Bachelor Assignment, SCS, University of Twente, 7-7-2017

Illumination for a finger-vein

Bart van der Scheer

Supervisors

Prof.Dr.Ir. R.N.J. Veldhuis

Ing. G.J. Laanstra

Abstract. Finger-vein images can be used as biometric, which can be used to identify a person. The veins can be made visible by infrared light. The most common way to do this is by using emitters above the finger, where light spreads around the bone and makes the veins visible. Then an image of the veins is made by placing an infrared camera below the finger. A design was made to apply light from different angles to the finger to find out how this compares to lighting from above. We found that when you illuminate the finger from the sides there is reflection of creases and features of the skin. The image is locally saturated due to strong emissions through the finger. This does not occur when you place the light source above the finger, and there is no light reflected in to the camera. When light is applied from below there is a lot of reflection which makes it impossible to see the veins, but lighting from the side makes a lot of veins visible comparable to lighting from above.

Table of Contents

Introduction

Finger veins are a biometric, a specific biological feature of a person like finger prints. Finger veins scanners already exist that use infrared light to make the veins visible so it can be used to identify a person. Finger veins are more secure than finger prints because they are not easily copied. It is used commercially already in banks and hospitals, which testifies to its reliability.

Previous research

The university has a finger-vein scanner that is used for research, which can be seen in Figure 1. This setup is used for research at the university to produce finger-vein images

This setup is used to produce a database of finger-vein images, used for testing and evaluating identification algorithms

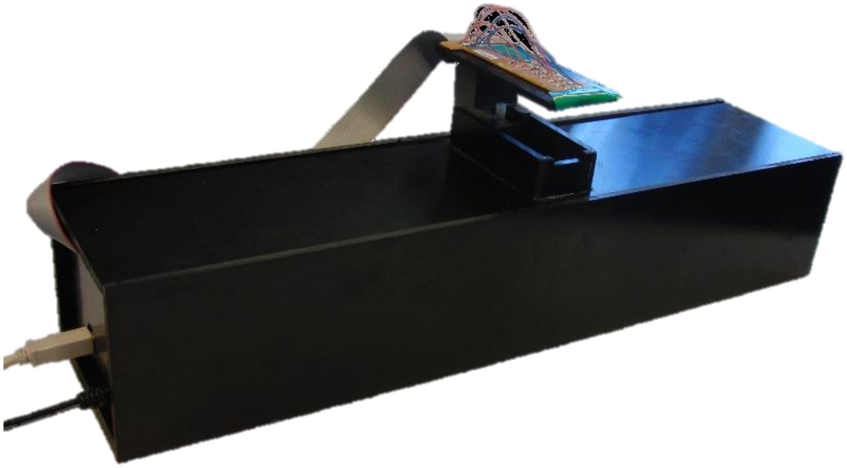

Figure 1 Finger-vein imaging setup used for research

Besides lighting from above it is also possible to illuminate from below or from the sides, which could increase vein uniformity. An open design would also make it more user friendly.

There is already a design by Hitachi that uses lighting from the sides[7]. It places the emitters horizontally next to the finger and light is transmitted through the finger. It claims to perform as well as their design with emitters above the finger.

The goal of this research is to make a design that can illuminate the finger from angles ranging between the horizontal axis, 20, 40, -20 and -40 degrees, then some commentary will be made about the result of each angle. And finally it will be compared with the previous setup that uses lights above the finger.

Research question

The research question is as follows,

Which angle gives the most uniform finger-vein illumination?

So in this research different angles will be experimented with and compared.

Materials and method

We want to find the best way of illuminating the veins inside the finger. The veins are located below the bone in the finger. There are some veins above the bone at the beginning of the finger which supply the extensor muscles, these muscles extend about halfway of the arm. But most of them are below the finger bone, supplying blood to the flexor muscles which contract the finger, these muscles can be seen in Figure 2, and the most veins are present where there are the most muscles. These are the veins that we will focus on. They extend into multiple smaller veins. Other research shows that these veins are unique in a way that they can be used for personal identification[4]

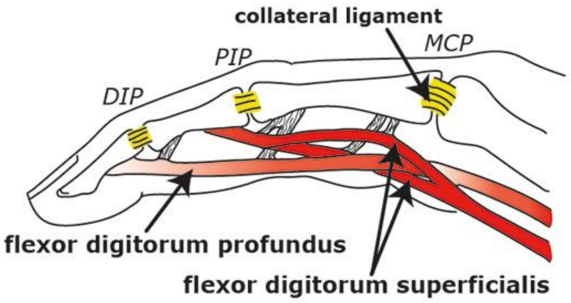

Figure 2: finger anatomy[1]

From this image it can be seen that most veins are present from the area of the fingernail to the knuckle. The veins in the tip of the finger are so dense that they are not suitable for imaging, and are not included in images.

Design

First of all a drawing was made. To get the most accurate angle a cylinder was used with multiple holes for the emitters. Figure 3 shows the final design to scale.

This design will replace the top part of the old setup and the camera will stay in the same position. So the two parts are interchangeable.

The larger the diameter of the cylinder, the more emitters can be fitted. With emitters that are 5mm in diameter and a cylinder of 6cm in diameter 5 different angles can be measured. From the horizontal axis of the cylinder 0, +20, +40, -20 and -40 degrees. The opening below is 3 cm by 6 cm, which covers the area of any finger and is the same as the previous setup. The cylinder is 5mm thick, this makes the construction more robust. Because of all the holes that are relatively close together it is prone to breaking during drilling.

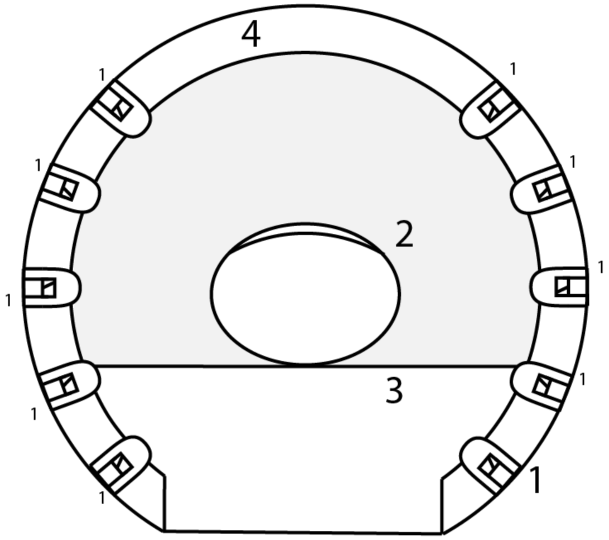

Figure 3 to scale cut section sketch of the final design

Legend of the parts in Figure 3:

1. LED sfh4550
2. Finger
3. Finger support, clear plastic
4. Casing, clear plastic

Construction

The design was realized in the workplace of the university. Mostly the milling machine was used to drill the shapes that were needed in the cylinder, see Figure 4. The emitters were soldered to wires and glued to the device, connected on the other end to the drivers. See Figure 5.

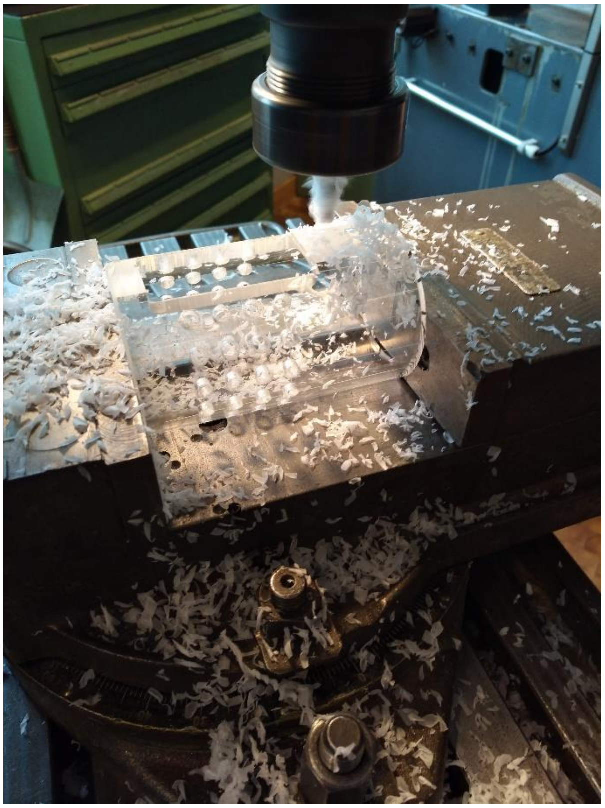

Figure 4 construction on the milling machine

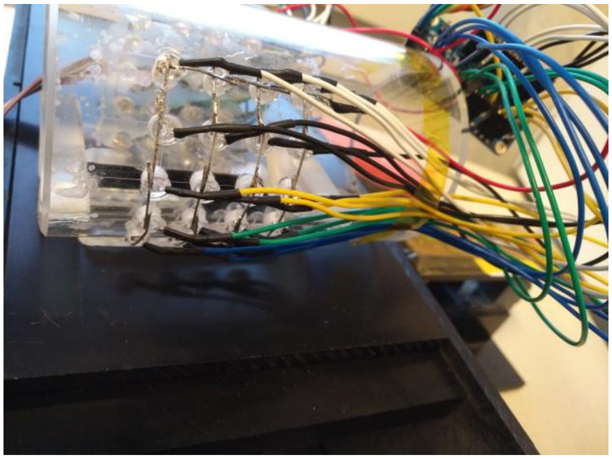

Figure 5 Side view of the finished design

Emitters

The infrared emitters that were chosen are the sfh4550. A total of 40 emitters were used. The wavelength of the light is 850nm. The reason these emitters were chosen is because of the wavelength, the light needs to be absorbed by blood in the veins, and blood mainly consists of hemoglobin, which has the highest absorption from 800-1000nm. The choice of 850nm is based on experimental data from previous research[3] to give the highest contrast. The diameter is 5mm.

It also has a comparably small (3 degrees) half angle which means that power is concentrated into the finger instead of radiated next to it which makes it more power efficient.

Power drivers

In total 40 emitters need to be powered, the previous setup could only power 8 emitters. Two 24 channel PWM drivers by Adafruit[5] were used to control all of them at the same time. Figure 6 shows the chip.

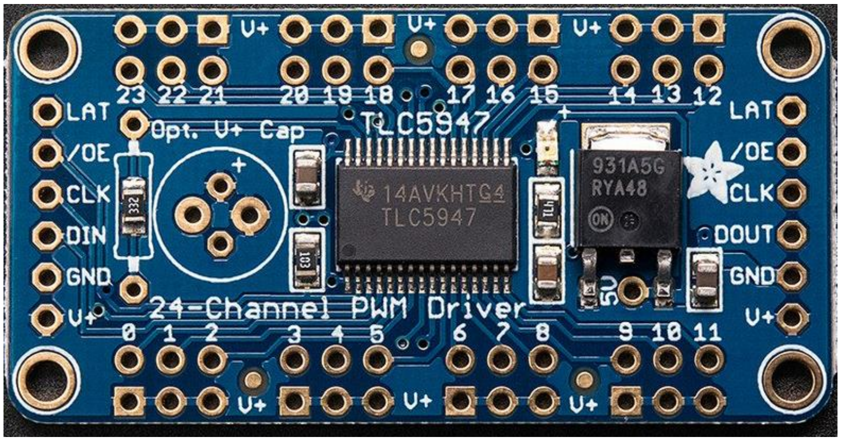

Figure 6 Adafruit 24-Channel 12-bit PWM LED driver with SPI interface

On the left side there are 6 input pins, on the right side there are 6 output pins. These are used to set the PWM value for each channel using the SPI, which will be explained later. Each device has 24 supply pins. Since the supply voltage is the same on each pin, only one wire for each driver is needed for this, all of the positive pin on the emitters are soldered together. There are 24 pins that control the PWM signal, these are connected to the negative pin of each emitter.

Figure 7 shows all these connections in one overview.

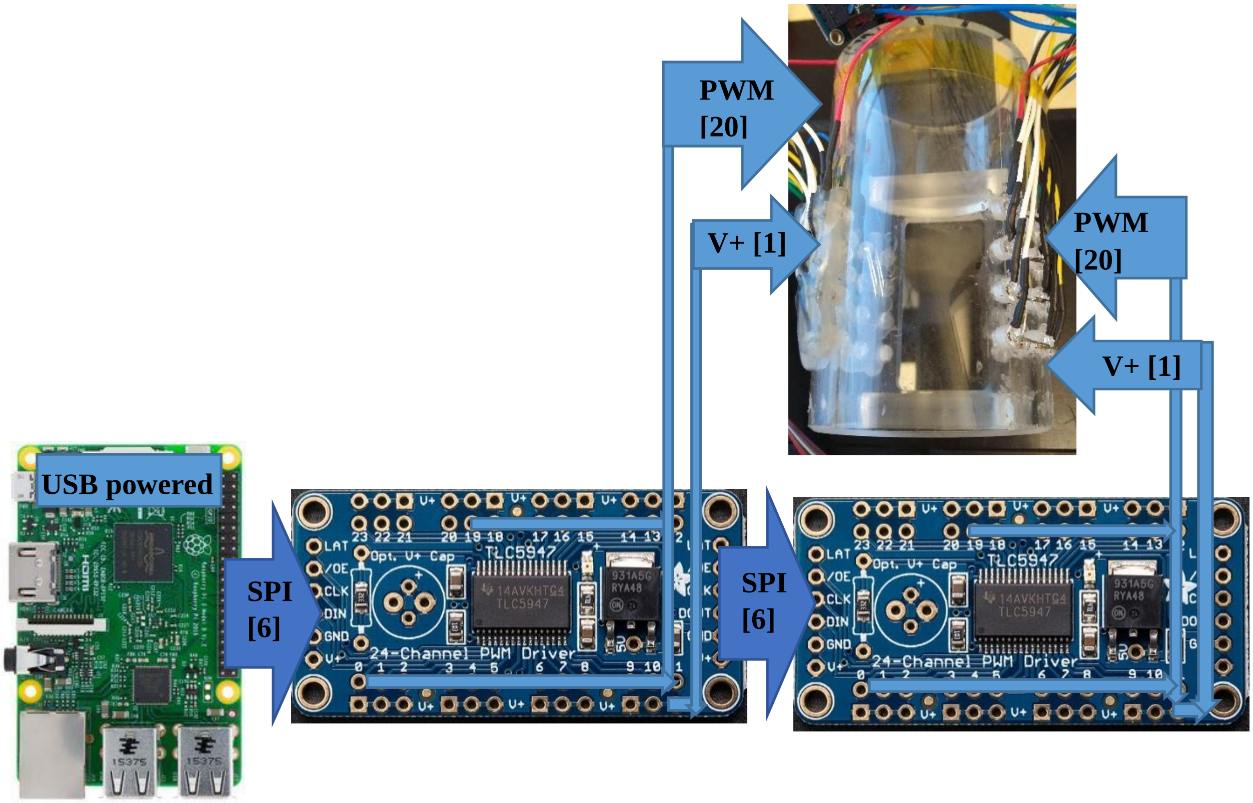

Figure 7 Overview of the connections that the power drivers have to the device and the controller

PWM outputs

See Table 1 for the channels number and each emitter that is connected to it. Driver 1 powers the left side, driver 2 powers the right side.

Driver 1 (left side)									
0: LED0	1: LED1	2: LED2	3: LED3	4: LED4	5: LED5	6: LED6	7: LED7	8: LED8	9: LED9
10: LED10	11: LED11	12: LED12	13: LED13	14: LED14	15: LED15	16: LED16	17: LED17	18: LED18	19: LED19
20: -	21: -	22: -	23: -						
Driver 2 (right side)									
24: LED20	25: LED21	26: LED22	27: LED23	28: LED24	29: LED25	30: LED26	31: LED27	32: LED28	33: LED29
34: LED30	35: LED31	36: LED32	37: LED33	38: LED34	39: LED35	40: LED36	41: LED37	42: LED38	43: LED39
44: -	45: -	46: -	47: -						

Table 1 Channel number and LED, colors indicate the row of LEDS, the thick bars indicate where the connectors separate

The wires are connected in pairs of 3. In case they are unplugged, they are marked with a red marker on the side of the connector, the red marker indicates the first emitter of the 3, so they can be connected in the right sequence. The wires have different colors, which shows the row that they correspond to. Blue is the bottom row, green the second, yellow the third, black the fourth, and white the top row. See Figure 5.

The output is a pulse width modulated signal, depending on the width of the pulses, more or less power goes to the emitter. The maximum power that is supplied is 30mA.

Because the register of each channel has a size of 12 bits, there are 2^{12}=4096 steps. So the current can be controlled in steps of 7.3μA.

SPI, communication with the driver

A controller is needed to communicate with the power drivers, the two drivers can be chained together to control both by a single interface. This is done by connecting the output pins of the first driver to the input pins of the second driver as shown in Figure 7.

The only requirement for the controller is that it has 4 controllable pins for the SPI communication, a voltage supply which can range from 5-30V[5], a ground pin, and an enable pin that is optional.

For this the Raspberry Pi model 1 was used. See Figure 8.

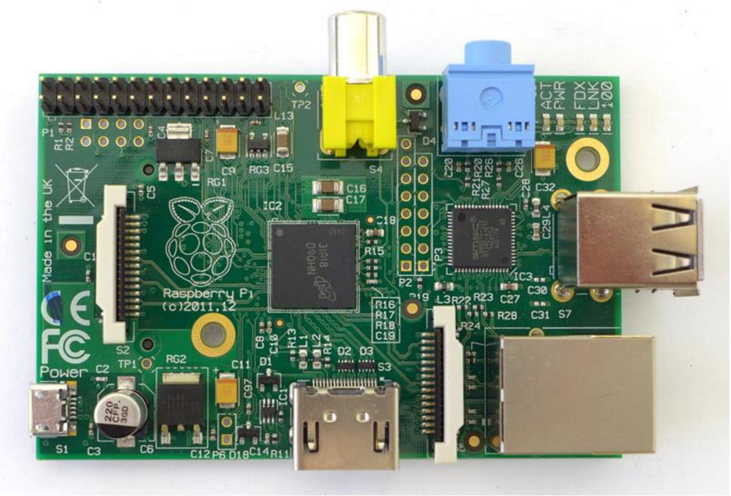

Figure 8 Raspberry Pi model 1[8]

The SPI (Serial Peripheral Interface) has many different types of applications[6]. For this report we will only focus on how it was used in the communication between the Raspberry Pi and the power drivers.

To illustrate what the function of each pin is, and how it is connected from the Raspberry Pi to the power driver, see Table 2.

Pin on the RPI	**Pin on the driver**	**Description**	**Function**
GPIO0	LAT	Latch	When this is high, the next set of data is ready. The old registers are emptied and set with the new data. Only then will the PWM output change.
GPIO7 (optional)	/OE	Optional Enable/Disable	When this is set high, all outputs are set to zero
GPIO3	CLK	Clock	The clock, a rising edge indicates that a next bit on the data pin is ready
GPIO2	DIN	Data	Bit value of the next bit
GND	GND	Ground	Connected to ground
V+ 5V	V+	Voltage supply	Supplies power to the chip

Table 2 Pins on the Pi and the power driver with their descriptions

A communication, where the data from a buffer is written to the device, would look as follows:

1. Clock is set LOW
2. Next bit in buffer is written to the data pin
3. Clock is set HIGH
4. Go back to 1 until the buffer is empty
5. Clock is set LOW
6. Latch is set HIGH
7. Latch is set LOW

The registers for each channel are 12 bit in size, meaning that the total buffer in our application consists of 12 bits * 48 channels = 72 bytes.

C program

To control the GPIO pins of the Raspberry Pi a program can be written in C or Python. For this project C was chosen. The reason for this is that there was already code available from Adafruit written in C that could be used as an example[5].

To make use of the GPIO pins from a C program, the WiringPi libraries need to be imported. The pin can then be set high or low by using the *digitalWrite* function with the pin number and either HIGH or LOW as arguments. To see which GPIO pin on the Pi corresponds to the number that the WiringPi library uses, see the WiringPi website page on pins[9].

The program creates a buffer by using the memory allocation function. The buffer is an array made of 48 (the total amount of channels) 16 bit variables. 16 bit variables are used while a register for the PWM value is only 12 bits. This is because there are no 12 bit variables in C. When the buffer is written, the 4 empty bits are skipped.

This buffer is filled by using the *setPWM* function, each channel has a value of 0 - 4095. The buffer can then be written to the driver by using the *write* function, the write function shifts through the bits in the buffer (after 12 bits it moves to the next variable skipping the 4 empty bits). Also the clock, data and latch pins are set as described earlier.

To compile the program, the following command was used (the C file in this case is called SPI.c):

```
gcc -Wall -o SPI SPI.c –lwiringPi –std=c++11
```

Arguments can be supplied to select emitters in the following order:

- Strength (integer): The power that is supplied to the emitter, ranging from 0 – 4095
  Default: 500
- Row (integer): The row of emitters that is powered, ranging from 0-4 (bottom to top)
  Default: 0
- Left side (Boolean): if the left side should be powered
  Default: 1
- Right side (Boolean): if the right side should be powered
  Default: 1

The program can be run by executing it using for example the command:

```
sudo ./SPI -2000
```

Individual channels can be adjusted by changing their PWM value.

The full code with comments can be found in the appendix.

Camera

The camera used is the BCi5 CMOS Camera by C-Cam technologies. The camera is capable of capturing images within the infrared band. The spectral sensitivity ranges from 400 nm to 1000 nm. The camera has low SNR and good contrast performance making it suitable for finger-vein imaging, see the datasheet of the camera and sensor[11] for the specifications. There is a mirror between the camera and the finger to make the device less high. The camera needs to be focused correctly by hand before using it.

Digital processing

To make the veins more visible in the image some processing can be done, an algorithm that can effectively enhance the contrast is AHE, adaptive histogram equalization which can be run from MatLab.

One downside of AHE is that it creates noise in homogeneous areas.

AHE was applied to the finger vein images that were captured.

It was done in MatLab with the following commands:

```
>> I = adapthisteq(rgb2gray(imread('Capture.PNG' )));
>> imwrite(I, 'Capture AHE.PNG' )
```

Results

Multiple images were taken of the same finger, varying the angles, 0, +20 +40 -20 -40 degrees at strengths of 10%, 25%, 50%, 75% and 100%. Then AHE is applied to enhance the contrast. Below the most interesting results are discussed.

Figure 9 shows the finger imaged with the previous setup. This can be used for comparison

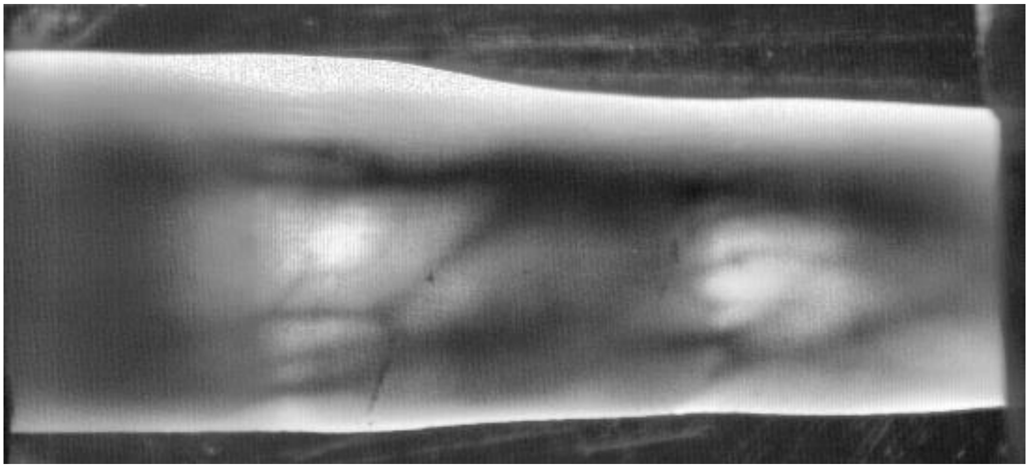

Figure 9 Vein image from the previous setup with lighting from above

The bottom row (-40 degrees) shows the most reflection, even when the power is very low. Figure 10 shows 10% power. No veins are visible here, only skin reflection.

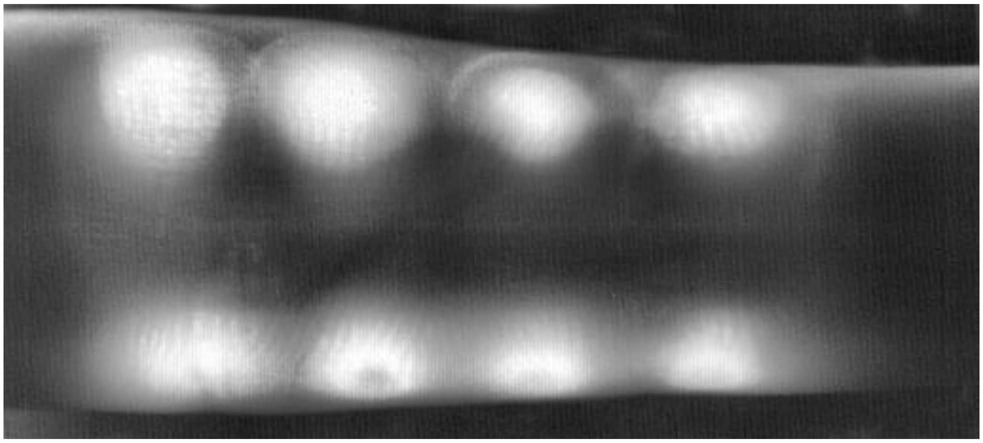

Figure 10 bottom row at 10% power

At -20 degrees veins begin to be visible, but there is some reflection, see Figure 11.

The light at the side of the finger is a homogeneous area of white light, which indicates that the camera is saturated. Higher power will make the saturation of the sensor spread across the whole finger.

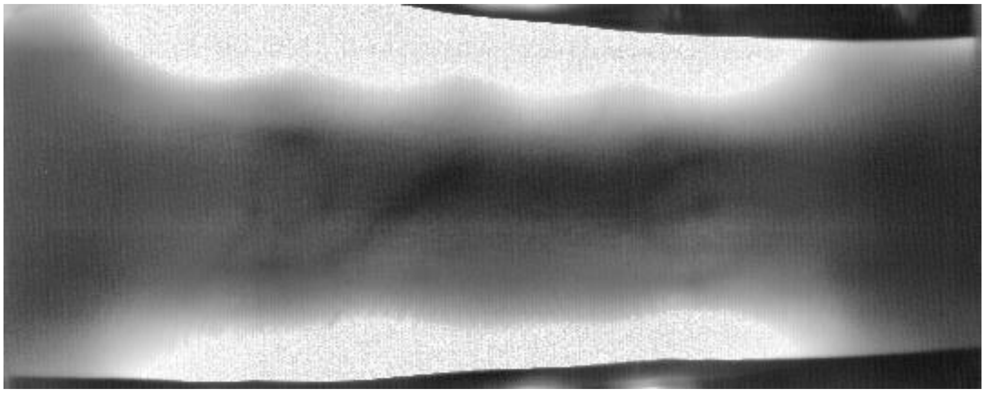

Figure 11 -20 degrees at 50% power

At a horizontal angle of 0 degrees veins are clearly visible, and saturation becomes less of an issue therefore the power can be higher, which is seen at Figure 12.

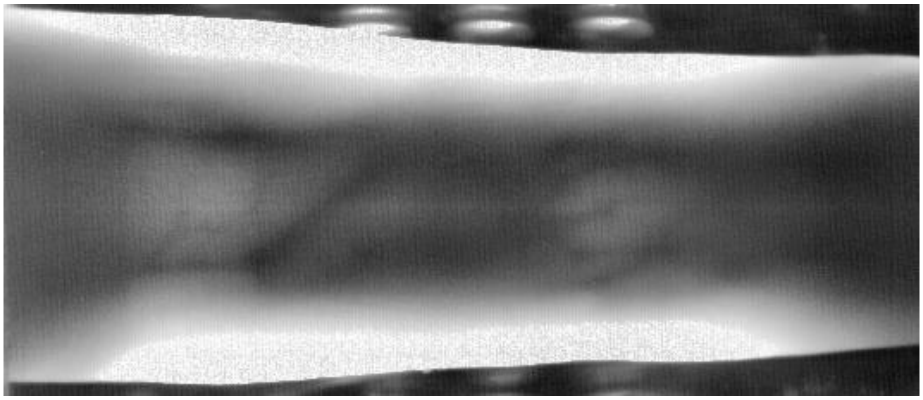

Figure 12 0 degrees at 100% power

Figure 13 shows the emitters at 20 degrees at 100% power, which out of the 5 angles gives the best visibility of the veins, there is still some saturation but it does not interfere with the veins significantly.

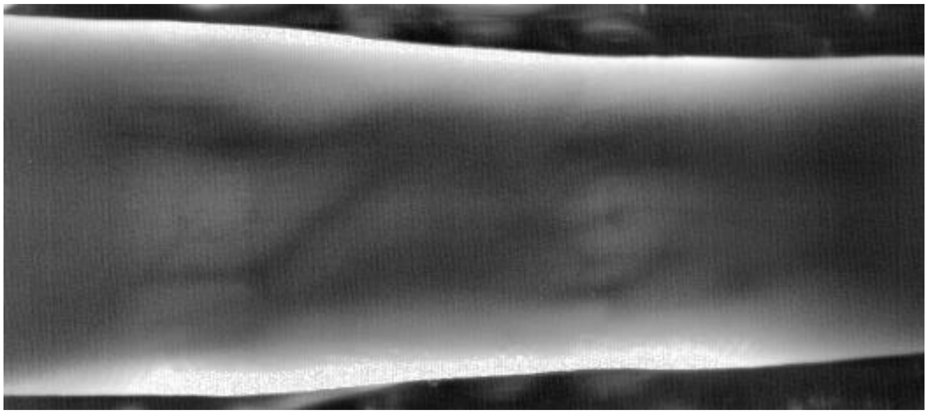

Figure 13 +20 degrees at 100% power

A 40 degree angle is almost comparable to lighting from above since there is almost no saturation. See Figure 14. The amount of veins are less than the previous angle, the image is less bright. Indicating that there is more light needed to illuminate the veins at a higher angle.

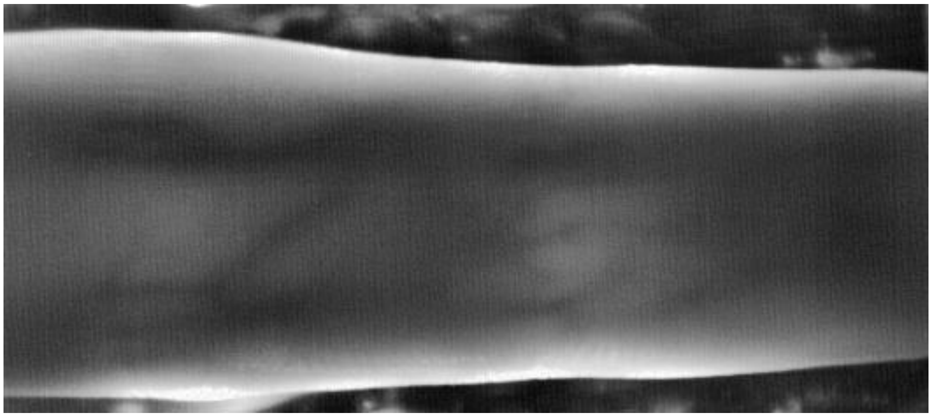

Figure 14 +40 degrees at 100% power

At the edge of the finger clearly there is a lot of reflection. Light were turned on from one side, and a piece of cardboard was held under the edge of the finger, see Figure 15, where the cardboard is at the bottom of the image. This shows that the light at the edges is not a reflection from the skin.

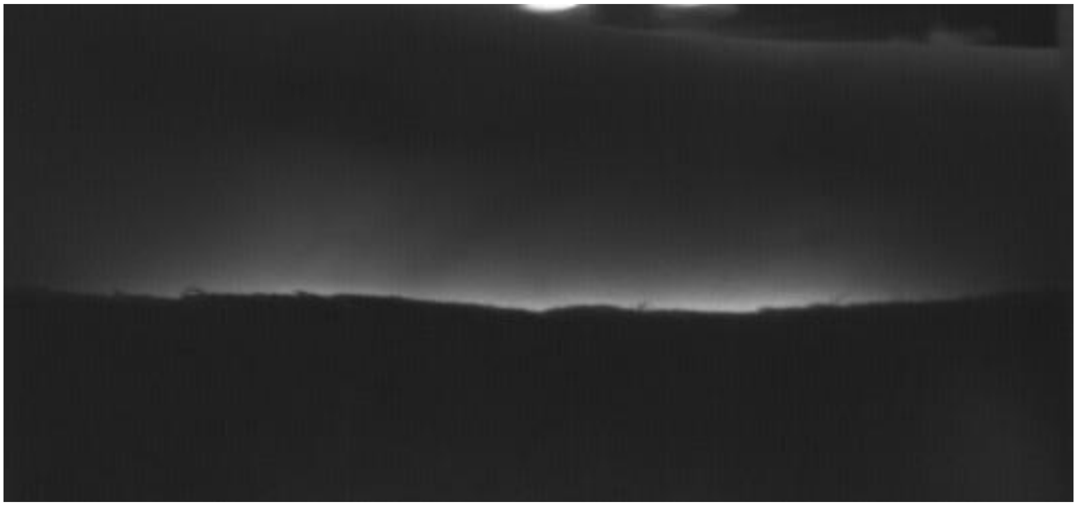

Figure 15 Illumination from one side with a cardboard at the bottom

In an attempt to get rid of the saturation of the camera at the edges of the finger, first one image was made with only the right side of lights, then another image was made with only the left side of lights. This was done on the horizontal angle of lights, with 100% power. Then these two images were merged. See Figure 18 for the result. The places where the saturation is removed do not have much added value because the light could not reach that far.

The merging was done by taking one side, cut out the saturated part, and then overlay it above the image from the other side. Then this was also done for the other side. See Figure 16 and Figure 17.

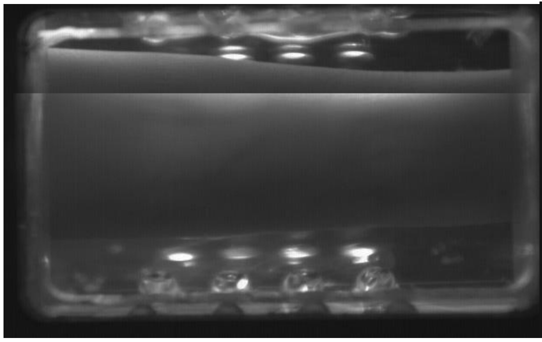

Figure 16 left side with saturation removed

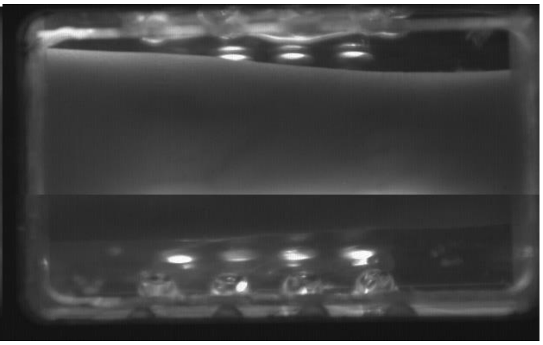

Figure 17 right side with saturation removed

These two resulting images were merged with MatLab,

```
>> merged = imfuse(imread('left side.png'),imread('right
side.png'),'blend','Scaling','joint');
>> mergedAHE = adapthisteq(merged);
>> imwrite(mergedAHE, 'mergedAHE.PNG')
```

It would have been better if instead of the cut in the image, a gradient was used to compensate the decay of brightness further away from the lights. Because now you can clearly see the edges were the image was merged giving a less uniform image.

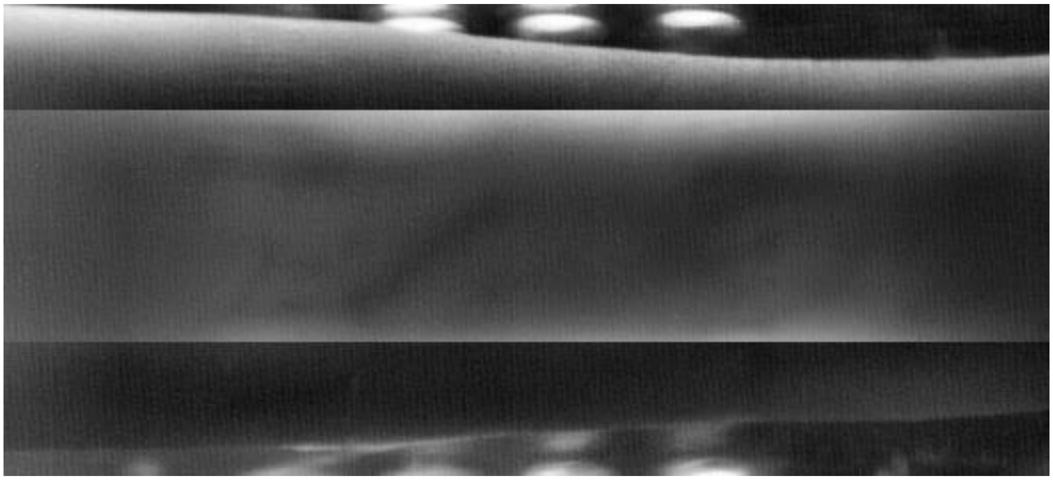

Figure 18 combined vein images from both sides

Conclusion

The problem with side lighting is the reflection of the skin and saturation of the camera from light on the side of the finger.

At the angles -40 and -20 degrees it is the reflection of the skin is a problem. AHE will also affect the skin, which makes it even more sensitive to the extra skin reflection

At the angles -20, 0 and 20 degrees the saturation of the sensor is a problem.

The higher the angle becomes, the more power is needed to illuminate the finger.

Vein visibility is comparable to an image from above, making an open design possible.

Research question

Which angle gives the most uniform finger-vein illumination?

An angle of 20 degrees showed the most uniform vein image.

Further work

It would be better to have a quantitative way of analysing the images. By using parts of the detection algorithms a quantitative measure for the amount of vein information that is visible could be calculated.

Noise middling could have been used to reduce noise, by taking multiple images and combining them.

The camera range could be changed, changing the saturation threshold of the camera.

A higher current PWM driver could be used. The emitters can take up to 100mA while the current chip only supplies 30mA, the TLC5947 could be replaced by a chip with a higher output current. If more than 100mA is needed the emitters will have to be changed.

Different angles could be combined to possibly get a more uniform image.

More work can be done on the digital image processing to remove the saturation on the sides.

It was suggested to use fibre optics to make a more directed beam of light, which might give more uniform images.

The skin reflection could be removed by making an image in the visible spectrum and then subtracting it from the infrared image, correcting the reflection, there is other research that shows this work with veins in the arm[2].

Finally it could also be tested on more people to have a larger collection of images. Properties like thicker skin, vein size and finger shape all influence the results.

References

1. Finger anatomy: http://www.startradiology.com/internships/emergency-medicine/hand-fingers/x-hand-fingers/
2. High-contrast subcutaneous vein detection and localization using multispectral imaging. Wang F1, Behrooz A, Morris M, Adibi A.
3. VASCULAR PATTERN OF THE FINGER: BIOMETRICOF THE FUTURE? SENSOR DESIGN, DATA COLLECTION AND PERFORMANCE VERIFICATION, B. Ton
4. T. Yanagawa, S. Aoki, and T. Ohyama. “Human finger vein images are diverse and its patterns are useful for personal identification”. In: MHF Preprint Series (2007). doi: 10.1.1.97.4421.
5. Adafruit 24-Channel 12-bit PWM LED Driver - SPI Interface - TLC5947 https://www.adafruit.com/product/1429
6. Serial Peripheral Interface Bus, Wikipedia, https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
7. Introducing Hitachi’s Finger Vein Technology, http://www.hitachi.eu/veinid/documents/veinidwhitepaper.pdf
8. Raspberry Pi model 1 Image, https://www.mbtechworks.com/wp-content/uploads/2015/05/RP-1-B-800x.jpg
9. WiringPi GPIO pins, http://wiringpi.com/wp-content/uploads/2013/03/gpio1.png
10. Histogram equalization, Wikipedia, https://en.wikipedia.org/wiki/Histogram_equalization
11. BCi5 CMOS camera datasheet, http://www.alrad.co.uk/datasheets/Imaging/BCi5.pdf

Appendix

C code for the Raspberry Pi

```
#include <wiringPi.h>
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>

uint16_t *pwmbuffer;

uint16_t numdrivers = 2;
uint8_t _clk, _dat, _lat;
6 e45
void write(void) {
  digitalWrite(_lat, LOW);
  // 24 channels per TLC5974
  for (int16_t c=24*numdrivers - 1; c >= 0 ; c--) {
    // 12 bits per channel, send MSB first
    for (int8_t b=11; b>=0; b--) {
      digitalWrite(_clk, LOW);

      if (pwmbuffer[c] & (1 << b))
        digitalWrite(_dat, HIGH);
      else
        digitalWrite(_dat, LOW);

      digitalWrite(_clk, HIGH);
    }
  }
  digitalWrite(_clk, LOW);

  digitalWrite(_lat, HIGH);
  digitalWrite(_lat, LOW);
}



void setPWM(uint16_t chan, uint16_t pwm) {
  if (pwm > 4095) pwm = 4095;
  if (chan > 24*numdrivers) return;
  pwmbuffer[chan] = pwm;
}
int main (int argc, char *argv[])
{
    //setup GPIO pins
    wiringPiSetup () ;

    //create the buffer
  _clk = 3;
  _dat = 2;
  _lat = 0;

  pwmbuffer = (uint16_t *)malloc(2 * 24*numdrivers);
  memset(pwmbuffer, 0, 2*24*numdrivers);
  pinMode(_clk, OUTPUT);
  pinMode(_dat, OUTPUT);
  pinMode(_lat, OUTPUT);
  digitalWrite(_lat, LOW);

    printf ("Starting\n") ;

    int strength = 500; //0-4095
    int row = 0; //0-4
    int leftside = 1;
    int rightside = 1;

    if(argc>1){
        strength = atoi(argv[1]);
        printf("Strength: %d\n", strength);
    }
    if(argc>2){
        row = atoi(argv[2]);
        printf("Row: %d\n", row);
    }
    if(argc>4){
        leftside = atoi(argv[3]);
        rightside = atoi(argv[4]);
        printf("Left side: %d, Right side: %d\n", leftside, rightside);
    }


    if(leftside==1){
        setPWM(row*4,strength);
        setPWM(row*4+1,strength);
        setPWM(row*4+2,strength);
        setPWM(row*4+3,strength);
    }
    if(rightside==1){
        setPWM(row*4+24,strength);
        setPWM(row*4+25,strength);
        setPWM(row*4+26,strength);
        setPWM(row*4+27,strength);
    }

    //Write the pwm buffer
    write();
    return 0 ;
}
```