 + 5 x 0 + 6 x 0

1 x 0 + 2 x 1 + 3 x 0 + 4 x 0 + 5 x 0+ 6 x 0

1 x 0 + 2 x 1 + 3 x 1 + 4 x 1 + 5 x 0+ 6 x 0

1 x 0 + 2 x 0 + 3 x 1 + 4 x 0+ 5 x 0 + 6 x 0

1 x 0 + 2 x 0 + 3 x 0 + 4 x 0+ 5 x 0 + 6 x 0) = 16 / 6 = 2,666667 ≈ **2,7**

These calculations are shown with the formula below:

X- coordinates

$$\text{Mean\_x} = \frac{1}{6}\sum_{x,y}^{6} x \times B(x,y)$$

$$\text{Mean\_x} = \frac{1}{6} \times 22 \approx 3{,}7$$

Y- coordinates

$$\text{Mean\_y} = \frac{1}{6}\sum_{x,y}^{6} y \times B(x,y)$$

$$\text{Mean\_y} = \frac{1}{6} \times 16 \approx 2{,}7$$

The coordinates for the center of gravity are calculated to be: **(3,7; 2,7)** the mean x, y, which also is illustrated in Figure 40.

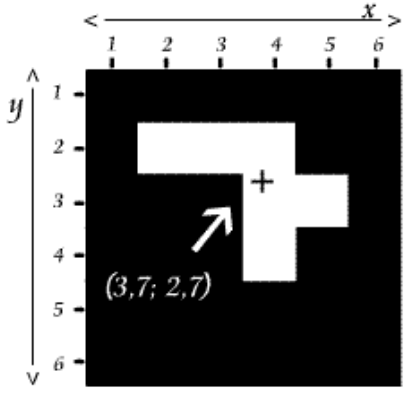


Figure 40: Center of gravity.

9.1.2 Computing the variance

The variance measures the variation of white pixel positions around the center of gravity[23]. This is computed through calculation of the average of all white pixels in x and y position of the image.

To illustrate computing of variance on the previous blob, use the white pixel coordinates from the previous equation and subtract them by the mean_x and y **(3,7; 2,7).** Then the output is powered by 2 (to avoid negative numbers) and summed together. Finally the total sum is divided with the number of white pixels (N). See the examples below.

$$\text{var\_x} = \frac{1}{N}\sum_{x,y}\left((x - \text{mean\_x}) \times B(x,y)\right)^2$$

var_x = ((2-3,7)^2 + (3-3,7)^2 + (4-3,7)^2 + (4-3,7)^2+ (5-3,7)^2+ (4-3,7)^2) / 6 $\Rightarrow$ 5,34 / 6 ≈ **0,89**

$$\text{var\_y} = \frac{1}{N}\sum_{x,y}\left((y - \text{mean\_y}) \times B(x,y)\right)^2$$

var_y = ((2-2,8)^2 + (2-2,8)^2 + (2-2,8)^2 + (3-2,8)^2+ (3-2,8)^2+ (3-2,8)^2) / 6 $\Rightarrow$3,44 / 6 ≈ **0,57**

The gray areas in the equation are coordinates for x and y.

9.1.3 Size of the surrounding box

The third parameter is calculation of the size of the box. When applying blob tracking, there is lack of information about the white pixels in the blob area. This is due to of information lost depending on the sigma (σ) range. The calculation below shows how to compute the size of the blob area, using the numbers from the previous example.

X- coordinates

$$\text{sigma\_x} = \sqrt{\text{var\_x}}$$

$\text{sigma\_x} = \sqrt{0,89} = 0,943398113 \approx$ **0,94**

Y- coordinates

$$\text{sigma\_y} = \sqrt{\text{var\_y}}$$

$\text{sigma\_y} = \sqrt{0,57} = 0,754983444 \approx$ **0,75**

[23] Slide 8: "*Blob tracking and applications*" by Volker Krüger

The box Mean_x ± 2sigma_x , Mean_y ± 2sigma_y contain 95, 44 % of all white pixels. The table below shows the percentage of white pixels at different range of sigma.

Range	CI
σ	68.26 %
2σ	95.44 %
3σ	99.73 %
4σ	99.99 %

Table 4: Standard deviation[24].

9.1.4 Corner positions of the box

The fourth and last parameter is the calculation of the corners of the bounding rectangle. The result from the calculations above is used. The formula for the corners are given below

Mean_x ± 2sigma_x, Mean_y ± 2sigma_y

In appendix A we have tried to use the above-mentioned formula on our previous example.
We have concluded from the calculation results in appendix A that the table of standard deviation is roughly true. Meaning that the percentage of white pixels within the box are approximate similar to the table above. However with the small amount of data in the binary image from our example there is a big difference from 2σ to 3σ, though with larger sets of data this would be impossible to see the difference.

9.2 Color image blob tracking

When tracking a color blob you first have to define the region for each color, through a *color histogram* or *Gaussian distribution*. This is necessary because a color is not always is the same. It comes in a number of variations.

[24] Slide 8: "*Blob tracking and applications*" by Volker Krüger

9.2.1 Color histogram

If a color histogram has been defined, every pixel in the input image is compared with the colors in the histogram. Inside the color histogram a column for each possible color in the image is created. These columns have different height depending on how many pixels exist within the color in the blob area. Thus the highest column represents a color pixel, which occurs most frequent within the blob. The output of the color histogram is a binary image. In other words if the color is highly registered in the columns in the histogram a white pixel is printed. If the color of the pixel is placed low in the columns of the histogram the output is a black pixel. Afterward the mean of the white pixels are computed, to find the center of the blob as explained earlier.

9.2.2 Blob tracking with the Gaussian distribution

The first step is to calculate and define the mean and the sigma values. The higher a variation of color the higher sigma is needed. Each pixel in the image is filtered and the output is a binary image. Then a pixel is considered to be less than sigma from the mean and the output is a white pixel. On the other hand if a pixel is judged to be larger than sigma from the mean the output is a black pixel. From of the binary image the center of gravity is computed as with binary blob tracking.

9.2.3 Region of interest (ROI)

The ROI in color images are computed by the average Red, Green, Blue values within the ROI. It is important that the ROI is defined so that it is not too large and not too small. If the ROI is too large it would create two blobs within the ROI, this would make it impossible to track the correct blob's center. If the ROI on the other hand is too small the movements in the image cannot be too fast, or else the ROI will have a hard time following the blob. Therefore the correct ROI has to be defined for the blob tracking to work optimal.

PART II:
The EyesWeb Experiments

In order to create the Human Remote Control (HRC) as described in the introduction, we have to create a number of experiments in EyesWeb, and through these gain a greater understanding of the problems involved. The theory explained in the first part of the report will be used as foundation.

This part of the report is divided into three steps:

- *The red hand experiments*, where the goal is to isolate the red hand using blob tracking and extract the coordinates from the blob.
- *Offline experiments (recorded footage),* the goal of this step is to implement the interaction with the movie clip (on/off, play, stop etc.).
- *Online experiments (live footage),* the aim of this step is to use the offline experiment patch and make it work with live input.

By isolating the different experiments from each other it is much simpler to identify and solve new and uncovered problems. If we were to tackle all the problems in one single experiment, we would properly still be working on the HRC this time next year.

All the milestone experiments[25] made in the process of creating the final online Human Remote Control patch will be described thoroughly herein. All the movie clips used and patches from EyesWeb are included on the accompanying CD-ROM.

A more general and detailed description of the different EyesWeb blocks used can be found in Appendix B.

[25] Milestone, meant as in, not including all the "what-does-this-block-do?" patches. They serve merely to gain experience with working with EyesWeb.

10 Problems to be considered

Before conducting the work with the experiments we tried to pin down likely problems that might influence our work, and the functions of the final patch. These problems are camera noise, resolution, change in illumination, incoming objects and the notorious “popcorn problem”.

Camera noise

When working with digital video and digital image processing in general, noise is always a parameter to be taken into consideration. Noise is not part of the ideal signal and may be caused by a wide range of sources, e.g. variations in the detector sensitivity (camera quality) and different environmental variations. Noise may cause problems when applying color tracking, because some of the pixels vary slightly in color value or intensity over time. There are different ways to get rid of noise, such as the median filter (described on page 35). But all pre-processing steps we apply to our digital video will slow down the process and we may run into problems if the EyesWeb patch gets too extensive.

Resolution

In addition to the noise problem, there may be a problem with the camera resolution. We will use a digital video camcorder with a resolution of 720x576 pixels and a frame rate of 25. This resolution is most likely too high, because of all the computations we will need to apply on the captured video.

Change in illumination

The illumination of an image is affected by the lightning and the reflections from different objects. If the lightning in the scene changes, objects will reflect more or less light and therefore change appearance. A change in illumination will cause all the pixels in the image to change intensity and color. This is a major issue when working with digital image processing. If the color of the red glove changes during the experiment, caused by illumination changes, the color tracking process might be difficult to perform. Therefore, we want to do the preliminary experiments in a closed environment, where we can manage the lightning ourselves. The problem of illumination is almost impossible to solve totally, therefore we don’t expect to solve this problem.

Incoming objects

If our experiments are disturbed by unwanted motion, such as a person walking in the background, it may give us problems when we want to extract the movements of the person wearing the glove. We want to know exactly where the glove is in the image, and how the position of the glove changes over time, so we can control a multimedia clip accordingly. But if a person or some other object disturbs this process, it will probably fail. Especially if the object has pixel values close to the color we want to track.

"The Popcorn Problem"

When will the system know it is being used or that the user is just eating popcorn? This is a problem we will have to solve in some way. We have to find a way to tell the system, whether it is active or not. Otherwise, the user can't do anything else with his hands while using the system, e.g. reaching for the popcorn bowl. A possible solution could be that the system only works when the user moves his hand in a certain area. Then, if the hand is outside this area, the system won't react to the movement of the hand.

11 The Red Hand Experiments

The following experiments were made to test out the color theory, including color spaces and blob tracking in connection with EyesWeb.

11.1 The Red Hand Experiments- Extended version

11.1.1 Experiment premise

This experiment was made as an extended version of a class exercise where the task was to single out the red hand of the "Micro-dance.avi" movie clip[26]. The exercise was made with background subtraction and we made a fairly successful result.

We then set out to recreate the same result but without using standard background subtraction. The entire background subtraction element of the process is left out because of the fact that we can't always rely on the being able to easily subtract the background[27]. We tried to choose a different path to achieve more or less the same result: isolation of the red hand.

11.1.2 Description of the EyesWeb patch

Patch filename: Red Hand - Extended Version.eyw
The movie clip used: Micro-dance.avi

The EyesWeb patch is fairly straightforward. The camera noise in the image is reduced with a 3x3 gaussian filter. Each of the RGB channel is separated and thresholded accordingly. The channel values are ANDed together and a median filter is applied to remove most of the remaining noise and the (now) binary blob is dilated to enhance its visibility.

The different threshold values are set to the following values:
Red = 100
Green = 45
Blue = 225

[26] Part of the standard EyesWeb program installation.
[27] Due to many reasons, e.g. moving background, no frame with none movement, etc.

These values where found by trial and error experimenting. The threshold value for the red channel could only differ by ±1. The green channel could differ by as much as ±10 and the blue channel could be in the range of 80-230 (but closer to 230 made the better result) with the end result would being more or less the same.

Some might argue that a median filter should have been applied after the dilation filter to remove all the remaining noise, but in this case the noise was so apparent in that the median filter is applied first. Otherwise the end result would have been an image with more than one clear object.
A median filter could also be added again after all this, but then it would only serve to "round" the dilated object[28]. When the patch is running, it is easy to spot where the red hand is in the moving image (see Figure 41 below).

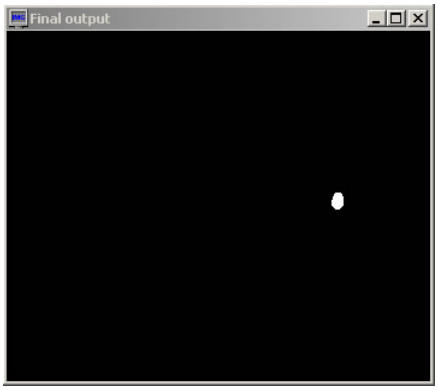


Figure 41: The Red Hand Experiment. Here shown with a second median filter following the dilation operation.

11.1.3 Summary

The end result of the extended version of the Red Hand Experiment was very satisfying. The only problem with the patch was some slight "flickering" in the final output. This was due to the threshold value being set so that the final output object in some cases was so small that the median filter would remove the object entirely.

But the question remaining was if the output result could be used in the experiments to come. This leads to the next experiment. Extracting coordinates from the moving red hand.

[28] EyesWeb can only work with square 2x2 structuring elements (Orally explained by supervisor Rune E. Andersen). A round structuring element would eliminate the need for a second median filter.

11.2 The Red Hand Experiments - Coordinates

In this experiment we continued on the extended version and set out to get EyesWeb to track the red hand that was now clearly visible.

11.2.1 Experiment premise

From the blob tracking we wanted to get the coordinates out so that we might be able to use these as variables in conjunction with controlling the movie clip.

11.2.2 Description of the EyesWeb patch

Patch filename: Red Hand - Coordinates.eyw
The movie clip used: Micro-dance.avi

In order for the color blob tracking block to operate successfully the color it is tracking have to be very easily singled out. Therefore the red color had to be separated from the rest of the color spectrum.

When applying color blob detection, we end up with a detection of the blob area (red hand) within a Region of interest (ROI). In the blob area it is possible to detect the center of the red hand, as seen in the images below.

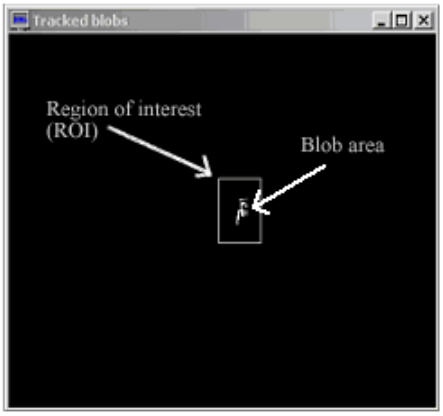


Figure 42: Region of Interest and the Blob Area.

We tried to adjust the ROI so that it fits the red hand in the image as shown below:

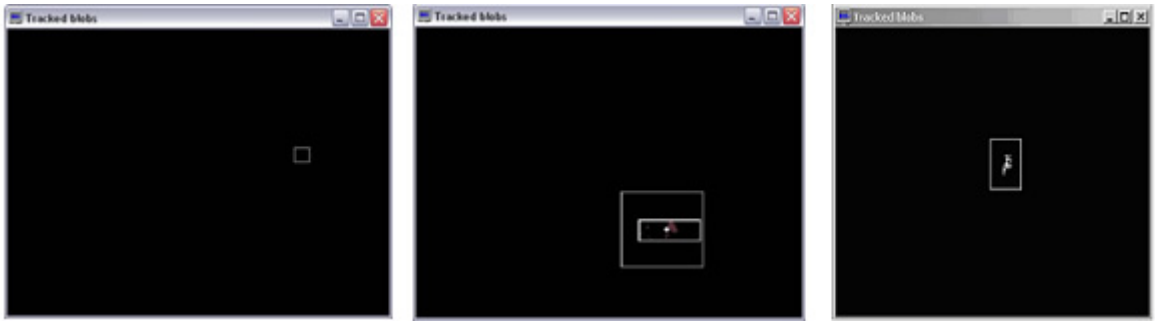


Figure 43: Different ROI sizes. Left: 15x15, middle 75x75 and right 35x50.

When the ROI is set to 15x15 it is too small, the patch has a hard time following the hand because of the red hands fast movements and disappears outside the ROI. If the ROI on the other hand was set to 75x75 it was too large and the ROI had problems with focusing on the center of the red hand because of the size of the ROI also included the red floor[29] as well. In our experiment the optimal size of the ROI is set too 35x50, where it follows the red hand closely.

11.2.3 Summary

By using color blob tracking in the experiment, useful coordinates are extracted. The extracting of these coordinates will make it possible to use the movement of the hands for further purpose.

We ran into a slight problem with the blob tracking block because it required a combined RGB signal and not a binary image like we had. However we used the Compose Channels[30] block to combine the three channels.

There were outputted coordinates that could maybe be used, but the overall patch did not perform as expected. It was still not entirely stable (the flickering from the previous experiment remained) and could very easily lose the center of the red hand.

[29] Or reddish floor in the Micro-dance movie clip. It is not entirely red but the color of the floor contains high red values.
[30] See appendix B for a detailed explanation of this block.

11.3 Conclusion on the Red Hand Experiments

These experiments took us only a little closer to the final goal of getting value out of a tracked blob that could be used in controlling a movie clip. Although we might have taken the wrong approach, they proved as experiments, to give valuable information about how the different tracking features of EyesWeb worked. The x,y coordinates from the blob tracking in the Coordinates experiment coincide with where the red hand is at, at a given moment[31]. The task is now, to take these coordinates further and transform them from being just that, coordinates, and into values that can control properties of a movie clip.

Before we go further with the interaction we have to stabilize the tracking, because without stability everything else is useless. The values from an unstable patch could cause the volume to rapidly go up and down and/or pause and fast-forward the movie clip continually.

[31] The coordinates are from the center of the blob (center of gravity) and not necessarily the exact center of the red hand.

12 Offline experiments

These experiments are built on the foundation from the red hand experiments.We will try out different experiments on a number of different backgrounds with various amounts of movements and different lighting situations.

12.1 Setting up the offline experiment

In order to have good uniformed footage we had to make the exact same set up for every environment we recorded in. We put the camera on a tripod to get good full view of the “actor”, and placed tape on the floor for the actor to know where to stand on each recording. We measured the distance from the actor’s feet to the wall, from the actor to the camera, and from the lens of the camera to the floor. In addition the zoom of the camera were the same for every recording as well. Therefore the setup was the same on every location. The set-up shown in Figure 44 below is from the auditorium at Erhvervsakademiet in Copenhagen called “Færgen” (The Ferry).

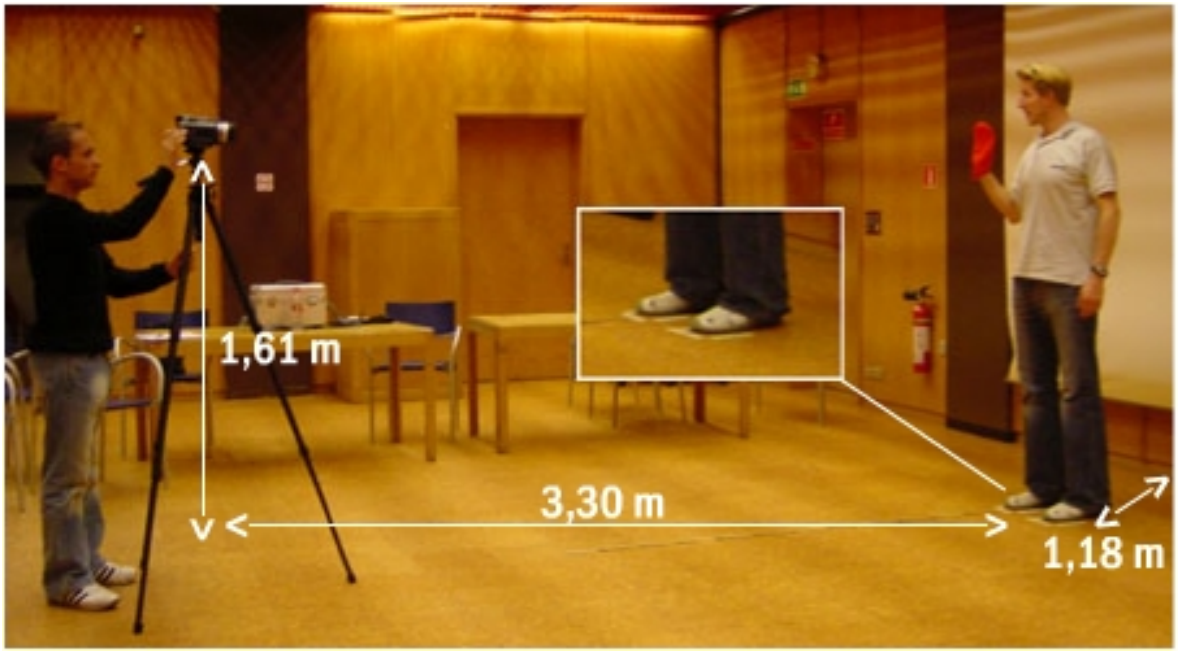


Figure 44: The Setup.

The actor moved a red glove upwards/downwards, sideways and by turns closed and opened the hand. These were all moves, which we were going to work with in EyesWeb, see also the user

manual in appendix C. We actually did recordings with both a blue but a red glove and ended up using only the red.

Concerning the experiments outside the color-tracking domain[32]

Even though we planned on tracking a color we did some experiments with motion tracking. This was rather complicated. This was due to the lack of refining what kind of motion to track, and we wanted merely to track the hand and *not* the whole arm and *not* the left arm eating popcorn etc. Moreover the shape of the hand could be a way to start experimenting but object recognition is hard and even if we succeed in recognizing the hand, the remaining problem would be the second hand and how to tell the computer to ignore one of the hands. Furthermore it would be nice to be able to scratch your leg with one hand and control the video with the other. So we concluded to stick to the color-tracking (wearing a red glove) domain and make it good.

12.2 Offline experiments – Stability

All the patches made with tracking the red hand all had the same problem. They were too unstable to use. This was a major problem that had to be solved and the first to be tackled in the offline experiments.

12.2.1 Experiment premise

Hopefully minor tweaking of the previous red hand patch could solve the stability problem. Combined with record footage made for our purpose it should make a more suitable patch.

12.2.2 Description of the EyesWeb patch

Patch filename: Offline Experiments - Test 9_STABLE-color-blob.eyw
The movie clip used: red_white-bg_02.avi

We intended to make the patch rock solid and some minor adjustments to the different blocks did the trick.

[32] See Offline Experiments – Test 1 to 8 on the CD-ROM.

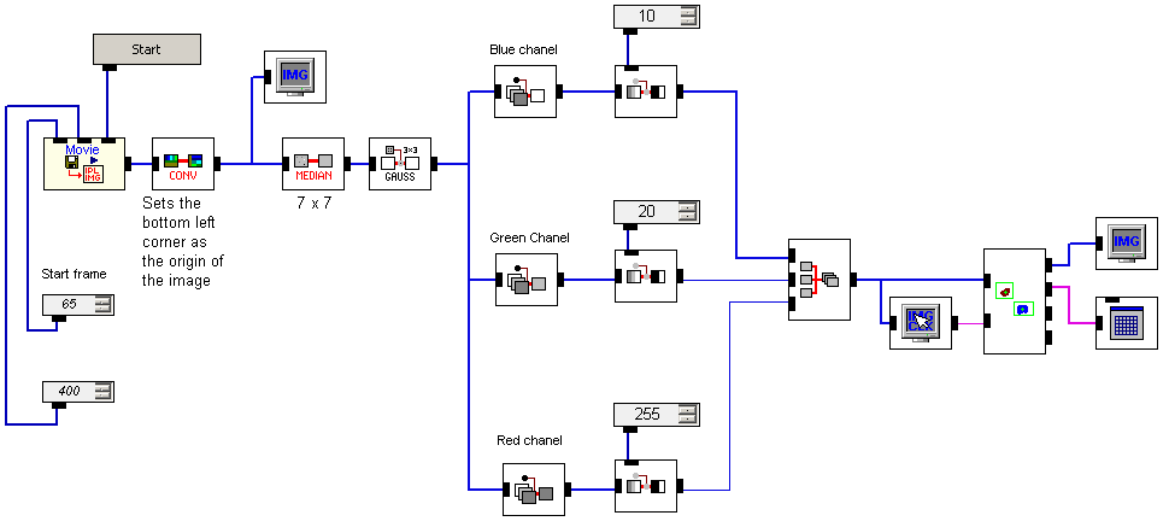


Figure 45: The entire "Offline Experiment – Test 9" patch.

The different threshold values were set to almost none blue and green (10 and 20 respectively) and full red (255). The output can be seen in Figure 46.

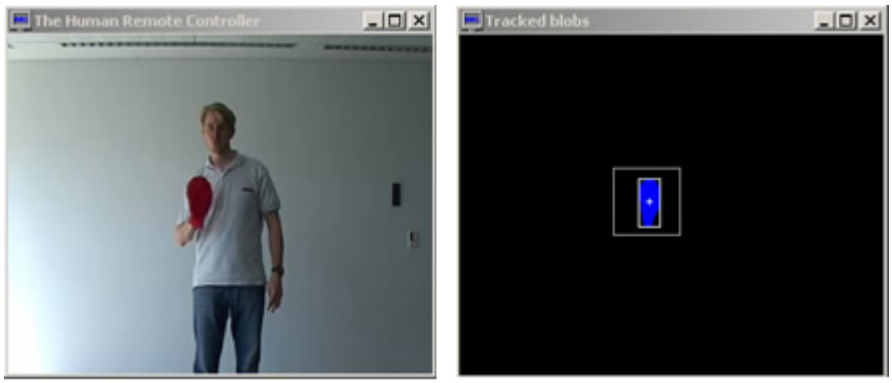


Figure 46: The output of the patch.

12.2.3 Summary

By playing around with different threshold on the three color channels and using different sizes of median filters and we ended a pretty good result on a specific movie clip.
However other movie clips wouldn't necessarily work because of the color selection block, which needs all three colors hence not isolating the red hand entirely. We have tested the patch on several movie clips and particularly one is interesting because the test person moves the hand in front of the dark blue jeans, and at that point the blob is unable to track the red hand. Blobs in the ROI is changed and follows the jeans. The same thing will happen with strong green trousers or a sweater etc.

A small note: Either we used the Convert block wrong or more likely, it doesn't work. It is set to convert the origin of the image to the bottom left corner but it never happens. The origin is still the top left corner. So the y coordinates are lowered when moving the hand upwards and vice versa.

12.3 Offline experiments – Interactivity

The next logical step was to use the coordinates from the previous stable patch to create some interactivity.

12.3.1 Experiment premise

The x,y coordinates from the blob tracking should prove sufficient material to create the desired functionalities. At this stage we will try to implement some, if not all, of the functionalities of the HRC. It was also time to create a way of solving the "Popcorn Problem" (the system's activation on/off).

12.3.2 Description of the EyesWeb patch

Patch filename: Offline Experiments - Test 14_STABLE-switch.eyw
The movie clip used: red_white-bg_03avi

The most prominent change in this patch was that the color blob tracking element was replaced with the binary blob tracking and using parts of the setup from earlier Red Hand Experiments.
At this point the change is of minor value, but later when we'll start on making the patch use less computer processing power because binary image processing requires fewer computations than full color images.

As stated earlier, the Convert block doesn't work so we had to invert the y coordinates in order to control the sound volume.

The system can either be active or inactive depending on the size of the hand. When the hand is open, the blob area is large and the system is active and react to the inputted x,y coordinates and opposite if the hand is closed. A zero or a one displays whether the system is on/off. The block sends the boolean data to the switch telling it to be true/false (see Figure 47 on next page).

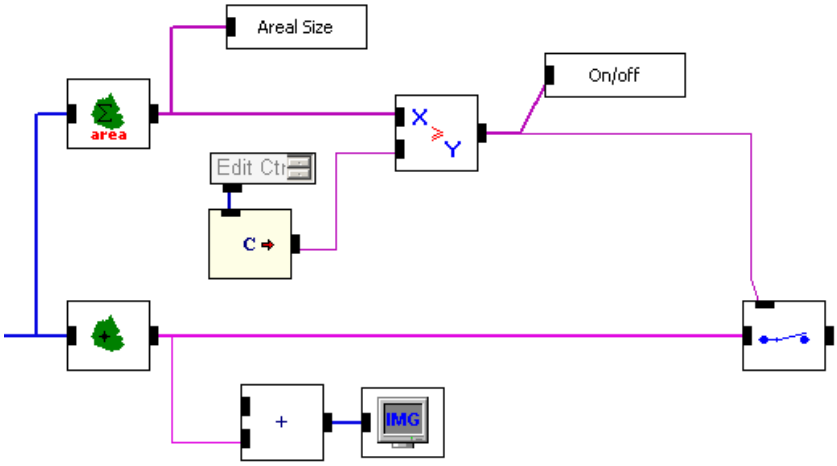


Figure 47: The activation of the system.

The play/pause video part of the patch is very simple and works like an “if sentence”, as follows: if the hand’s x coordinates are higher than 110 it pauses the movie clip and if the hand’s x coordinates are lower than 110 it plays the movie clip. The switch blocks are a simple way of controlling the commands of the MultiMediaFileReader block. The blocks that trigger the patch to play or pause are basically the same, varying only in what numbers that initiate the action.

We succeeded in making a more uniform patch, and now works on several different movie clips (check the accompanying CD-ROM it should work on following movie clips):
red_greenscreen_02.avi
red_white-bg_01.avi
red_white-bg_02.avi
red_white-bg_03.avi

It partially works on footage taken in bright sunlight see the following movie clips:
red_leaves_01.avi
red_leaves_02.avi

It won’t work on all the “ferry” clips because of the high reddish illumination present in the room and neither will it work on the movie clips with the background paintings in it.

12.3.3 Summary

All in all we found out that we needed to work on a patch with the HSI color model to make the patch function in different illuminations. In addition we have found a few flaws in the interaction, not that the patch is unstable in particular but the definition of the interaction. It is not possible to set the system to remember your moves and act on them accordingly. For instance if you turn up the volume to 50 and want to keep it there you close the system (close the hand) and it remains 50 but if you then open the hand elsewhere in the coordinate system, say lower, the volume will turn down. This is a problem because we want to make a flexible system, in which the user can control the volume and fast-forward the movie without letting one interaction influence the other.
To solve this problem we define a virtual coordinate system in which certain regions are interactive. See the user manual in appendix C.

The EyesWeb program gave us an unexpected limitation, hence there was not a single block that could both control a movie clip and its soundtrack. This meant that we had to create a separate sound control element of the patch. The blocks to control the movie and the sound don't have the same functionalities which resulted in that the movie and sound are never really in-sync.

12.4 Offline Experiments – Final Version

Patch filename: Offline Experiments - Test 20_FINAL.eyw
The movie clip used: red_white-bg_03.avi

12.4.1 Experiment premise

We had to solve the problem with the changing illumination. As of now the patch was only able to track the red hand in certain light settings.

MultimediaFileReader block in EyesWeb contain a bug that after some time "locks" the framerate value beyond control, so we had to find another way of controlling the framerate. The bug seems to be periodical and is not caused by anything specific[33].

[33] In the EyesWeb newsgroup for bug reporting (news://infomus.dist.unige.it/eyesweb.bugs) Barbara Mazzarino of the EyesWeb Group suggested that it might be a compatible problem between Microsoft DirectX 9.0 and EyesWeb 3.2.1, but we found that the bug is present in different version of DirectX as well.

As mentioned above it was necessary to create a system of retaining the volume level and not constantly turning the volume up and down.

Furthermore it was time to make the entire patch more “economical”, since the patch now have so many resource intensive operations (especially the filters) that it can slow down even a fast computer.

12.4.2 Description of the EyesWeb patch

To solve the illumination problem we change the color model from RGB to HLS[34]. By doing so we were able to control the hue, illumination and saturation separately, and threshold the channels the same way we thresholded the red, green and blue channels previously.
The patch can now operate in a much wider variety of environments with changing light. Almost all of our recorded footage with a red hand can now be used.

The periodic bang block can replace the framerate control and instead use the in-sync feature of the MultimediaFileReader block. The “bang” has to be inverted because of the very nature of it. The higher value inputted to the periodic bang block the slower the framerate. This is because that the value inputted is at what rate the bang should sound. The higher value means that there will be more time between the bangs and vice versa.
To calculate what value domain the periodic bang should operate it is needed to calculate the framerate per milliseconds. A standard movie clip has 25 frames per second. This equals 0,025 frames per millisecond, or 1 frame per 40th millisecond. A framerate of 100 equals 1 frame per 10th millisecond. The value domain has to go from 40 to 10, which equals the desired framerate from 25 to 100.

[34] HLS in EyesWeb corresponds to the HSI color model as described on page 21.

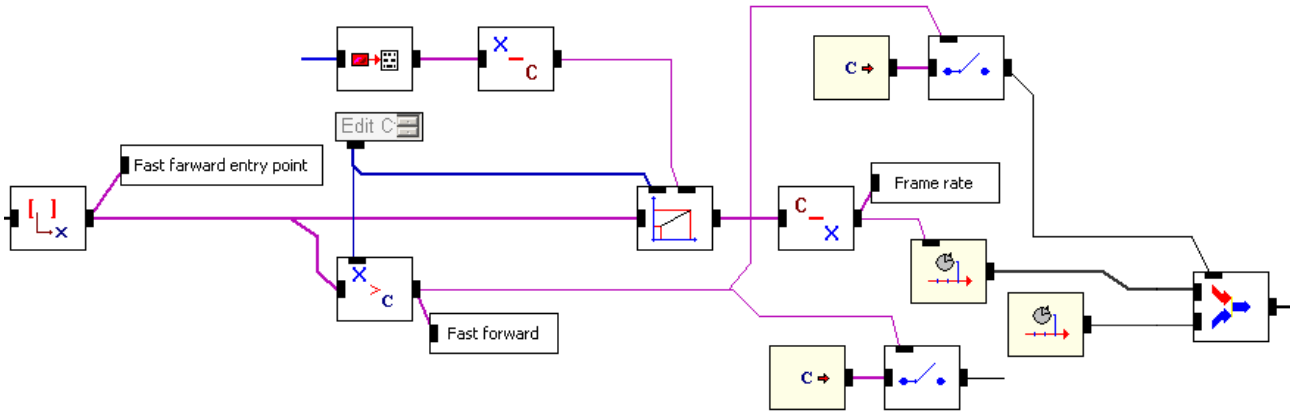


Figure 48: The "Fast Forward" part of the patch.

As part of the process in making the patch more economical we replaced all the logical operators.

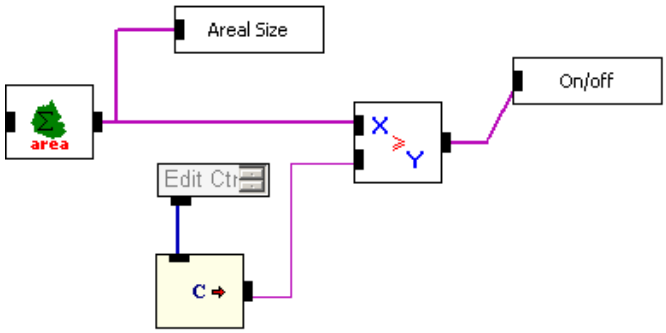


Figure 49: The logical operators: Before.

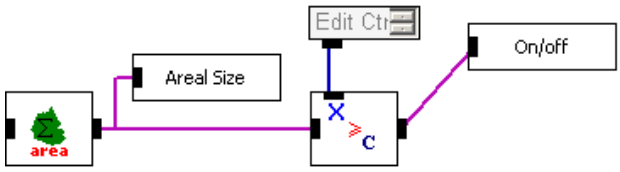


Figure 50: The logical operators: After.

The operation is exactly the same, but using fewer blocks, and although the Constant Generator block doesn't use that many system resources, the overall resource demand of the entire patch is reduced.

The initial noise removal section, consisting of the 7x7 median filter and the 3x3 gaussian filter, were reduced to a single 3x3 median filter. Another 5x5 median filter and 3x3 gaussian filter were

added after the inputted video image became binary, making fewer calculations needed for these filters.

A switch was added to the sound control part to “lock” the volume, once set. The principals behind this switch are the same as used creating the system activation. If the hand is above a certain y coordinate the system is reacting to volume adjustments otherwise not.

12.4.3 Summary

The patch is now stable and has become so economical that it can operate in real-time. It solves all the technical problems we have encountered during the process of creating the Human Remote Control in EyesWeb. Furthermore the problems stated in the section “Problems to be considered” on page 46 have all, but one, been solved. At this point Patch 20 is declared the final Offline Patch.

12.5 Conclusion of the offline experiments

We never solved the incoming objects problem. All the offline experiments documented fail if another red element is inserted. It is possible to control a certain degree of red elements and not allowing them to influence the process, but red elements of the exact same color as the red hand will always corrupt the experiment.

The illumination problems are partly solved. If the input is very dark or extremely bright the patch will not work. But this problem is almost impossible to solve.

Apart from these shortcomings the experiments worked much better than expected. We were able to track the red hand and with the x,y coordinates from this, control sound and video. This final version of the offline patches is very close to what we originally set out to create.

13 Online experiments

We will more or less try out the same experiments but with a live input primarily from a webcam. Our goal is to make the setup work so that anyone can try the Human Remote Control.

13.1 Setting up the online experiment

For the online experiment we started with feeding EyesWeb with a Digital Video (DV) input, which worked adequate, almost similar to the footage taken earlier. But since we wanted to test out our system without expensive equipment we ended up buying a webcam to experiment with. The webcam worked quite well, and placed on top of laptop we could do recordings on different locations.

Figure 51: Interaction through webcam.

One thing, which was entirely different, was the limited viewing area of the camera if sitting too close to the camera. Meaning that the “actor” didn’t have to move the red glove much before it was outside the camera’s view.

13.2 Description of the EyesWeb patch

Patch filename:	Online Experiments - Test 7_FINAL.eyw
Input:	Live feed from a webcam.

The patch is similar to the final offline patch. Here the input is just replaced with at live feed from a webcam. A zoom function is added as the only new element. The user can by moving his/her hand closer to the webcam, zoom in on the area where the hand is currently located.
It is a two-step zoom. The closer you move your hand to the webcam the more you'll zoom.

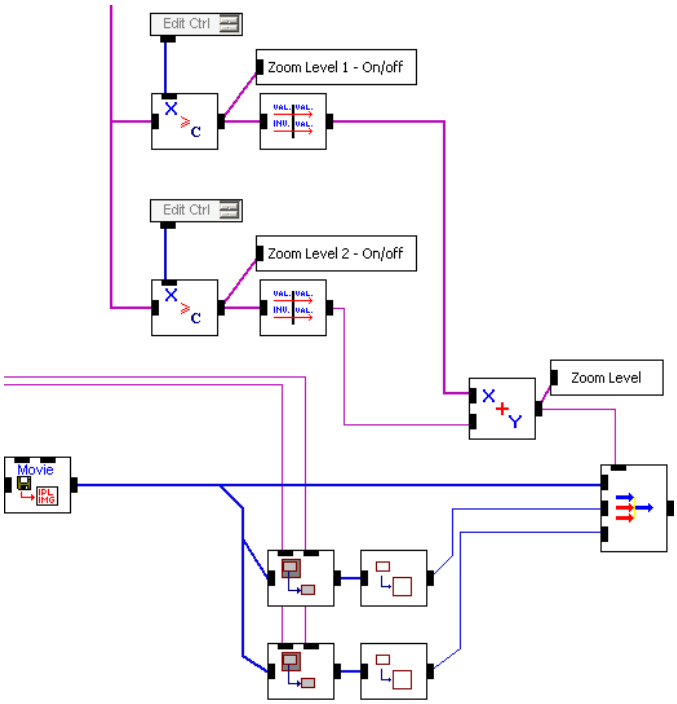


Figure 52: The zoom function.

13.3 Conclusion of the online experiments

Because of the solid foundation of the offline experiments the migration to online experiments was very easy and painless. However the patch has to be manually calibrated before it works in a given environment. This is partly due to the change in illumination on some of the locations. Also it is needed to calibrate the distance from the user to the webcam.

We have not to found a way to create a unified zoom function in EyesWeb. If the controlled movie clip is changed (image size) it is needed to adjust the blocks used to perform the zoom.

The "incoming objects" problem is still left unsolved.

14 Conclusion

During our project we have been working on the Human Remote Control. We have succeeded in developing a system, in which users are capable of controlling movie- and sound-clips using only a red glove. More specific, they can pause and play the clips, control the volume, zoom in/out and fast-forward. The system is limited to work inside EyesWeb only, but works with both recorded footage and live video from a webcam.

We have encountered some problems in the experiments, both expected but certainly also unexpected problems. Some of the unexpected were mostly small problems concerning the EyesWeb software, because the program is in development and contains some software bugs and limitations. For instance, it was not possible to import the soundtrack from a movie clip, so we had to use a separate sound clip in order to demonstrate the volume control. Meaning that the video and sound clips are not synchronized to each other, especially when the fast-forward is activated.

We succeeded in making a fairly unified system, which works in different kinds of illumination. However the illumination problem has not been completely solved. If the environment is very dark or light, the color tracking of the red glove was not possible. But the conversion from RGB to the HSI color model was the primary factor in partly solving the illumination problem.
Camera resolution was not as big a problem as expected. We thought it would consume a lot of computer power if the resolution was too high, but it was not the case. Maybe because we did an effort to make the patch more economical so it was less dependent on computer hardware.
Similarly, the noise problem was easily solved with the use of smoothing filters (median and gaussian). Going from the DV camera to the webcam, a slight increase in noise occurred but that did not prove to be a problem.

By making an on/off system function it was possible to eliminate the “popcorn problem”. By default the system is set to play in the middle of the interactive user space, so the user don’t have to concern himself about “wrong” movements in this area.
We never solved the problem of incoming objects except objects with other colors than red.

As declared in the problem statement, the aim was not to produce a final computer product but merely to make a prototype and learn from the process. Although we have reached a rather

satisfying result, we think we have a long way ahead of us. The HRC is a little awkward to use, but that problem could perhaps be solved with a more refined version of the glove and the interaction. Additionally the system would need an auto calibration of illumination of the environment and measure the actual distance from the user (remote) to the webcam.

Through the experiments we have used theory concerning digital filters. We have discovered that you cannot generalize and say that one filter or method is perfect for one particular purpose. It depends on the combination of filters and the given context.

Our project can easily be modified to accompany other purposes. Teachers could use it for controlling PowerPoint slideshows, and security officers could control monitors showing different surveillance cameras with their hands.

When all is said and done we are satisfied with the result of the project. We have learned a great deal about digital image processing and how to use EyesWeb to implement some of this knowledge.

The project is left with an open ending and hopefully our concept will serve as inspiration for others trying to create similar experiments. Much is needed before the HRC replaces the one from Minority Report, but it will definitely be possible before 2054.

15 References

15.1 Lectures

Lectures by Volker Krüger, Aalborg University Esbjerg.
All the lectures used are included in the enclosed CD-ROM.

#1: ”*About Images and Pixels*”.

#2: ”*About Colors*”.

#3: “*Computing with Images*”.

#4: “*Convolution and Applications*”.

#5: “*Applications and Geometry*”.

#6: “*Morphological Operators*”.

#8: “*Blob Tracking and Applications*”.

15.2 Books

Christensen, Marie and Fischer, Louise Harder: ”*Udvikling af multimedier*”. Denmark, 2001. Ingeniøren Bøger, ISBN: 8757122903.

Matlin, Margaret W. and Foley, Hugh J.: “*Sensation and Perception*”. Fourth Edition. USA, 1997. Pearson Higher Education, ISBN: 0205263828.

Gonzalez, Rafael C. and Woods, Richard E.: “*Digital Image Processing*”, First Edition. USA 1992. Addison-Wesley, ISBN: 0201600781

Gonzalez, Rafael C. and Woods, Richard E.: “*Digital Image Processing*”, Second International Edition. USA 2002. Prentice-Hall, ISBN: 0130946508

Russ, John C.: “*The Image Processing Handbook*”, 3rd Edition. USA 1998. CRC Press, ISBN: 0849325323

15.3 Internet resources

Hypermedia Image Processing Reference (HIPR2):
http://www.dai.ed.ac.uk/HIPR2/

Rune’s EyesWeb Notes:
http://cs.aue.auc.dk/~rea/eyesweb/

PHYS198/EECE226, a course on digital image processing:
http://academic.mu.edu/phys/matthysd/web226/L0228.htm

15.3.1 Research links

Laboratory of Computer Vision and Media Technology, Allborg University:
http://www.cvmt.dk/

FG-NET:
http://www.cvmt.dk/~fgnet/docs.html

Sebastien Marcel's Gesture Database Web Page:
http://www.idiap.ch/~marcel/Databases/main.html

Dr. Huang, Yu's Homepage:
http://www.ifp.uiuc.edu/~yuhuang/publish.html

Eye Controlled Media: Present and Future State, Datalogisk Institut, Copenhagen:
http://www.diku.dk/~panic/eyegaze/

The Image Group, Datalogisk Institut, Copenhagen:
http://www.diku.dk/forskning/image/

The Music Informatics Laboratory, Datalogisk Institut, Copenhagen:
http://www.diku.dk/forskning/musinf/

Appendix A: Corner positions

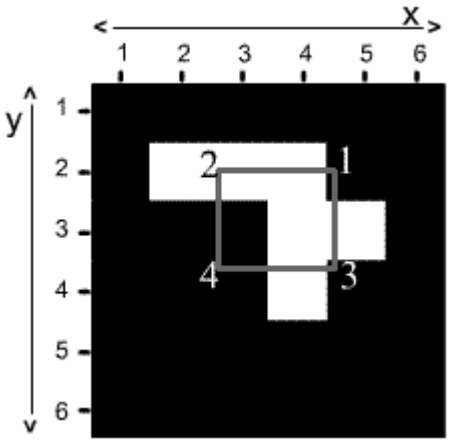
σ: 68.26 %

1 mean_x + sigma_x 3,7 + 0,94= 4,64

2 mean_x - sigma_x 3,7 - 0,94= 2,76

3 mean_y + sigma_y 2,7 + 0,69= 3,45

4 mean_y - sigma_y 2,7 - 0,69 = 1,95



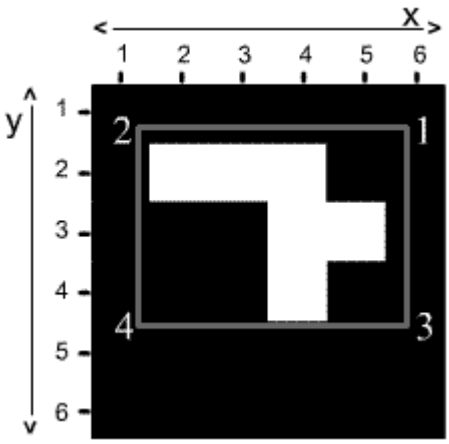
2σ: 95.44 %

1 mean_x + 2sigma_x 3,7 + 2×0,94 = 5,52

2 mean_x - 2sigma_x 3,7 - 2×0,94 = 1,88

3 mean_y + 2sigma_y 2,7 + 2×0,69 = 4,08

4 mean_y - 2sigma_y 2,7 - 2×0,69 = 1,32



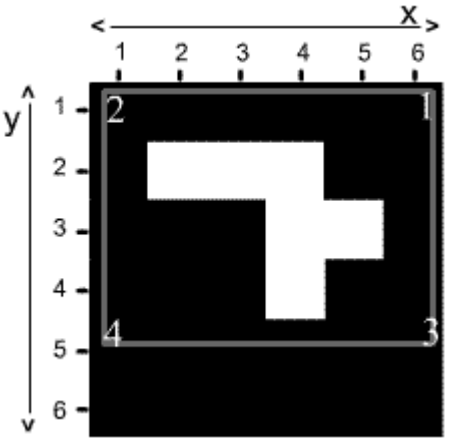
3σ: 99.73 %

1 mean_x + 3sigma_x 3,7 + 3×0,94 = 6,52

2 mean_x - 3sigma_x 3,7 - 3×0,94 = 0,88

3 mean_y + 3sigma_y 2,7 + 3×0,69 = 4,95

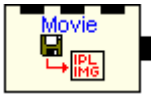
4 mean_y - 3sigma_y 2,7 - 3×0,69 = 0,45



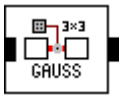
Appendix B: EyesWeb Blocks

Block: | **Name and function**

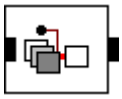
Patch Start:
Normally the first block of a patch. This patch command will enable the start function of the block it is connected to. E.g. to start a movie clip.



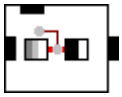
Multimedia File Reader:
Input > Imaging > MultimediaFileRead
This block loads the movie clip input. It is also possible to change the parameters of the playback in this block, e.g. the frame rate, start and stop frame, loop, etc.



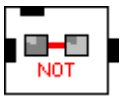
Linear Filter (gaussian filter):
Imaging > filters > LinearFilterFixed
A gaussian filter can be applied to smoothing/blurring the image, making the difference in the color values in the separate pixels less apparent, and to remove the initial camera noise.



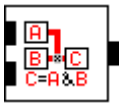
Extract Channel:
Imaging > Operations > ExtractChannel
The movie clip can be separated into the three screen colors RGB[35], one channel for each color. This way you can have total control over each channel. The same block can also be used to extract channels from other color models as HLS, YUV etc.



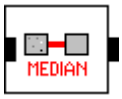
Threshold:
Imagine > Operations > Threshold
Each of the separated color channels then passed through a threshold to control how much of the specific input that is being passed on.



The NOT operation:
Imaging > Operations > MonidacLogicalOp
Through the NOT operator the values are being inverted and only the inverted values are allowed to pass.

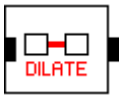


The AND operation:
Imaging > Operations > DyadicLogicalOp
This block is used to add two images together. The result is only true if both values are true.



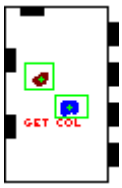
Median filter:
Imaging > Filters > NonlinearFilter
The median filter block is usually applied to remove remaining camera and image noise, giving a more clean output image. The filter can have a size of odd numbers 1x1, 3x3….7x7.

[35] The common way to express screen colors is in the order Red, Green and Blue (RGB) but in EyesWeb the order is usually BGR (Blue, Green and Red).

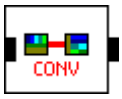


Dilation filter:
Imaging > Operations > MorphologicalOp
The block Dilate is applied to enlarge the foreground and shrinking the background. The structuring element used in EyesWeb is a square 2x2 element[36]. The size of the structuring element cannot be changed in EyesWeb, but the number of iteration can.

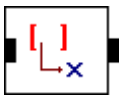
Click display:
Imaging > Output > ClickDisplay
By including the click display block the user gets the possibility of clicking with the mouse on the blob he or she wants to track.



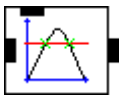
Color blob tracking:
Imaging > Operations > ExtractMultColors
The block is tracking the blob(s) and give x,y coordinates of its position in the spatial domain.



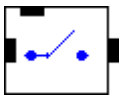
Convert:
Imaging > Conversion > Convert
This block is used to set the top left corner as the origin of the image; it is helpful when dealing with the coordinates. However the convert block is not functioning correctly as described in the summary on page 55.



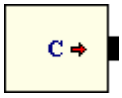
Get Entry:
Math > Matrix > GetEntry
The GetEntry block takes eithers the x or the y coordinates and passes them onto the next block.



Threshold Crossing:
Math > Scalar > ThresCrossing
The Threshhold crossing block acts like a cut-off. When the inputted value crosses the threshold a command is given (Boolean 0 or 1).

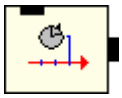


Switch:
Generic > Switch
The switch block activates the object it is connected to. The block is by default set to false and works like a Boolean operator. This block needs input in order to work, be that either values from a constant generator or a stream of values from other blocks.

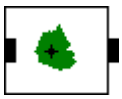


Constant Generator:
Math > Scalar > Input > Generator
The generator block's main action is to constantly send out values. The values can be set to be either real or integer values.

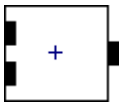
[36] Orally explained by supervisor Rune E. Andersen, Aalborg University, Esbjerg.



Periodic Bang:
Generic > PeriodicBang
This block can be used in a numerous ways. The block gives out a “bang” at a user-defined interval. The lower the number the faster.

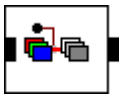


Baricenter:
Imaging > FeatureCalc > Baricenter
The baricenter block calculates the center of gravity of all white pixels in the input video and outputs coordinates according to this.

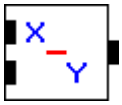


Draw Point:
Imaging > Draw > DrawPoint
The draw point block draws either a cross or a circle (optional) on the center of the block defined in the baricenter block.

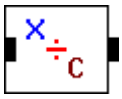
Area:
Imaging > FeatureCalc > Area
This block calculates the area of all the white pixels in an image and outputs this number.



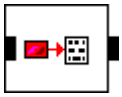
Convert Color Model:
Imaging > Conversion > ColorModel
The block converts the current color model to another. E.g from RGB to HLS.



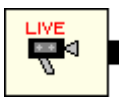
Binary Operator:
Math > scalar > BinaryOp
This block performs a chosen operation (add, subtract, division, multiplication and other basic arithmetic operations) on input 1 and 2.
E.g. The value in input 1 is subtracted the value from input 2.



Constant Operator:
Math > Scalar > ConstOp
This block works similar to the binary operator, but operates with a constant as one of the inputs.
E.g. The inputted value is always divided by 3.

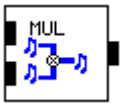


Get Description:
Imaging > GetDescription
Gets the description of the inputted image/video. This description can be the pixel height, depth, and width among many other things.

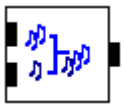


Live video:
Input > Imaging > Framegrabber
The FrameGrabber block enables EyesWeb to capture a live video feed from at video camera or webcam. The inputted feed is used in the exact same way as recorded footage.

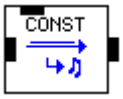
Audio File Reader:
Sound >Input > AudioFileReader
This block loads a specified sound clip. This can be either in MP3 or Wave format.



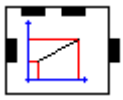
Binary Operator:
Sound > Operations > BynaryOp
This block performs a binary operation on the two input buffers (same type; length, number of channels, and sampling frequency. Different types of operations can be chosen.



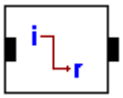
Compose Channels:
Sound > Convention > ComposeChannels
The ComposeChannels block converts the single channel control signal to a double channel control signal.



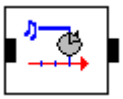
Generate External Clock:
Sound > Input > GeneratorExternalClock
This block generates the volume control signal.



Linear Rescale:
Math > Scalar > LinearRescale
This block converts the inputted x coordinates to real values between 0 and 1.



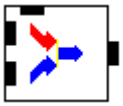
Converts Domain:
Math > Scalar > DomainConv
The DomainConv block converts the integer values to real values or vice versa.



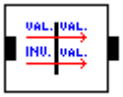
Extract From SoundStream:
Sound > Clock > ExtractFromSoundStream
This block extracts the clock signal from the audio device.

Wave Input:
Sound >Input > FixedBufferWaveInput
This block captures waveform audio from specified device. It is possible to choose a device, the buffer size unit and the sampling frequency.

Wave Output:
Sound > Output > FixedBufferWaveOutput
This block plays waveform audio on the specified device.



2 to 1 Input switch:
Unsupported > Generic > InputSwitch > 2 to 1 Sync
This is an input switcher. The output of this block is dependant on what channels is chosen (channel 0 or 1). There also exist a 3 to 1 and a 4 to 1 input switch.



Invalid Handler:
Math > Scalar > InvalidHandler
The Invalid Handler block is added to prevent the logical operator to become invalid if there are no white pixels to calculate the area from.

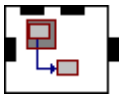
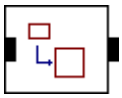
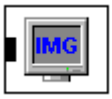


Image Extract:
Imaging > Operations > Extract
Extract a specified region of an image.

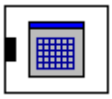


Zoom:
Imaging > Operations > Zoom
Scales the inputted image to a factor specified. The factor can be either only on the x or y axis or both.



Display:
Imaging > Output > Display
The block display is necessary to view movie clips.

Scalar Display:
Math > Scalar > Display
Displays the scalar values.



Matrix Display:
Math > Matrix > Output > Display
This block is showing the output as a matrix.

Appendix C: User manual for the HRC

To operate the system you have to activate the system on and this is done by opening the hand. And you close your hand to close the system.

Figure 53: Open system.

Figure 54: Closed system.

The Figure 55 below shows the areas in which the user can interact with the system still depending whether the hand is open/closed. Within some areas it is possible at the same time to control the volume and play the movie clip. In the middle of the figure there is a large gray square, and in here the movie plays by default. Another feature in the system is a 2-step zoom-function. Moving the open hand towards the screen (the webcam) makes it possible to zoom in/out of the picture.

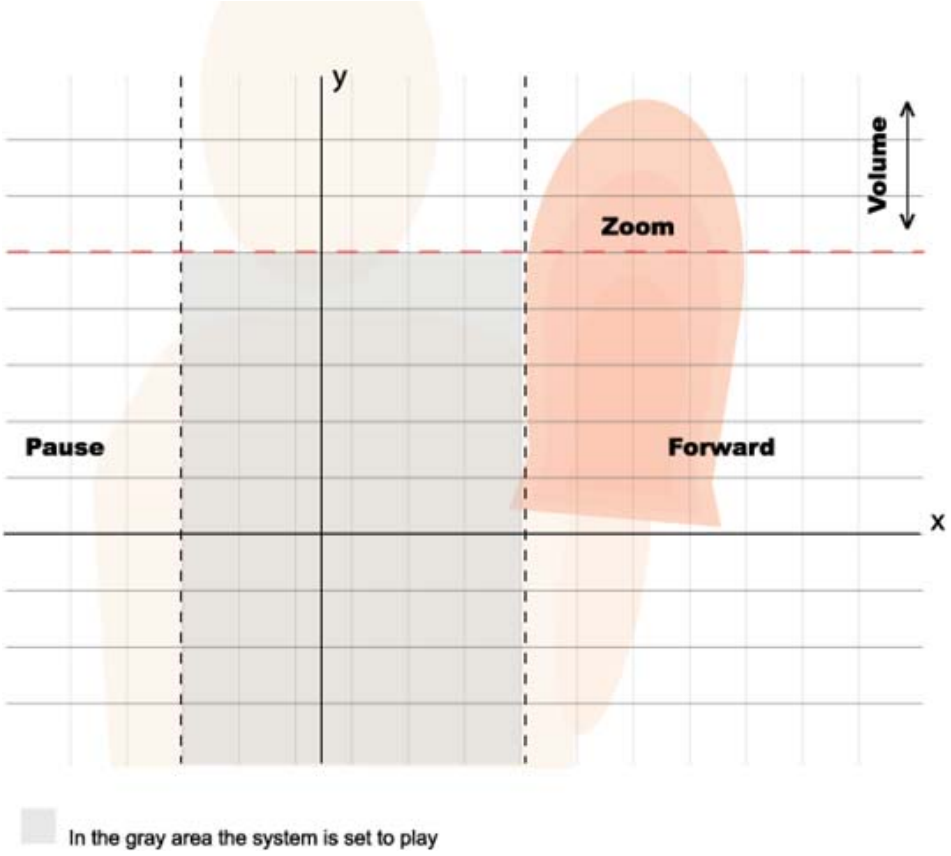


Figure 55: The interactive coordinates system.