Bachelor's project

Image-based quality evaluation of otoscopy images

Freja Rindel Peulicke, 185393

med vejledere
Rasmus Reinhold Paulsen
Josefine Vilsbøll Sundgaard
Søren Laugesen (ekstern)

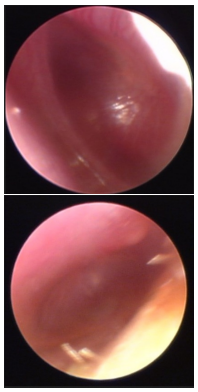
DTU Compute
Technical University of Denmark
October 23, 2021

Abstract

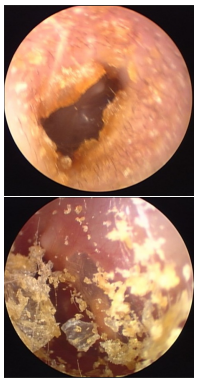
Contents

1 Introduction

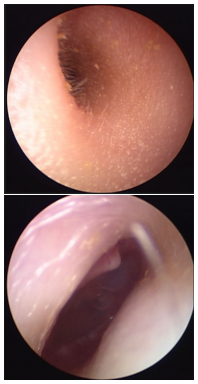
Correctly determining if a person has inflammation in the ear can be difficult. Using image analysis and machine learning on a good data set, the expectation is that it is possible to reduce the amount of antibiotics used on ear inflammations, as diagnosing correctly would be much easier.
Creating this data set requires sorting away pictures of poor quality. The images are taken with an otoscope, which is a device that is placed in the ear canal from the outer ear, and makes it possible to see the eardrum. Some possible complications with the otoscopy images are:



(a) The image is out of focus



(b) There is too much ear wax to see the membrane



(c) The membrane is not visible

Figure 1: Three primary challenges of filtering away otoscopy images of poor quality.

In this project, the aim is to construct an interactive program to help a doctor take good otoscopy images of the ear drum. A data set of 153 images is used to determine the be best possible methods for detecting the three challenges in figure 1.
The first challenge to be examined is detecting if the image is blurred 1a. Detection of blur is one of the most challenging problems to solve in the field of image processing[**FM**]. A selection of some promising algorithms from [**FM**] and [**JNB**] will be tested and evaluated to integrate the best performing one in the final program.
Next, different methods for determining if there is too much earwax 1b will be examined after which the last problem 1c will be examined.
After this, an evaluation of the design, implementation and performance of the final program will be reviewed.

1.1 Previous Work

1.1.1 Articles on no-reference sharpness-metrics

13 different no-reference algorithms for detecting image sharpness are outperformed by another algorithm using probability summation and JNB (just noticable blur) introduced in [**JNB**]. The algorithms are tested on two types of test-datasets: One containing the same image with different blur-values. Another with images of different content and blur-values. Results of the tests are, that the 13 algorithms do not perform very well for images with different content. The algorithms are tested and compared to one another by using the results of experiments made on humans detecting blurriness on the same images. The JNB-metric works by scanning the image with an edge detector, Sobel in this case. Afterwards the no. edge pixels, N, are calculated for each 64*64 block of the image, excluding smooth blocks - having $N <$ `threshold` - from further evaluation, as it is assessed that there must be some edges to measure blurriness on. On each block the contrast, edge widths and probabilities of blur distortion are calculated, and the overall distortion is calculated. The sharpness measure is the inverse distortion normalized over all the blocks.

A metric, Frequency Domain Image Blur Measure, with the purpose of quantifying the quality of blurred images is presented in[**FM**]. It is shown that this algorithm in certain cases outperforms the JNB metric[**JNB**][1] and the CPBD[**CPBD**][2] metric, that are presented as some of the best image sharpness metrics so far. The metric works by transforming the image into frequency domain, as it is observed that *when the blur in an image increases the number of high frequency component in the images decreases*[**FM**]. An experimentally estimated threshold is used to calculate the number of high frequency components in the image, and this is then used to calculate the final score. The time complexity of FM is calculated to be $O(nlogn)$, where $n =$ `no.pixels_in_image`, as the time complexity of the fast Fourier transformation is $O(nlogn)$ and all of the following steps takes $O(n)$ time.
The three metrics were tested on images containing Gaussian blur and motion blur. FM performed well in both tests, JNB performed well on Gaussian blur and CPBD performed well on motion blur. Applying FM on images where blur removal was applied, it agreed well with the blur before and after.

A proposed metric for detecting blurriness in a no-reference image, as there is an increasing interest. The blurriness can emerge when the image is first taken, during processing or compression. There exist other metrics including JNB[**JNB**] that correlates well with subjective scores on some images, though not images where the blur in foreground and background is very distinct.
The probability of blur detection is calculated on the edges as in JNB[**JNB**]. Also, the concept of just noticeable blur defined in JNB is the basis of the

[1] Just noticeable blur
[2] Cumulative probability of blur detection

CPBD metric, and the image is divided into smooth/non-smooth blocks as in JNB. After dividing into blocks the cumulative probability of blur detection is calculated as the probability of blur detection being below the just noticeable blur.
The metric is discretized into five different quality classes. For this, the MOS[3] and CPBD scores of a selected set of images were used. Tests were performed on Gaussian blurred and JPEG2000 compressed images. The CPBD metric performs better than the discrete CPBD metric in all the tests. They both perform better than the JNB metric and marziliano metric, that were tested too.

*Proposing sharpness metric for images with noise. No need to estimate or remove noise before evaluating sharpness. Using Perceptual blur metric[***jnbm03***]. Same explanation of 9 of the metrics from [***JNB***]. "discrete dyadic wavelet transform (DDWT) of the image [...] lowpass filter [...] highpass"*

A method for autofocusing an electron microscope. Using variance over the whole image, the sharpness is measured. The method is considered fast, as previous methods calculated the Fourier transformation, which took long time. The proposed algorithm only calculates variance ...argumentation in article.[**jnbm05**]

*Thesis on autofocusing on scannings from electron microscope.[***jnbm06***] Including sections on developing, implementing and testing four different algorithms on detecting sharpness of which two are the autocorrelation-based metric and the derivative-based metric listed in table1.*

Presentation of full- and no-reference blur-metric and a full-reference ringing metric. The blur-metric presented finds vertical edges, then filters noise away using a threshold on the gradient image. The width of the edge is then counted and the average edge-width then determines the sharpness of the image. The metric can be extended to finding all horizontal edges as well.[**jnbm07**] In the article, the edge detection algorithm is the bottleneck, as they used a lightweight (sobel filter) algorithm. Using a more powerful edge-detection algorithm can improve the precision of the metric, but will reduce the speed.

*Autofocus. constructing different images to test on: sinusoidal, random noise and (true) blurred digitized images from a microscope.[***jnbm08***] analyzed 9 autofocus-methods.*

Cellular logic measures: Transforming grayscale image into 3d bit-voxel (opacity) images. using highpass-filter with a factor.

`Spectral analysis`: "Sharp edges and fine details correspond to high spatial frequencies, and large objects with slowly changing gray levels correspond to low spatial frequencies. Therefore, we can expect focused images to exhibit more power at high frequencies than at low frequencies."

[3]Mean opinion score: human judgement of overall quality of the image.

Power measures: "Power can be defined for images", "the AC power component increases as the image sharpens"

Variance: "AC power of an image is directly measured by computing the variance of the picturepoint values", "$F_{var} = EG^2 - (EG)^2$*"*

Brenner's methods: "as an image comes into focus, differences between a picturepoint and its neighbor two points away increase"

Histogram measures: histogram on grayscale-values. "For most in-focus images, the histogram contains occurrences in many bins". Defocus -> decreased no. bins in histogram with different grayscale and vice versa. However, this does not apply to all images.

Range: "difference between the maximum gray level and the minimum gray level". Maximum and minimum gray level present in image. It is assumed that the range will increase as the image comes into focus.

***Mendelsohn and Mayall's histogram method** (Histogram threshold): Wheighed sum of all picture points in histogram bins above a threshold chosen to be near the mean graylevel value. It is computed as the sum of all graylevels multiplied with the current graylevel's amount of occurrences.*

Mason and Green's histogram method: Same as above except for the way that the threshold is chosen: "weighed the importance of picturepoints by estimates of the gradient at that point".

Histogram measure entropy: " For an in-focus image, the information content is usually higher because the probability of occurrence of each gray level is low", "The entropy function is a measure of information content", "entropy has fundamental problems when used to measure the relative focus of those images that fill few bins when in focus"

The two best measures were the the power measures. The worst ones were using the grayscale histogram.

[**jnbm09**]
[**jnbm10**]

Attempt to formulate a robust autofocusing algorithm.[**jnbm11**] Comparing eight different sharpness metrics. They are all described shortly and it is explained that they are more or less similar in pairs: amplitude and variance, Tenengrad and Laplacian, fast Fourier transform and sum-modulus-difference, histogram entropy and histogram of local variations. Also some searching algorithms for finding the global maximum of the sharpness function are evaluated. The variance method and the Tenengrad are recommended as sharpness functions in the conclusion.

Histogram frequency-based metric for detecting blur in a JPEG or MPEG image.[**jnbm12**] A method for detecting blur in an image and in videos is proposed and tested. Images in JPEG and MPEG file format are compressed. The compression-calculations of the DCT[4] are exploited. "based on histograms of non-zero DCT occurrences". To ignore noise, a threshold is set when creating

[4]discrete cosine transform

the DCT-histogram. All values below the threshold are set to 0. "The idea of the blur estimation algorithm is then to examine the number of coefficients that are (almost) always zero in the image, i.e. to count the number of zeroes (or nearly zero values) in the histograms". A weighting-grid is applied to make the final quality measure. (Is lightweight[**jnbm12**]: instead of calculating over the whole image (256*256 pixels) it is calculated on the equally divided 8*8 blocks of the image)

*A method for measuring image sharpness based on the wish of developing a method for sharpening the image is proposed in [***jnbm13***]. "employs localized frequency content analysis in a feature-based context". Fourier transform[***jnbm13***]. Feature based - require well-defined model e.g. single star in astronomical image. Highpass and bandpass filters are applied in 1D to make the implementation more light-weight, thus unfortunately the diagonal edges appear more blurry than horizontal and vertical edges. Also, it is determined, that small parts of the feature regions are representative for the whole feature. "implement sharpness estimation WFmSh(11), with HP(x)m, and BP(x)m (8) implemented as IIR filters (9), and substituting (7) into (6)"*

*"based on the digital image power spectrum of normally acquired arbitrary scenes"[***jnbm14***]*

Detection of gradients in image. Using canny edge detection to find the edges. Calculating width of edge by for all edge pixels calculating in both directions (+ and -) of the gradient the amount of pixels going respectively up or down (following the gradient). The average edge-spread is used in the proposed image quality score together with some parameters, determined by training on a dataset.[**jnbm17**]

In 1991 the **JPEG method** for digital image compression of still continuous-tone images was proposed in a paper [**jpg**] in hope that a standard compression form would enable better performance in compressing and storing images digitally, as it would be possible to develop hardware for the common standard, and lower the cost of this specialized hardware.
The JPEG coding of the image is based on the DCT, that is applied to 8×8 blocks of the image. The DCT does not provide any compression of the image data. After the application of the DCT, each of the 64 DCT-values are quantized by dividing with a corresponding Q-Table (quantization-table) element. The 64-element Q-Table is given to the encoder as input, and should be designed with great care, as the quantization decreases the amount of information in the image. In the final compression step, entropy coding, there is no information loss. Based on the statistical characteristics of the image its data is encoded more compactly[5].
The decompression of JPEG images is also described. The article later describes

[5]This may be done using Huffman coding or arithmetic coding.

typical applications of JPEG to examine how it can be used, what its potential is and the work it requires to implement its use into society.

2 Otoscopic Images

2.1 Quality problems

3 Blurred images

Detecting sharpness/blurriness of an image is a very central challenge in image quality assessment (IQA). Three different types of IQA-methods can be classified as:

- Full reference
  In this type of IQA, a reference image of perfect quality is given. This image will be distorted to varying degrees, and the goal is then to provide an algorithm, that can detect the degree of distortion.
- Reduced-reference
  Here, a distorted version of the reference image is given together with a few informations on the perfect quality image. These informations can vary depending on the algorithm.
- No-reference
  No reference image is given. Only an image of unknown quality. Again, the challenge is to provide an algorithm for determining the degree of distortion.

Detecting blur in this project, will be the no-reference challenge, as the final metric will be used live for detecting sharpness in the original image. This is considered the hardest problem of all three, and a perfect algorithm doesn't exist today.

3.1 Description of algorithms

In [**JNB**] and [**FM**], a total of 16 different no-reference sharpness-metrics are mentioned. As a full test and implementation of all of them will be quite comprehensive, a few of them will be selected for testing. In table 1 the metrics are listed in the first column. The parameters that the metrics are measured on are listed in the top row. They represent different methods used in construction of the algorithms and situations they might perform well in. The parameters can be described as follows:

Probability of blur-detection: Dividing the image into blocks. Using a formula on each block to determine the probability of the block being blurred...
Edge detection: Using an algorithm - e.g. Canny or Sobel edge detector - for

finding the edges in the image. The sharpness of the edges is often measured to estimate the sharpness of the image. laplacian second order???
Training data: The metric contains constants, that are determined using a dataset.
Different contents: The algorithm performed well in tests on determining sharpness of images containing different content.
Fourier Transformation: The frequency domain of the image is used in determining its sharpness. Transforming an image into its frequency domain is quite computationally hard.
Grayscale gradient: In the image, the gradients of the are calculated. These can be helpful in calculating edge thickness, and determining if there are sharp differences in the image. If there is much noise, using the gradient can unfortunately result in an estimation of an image being sharp even though the real motif isn't.
Grayscale histogram: The normalized or un-normalized probability density function of the grayscale values of the image. As a completely blurred photo would approach a single gray color, and thus lose information, it is generally assumed, that sharper images tend to have more different grayscale values and not very many occurrences of each value, whereas more blurred images tend to have more middle grayscale values, and more occurrences of each value.
However, this is not always the case. E.g. one could easily have a sharp image with very dark and light gray values and almost none of the middle gray values. As this assumption affects the accuracy of the sharpness-measure, one could assume, that this is the reason for the histogram-based metrics to be less popular.
Grayscale variance: Calculated as the variance over the whole grayscale image. As the image becomes more blurred, the grayscale values will become more similar, thus decreasing the variance.
Noise robust: The correctness of the sharpness measure generally is not affected by noise.

For all of the algorithms, it is noted whether they perform good in the situation or use the concerned method.

Sharpness metric	Probability of blur-detection	Edge detection	Training data	Different contents	Fourier Transformation	Grayscale gradient	Grayscale histogram	Grayscale variance	Noise robust
Variance[**jnbm05**] 1982	no	no	no	no	no	no	no	**yes**	**yes**
Auto-correlation-based[**jnbm06**], 2000	no	no	no	no	**yes**	no	no	no	ok

Derivative-based[**jnbm06**], 2000 laplacian	no	no	no	no	no	**yes**	no	no	no
Perceptual blur[**jnbm07**], 2004	no	**yes**	no	no	no	no	no	no	ok
Frequency threshold[**jnbm08**]	no		no	no					
Kurtosis[**jnbm09**][**jnbm10**], 2004**	no	**yes**	no	no	**yes**	no	no	no	no/ok
Histogram threshold[**jnbm08**], 1991	no	no	no	no	no	no	**yes**	no	ok
Histogram entropy-based[**jnbm11**], 2001	no	no	no	no	no	no	**yes**	no	ok
Histogram frequency-based[**jnbm12**], 1999	no	no	no	no	(**yes**[6])	no	no	no	ok
Shaked-Tastl[**jnbm13**], 2005	no	**yes**	no	ok	no[7]	**yes**	no	(no)	ok
Image Quality Measure[**jnbm14**], 1992	no	no	no	ok	**yes**	no	no	no	ok
Noise Immune Sharpness[**jnbm03**][**jnbm07**], 2005	no	**yes**	no	no	no	no	no	no	**yes**
No-reference Blur[**jnbm17**], 2003	no	**yes**	**yes**	no	no	no	no	no	no
Just Noticeable Blur[**JNB**], 2009	**yes**	**yes**	no	**yes**	no	no	no	no	ok

[6] The images need to be in JPEG or MPEG compressed format. That is, the DCT (Fourier-related transform) has to be calculated before applying the algorithm.

[7] *we use spatial filters rather than a full 2D Fourier transform*[**jnbm13**]

CPBD[8][**CPBD**], 2010	**yes**	**yes**	no[9]	**yes**	no	no	no	no	ok
Frequency Domain[**FM**], 2013	no	no	no	**yes**	**yes**	no	no	no	ok

Table 1: Different no-reference metrics for detecting blur in an image, what methods they use and what situations they perform well in.

The following sharpness-metrics will be gone forward with: Variance, Derivative-based, Histogram frequency-based, No-reference blur and Frequency domain. The first one, variance, is a very simple metric, that will be included to create a good baseline to compare performance and accuracy of the other metrics included. The derivative-based and frequency domain are some classical methods[] and are therefore included. The no-reference blur-metric is included as it uses a training dataset, and as the images are all of ear canals, it would be very interesting to see if this metric has better accuracy. The last one, histogram frequency-based metric, is included, as it is developed to function well on MPEG (movie) images, and therefore might evaluate the images very quickly without compromising on accuracy.
These metrics will be tested using a few different test-methods: sensitivity/specificity. Accuracy, f-score. Area under the roc...
Sklearn[10] has implementations of these tests, that will be used in the project.
When testing, it will be assessed whether a few of the metrics should be used together in the final program, as they might perform better that way.

3.1.1 Analysis of running time

3.1.2 Cumulative probability of blur detection

The used implementation of the CPBD metric[**code˙CPBD**] is based on the concept of JNB defined in [**JBN**]. JNB is explained: *The "just noticeable blur" (JNB) is the minimum amount of perceived blurriness around an edge given a contrast higher than the JND*[**JNB**], where JND is the just noticeable difference *the JND is the minimum amount by which a stimulus intensity must be changed relative to a background intensity in order to produce a noticeable variation in sensory experience*[**JNB**]. The JNB have been experimentally determined in relation to the contrast in an image.

The metric starts out by splitting the input image into blocks of $64 \cdot 64$

[8] Cumulative probability of blur detection

[9] Training data is used to determine the discrete quality classes, but not the continuous scores of the metric.

[10] `https://scikit-learn.org/stable/modules/model_evaluation.html`

pixels. Canny edge detection by *sci-kit*[11] is used on the whole image to classify the blocks as edge or non-edge blocks.

The canny edge detector[12] takes as input the output of the **sobel edge detector**, which works by applying a 2-dimensional Gaussian filter with $\sigma = 1$ to the image to smooth out noise[13]. The Gaussian filter is applied first in one direction and afterwards in the other[14]. This takes $O(w_g \cdot N \cdot M + h_g \cdot N \cdot M)$[15], where w_g and h_g are respectively the width and height of the Gaussian filter. After this, the horizontal and vertical sobel filter

$$sobel_h = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$sobel_v = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

are applied to all the pixels in the image, that is $O(N \cdot M)$. Now the canny edge detector can thin out the edges to be 1 pixel wide. This requires going through all found edges, which in the worst case is $O(N \cdot M)$ edge pixels. To thin the edges, each edge point will be traversed a constant number of times, c: first to sort the edges into horizontal, vertical and diagonal edges, and then compare their values to the reversed and normals to see if they are greater in those directions. The points for the edges are picked using interpolation. All of this takes $O(c \cdot N \cdot M) = O(N \cdot M)$ time.

Lastly in the canny edge detector, a hysteresis thresholding is performed, where all the edge pixels below a low threshold are discarded, and all the edge pixels above a high threshold are included. Edge pixels that are 8-connected to the high threshold edges, are included in the final edges as well to store as much relevant information as possible. This can be implemented in $O(N \cdot MlogN \cdot M)$ time according to [16]. Thus, the total running time of canny will be $O(N \cdot MlogN \cdot M)$.

In the article[**CPBD**], now would be the time where the edge pixels were counted, and the smooth blocks were discarded. However, the implementation[**code˙CPBD**] will at this time **compute the edge widths**, $w(e_i)$, instead. This is done using the sobel edge detector[**code˙CPBD**] described earlier, as the edges in the

[11] `https://scikit-image.org/docs/dev/api/skimage.feature.html#canny`, accessed October 5, 2021

[12] `https://scikit-image.org/docs/dev/api/skimage.feature.html`, accessed October 4, 2021

[13] `https://scikit-image.org/docs/dev/api/skimage.filters.html?highlight=gaussian%20filter#skimage.filters.gaussian`, accessed October 4, 2021

[14] `https://github.com/scipy/scipy/blob/9492369e7f76c8206ac1c375e75375ca7410a7e7/scipy/ndimage/filters.py`, accessed October 7, 2021

[15] depends on 1D+1D or 2D. It is 1D+1D

[16] `https://stackoverflow.com/questions/17458237/time-complexity-of-canny-edge-detector#:~:text=The%20final%20step%20works%20by,O(mn%20log%20mn)`, accessed October 5, 2021. examine this closer

canny edge detector have been reduced to a width of 1 pixel. In the implementation, all $w(e_i)$ are calculated in $O(N \cdot M \cdot 101) = O(N \cdot M)$. This includes calculating the gradients of the image, $O(N \cdot M)$. The angles of the edges calculated in $O(N \cdot M)$. The angles are used for calculating edge widths in $O(N \cdot M \cdot 3 \cdot 101) = O(N \cdot M)$.

Next step in the implementation is to loop over all the $\frac{N \cdot M}{64}$ blocks in the image. Here, the edge pixels are counted, and a threshold of $t = 0.2\%$ is now used to determine if the block is an edge-block. That is, if the number of edge pixels in the image detected by the canny edge detector is above 0.2% of all the pixels in the block, the block is considered an edge block, and otherwise as a smooth block. The counting requires looking at every pixel in the block, that is $block\_size = 64^2$.

If the block is considered an edge block, the **JNB edge width**, $w_{JNB}(e_i)$, is determined by calculating the block contrast $C = max(grayvalue) - min(grayvalue)$ and from this determining the JNB as in [**JNB**] from the experiments mentioned earlier. Finding the grayscale values for the contrast takes $O(64^2)$ time.

Now, the **probability of blur detection** for each edge e_i can be computed. This is done using the following equation

$$P_{BLUR} = P(e_i) = 1 - e^{-|\frac{w(e_i)}{w_{JNB}(e_i)}|^{\beta}}$$

mentioned in [**CPBD**] and explained in [**JNB**], where $\beta = 3.6$.

The probability distribution function of P_{BLUR} is updated in $O(64^2)$ time, as the last step in the loop on the blocks.

The final step of the metric is to **calculate the CPBD**. The equation from [**CPBD**]

$$CPBD = P(P_{BLUR} \leq P_{JNB}) = \sum_{P_{BLUR}=0}^{P_{BLUR}=P_{JNB}} P(P_{BLUR})$$

is used, where $P(P_{BLUR})$ is the value of the pdf of a given P_{BLUR}. Only the first 64% of the pdf are summed, as *It should be noted that when* $w(e_i) = w_{JNB}(e_i)$ *then* $P_{BLUR} = 63\% = P_{JNB}$.[**CPBD**]

This will provide a total time complexity of

$$O(canny + sobel + |blocks| \cdot (count\_edge\_pixels + contrast + update\_pdf) + sum)$$

$$= O((N \cdot MlogN \cdot M) + (N \cdot M) + \frac{N \cdot M}{64} \cdot (3 \cdot 64^2) + 64)$$

$$= O(N \cdot MlogN \cdot M)$$

3.1.3 Histogram frequency-based metric

The Histogram frequency-based metric[**jnbm12**] bases its solution on images and videos in the file format of JPEG and MPEG, as it is estimated that the

compression of the images can be exploited for fast and effective blur detection. The implementation used[**code˙HF**] is implemented in `C`.

The JPEG format compresses images[**jpg**] by i.a. applying the DCT[17] to 8×8 blocks of the image.

DCT: should the formula be included?

The output of the DCT is a matrix, where the top left cell is called the DC[18], as it does not depend on the index, as $x = 0$ and $y = 0$. The remaining 63 cells are the AC[19] cells. Precisely this DCT computation is what is exploited in the metric, as *The DCT coefficients used within MPEG are intended for compression and are deeply related to the image content. Basically, they reflect the frequency distribution of an image block*[**jnbm12**]. It is also known from i.a. [**FM**][20] that there is a connection between frequency and blurriness of an image. In the histogram frequency metric this is used by looking at the absence of high AC coefficients.

In the implementation[**code˙HF**] a histogram (an array of length 64) is initialized to all 0's. Then all 8^2 luminance blocks[21] in the image are traversed and the DCT values that are greater than a threshold *minDCTValue* are inserted into the histogram by adding a 1 to that index. The *minDCTValue* threshold is used for filtering away image noise. All DCT values below this threshold remains 0 in the histogram.
After this, all the values of the histogram will be reviewed using another threshold, *maxHistValue*. If `histogram_value < maxHistValue · histogram_first_value` the histogram bucket will be discarded, otherwise its weight of a wheighting grid defined in [**jnbm12**] will be added to *blur*, initialized to 0. The final metric is calculated by

$$HF = 1 - \frac{blur}{weight\_total}$$

where *weight_total* is the sum of all weights in the weighting grid. A lower result would mean a more blurred image, while a greater result would mean a sharper image.

This gives a time complexity of $O(\frac{N \times M}{8} \cdot 64 + 64) = O(N \cdot M)$.

In the article[**jnbm12**] the metric also loops over all macroblocks[22]. However, this has been excluded in the implementation[**code˙HF**].

3.1.4 Frequency Domain Image Blur Measure

The FM metric is based on the observation that *when the blur in an image increases the number of high frequency component in the images decreases*[**FM**]. Thus, the metric works by transforming the image into frequency domain and

[17] discrete cosine transform
[18] constant component
[19] alternating component (add more detail?)
[20] *when the blur in an image increases the number of high frequency component in the images decreases*
[21] explain
[22] explain

calculating the no. high frequency components in the image above a threshold. This number is then used to evaluate the sharpness of the image. The sharper the image, the higher the score.

In the implementation[**code˙FM**] the Fast Fourier Transform (FFT) is calculated on the input image. This takes $O(N \cdot MlogN \cdot M)$[**FM**], which is also the total time complexity of FM. Then the FFT is shifted so the 0-frequency is centered in $O(N \cdot M)$ time. The maximum, M, of the absolute value of all Fourier coefficients are calculated $M = max(AF)$, also in $O(N \cdot M)$ time, where $AF = |centered\_Fourier|$.

The final quality assessment is calculated as

$$FM = \frac{T_H}{M \cdot N}$$

where $T_H = |AF > t|$ and $t = M/1000$ is the mentioned threshold.

3.1.5 Variance of the Laplacian

This metric is implemented in python[**code˙LV**] and based on a blogpost[23] referring to an article[24] from 2000. The idea is, that taking the laplacian of an image will leave the edges in the image visible. Calculating the variance of the laplacian image will provide a score, that can be compared to an experimentally set threshold. If the score is below the threshold, the image is blurred and vice versa. The threshold may vary considerably depending on the image data set.

If the image contains colors, that is, it is stored in 3 dimentions, it is initially converted to grayscale. This takes $O(N \cdot M)$ where N and M are respectively the horizontal and vertical no. pixels in the image. After this, the laplacian filter[**code˙LV**]:

$$laplacian\_filter = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

is applied to the original image. This is also applied to all pixels in the image, and thus takes $O(N \cdot M)$ time. Next the variance:

$$\sigma^2 = mean(|a - mean(a)|^2)$$

where

$$mean(x) = \frac{\sum x}{|x|}$$

is applied on all the laplacian values. Thus, the time complexity of σ^2 is $O(N \cdot M)$. Finally the variance is compared to the threshold, $O(1)$. The complete time complexity of the metric is thus

$$O(N \cdot M) + O(N \cdot M) + O(N \cdot M) + O(1) = O(3N \cdot 3M) = O(N \cdot M)$$

[23] `https://www.pyimagesearch.com/2015/09/07/blur-detection-with-opencv/`, accessed: October 4, 2021

[24] `https://decsai.ugr.es/vip/files/conferences/Autofocusing2000.pdf`, accessed: October 4, 2021

3.2 Performance tests

In the classification of sharpness of the otoscopic images of the eardrum, it is more important that no blurry images are classified as sharp than that sharp images are classified as blurry, as the goal is to create a dataset consisting of good images. In order to examine the performance of the implemented blur-metrics, the following test methods will be used.

3.2.1 Confusion matrix

The confusion matrix displays the correctness of the output of a classifier given a binary ground truth. There are four output possibilities:

1. True positive, **TP**: when a positive output is also positive in the ground truth.
2. True negative, **TN**: when a negative output is also negative in the ground truth.
3. False positive, **FP**: when a ground true negative element is classified as positive.
4. False negative, **FN**: when a ground true positive element is classified as negative.

The confusion matrix contains the sum of these different outputs in a matrix as:

Ground true negatives	TN	FP
Ground true positives	FN	TP
	predicted as false	predicted as true

This is also defined in [**img˙analysis**][25] and [**ML˙book**][26]. The output of the classifier is in our case continuous, thus a threshold is needed to determine a binary output. A good approach to determining a threshold is using the ROC curve.

3.2.2 Area under the ROC curve

As the ground true classes may vary in size, the classes are normalized to provide a valid starting point to choose a threshold from. The normalized values are called the true positive rate (TPR also known as sensitivity) and the false positive rate (FPR, also known as probability of false alarm) defined as:

$$sensitivity = TPR = \frac{TP}{TP + FN} \tag{1}$$

[25] chapter 7.4.3
[26] chapter 16.2.1

and

$$FPR = \frac{FP}{FP + TN} \tag{2}$$

as in [**ML˙book**][27]. The receiver operating characteristic curve (ROC curve) is a curve describing the TPR on the y-axis and the FPR on the x-axis given a varying threshold. Labeling all outputs of the classifier above a threshold t as positive and vice versa, choosing a threshold t that is less than all the outputs of the classifier, will provide only positive outputs, thus providing a TPR and FPR of 1. On the other hand, a t greater than all outputs will provide a TPR and FPR of 0. Assuming that the classifier is better than random, thus generally classifying the inputs as negative when negative and vice versa, the area beneath the ROC curve will be between 0.5 and 1. This is because, if the inputs are classified as described, increasing t it will move over the negative labeled outputs faster than the positive labeled outputs.insert diagrams

Knowing this, the area under the curve (AUC), can be calculated as the integral of the ROC curve and used for measuring performance. If the ROC curve goes through $(0, 1)$, the AUC will be equal to 1, and the TPR and FPR will be respectively 1 and 0, thus all the true positives will be labeled as so, and there will be no false positives. If, however the AUC equals 0.5, this will indicate that the ROC in average lies on the straight line between $(0, 0)$ and $(1, 1)$. If the ROC goes through $(0.5, 0.5)$, the TPR and FPR will be equal to 0.5, and thus the classifier will perform as good as a random classifier.

3.2.3 Sensitivity / specificity

Sensitivity has already been defined in equation 3.2.2. This is a measure on how many positives are classified as positives compared to the ground truth number of positives. The specificity (TNR, true negative rate) is defined as the normalised true negatives:

$$specificity = TNR = \frac{TN}{TN + FP} \tag{3}$$

In the case of evaluating sharp otoscopic images, this measure is not as important, as mentioned in 3.2. However, it still provides a good measure of how well the metric tend to perform.

3.2.4 Accuracy

The accuracy of the metric is defined as

$$accuracy = ACC = \frac{TP + TN}{TP + FP + TN + FN}$$

That is, it is a measure of how many inputs are correctly classified compared to the total number of inputs.

[27] chapter 16.2.1

3.2.5 F1-score

The precision (positive predictive value (PPV)) is calculated as

$$precision = PPV = \frac{TP}{TP + FP}$$

This is a measure on how precise the positive outputs of the classifier are, which in this case is a very important measure, as it describes the false positives, that, as explained, are the results that should be minimized.

F_1-score is defined as the harmonic mean of sensitivity and precision. That is:

$$F_1 = \frac{2}{PPV^{-1} + TPR^{-1}} = 2\frac{PPV \cdot TPR}{PPV + TPR} = \frac{2TP}{2TP + FP + FN}$$

This is another way to measure the accuracy of a binary classifier, though without including the true negatives. The score will be between 0 and 1. 0 if either PPV, TPR or both are 0, and 1 if PPV and TPR are both 1, that is they classify perfectly compared to the ground truth.[28]

3.3 Generation of synthetic dataset

To extend the image data, the images have been mirrored and a Gaussian filter has been applied to the sharp images.

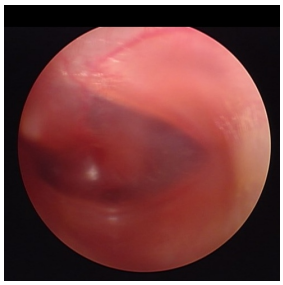
3.3.1 Rotate...

insert images

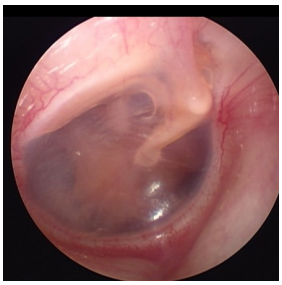
3.3.2 Gaussian filter

To the sharp original and mirrored images, a Gaussian filter (described in [**img˙analysis**][29] with a kernel size of 15×15 and different values of σ have been applied.

[28] Both equations need a source
[29] chapter 5.2



(a) Naturally blurred image for comparison with the synthetically blurred images.



(b) A natural image with no problems.

(c) The sharp image (2b) with applied Gaussian filter with $\sigma = 1.5$.

(d) 2b blurred with $\sigma = 3$.

(e) 2b blurred with $\sigma = 6$.

3.4 Results

The continuous CPBD metric[**CPBD**], the Histogram frequency-based metric (HF) and the FM metric have been evaluated on the original dataset. They all output a score in the interval $[0, 1]$. The implementations of the CPBD and FM metric are both in python, while HF is implemented in `C`. This likely has a thing to say in how fast the algorithms run.

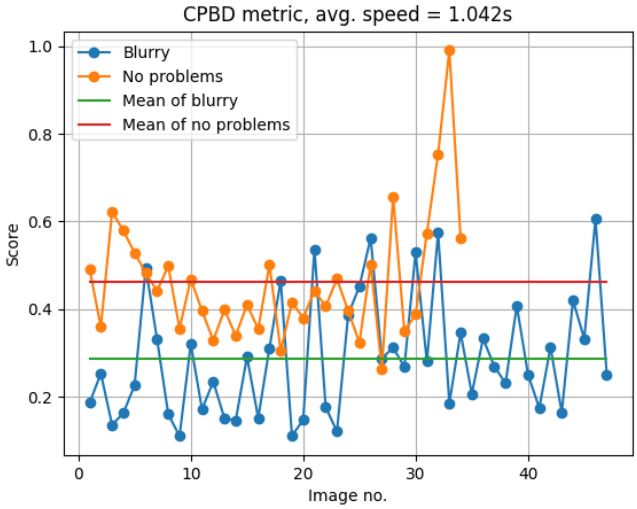


Figure 3: Results of running CPBD on the blurry and the good images.

Figure 3 displays the performance of the CPBD-metric. Fortunately, as can be seen on the graph, the blurry images tend to have lower score than the images without problems. The average time on evaluating an image is 1025.95ms, calculated as `avg.speed` $= \frac{sum(time)}{no\_images}$, which is quite slow. However, one second might not be too slow when interactively evaluating a single image at a time.

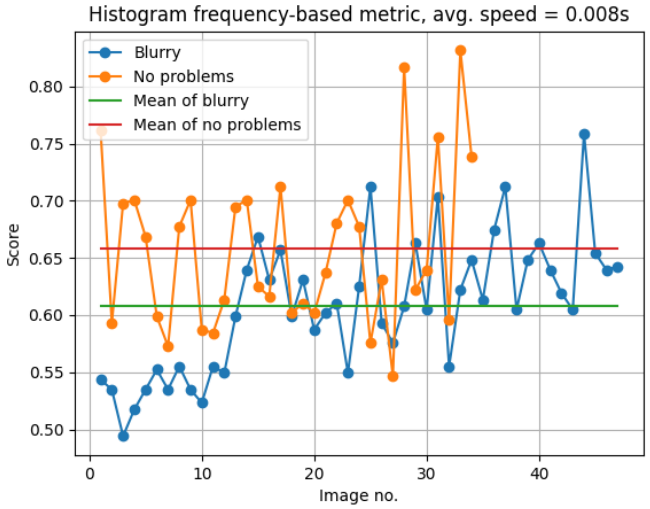


Figure 4: `avg.speed` $= \dfrac{0.306\text{s}}{47+34} \approx 0.008\text{s}$. The sharper the image, the lower the metric is. Since the metric is measured in percentage and thus lies between 0 and 1, the output has been normalized so that the output is still between 0 and 1, but the sharper images will have lower score. It is normalized by the simple `norm` $= 1 -$ `result`.

As can be seen in figure 4, the HF metric immediately seems to perform reasonably well, but not as good as CPBD. There is only about $0.66-0.61 = 0.05$ between the two means in HF, whereas in CPBD there is about $0.48-0.3 = 0.18$. However, the speed of this metric is much lower, 0.008s pr. image than the 1s from CPBD.

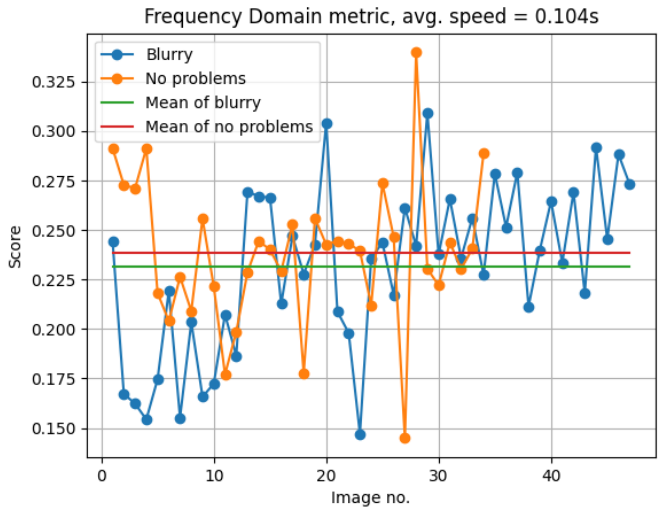


Figure 5: `avg.speed` $= \frac{4.694\text{s} + 3.690\text{s}}{47 + 34} \approx 0.104\text{s}$. The sharper the image, the higher the metric.

The initial impression of figure 5 is that the results are almost random. The distance between the two means is around $0.240 - 0.232 = 0.008$, which is much lower than the distance in both the HF metric and the CPBD metric.

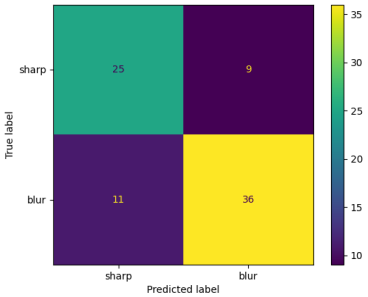
To enable binary evaluation of an image, the two means of respectively figure 3, 4 and 5 are used to determine a threshold. The threshold is calculated as $threshold = min(mean1, mean2) + |\frac{mean1 - mean2}{2}|$ to account for the different amount of sharp and blurry images in the dataset. Scores above the threshold will be evaluated as sharp and vice versa.

```
freja@GLaDOS:~/Documents/DTU/bachelor/software$ python3 FM/output/visualize_output.py
mean = 0.235
freja@GLaDOS:~/Documents/DTU/bachelor/software$ python3 Histogram-Frequency-based/output/visua
lize_output.py
mean = 0.633
freja@GLaDOS:~/Documents/DTU/bachelor/software$ python3 cpbd_run.py
mean = 0.375
```

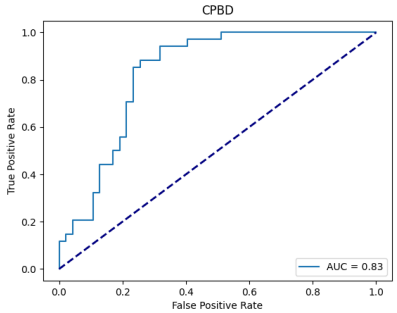
Figure 6: The mean values (thresholds) of all the results from the metrics in order Frequency Domain5, Histogram frequency-based4 and CPBD3

These means will be used as thresholds to provide a binary measure of the images: if the image lies above the threshold, it will be classified as sharp and vice versa.

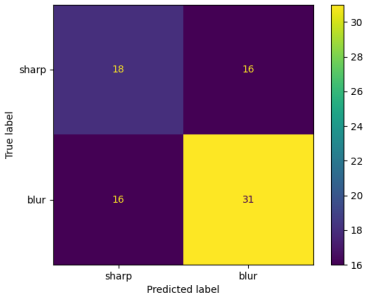
The classification into sharp and blurry images by the three metrics are plotted in confusion matrices and area under the ROC curves. In the confusion matrix, the true positives in the top left corner are the ground true sharp images evaluated as sharp by the metric. The lower right square represents the true negatives, that is the number of ground true blurry images evaluated as blurry. False positive and false negative are contained in respectively the lower left and the top right.



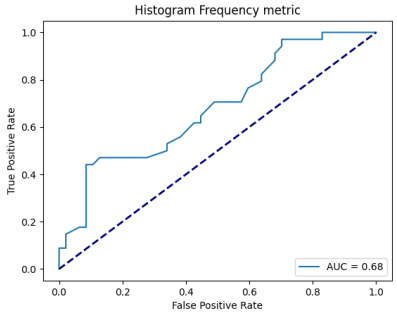
(a) Confusion matrix of CPBD results. The true positive and true negatives make up the majority of the results.



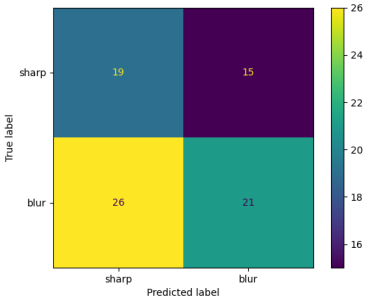
(b) The AUC of CPBD has a score of 0.83. This is a quite good result.



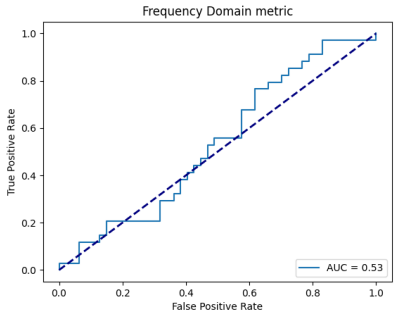
(c) Most of the results from the HF metric lies in the true positives and negatives. However, still a rather big amount lies in the false positives and negatives.



(d) This metric does not perform very well, however better than random.



(e) Many of the blurry images are evaluated as sharp by the FM metric.



(f) This metric seems to provide almost random results, that is the metric seems to perform very badly on the current type of images.

From the above plots, it is clear that CPBD an HF performs better than

FM. These metrics will be combined...

4 Choosing the best metric

In this section, the four metrics will be given different parameters to make them perform their best.[30]

4.1 Tweaking parameters

Some of the chosen metrics have constants that can be adjusted to optimize the correctness of the output scores. This concerns the histogram frequency-based and the frequency domain metrics.

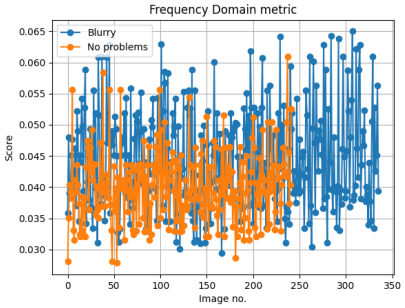
The best possible constants have been chosen experimentally, looking at the AUC value of the outputs when running the metric on the training data set. The constants will be chosen between those that provide the highest AUC values on the training data set both including and excluding the Gaussian blurred images.

4.1.1 FM

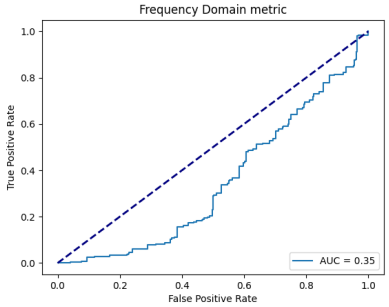
In the frequency domain image blur measure, the threshold $t = \frac{M}{1000}$ is experimentally estimated [31]. This will also be done in the following, where the constant for division (1000) is varied.

[30] rewrite

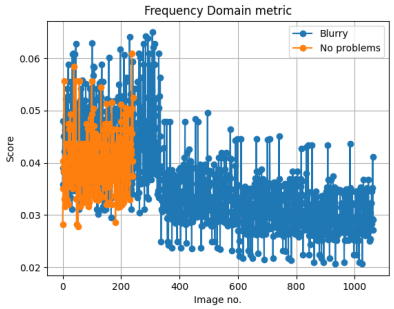
[31] *Experimentally it is observed that this particular threshold value gives a fairly accurate sense of image quality*[**FM**]



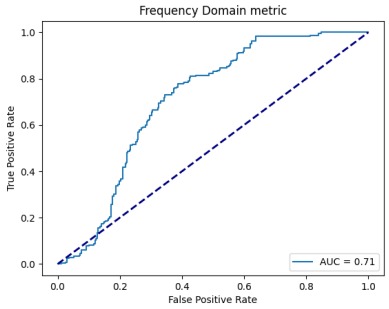
(a)



(b) The worst performance with constant 102 and no Gauss, having AUC= 0.35.

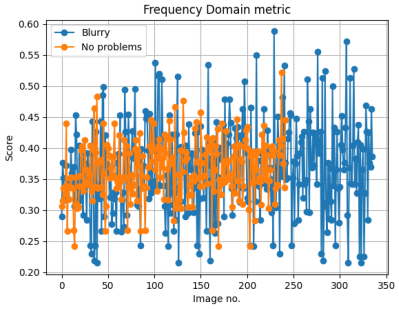


(c)

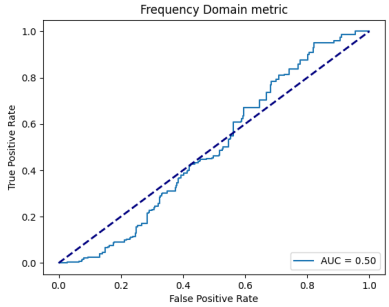


(d) The performance with constant 102 on the training set including Gauss, having AUC= 0.71.

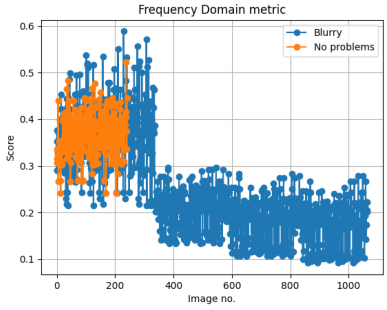
Figure 8: The output with a constant of value 102



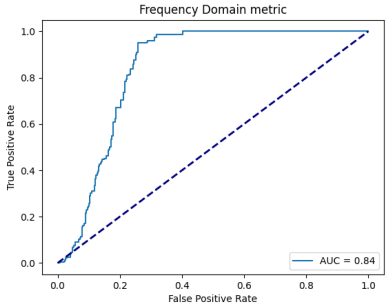
(a)



(b) The AUC= 0.5 on the training set without Gauss with the constant 2202.



(c)



(d) One of the best performances with Gauss is with the constant 2202, having AUC= 0.84.

Figure 9: The output with a constant of value 2202

The Gaussian blurred images will not be taking into account when choosing the constant for the metric, as they distort the result. This can be seen in the above images, where the metric performs excellent on the training set including the Gaussian images, while at the same time it performs quite badly on the training set without. The AUC in the tests on the training set without Gauss did not manage to exceed 53%, however, the lowest value was 35%, which gives an AUC of $100\% - 35\% = 65\%$. However, doing this, the metric would classify the Gaussian blurred images as sharper when applying more blur.

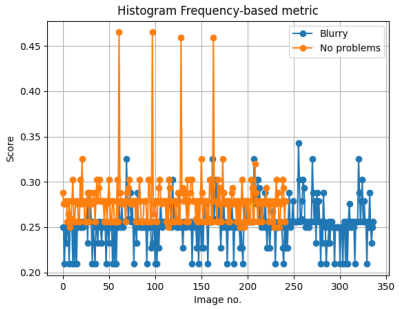
As this metric performs worst of all 4as can be seen..., it will not be taken into account in the following sections.

4.1.2 HF

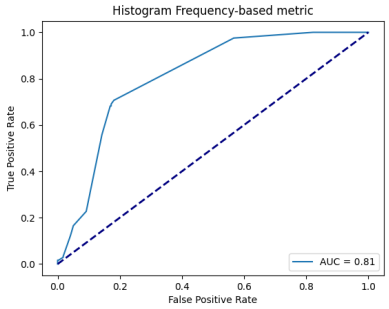
4.2 Metric performance rewrite...

Following are statistics on the three metrics. In the histogram frequency-based metric, the two thresholds, $minDCTValue$ and $maxHistValue$, can be ad-

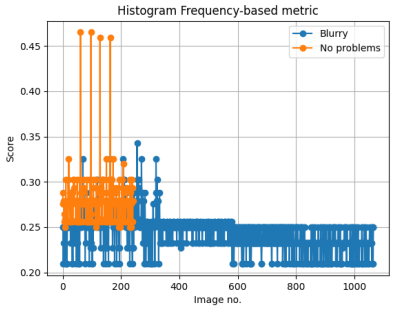
justed for optimizing the output.



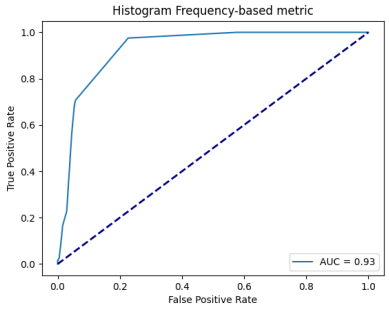
(a)



(b) One of the two best performances without Gauss have constants $minDCTValue = 1$ and $maxHistValue = 0.085$, having AUC= 0.81.

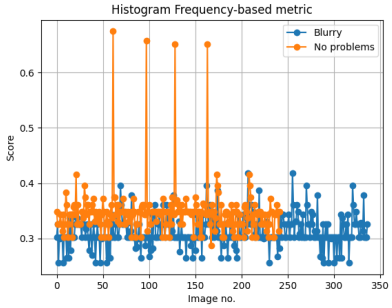


(c)

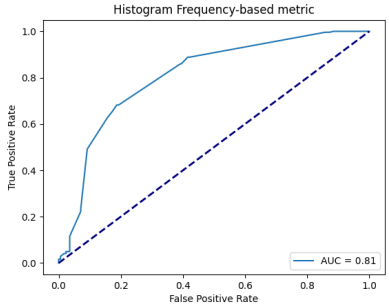


(d) The output on the training set including Gauss on constants $minDCTValue = 1$ and $maxHistValue = 0.085$, having AUC= 0.93.

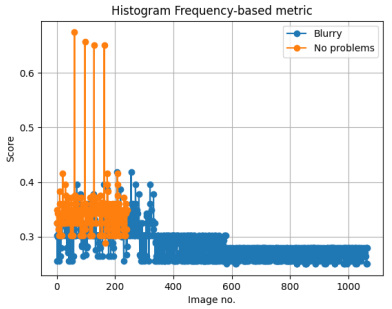
Figure 10: The output with $minDCTValue = 1$ and $maxHistValue = 0.085$.



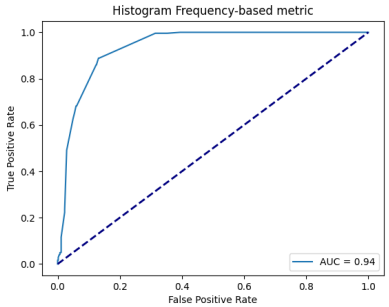
(a)



(b) One of the two best performances on the training set with $minDCTValue = 0$, $maxHistValue = 0.115$ and without Gauss, having AUC= 0.81.



(c)



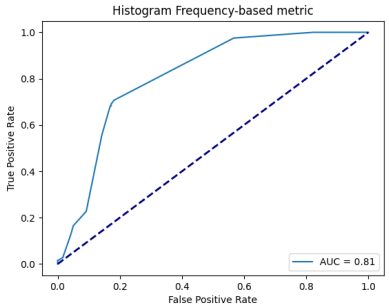
(d) The best performance with Gauss is with $minDCTValue = 0$ and $maxHistValue = 0.115$, having AUC= 0.94.

Figure 11: The output with $minDCTValue = 0$ and $maxHistValue = 0.115$.

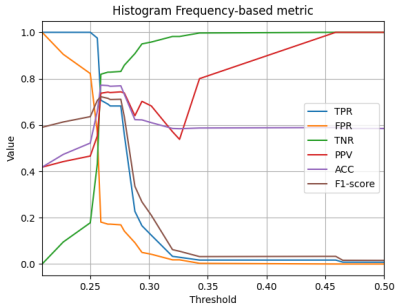
The correctness of the output of the algorithm doesn't seem to be distorted by the Gaussian blurred images. In figure 11 the ROC curves with highest scores are displayed. As can be seen, the output of running the algorithm on the training set without Gaussian blurred images performs very well in two different situations. The correctness is also very high and almost identical for the results including the Gaussian blurred images. However, it is a little better with the lower $minDCTValue = 0$, whose purpose is filtering away noise. That is, when no noise is filtered away.

Knowing this, the $minDCTValue = 1$ and $maxHistValue = 0.085$ are chosen for the metric, as the purpose of the metric is not detecting Gaussian blur, but blur in non-manipulated images, where noise could occur, and as the worsening of the results on the Gaussian blurred images is insignificant.

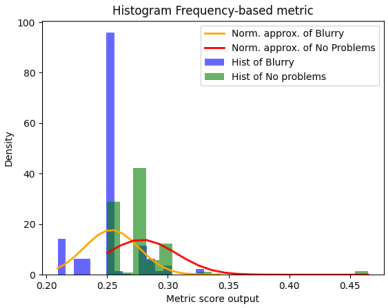
4.2.1 Histogram frequency-based metric



(a)



(b)

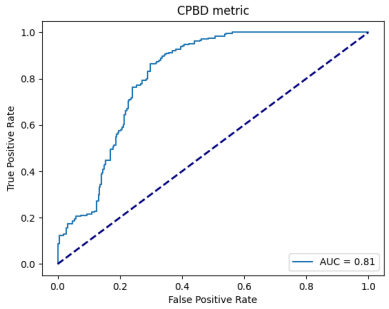


(c)

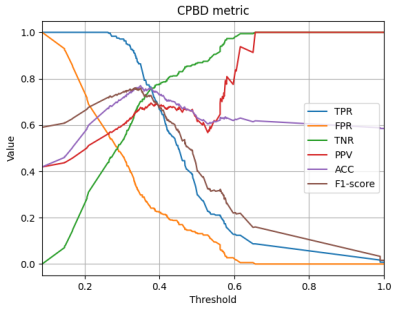
Figure 12

4.3 Best version of other metrics...

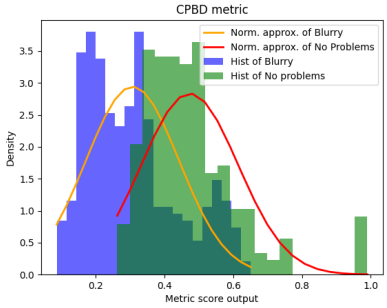
4.3.1 CPBD



(a)



(b)

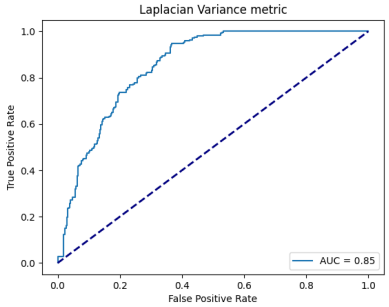


(c)

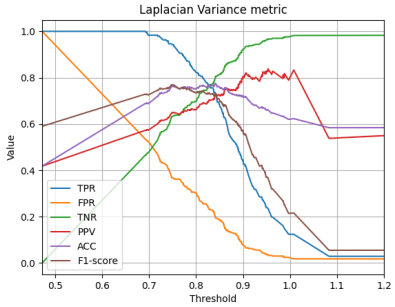
Figure 13

4.3.2 Variance of Laplacian

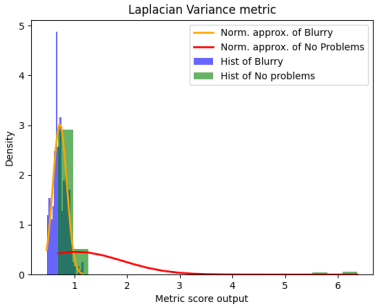
JPG



(a)



(b)

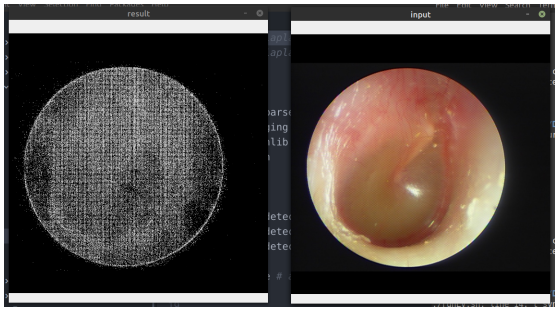


(c)

Figure 14

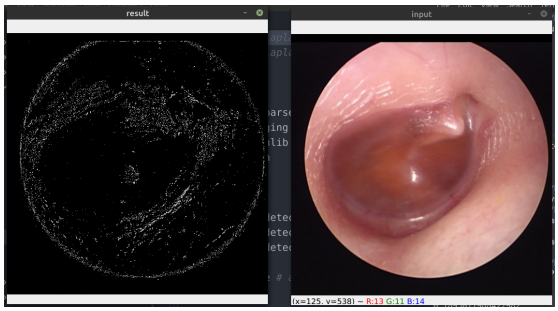
PNG
Outlier

(a) This image recieves a score of 5.7, whereas most of the sharp images scores around 1. By looking closely at the image, one can see a grid-like structure on the image pixels, which explains the output in figure 15b. add to bilag



(b)

Figure 15



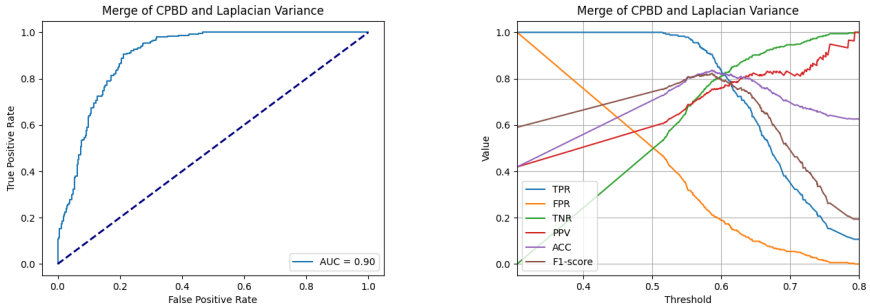
(a)

4.4 Combining the metrics

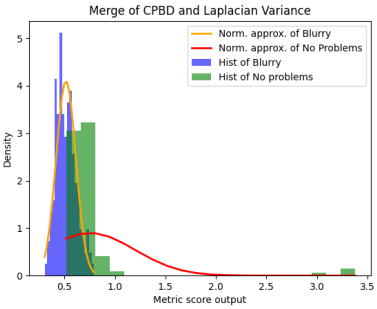
The three metrics, CPBD, Variance of Lapracian and Histogram Frequency-based metric, will be combined to form 4 new metrics. This is done by combining the outputs of the metrics as $m_{new} = \alpha \cdot m1 + (1 - \alpha) \cdot m2$, where $m1$ and $m2$ are two different metrics.

4.4.1 Merge of CPBD and Variance of Laplacian

Highest AUC = 90%



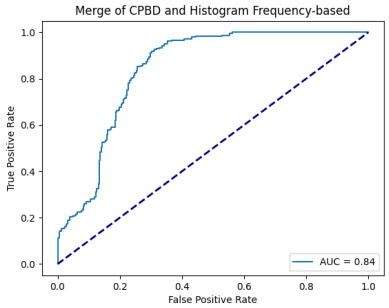
(a) (b)



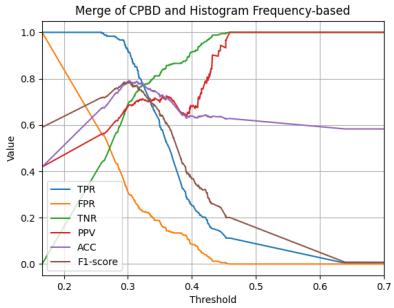
(c)

Figure 17

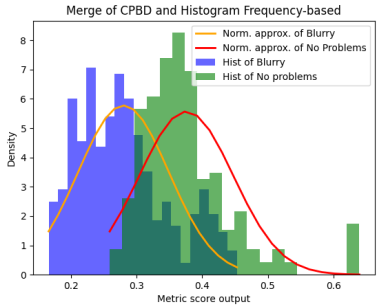
4.4.2 Merge of CPBD and Histogram frequency-based metric



(a)



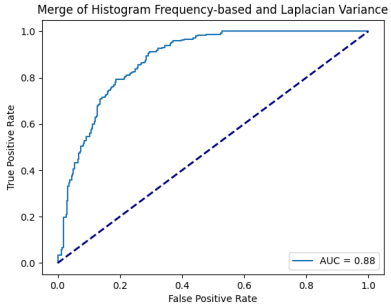
(b)



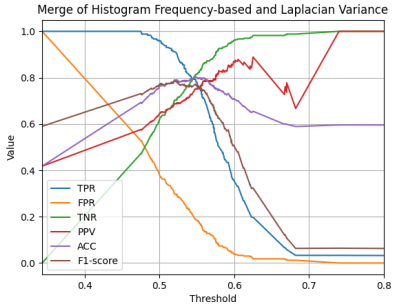
(c)

Figure 18

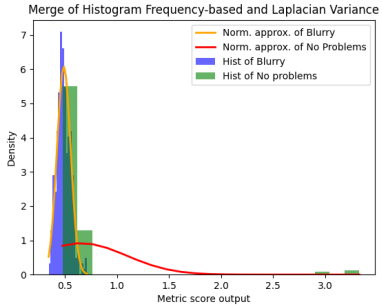
4.4.3 Merge of Histogram frequency-based metric and Variance of Laplacian



(a)



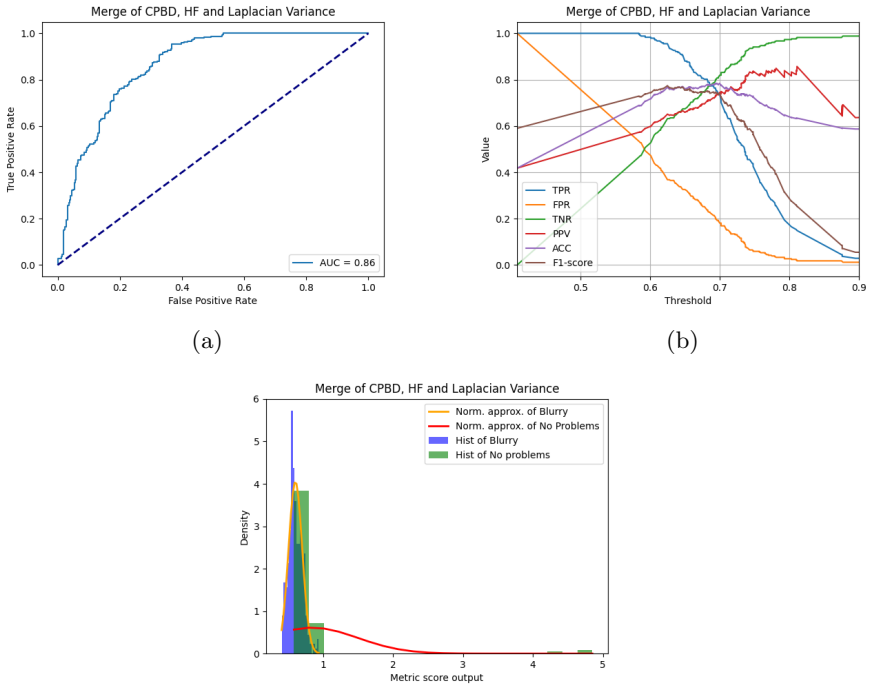
(b)



(c)

Figure 19

4.4.4 Merge of CPBD, Histogram frequency-based metric and Variance of Laplacian



(a) (b)

(c)

Figure 20

4.5 "Training" the metrics

The goal is to avoid false positives (**FP**: classified as sharp when blurry), as far as possible, as these can distort the results from the data later on, while still achieving some positive outputs.

Choosing the threshold:
Let the user choose an acceptable TNR (specificity), e.g. 98%.
After this, find the corresponding threshold and choose the metric with the highest accuracy, as it will provide a possibility that some images will be "accepted", that is, classified as sharp.

We could also calculate summed distance, d, of some rates... TPR (sensitivity), F1-score, precision... to the top and bottom border (1 and 0) according to which border is desired for that particular rate. Then choose the metric with smallest d.

- maximize TNR (how many are blurry when blurry)

- maximize TPR (we would still like some possibility that images are classified "sharp")
- maximize PPV (part declared sharp when sharp)
- high f1-score for general score + do not produce FP (to not filter away too many negatives)

The following graphs displays output on the training data set excluding the Gaussian blurred images.

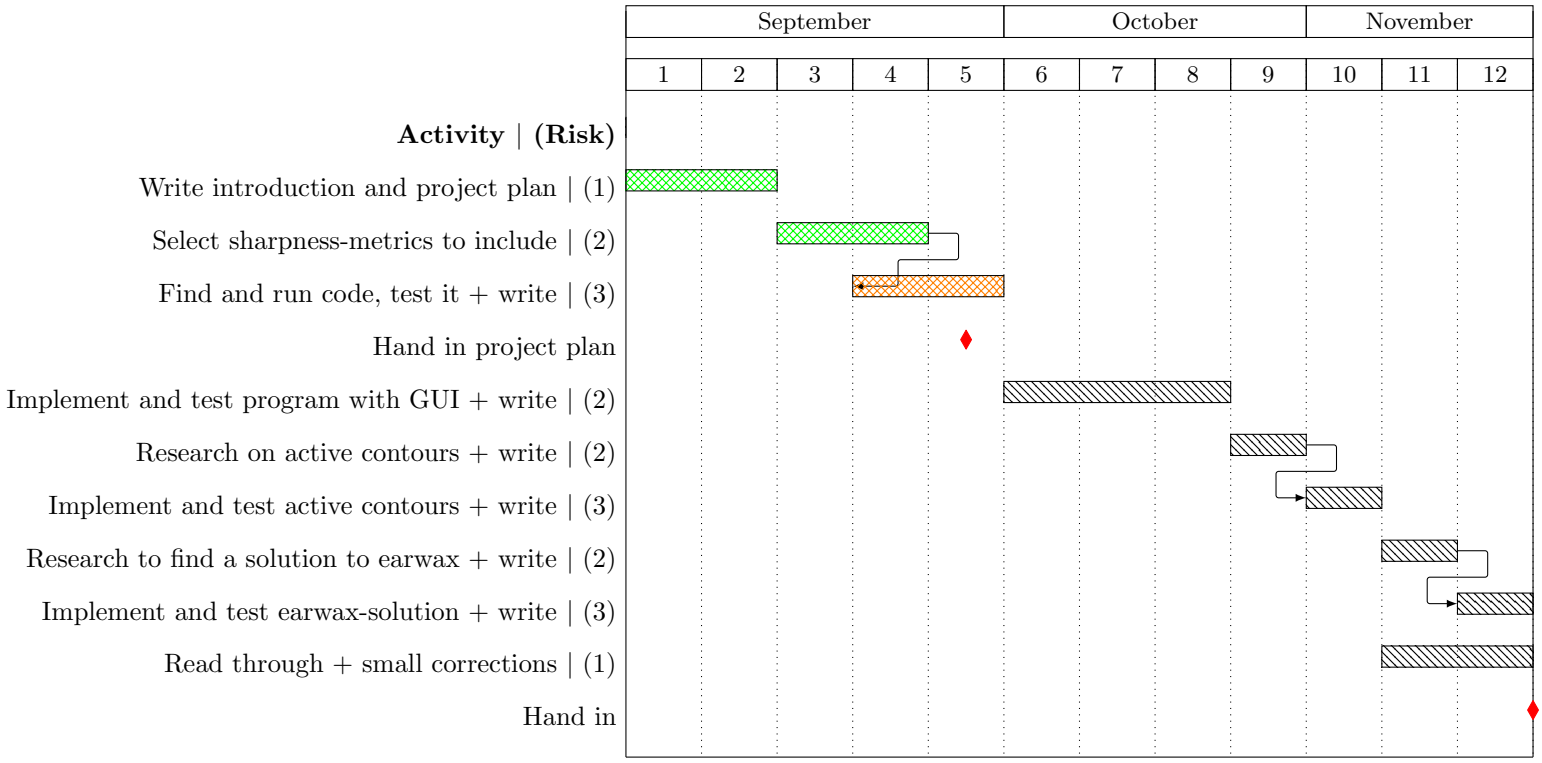
5 Implementation

Using python, numpy
Could use C or C++ for image processing (matrix computation) and call from python. However, this is what numpy has already implemented.

Implementing image-processing algorithms in Jupyter and later copying them into the final program.

6 References

Overall Project Plan



Møde 7

Spørgsmål

- *Find betragtninger over køretider!* betragtninger?

Noter

png i stedet for jpeg i lv lav boxplot tune alpha-værdig (normaliser) med merge metrics for at se hvornår den scorer bedst (vægtning)

Møde 5

Skriv til Josefine: møde med Interacoustics. Video af otoskop, så min model kan sige til, når billedet er godt.

Lav plot af hvordan metricken performer på forskelligt gausiske blurrede billeder
Større gaus-matrix

Møde 4

Spørgsmål

- Tips til at finde implementationer? Har ikke haft held med det...
- Træningsdatasæt: syntetiske billeder? (skal vist ikke teste korrelation). Er det så bedre med syntetiske billeder også?
- Josefine mener, at jeg ikke behøver komme igennem alle tre kvalitetsproblemer. Er det vigtigst, at jeg får gennemarbejdet billedanalysedelen, eller at jeg også får lavet et program?
- Ser min plan urealistisk ud? Den skal afleveres omlidt - er det bare en formalitet?

Noter

Få den fundne kode til at køre
confusion matrix. beregn relevante scores
ikke overtræne på data - lav syntetisk datasæt
Det er nok at have et gennemarbejdet bachelorprojekt kun om blur...

Møde med Josefine 1

Spørgsmål

- Er parametrene i tabellen relevante i forbindelse med at udvælge relevante algoritmer til blur-detection?
- Hvilke algoritmer skal testes? F.eks. vil jeg gerne teste de to med træningsdata, da jeg går ud fra, at de fleste billeder vil være relativt ens.
- Hvilke metoder til test ville være bedst? F.eks. area under the roc curve, korrelation, accuracy
- Hvordan skal test-datasættet laves? Forskellige metoder bl.a. gaussian. Skal de være gradvist mere uskarpe, eller skal der være en tydelig grænse?

Noter

simple variance, prøv laplace/derivative-based + frequency domain (klassiske)
Skal de kombineres?
sensitivity/ specificity - specificer accuracy overall (F1-score)
python library til test: sklearn, metrics

Møde 3

Spørgsmål

- Møde i uge 42?
- Skal jeg tilføje en række i tabellen over, om paperet siger, at algoritmen er hurtig?

Noter

Lav forklaring af søjler i tabellen
møde med josefine - hvad synes
area under the roc curve
korrelation
skriv til rasmus om accuracy mål (kapitel i bog) - hvis josefine siger god
hvordan laves test dataset

Møde 2

Tabel over alle forskellige algoritmer
Hvilke er gode til hvad? Beslut, hvilke der skal bruges på baggrund af det.
Skab kunstigt dataset. Afprøv matlab-implementationen af JNB.
Skriv lidt mere (ja resume) til hver artikel. Bliver integreret som et underafsnit i introduktion. Referer til artiklerne herfra senere.

(feedback til lars: skriv gerne en "todo" af hvordan man får noget op at køre. Til appendix)

Møde 1

openCV, pillow, scipy
undersøg flere blur-algoritmer. Hvilken virker bedst?
Ugerapport i bachelor-afhandlingssprog template på ugerapport på hjemmeside
1. projektplan
2. risikoanalyse (for at blive forsinket i processen) - hvad er plan B?
3. Skriv egen forståelse - vis problemerne (billeder), undersøg algoritmerne (tid,

præcision) (lav et framework)
Introduktion

Indsæt læste artikler i "previous work"
Screendump af interessante billeder til ugerapport
Hvor langt er vi kommet i forhold til planen?

Diskussion/resultater - hvordan kan resultaterne bruges?

Informationer om projekt af typen Bachelorprojekt:

Dansk titel: Evaluering af kvaliteten af otoskopibilleder med billedanalyse
Engelsk titel: Image-based quality evaluation of otoscopy images
Forklaring/indhold(DK): læringsmål:

- Beskrive hvordan otoskopibilleder bliver taget
- Beskrive typiske problemer med kvaliteten af otoskopibilleder optaget i et klinisk miljø
- Vælge eksempler på billeder hvor der er typiske kvalitetsproblemer
- Implementere, teste og evaluere billedanalysealgoritmer, der kan detektere og kvantificere typiske kvalitetsproblemer i otoskopibilleder

Evaluering af otoskopibillede-kvalitet

Målet med projektet er at lave et værktøj, der kan afgøre, om billedkvaliteten af et otoskopibillede af trommehinden er god nok til at anvende billedet til at stille en diagnose. I første omgang er målet af identificere følgende problemer:

- Uskarpt
  § De, K., & Masilamani, V. (2013). Image sharpness measure for blurred images in frequency domain. Procedia Engineering, 64, 149-158
  § Ferzli, R., & Karam, L. J. (2009). A no-reference objective image sharpness metric based on the notion of just noticeable blur (JNB). IEEE transactions on image processing, 18(4), 717-728.
  § Ferzli et al. har en grundig gennemgang af tretten forskellige metoder til skarphedsvurdering med referencer
- Ørevoks
  Kig på farverne i billedet, da ørevoks er gulligt. Fravælg f.eks. billeder med for meget gult i billedet
- Membranen er ikke synlig på billedet
  Active contours eller anden simpel segmenterings algoritme til at segmentere membranen. Hvis ikke membranen kan findes i billedet, eller hvis segmenteringen viser at membranen er meget lille i billedet kan man konkludere at billedet ikke er godt nok

Afslutning af projektet: at lave det til et lille program med en GUI. Her vil man kunne uploade sit billede nemt med en load-knap, og så få feedback på billedet. Feedback kunne være en besked om hvilket problem programmet har detekteret – måske også i hvilken grad, og så en besked om at lægen skal tage et nyt billede uden disse problemer.

Josefine har forberedt en database med billeder med de forskellige problemer med 47 uskarpe billeder, 51 billeder med ørevoks, 21 billeder hvor membranen ikke er synlig på billedet, samt 34 billeder som lever op til den forventede kvalitet uden disse problemer.

Læringsmål

- Describe how image data is represented in a computer system
- Analyze, test and evaluate the running time of the used image processing algorithms
- Apply and analyze the graph algorithms that are used in active contours algorithm
- Establish a requirement specification for a software interface to the developed image analysis algorithms
- Design, implement and evaluate a user interface that can be used for quality evaluation of otoscopic images
- Plan and execute a test based on the requirement specification and report the result

Requirement og test kan være fokuseret på hvad en typisk bruger kunne ønske sig at af en brugerflade.
"test" - måling af køretider via en form for timer.
"Analyze" - er om algoritmen er O^2, O(N log(N)) osv.
"evaluate" - mere højniveau evaluering om fordele og ulemper ved algoritmevalg i forhold til køretid og brugsscenarie. F.eks.: er det ok (givet requirement) at en læge skal vente 10 minutter på et bedre svar end med en knap så præcis algoritme, som kan give et svar på 5 sekunder?