Faculty of Informatics
Masaryk University Brno

Image composition

Master degree Thesis

Vladimír Ulman

2003

I declare this thesis to be my original work that I have written singly. All the sources and literature that I have used I cite properly and provide full link to its source.

I would like to express thanks to my thesis supervisor Doc. Matyska for providing me with basic literature and giving me this way very good point to start, also for helping me organise my ideas when writing this thesis. Thanks belongs also to my co-supervisor MUDr. Feit for testing the program and for all fruitful notes to it. And especially I must thank my parents for supporting me during my entire study, I couldn't get such far without them.

Abstract

Image composition is a process in which small pictures are composed together into one single picture. It is used whenever the sensor that is used for obtaining pictures is simply not capable of producing final image of desired size. There are several reasons for this to happen. Mainly it is the matter of physics, sometimes such adequate technology exists but it is too expensive. Typical example of the latter is billboards. They look like a single picture but in fact they are composition of smaller ones. These are easier to produce and handle.

But how does a man compose picture from smaller ones? He should know at least their relative positions so he knows which pictures should be composed together. He does his job using a process called image *registration*. This is the basic of each composition. Registration gives us important information how to align overlapping parts of images so these two seems as one image. Images need not necessarily overlap. In special (and rare) cases this information can be retrieved only by comparing pixels from left image with corresponding adjacent pixels from right image.

In fact there exists lot of various techniques to match two images. In this paper we will describe one approach how to compose images. This approach isn't based on feature-extracting and won't need any pre-segmentation of images. It is working directly on the image data using intensity values of pixels present in overlapping parts of both images. Several known statistical methods will be presented here. The purpose of these methods is to retrieve valid align information directly from the data. Images will be little overlapping and won't suffer from heavy distortions (just like in the billboard example), their relative positions will be known in advance.

Finally we will briefly describe structure of program which is correctly composing given input images into single one. Some optimisation techniques will be shown too as well as some hints of image fusion when two images are not identical in their common overlap.

Keywords:

Image composition, Image fusion, Image matching, Pattern matching, Sum of absolute valued differences, Stochastic sign change, Normalised correlation coefficient, Correlation ratio, Mutual information, Uni-modal matching

Contents

Chapter 1

Introduction

Preface

Computer graphics is one of the most studied subjects in information technologies. It's literally everywhere around us. It usually serves as an information channel from some device or set of devices to human. This information is then graphically presented, it doesn't have to be only via some tables in windows on someone's computer screen. As an example we can state image processing in medicine where typically an image from inside of human body is taken and then presented on doctor's screen or somehow printed.

Important fact on this matter is that this information is often processed before it is presented. The image composition occurs often in the process of graphical information manipulation. It is used when ever the sensor that is used for obtaining images is simply not capable of producing final image of desired size. And so the final image must be composed from smaller ones which are scanned separately. Generally it can be said that image composition as a part of computer graphics processing can be found mainly in optical systems. In such systems the optical properties of lens don't allow us to scan large images. The problem is solved by scanning only that part of image in which the lens won't affect the scanning process much and hence the registering process too. The desired area is then registered part after part and finally composed together.

Examples of image composition can be found in almost every industry, notably in medicine. Example of everyday use around us might be in recent time the digital cameras when taking panoramatic pictures. The latter is usually done by taking sequence of images and then using computer software to stitch them together in one panoramatic shot.

Uni-modal and multi-modal compositions

Composition is often divided into two basic classes in accordance with the way in which the individual images where scanned. There are uni-modal and multi-modal image compositions. Uni-modal image composition means that all images were obtained using the same device or devices from the same class. Class of devices in this context contains all such devices which have very similar properties important for image scanning, for instance illumination, contrast, distortion, noise, etc.

Multi-modal image composition on the other hand means that some images in composition are scanned using some sensor and some are scanned using some other sensor.

Sensors in multi-modal image compositions are not from the same class. Therefore images are obtained often in different scale. Contrast and brightness usually aren't the same too, most importantly in this case same objects are visualised differently and thus some colour conversion should be performed. A classical example of multi-modal composition: Bones are shown in dark colours in MR[1] pictures while they are in light colours in CT[2] pictures.

This automatically precludes use of matching straight the raw data. Registered image is represented in computer as a sequence of pixels each holding light intensities of typically red, green, and blue colours, very often it is enough to represent images only in grey-scale (luminance of each pixel can be expressed in numbers ranging from 0 to 255, the more the brighter). In this example we must therefore firstly find some conversion function. This function should define correspondence between values of the same object from one modality to the other one. This doesn't have to be an easy task, mainly when full-automatic image composition is desired.

But nothing is lost, there exist other approaches to this problem. Nevertheless these usually make use of image segmentation which is again nothing easy. The purpose of segmentation in this case is to extract objects which are present in both images (bones in this example) with their positions, then somehow try to match images using gained information. These images should have objects pictured likewise and shouldn't probably picture anything else. It is obvious that multi-modal image composition is sometimes far more complicated then uni-modal image composition. Uni-modal image composition doesn't have to take care of notable differences between values representing same objects.

Matching

Composition could be well thought as of 2D pattern matching. Consider composition of two images. Left is called the reference (sometimes it is also called model) image, right is usually called the registered image. Images are typically overlapping. Composition means firstly find some transformation of coordinates and secondly use this transformation when creating the final image. The first stage, the correct-transformation searching, could be imagined as pattern finding (the overlap of the registered image) in the data (the overlap of the reference image). We will return to this later.

The desired property of transformation of coordinates is clear. It must uniformly establish the bindings for every pixel from reference image with its counterpart in the registered image. Very often two following facts are true when composing two images (sadly even in uni-modal case): the values in gray-scale representation of same object aren't identical, images in their common overlap don't correspond exactly. This is bad news both for finding the correct transformation and for final stitching. The transformation search process must be a little bit fuzzy. This means that it must not search for exact solution, for exact transformation that will map given pixel onto its identical counterpart. When final stitching is in progress special care must be taken of hiding the border where the first image ends and where the second image continues. The sutura will be noticeable.

[1] Magnetic resonance
[2] Computed tomography

Chapter 2

Overview of the thesis

Chapter 1 presented the image composition problem. We have stated that it is a computer graphics problem, when is it used and we gave some examples. Then we have stated basic problems that follows this issue.

Chapter 2 is this chapter.

Chapter 3 goes more into detail about the matching principles. It will also introduce notation that is used throughout this document. We will describe here given optical system which was used for image scanning. Image composition problem was studied and results published here were applied on this system.

We will also specify the alignment evaluation methods that were implemented and tested. It will try to give some information regarding their ideas behind, complexity, implementation help and range of cases where this method could be successfully used.

Also in this chapter we will compare selected alignment evaluation methods, for this purpose we will define tool which should describe the process of matching of given two images. Then we will try to show the behaviour of these selected methods using this tool. Finally in this chapter we will compare the speed of selected methods.

Chapter 4 is about some speed-up hints in process of image composition. Two optimisation techniques will be shown here. These techniques will operate on the correct-alignment search process. Then section about image fusion concerning hiding of sutura after the transformation of coordinates is found.

Chapter 5 is conclusion.

Chapter 3

Matching

3.1 Alignment and basic approaches

In the introduction we have learnt that image composition is a fusion of given images into one single image. This is done separately for each pair of images that have to be composed. So for each pair the image composition consists of matching their common overlap and then stitching them together according to the discovered transformation of coordinates. The most important part is the first one, the matching. Matching is a process in which we are trying to retrieve the correct alignment. We can do it in many possible ways, next sections will cover this more. By now we will try to concentrate on the notation through which the alignment is defined.

Alignment is defined by certain parameters values of transformation of coordinates. This transformation depends on the kind of distortion that is present in the images. We are interested only in spatial distortions, i.e. such distortions that actually move objects in overlap or change their size or shape.

For instance suppose we are composing two images, the left and the right one. Suppose these two images were obtained using one sensor (uni-modal composition). This sensor will have some error on the left edge of its view port. Let us say that sensor is optical microscope which has scratch on its lens or more likely the lens will be pour quality and so it will not maintain same proportion on its left and right edge. This will lead, for example, in a line in the overlap of left image (overlap will be at the right edge of image) that will not look like a line in the overlap of the right image (this overlap will be at the left edge and so it will be affected by mentioned error on sensor). It is probable that the line and the curve which should represent the same object won't even have the same length. The scientists will certainly find some more realistic examples of spatial distortions from their experience.

From this example it could be also noticed that we must first decide (or experimentally find out) what type of distortion (or what mix of distortions) is present in the given system. The reason is both obvious and important: According to the distortion we select the transformation of coordinates. If spatial distortions are absent then it is enough to describe alignments only with transformation which only shifts coordinates by some constant. When multi-modal composition is employed it frequently happens that images are scanned in different scales, then some linear transformation comes to play.

Obviously "constant-shift" transformation can be replaced with linear transformation which can be again replaced with some more sophisticated one and so on. This way we can use only some general transformation for describing every possible alignment. But there's a catch. The matching process is a search through given parameter space. The more general transformation we employ the bigger parameter space we must search. For this reason to keep matching process feasible we are trying to use the simplest transformation possible and usable. This is important in order to develop successful system which is both fast (parameter space is small) and reliable (selected transformation is usable). These facts lead to contradictory requirements and the decision what transformation to use is often a matter of compromise.

3.1.1 Notation

We will represent images with a function of two arguments. These two arguments will be mapped to the specific gray-scale value. Arguments denote the position of some pixels in the given image, we will use Cartesian coordinate system with top left corner of the image at position $[0, 0]$. Obviously this function will be dedicated to some certain image and will define for every valid combination of arguments its value conforming to the value of pixel at the given position in the given image. The first argument will be the x coordinate and the second argument will be the y coordinate. The range of both arguments will be according to the dimension of the entire image. The reference image we will denote with function $A(x, y)$, the registered image we will denote with function $B(x, y)$. Using this notation while given I and T, alignment can be formulated

$$A(x, y) = I(B(T(x, y, P))) \tag{3.1}$$

where I is the intensity conversion function and T is the transformation of coordinates.

Intensity conversion function is typical for multi-modal composition where same objects are displayed using different gray-scale values. Clearly we can assume I to be the identity function in cases where intensity conversion is not needed. The transformation of coordinates is clear, for given position in reference image it returns the corresponding position in the registered image according to the given P. Parameter P defines the alignment and in fact the correct value of P is what we are searching for when performing the image matching. During composition of one matrix of images the transformation is supposed to be of the same type, denoted $T(x, y, P)$. Assume the transformation formalism is some formula with parameters (bundled in one complex parameter P) and two input variables x and y.

For the sake of simplicity we will assume that arguments of variables x and y (respective the position) will range only from the overlap. But the overlap is defined by certain alignment, i.e. by P. This concludes that range intervals $X_A(P)$, $Y_A(P)$, $X_B(P)$, $Y_B(P)$ are dependent on the P and we can therefore finish the definition of alignment with

$$\begin{aligned} T(x, y, P) &: X_A(P) \times Y_A(P) \to X_B(P) \times Y_B(P), \\ card(X_A(P)) &= card(X_B(P)), \\ card(Y_A(P)) &= card(Y_B(P)). \end{aligned} \tag{3.2}$$

This only states that transformation of coordinates is a *function* from Cartesian product of two sets/ranges to another Cartesian product of two sets/ranges where sets for x co-

ordinate respective for y coordinate are exactly the same size (the number of pixels from each image in their common overlap must be obviously the same).

Valid pixel coordinates are discrete (natural numbers). Depending on the transformation of coordinates it may occur that for certain positions in reference image the computed position won't be natural number. In such cases the value of pixel at computed position cannot be retrieved directly from the image. Image re-sampling method must be used. In image re-sampling in general three different approaches can be applied, namely: The nearest neighbourhood, the bilinear interpolation and the cubic convolution. They possess their own characteristics but mainly vary in computational time and precision.

3.1.2 Transformations

The type (or class) of transformation of coordinates is closely related to the spatial distortion present during image acquisition. See [5] or preferably [3] for more detailed discussion about transformation of coordinates and its use in image composition.

We will mention here only the so-called global transformations. The property of this transformation is that there is exactly one transformation formalism for every point in the overlap. Local transformation on the other hand have separate transformation formalisms for each local area in the overlap. For example lower part of the overlap is spatially distorted differently than the upper part. It is impossible to describe such behaviour using only one formalism without any *if* conditions present in the underlying formula. Hence it can be stated that global transformations of coordinates have no *if* conditions in their formalisms. The most used global transformations in 2D are linear, affine, projective and polynomial.

Linear and affine transformation

Transformation of coordinates T is linear if and only if

$$T(x_1, y_1, P) + T(x_2, y_2, P) = T(x_1 + x_2, y_1 + y_2, P) \tag{3.3}$$

$$c \cdot T(x_1, y_1, P) = T(c \cdot x_1, c \cdot y_1, P) \tag{3.4}$$

and T is affine if $T(x, y, P) - T(0, 0, P)$ is linear. Affine transformation for equation

$$(x_B, y_B) = T(x_A, y_A, \{c_{11}, c_{12}, c_{13}, c_{21}, c_{22}, c_{23}\}) \tag{3.5}$$

can be rewritten as

$$\begin{pmatrix} x_B \\ y_B \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \begin{pmatrix} x_A \\ y_A \end{pmatrix} + \begin{pmatrix} c_{13} \\ c_{23} \end{pmatrix}. \tag{3.6}$$

This transformation contains all basic 2D operations such as shift the position by constant c_{13} at x coordinate, by constant c_{23} at y coordinate and s-times longer the distance of given point from the origin while turning this point anti-clockwise around the origin by angle ϕ. The latter is encoded in the following equations

$$\begin{aligned} c_{11} &= s \cdot \cos(\phi), & c_{12} &= -s \cdot \sin(\phi), \\ c_{21} &= s \cdot \sin(\phi), & c_{22} &= s \cdot \cos(\phi). \end{aligned}$$

All geometric shapes are preserved, for example triangles remain triangles after the affine transformation.

We will skip rewriting the formal notation for all next transformation of coordinates. They all will be similar to the equation 3.5 except the number of constants in parameter P. We will only describe equations which define the result of $T(x, y, P)$.

Projective transformation

The projective transformation is designed to "turn" the plane of registered image such way that both planes of registered and reference images face the same direction. This often happens when scanning same objects from different view ports, it is a special case of image composition. Typical case of image composition is stitching images into a single one. Hence this transformation is employed in converse cases when scanning adjacent images from the very same point. This situation can be imagined as taking pictures of a sphere from its centre. Manipulation of image planes either reference and/or registered is called rectification. When creating a composition of more than 2 images then all images should be rectified so their image planes are tilted the same direction. This is necessary both for searching the correct alignment and for final image stitching. For more details about this topic, please refer to [3, 7, 21] or alternatively [10].

The projective transformation is a 3D thing in spite of the fact that images are 2D themselves. We must therefore know in advance (or find it out, see [21]) the relation between these two images in term of Euler angles. Euler angles are three angles α, β and γ each corresponding to its axis x, y and z, resp. This says turn the given plane by α degrees around the x axis and then do the similar thing around the y and z axis. This behaviour can be well summed up in the following rotation matrix

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \tag{3.7}$$

in which the following equations hold

$$\begin{aligned}
c_{11} &= \cos(\beta)\cos(\gamma), & c_{13} &= \sin(\beta), \\
c_{12} &= -\cos(\beta)\sin(\gamma), & & \\
c_{21} &= \sin(\alpha)\sin(\beta)\cos(\gamma) + \cos(\alpha)\sin(\gamma), & c_{23} &= -\sin(\alpha)\cos(\beta), \\
c_{22} &= -\sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma), & & \\
c_{31} &= -\cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma), & c_{33} &= \cos(\alpha)\cos(\beta), \\
c_{32} &= \cos(\alpha)\sin(\beta)\sin(\gamma) + \sin(\alpha)\cos(\gamma). & &
\end{aligned}$$

In the given scanned image it is assumed that all points in it are in equal distance from the camera. Hence we can expand the image coordinates into 3D by putting the z_1 coordinate equal to 1 in the equation 3.7. Imagine we are in 3D space. The camera sensor is in the origin and the image is in plane perpendicular to axis z at the distance of 1. We want to be in the very same situation in 3D space but this time with the new sensor in the origin (with that sensor for which the rectification is processed). The rotation matrix will turn the image plane so it is tilted the requested direction. To get the final image we must deal with perspective. That will bring the coordinates back to 2D by fixing the z coordinate

to 1 and recomputing the x and y coordinates according to the scheme

$$\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} \quad \rightarrow \quad \begin{pmatrix} x_3 \\ y_3 \\ 1 \end{pmatrix}$$
$$\text{by setting} \quad x_3 = \frac{x_2}{z_2}, y_3 = \frac{y_2}{z_2}.$$

The projective transformation is computed using these two equations

$$x_B = \frac{c_{11}x_A + c_{12}y_A + c_{13}}{c_{31}x_A + c_{32}y_A + c_{33}}, \tag{3.8}$$

$$y_B = \frac{c_{21}x_A + c_{22}y_A + c_{23}}{c_{31}x_A + c_{32}y_A + c_{33}}. \tag{3.9}$$

Polynomial transformation

The last case, polynomial transformation of coordinates, is used for static spatial distortions where a general transformation is needed. It is usually formalised using bivariate polynomial transformation

$$x_B = \sum_{i=0}^{m}\sum_{j=0}^{i} c_{ij}x_A^i y_A^{j-i}, \tag{3.10}$$

$$y_B = \sum_{i=0}^{m}\sum_{j=0}^{i} d_{ij}x_A^i y_A^{j-i} \tag{3.11}$$

where c_{ij} and d_{ij} are constants defining certain transformation. As can be seen from the equations the capability to model the distortion is controlled by the order of polynomial m. The higher m the more precise such transformation can be and also more complex to compute. Not to mention how to construct such transformation (define constants) while polynomials in higher orders can be sometimes too tricky to handle.

There exists also others more complicated transformations of coordinates. These are mostly designed for particular distortions. They differ in their expressibility power, computation burden and mainly in parameter space (it is the size of structure which is behind the letter P in our notation). Generally holds that the more complicated distortion the given transformation can handle the bigger parameter space it has. Next section will cover basic approaches for searching the parameter space.

3.1.3 Basic principles of matching

There exists several matching techniques. These can be divided into two groups. The first group we will call for convenience the feature-based techniques while the second group we will call the statistic-based techniques. Both groups are searching the parameter space of an a-priory known transformation of coordinates although it may not seem to.

The feature-based techniques group represents a heuristic approach. These techniques are trying to find the alignment straight from the data using some known property. The input is overlap of reference and registered images, it is pre-processed trying to extract

or emphasise some feature which should be common to both images. This feature is then mapped from the reference image to the registered. The mapping is described with alignment (transformation with given P) which is the result. In this group the image preprocessing is nothing rare, sometimes even special marks are inserted into the images.

The techniques found in the second group usually can work straight on the raw data. This group represents a statistical approach. These techniques are trying all possible alignments, each alignment is then evaluated. Evaluations are compared while trying all possible alignments, this is performed until the best alignment is found. The evaluation should describe how appropriate this alignment is, so the resulting alignment is the one which describes the most appropriate case. The definition of alignment evaluation will be described later in section 3.3.

Feature-based techniques

Each approach has its own pitfalls. Feature-based techniques are usually two step processes. First such technique must extract some feature from the raw data. It is usually not an easy task, mainly when contrast of image is low. The image resolution is also a crucial factor in this case. When for example control points are inserted into a scanned sample then low resolution scan may picture each control point using a few pixels. This point can easily get lost among the information presented and expressed in surrounding pixels. A solution may be in enlarging the control point in the sample or changing its colour so it becomes more different even in the pour contrast scan. Pour contrast is bad for segmentation methods where extracting given feature may become impossible for reliable automatic image composition. In such cases a human interaction may help. The human operator will show the computer the feature in the image and the computer will continue with extracting the feature or continue directly with the second step.

Either way the information extracted should be accurate because the second step will continue with this information and eventual errors will accumulate. For example if the second step is computing the gravity centres of segmented control points then accuracy in segmentation should be high. It depends very much on the borders of points. If the segmentation process failed to extract these control points exactly with their borders then the variation present in the shape of points (the difference between segmented points against their original shape) would have unpleasant impact on the resulting gravity centre. This would result in slightly different alignment than the appropriate one.

The second step in feature-based techniques is estimating the alignment. This is managed depending on the kind of information extracted or on the image pre-processing made on the data from the first step. The second step algorithm should cope with little imprecision, sometimes it may happen due to this imprecision that there will exist more then one solutions. Program may log them for human operator leaving him/her to make the final decision. Usually the first step is more complicated than the second one.

These methods are usually aimed against the difficult distortions where the parameter space is very large to search it. Or sometimes adequate alignment valuation cannot be found and so the comparison between two alignment for which one is more suitable cannot be determined. Often approach is in re-sampling the whole data into smaller sizes (half dimension can be big help) where the extracted feature is clearly significant in the image and is the main object that should be aligned (the rest is background and is unimportant).

Statistic-based techniques

The statistic-based techniques can operate directly on the raw data. The operation in this context means computing some statistics of data (of the overlap) typically sample mean, variance or histogram. These statistics of the overlap from the reference image and of the overlap from the registered image are compared and the result is numerically expressed. These numbers serve as valuations of given alignment. The matching is then done through evaluating all possible alignments and the result is the alignment with the best (usually the highest) valuation.

These methods depend only on the valuation method employed. It doesn't have to be only the statistics of overlaps separately, it can be of course (and mostly is) the statistic computed from both overlaps together. Example: Evaluation of alignment expressed as the mean from differences in values of corresponding pixels. The trend is to create such statistics in which adjacent alignments have similar valuation and for which holds (without losing of generality) the higher valuation the better is the alignment.

This is good for optimisation techniques. Their task is to find the optimal alignment (represented by global maximum valuation) without going through the whole parameter space. They usually manage this as the next level processes which call the evaluation methods and then make decisions, sometimes they make use of special forms of given evaluation method to get results faster. For instance when optimising using a gradient descend technique (well, when higher valuation is the better one then this technique should be implemented and called gradient *ascend*) then we do not need to compute several valuations and from their differences decide the next step (estimating the value of derivation) but instead we can (if possible) compute directly the value of derivation of the given method.

These statistic-based techniques have good results for various image composing applications. They can handle low contrast as well as spatial distortions. They are mostly usable only on such distortions which can be handled by affine transformation of coordinates. Sometimes the image pre-processing is needed, for instance in cases where the background fools the evaluation. In such cases it also might help switching to different evaluation method. Significant help can be again achieved by re-sampling input images into smaller ones. The reason is simple, even if evaluation methods are often computationly simple the major factor influencing the speed of matching is the amount of data that has to be processed. When image matching is done in opposite to classic pattern searching the problem of data volume becomes more obtrusive while the bigger is the overlay the bigger is the amount of data that has to be evaluated *and* the bigger is the amount of possible alignments. So there is something like quasi-quadratic slow-down.

3.1.4 Examples of matching methods with various approach

Matching using control points

This technique finds distinctive points in the common overlap in both images and store them in two sets of points, each set corresponding to one image. These points can be either artificial (intrinsic), i.e. manually inserted into pictures acting as markers, or these can be extrinsic — relevant to the data itself. In the first case the shape of points is selected to be unique in the whole image and also the coordinates of these points should be easily extracted. Typical examples are wholes drilled into a scanned sample or in medical imaging

we can mention fudicial markers like plastic "N" shaped tube filled with $CuSO_4$ in MR scans. In the case of extrinsic markers these are selected to be again easily determined in the scan, to be rigid and stationary. Usually these are identifiable landmarks, anatomical structures or mainly easier objects like line intersections, corners or centres of gravity of closed-boundary objects.

The markers should generally be invariant to the distortions, for example having control points as small dark markers in distortion in which dark noise is present isn't probably the best choice. Sometimes artificial markers are placed on the sensor itself. Thus the matching process becomes closer to the calibration process. But markers should be still reasonably selected. Finally after creating two sets of control points these should contain similar number of elements.

Now when we have two sets of coordinates of control points we must find the certain values of P (find the certain alignment) at given transformation of coordinates. We assume that the given transformation is capable of describing the spatial distortion that is present during image scan. The following example algorithm for points mapping will be functional only for affine transformations.

We will need two-dimensional array, one dimension will represent points from one set and the second dimension will represent points from the remaining set. This array we will call the accumulator array and for this purpose we must number the control points in each set to make it a classical number indexed array. The first step is zero the whole accumulator array and then create every possible triangle from points from the first set and simultaneously from the second set. Compare each pair of triangles using the circularity measure. The circularity measure K is invariant to rotation, scaling and also to translation. It is defined as

$$K = \frac{2|c_{11} \cdot c_{22} - c_{12} \cdot c_{21}|}{c_{11}^2 + c_{12}^2 + c_{21}^2 + c_{22}^2} \tag{3.12}$$

where c_{11}, c_{12}, c_{21} and c_{22} are constants from given affine transformation which can be determined from the coordinates of points which are creating measured triangle pair. One triangle defines a circle, the circle will be inscribed into this triangle. This measure should describe how easily one can inscribe the very same circle into the second triangle. As the K gets closer to 1 the easier it is. Value of 1 means complete conformity.

Every element in the accumulator array increment by 1 if and only if its position in the accumulator array represents some pair of corresponding vertices of the triangle pair which in addition has $K > 0.9$. When this is done for all possible triangle pairs the next step is go through the accumulator array and zero all such elements which value is less then predefined threshold. The value of threshold should be first experimentally determined. This step should eliminate the possibility of mismatched pairing of control points. After this again scan through the accumulator array this time row by row zeroing all elements in the row except the one with the highest value, then do the same thing but column by column. The algorithm is finished leaving non-zero elements in the accumulator array at the coordinates which define control points that correspond one another.

This approach is typical representative of the feature-based group of matching techniques. The control points can be determined differently, that was very briefly explained in the paragraph beginning at the page 13.

Matching using correspondence of borders

This method begins with segmentation of objects from input images, the segmentation from common overlap. We must extract the frontiers of objects which are distortion invariant and present in both images. This frontiers must be remembered somehow and from this information we will get the certain values of parameter P (certain alignment).

One solution is to use the general Hough transformation for isolating the border. Usually each pixel is stored in Cartesian coordinates but this method requires to recompute them into polar coordinates, i.e. coordinate is defined through distance and angle from the given reference point and the axis. In image composition we can use the origin of coordinates as the reference point and the x axis as the reference axis/direction. The border in the reference image is stored pixel by pixel in dependence on the reference point, in the reference image this point is the origin. This way we get characteristic of given object in reference image. For the sake of this explanation let's assume that this characteristic is ordered somehow (usually by angle), thus we can number each pair (distance and angle) describing each pixel from that object.

Now we will use this characteristic from reference image and apply it on every pixel (respective its coordinate) in the border of the same object in registered image. This means for given pixel in registered image and for every element (distance and angle) of this characteristic compute the reference point in the registered image such that from this reference point we can get to the given pixel coordinate using given distance and angle. The positions of computed reference points are then used in accumulator array (which had to be initialised beforehand).

The accumulator array is two-dimensional, the position in accumulator array corresponds uniquely to some position in the registered image. So every computed reference point will increase some element in the accumulator array. The transformation of coordinates is in this example translation (every coordinate is shifted by some constant). The optimal alignment is expected to be that one which translates the origin from the reference image into the reference point in the registered image with the highest accumulated value.

There exist several other methods using similar principle, for instance the active contour models (also called snakes). Model can be a line that can curve itself depending on its parameters, splines are very popular in this context. This line is initiated inside some object and it is trying to get alongside this object. The shaping of the line is controlled via some equations. These equations preserve the smoothness of the line but these equations are under influence of the object. When the equality holds, the line is alongside the object as tight as its smoothness will permit. Advantage is in the reduction of information while each line (respectively the object in the image) can be well described using a few parameters. Few objects are described this way and finally some parameter comparison takes place returning the alignment.

Matching using Fourier transformation

Fourier methods pose different approach to solution of matching problem but still with similar characteristics. They do kind of statistical investigation of data (that's the similar characteristics) but they do it in the frequency domain (and that's something new). The idea comes from facts that spatial translation, rotation, scale or reflection have their counterparts in frequency domain.

Advantage is also in the speed when using fast Fourier transform and mainly in robustness against frequency dependent noise. Imagine satellite scanning of Earth and then transmitting analog data signal back to ground where secondary data processing is held including image composition. During transmission it can happen that some periodic noise will interfere the transmitted signal and introduce this way a frequency dependent noise in the retrieved image.

Favourite example for image matching using Fourier transform is phase correlation. This matching process can find parameters of translation as a transformation of coordinates. It relies on the translation property often called the shift theorem. Let us assume we have two overlaps of images. Following the established notation we will denote them $A(x_A, y_A)$ and $B(x_B, y_B)$ where x_A ranges from $X_A(P)$ and similarly $y_A \in Y_A(P)$, $x_B \in X_B(P)$ and $y_B \in Y_B(P)$. Remember overlap is dependent on the alignment which is defined via parameter P. For both functions A and B we have their Fourier counterparts $F_A(\omega_x, \omega_y)$ and $F_B(\omega_x, \omega_y)$. The shift theorem declares that the following two equations are equivalent

$$A(x, y) = B(x + c_1, y + c_2), \tag{3.13}$$

$$F_A(\omega_x, \omega_y) = e^{-i(\omega_x c_1 + \omega_y c_2)} F_B(\omega_x, \omega_y). \tag{3.14}$$

The equation 3.13 is almost the equation 3.1 (on page 9) with the difference that the transformation of coordinates symbol T was substituted with the translation in which the shift is done with vector (c_1, c_2). The second and final difference is the removal of intensity mapping function I. Under certain circumstances the I function could be left there, the phase correlation can be successfully applied on matching images which were obtained from different sensors (multi-modal image composition). The symbol i in equation 3.14 is the complex unit ($i^2 = -1$).

Equation 3.14 states that two images translated by some constant vector have the same magnitude but differ in the phase. The normalised cross power spectrum is hence given by

$$\frac{F_A(\omega_x, \omega_y) \cdot F_B^*(\omega_x, \omega_y)}{|F_A(\omega_x, \omega_y) \cdot F_B^*(\omega_x, \omega_y)|} = e^{-i(\omega_x c_1 + \omega_y c_2)} \tag{3.15}$$

where upper index * denotes the complex conjugate (when $z = a + ib$ then conjugate is $z^* = a - ib$). Now compute the inverse Fourier transform on left side of equation 3.15 and we get the Dirac δ-function.

Well, this with a few assumptions about integrability and so on is not applicable on images. That is to say images are discrete. Never mind we can still use discrete Fourier transforms with further assumptions, notably periodic extension of images outside their support sets $X_A(P), \ldots, Y_B(P)$. The Dirac δ-function then becomes a unit pulse. That should be localised at the coordinates (c_1, c_2). This is how we can determine the optimal alignment using Fourier transform.

Previous paragraphs were a small excursion into a whole variety of matching approaches and algorithms. All of the algorithms presented here are mostly extensible into more complicated transformations of coordinates. Nevertheless with speed penalty and also more complicated implementation. We have omitted typical members from statistic-based family of matching algorithms. We will mention the optical system in the following

section which was used for testing and then in the following section we will explain five matching methods that were implemented. The program which is part of this master degree thesis was constructed at first to fulfil the matching process which is outlined in the next section. It was decided to use statistic-based methods and so we will describe them later a little bit more detailed.

3.2 The system in Brno Faculty Hospital — Bohunice

The practical part of this master degree thesis is to develop a program that will successfully compose images. The test data were obtained at Department of pathological anatomy which is located at Brno Faculty Hospital — Bohunice. The department belongs to Faculty of Medicine, Masaryk University in Brno.

The images are scanned using the Lucia DI system (provided by Laboratory Imaging Ltd., Prague). This system controls the digital camera Nikon DXM 1200, microscope Leica DMLB with lens HC PIApo 10/0.4, HC PIApo 20/0.7, HCX PIApo 40/0.85 CORR, HCX PIApo 100/1.35 Oil Imm and finally the scanning table Märzhäuser 2D. The high resolution images are first scanned part after part in meanderic manner, i.e. begin with left top image, scan it, move right, ..., scan the rightmost image, move down, scan it, move left, ..., scan the leftmost image, move down, scan it, move right and so on. Images are scanned with overlap, its volume can be roughly controlled via the percentage of overlap from the whole image. The Lucia DI system automatically takes care of focusing each image. When 3D reconstruction is needed then this system takes a few images at different focus distances, then compose them together using mainly contrast parts from images. This way after "a longer while" we receive serie of images, each image is typically 1232×972 pixels in 24-bit colour depth. Image file format is `TIFF` not-compressed, i.e. the size of one file is approximately 3.4MB. The scan batch is then described using configuration file and composed together. The routine of image composition in Bohunice is also described in [6], input file formats are described in [15, 9].

From this characterisation it is clear that the problem of composing given images is uni-modal. The system is not much influenced by the ambient light, furthermore the system is situated in special room which is without windows and hence dark during the scanning process. Thus the illumination variation is very low if any. The problem arise with contrast. The cause is the strength of scanned sample which is not constant in the whole sample. There exist several peaks depending on the kind and texture of the sample. The system tends from time to time to focus on these peaks resulting in pour quality image (the rest of image is majority of image and it is not focused). The system tries to recognise such situations and asks the operator whether it should start the mentioned 3D reconstruction. There are usually a few images in each image composition with slightly different focus distance, so the contrast in images is variating. One won't usually notice it when looking at the zoom out composition. For these reasons the intensity conversion function was supposed to be identity and therefore was excluded from any computations.

The scanning table is very accurate. Hence the translation was good enough to serve as the transformation of coordinates. The accuracy of scanning table was a big help when employing the optimisation techniques in the final level of program development.

Spatial distortions are present mainly due to lens properties. When two images are

aligned by the upper part of their common overlap then the lower part is misaligned by a few (less then ten) pixels. The overlap is very large, usually from 5% to 10%, i.e. in horizontal matching the overlap dimension is from 61×972 to 123×972 pixels. Such amount gives enough information for optimal alignment even when the ideal alignment isn't possible. The misalignment and sometimes the contrast difference cause the sutura to be noticeable. The problem is solved using the sutura camouflaging routines which is run when the image stitching is in progress.

It was decided to use statistic-based matching methods because of the nature of biomedical images. These contain tiny parts of human skin or organs, lots of cell with different textures and plenty of inter-cell space filled differently. We believe that any kind of segmentation when fully automatic composition is requested won't be at least manageable if possible. The probability of misbehaving of matching method which makes use of segmentation was too high. In addition the spatial distortion isn't too bad and the variance in illumination and contrast is acceptable for computing basic statistics directly on raw data. Latter in Chapter 4 we will see that this approach paid off.

3.3 Matching evaluation methods

3.3.1 Statistic-based approach, revisited

In the previous section we have learnt about the specification of given image composition problem. We have learnt that the transformation of coordinates $T(x, y, P)$ will be simply the translation and that the intensity conversion function is the identity. Thus the basic equation for alignment 3.1 (on page 9) will be rewritten into

$$A(x, y) = B(x + c_1, y + c_2). \tag{3.16}$$

We have dropped the I function because we've assigned it to be the identity function and the semantics of the identity function allow us to do that. Just to make clear, under the term identity function we mean the following scheme for any arbitrary set Set:

$$\begin{aligned} I &: Set \to Set, \\ I(s) &= s, \qquad \forall s \in Set. \end{aligned}$$

We are aiming to use the matching methods from the statistic-based family. According to its definition we must firstly define the alignment evaluation function. This function will accept the shift constants c_1 and c_2 as its input and will result in some number val. The domain of val will be various but always it will be some basic mathematical set of numbers (or its sub-set), mainly it will be natural or real numbers. This have the property that we can always compare two such numbers and we can always say which one is larger, that is very important for our purpose. We will define some certain valuation functions in this section and denote them according to their names. But for these paragraphs we will denote them with some general symbol V (abbreviation for term *valuation*). Let the symbols $Align_1$ denote the alignment given by c_1 and c_2 and similarly $Align_2$ for c_1' and c_2'.

The equation 3.17 is crucial for the matching process

$$V(c_1, c_2) > V(c_1', c_2') \quad \Longleftrightarrow \quad Align_1 \text{ is closer to the optimal then } Align_2 \tag{3.17}$$

and must be held for every valuation V in order to let the matching process be successful. Whenever the alignment evaluation function returns higher number then the more suitable alignment we have found. Hence the optimal alignment written as a pair (C_1, C_2) should satisfy the property

$$(C_1, C_2) = \max_{(c_1,c_2)\in P'} V(c_1, c_2). \tag{3.18}$$

The P' denotes the parameter space of all possible alignments.

We use here a quite general notation because behind the symbolics there is a lot of work and also lot of relations. For example we state here that the matching is *just* finding the maximum evaluation. Obviously we can stitch two images using only one alignment (i.e. using only one transformation of coordinates), because we are reasoning about the global transformations of coordinates. Hence we preclude stitching the upper part of two images using some alignment and simultaneously stitching the lower part using some other alignment. But what shall we do when there will be two (or more) alignments having both the maximum valuation? In practical solutions we will probably also have to work out the P' searching for the maximum/optimal.

For the sake of explanation and introducing the notation we had to simplify. There are some relations behind the property 3.18. The first and the most remarkable is that virtually everything is depending on the two matching images, the reference image $A(x, y)$ and the registered one $B(x, y)$. These two define the general parameter space for P (P is an element from that space) in general matching problem and the translation parameter space P' in Bohunice's case. That's a little bit tricky while in fact there are two things behind the term overlap.

The image composition was introduced as an image matching and then an image stitching process. The matching mostly implies the overlap of images. The system in Bohunice can control how much the overlap will be in percentage from the whole image. This is the first meaning of overlap and in fact it defines the largest possible overlap of images, the optimal one can be (and usually is) smaller. We use this information for creating the P' — the space of all possible alignments regarding given two images. But images can generally be of different dimensions (even the reference and registered images need not to have the same dimensions). This is why the P' is dependent on the images $A(x, y)$ and $B(x, y)$.

Then the second meaning of overlap is the actual overlap when some alignment is given. This means such part of reference image in which every pixel has its own counterpart in the registered image (and vice versa). This is dependent on the exact values of P and the given transformation of coordinates as outlined on page 9.

So it looks like the overlaps and alignments are related in a circle-definition, overlap depend on given alignment P which is from all alignments space P' which is again dependent on the overlap. But that's not true because of the duality of the term overlap in our context. We will use the ranges $X_A(P)$, $Y_A(P)$, $X_B(P)$, and $Y_B(P)$ which define the intervals for the actual overlap at given alignment. From this point on we will note the alignment by pair/vector (c_x, c_y). This defines some certain alignment from the space of all possible alignments P', but we won't need the notation for space P'.

Finally the valuation V is dependent on the given alignment and the given images because it needs to compute the actual overlap and estimate the "optimality" of it. Exact notation would require function nesting resulting in plenty of parentheses or indices not

mentioning the definitions of domains which are again dependent.

It is not hard to figure out how to compute all possible overlaps (the second meaning) from given two images (respective their dimensions) and the overlap (the first meaning) in percentage. For review of 3.16 the pixels at alignment (c_x, c_y) are in correspondence

$$A(x,y) = B(x+c_x, y+c_y) \tag{3.19}$$
$$\text{where } x \in X_A(c_x,c_y), \quad y \in Y_A(c_x,c_y) \tag{3.20}$$
$$\text{and } x+c_x \in X_B(c_x,c_y), \quad y+c_y \in Y_B(c_x,c_y). \tag{3.21}$$

The pair (c_x, c_y) replaces the general alignment defined previously by parameter P.

Acknowledgements at this point belong to [4, 14] for good help at remainding me the basics of statistics. The summary of selected valuation methods can be also found in [5].

The n-pass test is every such test that needs for its evaluation exactly n-times examine all the data from given overlap.

3.3.2 The stochastic sign change

The stochastic sign change (abbreviated to SSC) is the first and the easiest test for similarity. It works on the raw data without any pre-computations. It is simply one-pass test which is counting the changes of sign in the series. The design of this method presumes that the more sign changes there'll be the more similar these two images are in their common (tested) overlap.

The series consists of differences from values of corresponding pixels. The sign change is considered as every such situation in series in which the sign of predecessor turns into the sign of successor in one of the following manners:

1. change from strictly positive (above zero) to zero or less,
2. change from strictly negative (bellow zero) to zero or more,
3. "change" from zero to zero.

These situations don't correspond exactly to the name of the test, foremost because of the third option. The reason for this is the situation in which we are evaluating two exactly same overlaps. In this situation the difference from values of corresponding pixels will be always equal to 0. How much sign changes will be there? Obviously there will be no sign change but in most others alignments there will be at least one pair of corresponding pixels that will differ more or equally to 1, except the matching of two exactly one-coloured images. This would imply that mostly every non-optimal alignment will yield more sign changes then the optimal one. Thus the optimal alignment will never be found in such situations. For situations like this the SSC was enhanced preserving the situations in series where the exact sign change occur.

The "enhanced" SSC is defined using the help series $ssc(i, c_x, c_y)$ for given alignment (c_x, c_y) as

$$ssc(i,c_x,c_y) = A(x,y) - B(x+c_x, y+c_y), \quad i = 1 \ldots N, \tag{3.22}$$
$$\text{where } x = i \,\%\, card(X_A(c_x,c_y)) + \min(X_A(c_x,c_y)), \tag{3.23}$$
$$y = i \div card(X_A(c_x,c_y)) + \min(Y_A(c_x,c_y)), \tag{3.24}$$
$$N = card(X_A(c_x,c_y)) \cdot card(Y_A(c_x,c_y)). \tag{3.25}$$

The N is the total amount of pixels in overlap of images $A(x, y)$ and $B(x, y)$ at given alignment (c_x, c_y). It is computed from the cardinality (size) of range intervals of permitted (or equivalently of presented) coordinates in overlap. The % stands for modulo operation (the reminder after dividing) and the $\div$ stands for dividing without reminder.

The equations 3.23 and 3.24 summarise few implicit facts. First the range interval either for x or y coordinate is "discretely continuous," i.e. for example in range of natural numbers from 1 to 10 there are no numbers missing (no holes), the sequence is complete. Secondly the numbers in range interval can be linearly ordered and therefore we can determine the minimal number in every such interval. In these equations we have presented that we can uniquely number all pixels present in overlap, start from bottom left corner of the overlap and assigning increasing numbers while moving "as we read" but up.

The series $ssc(i, c_x, c_y)$ can be created variously, it really doesn't matter in what order we are inserting the values of pixel pairs into the series. It doesn't even matter in what order we are computing the differences, we can even switch the order during evaluation of given alignment. Important is when image matching that orders either of inserting values or computing differences are preserved and therefore the very same for every valuated alignment.

Next equation will be finally the definition of the stochastic sign change measure of similarity

$$V(c_x, c_y) \equiv SSC(c_x, c_y) = \frac{1}{N} card(\{i|\ ssc(i-1, c_x, c_y) \cdot ssc(i, c_x, c_y) \leq 0, i = 2 \dots N\}). \tag{3.26}$$

From this point we will drop the obvious prefix $V(c_x, c_y) \equiv$.

In the definition of SSC there is denominator (the fraction $\frac{1}{N}$) which provides kind of normalisation. The reason for that is again simple. The bigger is the overlap then it can possible more often occur a sign change. If the optimal alignment represents small overlap then misalignment can easily happen and in fact in practical testing pretty often happened. Denominator proved himself to be a good solution. Disadvantage of this solution is in extremely small overlaps where the strength is less then 5 pixels (recall that the system in Bohunice creates overlaps with the strength usually more than 50 pixels). At these small overlaps the number of sign changes in combination with denominator smaller then usual tends to raise the evaluation. Put in other way, at large overlaps the number of sign changes in combination with significantly higher denominator tends to lower the evaluation. In praxis the result was always misalignment until the small overlaps were simply forbidden (such alignments were excluded from the searched alignment space).

The domain of possible evaluations is $\langle 0, 1\rangle$. This method presumes that the reference and registered images are identical except for non-correlated, additive noise with the zero mean value and symmetric probability density function. This is clear assumption when looking at the definition of SSC. Moreover the bigger difference in brightness of matching pictures the less sign changes will occur (provided the order when computing the differences is not alternating) because the values of paired pixels will draw apart. This way the sign changes will occur sparsely and their expressibility in term of similarity will become more poor. Finally at some level of difference in brightness the SSC will lose completely the ability to point out optimal alignment. The similar will happen when the noise won't be centred at zero value.

3.3.3 The sum of absolute valued differences

This evaluation (abbreviation is SAVD) seems to have its root in the least-square criterion. The similarity will be more distinct after the definition. The definition of SAVD is

$$SAVD(c_x, c_y) = MAX - \frac{1}{N}\sum_{i=1}^{N} |A(x,y) - B(x+c_x, y+c_y)|. \tag{3.27}$$

The definition of variables x, y and N holds from equations 3.23, 3.24, and 3.25.

The MAX value is here only for "cosmetic" purposes. The subtrahend is the core of SAVD method. To match our definition for matching as finding the alignment with *maximum* evaluation we had to equip the SAVD valuation core (the subtrahend) with the minus sign. The best alignment is then achieved when the sum is equal to value 0. Any sub-optimal alignment will differ at some point (pixel pair) resulting in strictly above zero value of the sum. In the end "after minus sign" the SAVD will behave the way we want to. Constant MAX is then simply the shift to positive values of valuation and should be set to the maximal possible value of pixels (typically to number 255 for 8-bit colours).

As mentioned the $SAVD$ suggests the least-square criterion which is very popular measure of similarity in computer science. The square function which purpose is in fact to turn the negative values into positive ones is substituted with the absolute value function. The advantage after this exchange is less sensitivity to outliers — corresponding pixel pairs which differ notably more then the rest. This improves the similarity evaluation when the noise is present provided the noise won't over-buzz the image itself (the image won't be over-much noisy). The noise won't affect the total sum as much as it would affect in least-square criterion leaving this way the noise-free pixels to control the total value of the sum. This is in correspondence with our expectation that the ratio "pixels influenced by noise against the noise-free ones" is far less than one. The subtract is again divided by the overlap size for the same reasons as in the stochastic sign change criterion. The narrow overlap restriction holds.

This method's domain of possible evaluations is $\langle 0, MAX\rangle$. It is slightly faster than SSC and more reliable too. In the meantime it is the main method used in Bohunice because its speed, desirable properties for further optimisation techniques and reliability. The matched images should be identical, small variance in brightness and noise is acceptable, i.e. the matching shouldn't fail. The tested amount of noise wasn't *really* much but it was enough to consider these images to be useless, see Appendix for example pictures.

3.3.4 The normalised cross-correlation coefficient

Also known as Pearson r-coefficient sometimes also referred as linear or product-moment correlation. It is also very basic similarity measure which can be found in every literature about statistics, helpful was [4], the "alias" terms come from [14], technical help came from [2]. This is a two-pass evaluation.

The definition is

$$A^*(c_x, c_y) \quad = \quad \frac{1}{N}\sum_{i=0}^{N} A(x,y), \tag{3.28}$$

$$B^*(c_x, c_y) = \frac{1}{N}\sum_{i=0}^{N} B(x + c_x, y + c_y), \tag{3.29}$$

$$NCC'(c_x, c_y) = \frac{100\sum_{i=0}^{N}(A(x,y) - A^*(c_x,c_y))(B(x+c_x, y+c_y) - B^*(c_x,c_y))}{\sqrt{\sum_{i=0}^{N}(A(x,y) - A^*(c_x,c_y))^2 \sum_{i=0}^{N}(B(x+c_x, y+c_y) - B^*(c_x,c_y))^2}}. \tag{3.30}$$

A little confusing but yet the equations 3.23, 3.24, and 3.25 hold, definition 3.28 and 3.29 specify the sample means (average) from the given overlap. The number 100 is there only to enlarge the domain of possible evaluations.

Nevertheless there is an apostrophe in the name NCC' which means that this is not the final definition. Given definition involves two-pass computation, in the first pass we must compute the average values and then in the second pass we can estimate the optimality of alignment. This takes time. There exists one-pass variant which allows us to save time for computation, on the other hand it also introduces some computation inaccuracy. Probably depending on the implementation but the announced inaccuracy hasn't shown us to be a crucial problem. There was some but the differences it created were very decent and for the matching and optimising unimportant.

How to get one-pass algorithm? First notice that $A^*(c_x, c_y)$ and $B^*(c_x, c_y)$ are constants in equation 3.30, using this information we can rewrite similar sums

$$\begin{aligned}
\sum_{j=1}^{N}(a_j - A)(b_j - B) &= \sum_{j=1}^{N}(a_j b_j - a_j B - A b_j + AB) = \\
&= \sum_{j=1}^{N} a_j b_j - \sum_{j=1}^{N} a_j B - \sum_{j=1}^{N} A b_j + \sum_{j=1}^{N} AB = \\
&= \sum_{j=1}^{N} a_j b_j - B\sum_{j=1}^{N} a_j - A\sum_{j=1}^{N} b_j + ABN, \qquad (3.31) \\
\sum_{j=1}^{N}(a_j - A)^2 &= \sum_{j=1}^{N}(a_j^2 - 2a_j A + A^2) = \sum_{j=1}^{N} a_j^2 - \sum_{j=1}^{N} 2a_j A + \sum_{j=1}^{N} A^2 = \\
&= \sum_{j=1}^{N} a_j^2 - 2A\sum_{j=1}^{N} a_j + NA^2 \qquad (3.32)
\end{aligned}$$

where the A,B are constants and a_j,b_j are variables. Constants represents the averages from both sides (images) of overlap, hence for averages hold

$$A = \frac{\sum_{j=1}^{N} a_j}{N}, \qquad B = \frac{\sum_{j=1}^{N} b_j}{N}.$$

These two equations put into the equations 3.31 and 3.32, we get

$$\begin{aligned}
\sum_{j=1}^{N}(a_j - A)(b_j - B) &= \sum_{j=1}^{N} a_j b_j - B\sum_{j=1}^{N} a_j - A\sum_{j=1}^{N} b_j + A\frac{\sum_{j=1}^{N} b_j}{N}N = \\
&= \sum_{j=1}^{N} a_j b_j - B\sum_{j=1}^{N} a_j = \sum_{j=1}^{N} a_j b_j - A\sum_{j=1}^{N} b_j, \qquad (3.33)
\end{aligned}$$

$$\sum_{j=1}^{N}(a_j - A)^2 \quad = \quad \sum_{j=1}^{N} a_j^2 - 2A\sum_{j=1}^{N} a_j + N\frac{\sum_{j=1}^{N} a_j}{N}A = \sum_{j=1}^{N} a_j^2 - A\sum_{j=1}^{N} a_j. \tag{3.34}$$

In equation 3.33 we can choose one from two equal forms of numerator of the final normalised correlation coefficient. We expect the similar image quality at both sides of overlap and therefore we can make choice arbitrary.

Now the one-pass algorithm for NCC must compute following five sums

$$S_1 = \sum_{i=0}^{N} A(x,y)B(x+c_x, y+c_y), \quad S_2 = \sum_{i=0}^{N} A(x,y), \quad S_3 = \sum_{i=0}^{N} B(x+c_x, y+c_y),$$

$$S_4 = \sum_{i=0}^{N} A(x,y)^2, \quad S_5 = \sum_{i=0}^{N} B(x+c_x, y+c_y)^2.$$

The averages can be computed from $A^*(c_x, c_y) = S_2/N$ and $B^*(c_x, c_y) = S_3/N$, then

$$NCC(c_x, c_y) = (S_1 - B^*(c_x, c_y)S_2) \,/\, \left(\frac{\sqrt{S_4 - A^*(c_x, c_y)S_2}}{10} \cdot \frac{\sqrt{S_5 - B^*(c_x, c_y)S_3}}{10} \right). \tag{3.35}$$

This is the definition of one-pass NCC, the range of possible evaluations is $\langle -100, 100 \rangle$. The optimal alignment is reached for the 100.

This evaluation measures the extent to which the values of corresponding pixels are "proportional" to one another. The term proportional in this context means linearly related. The higher is the NCC the better can be every pixel pair from overlap described with equation

$$A(x,y) = d_1 \cdot B(x+c_x, y+c_y) + d_2. \tag{3.36}$$

while using only single value for each constant d_1 and d_2 (constants are independent on the position in the overlap). This means that the NCC should work when additive noise is present with expectation of d_2 and symmetric probability density function. Multiplicative noise with expectation of d_1 should cause no problems too. In Bohunice we have the situation $d_1 \approx 1$ and $d_2 \approx 0$, i.e. image brightness is roughly identical. Nevertheless when for example the left side of sensor brings more brightness into images than the right part, the result is the different brightness when image matching. The NCC evaluation should still find the optimal alignment in such cases.

The previous three evaluation methods were "exact-colour sensitive." These were estimating their view of similarity at given overlap by strictly comparing corresponding pixel pairs from given overlap. Images therefore had to be colour similar in the first place. This means that some object from overlap had to have the same texture in term of its relief and colour design. Selected three methods perfectly present three steps of accepting the difference in image illumination. The SSC must have identical both images which should be matched, low noise and no difference in brightness. The SAVD accepts low noise and small difference in brightness. By the difference in brightness we mean that there is some difference in values of pixel pairs and that this difference remains constant in the whole

overlap, pixel values from one image are all shifted constantly. In other words the pixel value mapping function from values in $A(x, y)$ into values from $B(x + c_x, y + c_y)$ should be simply the constant shift. The NCC may have than similar certificate as SAVD except the difference in pixel pairs which doesn't have to be necessary the same but still the pixel value mapping function must be linear. This features pre-determine the SSC, SAVD and NCC for uni-modal image composition respective uni-modal matching when the intensity conversion function I (see equation 3.1 on page 9) is considered to be the identity.

3.3.5 The correlation ratio

The correlation ratio (abbreviation is CR) is the first representative of statistic-based approach which is indeed computing the basic statistics variables during evaluation. This concept will allow us to use such evaluation methods for multi-modal image composition *without* the need for estimating the intensity conversion function. In addition to previous paragraph this evaluation method further extends the possibility of NCC. While the NCC can detect only linear dependency of pixel values from overlap, the CR is capable to detect arbitrary functional dependency, i.e. the mentioned pixel value mapping function can be arbitrary.

The correlation ratio evaluation is the main theme of [13], it is there compared to the Woods criterion and also to the mutual information measure which will be described in the next subsection. The [12] is in fact a report of the [13] concerning the image registration, this report suggests the CR straight for multi-modal problems.

The idea behind this alignment evaluation function is firstly estimate the "dependence" of the registered image on the reference image and then secondly quantify this dependence. The term dependence means in fact the pixel value mapping function. We would like to find the function ψ^* which satisfies the formula

$$\psi^* = \min_{\psi} Var[B(x + c_x, y + c_y) - \psi(A(x, y))] \tag{3.37}$$

in which ψ is the pixel value mapping function, the variables x and y ranges from $X_A(c_x, c_y)$ and $Y_A(c_x, c_y)$ (see 3.2 on page 9) in dependence on the given alignment (c_x, c_y), Var is symbol for statistic variance (the second order central moment). The minimum goes over all possible pixel value mapping functions.

Let's denote with J the set of only all possible pixel values, typically $J = \{1, \ldots, 255\}$. Then for the function ψ it holds $\psi : J \to J$. The theory gives us the result such that

$$\psi^*(J) = E[B(x + c_x, y + c_y)|A(x, y)], \tag{3.38}$$

$$\psi^*(j) = \sum_{i \in J} (i \cdot p(i|j)) \tag{3.39}$$

where E is the conditional expectation statistics (the first order initial moment) and $p()$ is the conditional probability function. Variables x and y range again over the entire overlap area. Equation 3.38 states what's the meaning/idea of equation 3.39. The pixel value mapping function which represents the best dependency is the expectation of the conditional probability function. The argument of $\psi^*(j)$ is the condition at which the probability is given. It is the value of the pixel in the reference image.

So this is the first part, we have the dependency estimate. Now we must quantify how appropriate it is in term of optimality of alignment. We already know that the ψ^* is the best dependency estimate for the current alignment situation. We will start with the total variance theorem

$$\begin{aligned} Var[Y] &= Var[E[Y|X]] + E_X[Var[Y|X=x]] && (3.40) \\ \text{where } \forall(\phi : J \to J) : E_X[\phi] &= \sum_{j \in J}(\phi(j) \cdot p(j)). && (3.41) \end{aligned}$$

Symbols Y and X are the discrete random variables ranging from J and representing both sides of overlap, E_X is just help operator.

We can think of the total variance theorem (equation 3.40 is the theorem, 3.41 is help to support the theorem) as of the energy conservation function. The first term $Var[E[Y|X]]$ measures the part of Y which is predicted by X, the second term measures the part of Y which is functionally independent of X. The total variance theorem is similar to the orthogonality principle

$$Var[Y] = Var[E[Y|X]] + Var[Y - E[Y|X]]. \tag{3.42}$$

The second term in the right hand side from equal sign is equal to the "functional independent" energy and is also a part of the equation 3.37. This should clarify the connection between the first and the second part of the idea behind the CR evaluation.

The correlation ratio is defined as the comparison between the "total energy" and the "explained energy" of Y. It could possibly be also defined only as the "explained energy" but this concept would introduce similar problems as were before the normalisation of SSC and SAVD.

From the orthogonality principle we can notice that $Var[Y - E[Y|X]]$ can be low for two reasons. Firstly the Y is well "explained" by X and therefore there is not much left for this term to fill up the $Var[Y]$. Or secondly Y itself gives little information, i.e. $Var[Y]$ is low. The second reason is a problem, $Var[Y]$ may be arbitrary low depending on the overlap volume. The "normalisation" helped us again and hence we define the correlation ratio for two random variables X and Y as

$$\eta(Y|X) = \frac{Var[E[Y|X]]}{Var[Y]} \tag{3.43}$$

or equivalently

$$1 - \eta(Y|X) = \frac{E_X[Var[Y|X=x]]}{Var[Y]}. \tag{3.44}$$

Notice that the correlation ratio is not symmetric. Because of the role of *functional* dependency in the concept of CR. Simply imagine we have the pixel value mapping function in which two distinct values of pixels from the reference image are mapped onto one single value of pixel from registered image. How will the inverse function look like? There won't exist any but the expectation of variance of Y through values of $X = x$ will overcome this, the penalty is the higher variance which may result in different evaluation of given alignment. Therefore when matching two images the "dependency direction" must be preserved for all tested alignments.

The correlation ratio can be computed in one-and-one pass, which means firstly scan once the whole overlap area and compute the conditional variance and the total variance of the pixel values of registered image $B(x, y)$, the second pass is for evaluation of the E_X operator using conditional statistics. We have explicitly mentioned this second step to highlight the difference of this final step between NCC and CR. The NCC have also a final computation after scanning the entire overlap but this computation is rather simple and does not consume a lot of time. The CR on the other hand needs to compute with every possible pixel value (resulting in a cycle when programming), we have denoted a set J containing only all possible pixel values.

We will choose the definition 3.44 for implementing the CR valuation method. We will implement the right hand side from equal sign, the fraction. This represents the "one minus CR," so in order to get the CR we must compute "again" the "one minus" operation. We will also use similar trick with variance as we did when implementing the NCC. First step is to compute

$$S_3 = \sum_{i=0}^{N} B(x + c_x, y + c_y), \quad S_5 = \sum_{i=0}^{N} B(x + c_x, y + c_y)^2, \tag{3.45}$$

$$S_{3j} = \sum_{i=0\ldots N, A(x,y)=j} B(x + c_x, y + c_y), \tag{3.46}$$

$$S_{5j} = \sum_{i=0\ldots N, A(x,y)=j} B(x + c_x, y + c_y)^2, \tag{3.47}$$

$$N_j = card(\{i | A(x, y) = j, i = 0 \ldots N\}), \tag{3.48}$$

these represent the first pass over the entire overlap. Sums S_3 and S_5 can compute the total variance of $B(x, y)$ in one pass, similarly S_{3j} and S_{5j} can do the same job for conditional variances. The latter ones are computed from all such pixels in the registered image which are counterparts of all pixels from the reference image that have its value equal to j. The notation $i = 0 \ldots N, A(x, y) = j$ should capture that as the x and y are functions of i. Do not forget that the equations 3.23–3.25 (on page 21) still hold for variables x, y and N regarding given alignment (c_x, c_y). Finally the definition of the correlation ratio is

$$CR(c_x, c_y) = 100 - 100 \frac{\sum_{j \in J} (S_{5j} - S_{3j}^2 / N_j)}{S_5 - S_3^2 / N}. \tag{3.49}$$

The range of possible evaluations is $\langle 0, 100 \rangle$ with the higher valuation the more optimal alignment. When the ideal alignment is tested (both sides of overlap fit exactly) then the variance in numerator will be always equal to 0. Hence the whole numerator will be equal to 0 and so $CR = 100$. The method is still rather fast and robust. Because of the CR evaluation is considered to be an extension of NCC in term of more complicated functional dependency it can find, the restrictions regarding the noise volume hold as for NCC.

3.3.6 The mutual information

The mutual information (abbreviation MI) as a criterion of similarity measure is based on the information theory. It also uses probabilities to estimate the optimality of alignment, probabilities are also compared among each other. This time the comparison is not in

term of functional dependency but instead in term of statistical independence. The basic literature about MI can be considered [18]. Then there are few short explanations of the method regarding the image registration [16, 17], technical reports [19] and notably [8]. The idea of estimating probabilities using their joint histogram in this evaluation method we have learnt from [20].

We can think of the image data from their common overlap as of a sequence of trials. All possible trial results will be numbered, the set J will again contain only all possible trial numbers, these would be the all possible pixel values. This way we can imagine the U to be the discrete random variable with the probability density function P_U. Similarly the V will be discrete random variable with P_V given by the overlap of registered image. U is given by the reference image. The goal is to estimate their joint probability density function P_W so that P_U and P_V are the marginal ones. W is the discrete random vector $W = (U, V)$.

For the sake of simplicity we will again drop from writing indices which should denote that U, V, and W are alignment dependent. It is also clear from their definitions (defined by overlap, overlap is defined by alignment).

The measure of optimality of alignment is based on the following two ideas. Two random variables are statistical independent when the equation 3.50 holds,

$$\forall u, v \in J : \; P_U[U = u] \cdot P_V[V = v] = P_W[U = u \wedge V = v]. \tag{3.50}$$

And also two random variables are statistically maximum dependent when the equation 3.51 holds,

$$\forall u, v \in J : \; P_U[U = u] = P_V[V = v] = P_W[U = u \wedge V = v]. \tag{3.51}$$

The mentioned literature defines the mutual information between random variables U and V according to the theory of information the way we are used to, that is

$$MI(U, V) = I'(U|V) = I'(V|U) = H(U) + H(V) - H(U, V). \tag{3.52}$$

The $I'(U|V)$ denotes the information measure from theory of information where the amount of information that V has about U is expressed as the Shannon entropy of U subtracted by the conditional Shannon entropy of U provided V.

In [5] on pages $24, 25$ we can find another definition of mutual information

$$MI(U, V) = \sum_{u,v \in J} P_W[U = u \wedge V = v] \log \frac{P_W[U = u \wedge V = v]}{P_U[U = u] \cdot P_V[V = v]}. \tag{3.53}$$

This measures the mutual information between U and V via the degree of dependency using the Kullback-Leibler distance between the numerator and denominator of the logarithm fraction. Which is in correspondence with the idea explained by equations 3.50 and 3.51.

Both definitions of MI are equivalent, i.e. equation 3.52 can be converted into 3.53 and vice versa. If we rewrite the formula in 3.53 (the part with logarithm) and apply the Bayes rules for conditional probability, we will receive the Shannon entropies as in 3.52. The opposite direction is available too.

The base of logarithms should be the same. The value of base doesn't matter so far as it remains constant during the entire image registration. Also notice the symmetric of mutual information in opposite to the correlation ratio.

Both definitions make use of the estimation of probability density functions. These can be determined either from joint histogram (as in [20]) and then simply computed by the equation 3.53. Or, Paul Viola in [18] has presented different approach, the probabilities were estimated using the Parzen-window technique. This Parzen estimator is developed for estimating the probability density functions of continuous random variables. These can be then used when differentiating the MI which is performed in order to get better performance when searching the parameter space using the gradient descend optimisation technique. We have implemented the Parzen estimator to estimate the probabilities of discrete random variables. This solution works but requires a lot of data in order to be pretty accurate. The Parzen-window formula can be well embedded into the formula of entropy present in MI given by equation 3.52.

Probability density functions from joint histogram

This is the one-and-square pass (will be explained) evaluation. In the first pass we must determine the joint histogram from the entire overlap of both images. At the given alignment (c_x, c_y) we define for every $j, k \in J$

$$\begin{aligned} A_j &= card(\{i|A(x,y) = j, i = 0 \dots N\}), && (3.54)\\ B_j &= card(\{i|B(x + c_x, y + c_y) = j, i = 0 \dots N\}), && (3.55)\\ AB_{j,k} &= card(\{i|A(x,y) = j \wedge B(x + c_x, y + c_y) = k, i = 0 \dots N\}). && (3.56) \end{aligned}$$

The definitions for x, y, and N hold (see page 21), the probabilities are then simply for every $j, k \in J$

$$P_U[U = j] = \frac{A_j}{N}, \quad P_V[V = k] = \frac{B_k}{N}, \quad P_W[U = j \wedge V = k] = \frac{AB_{j,k}}{N}. \tag{3.57}$$

After this we can compute the "square" pass, the reason for calling it this way is simply to emphasise that we will compute through the entire joint histogram, i.e. one pass times one pass (the square) through the J set (through the all possible pixel values). This method is therefore a little bit slower then CR. Nevertheless as the size of overlap increases the latency that MI will remain the same because it does not depend on the overlap size but instead on the colour depth of images.

We will put the equations in 3.57 into the equation 3.53, then arrange it a little and we get the definition for MI at given alignment (c_x, c_y) of reference image $A(x, y)$ and registered image $B(x, y)$ as

$$MI(c_x, c_y) = \frac{1}{N} \sum_{j,k \in J} \left(AB_{j,k} \cdot \log \frac{N \cdot AB_{j,k}}{A_j \cdot B_k} \right). \tag{3.58}$$

Probability density functions using Parzen estimator

This subsection aims to explain the background of definition 3.52. The mutual information given by Shannon entropies

$$H(U) = -\sum_{u\in J} P_U[U=u]\log(P_U[U=u]), \tag{3.59}$$

$$H(V) = -\sum_{v\in J} P_V[V=v]\log(P_V[V=v]), \tag{3.60}$$

$$H(U,V)=H(W) = -\sum_{u,v\in J} P_W[U=u\wedge V=v]\log(P_W[U=u\wedge V=v]). \tag{3.61}$$

It can be shown (refer to [18]) that the entropy can be approximated from a sample of pixels drawn from the overlap. The idea is just that the average from given sample trials of given random variable approximates the expectation of this random variable, if the sample is large enough (the bigger the better). The free choice of sample is possible thanks to the fact that expectation of sample averages is equal to the expectation of the whole variable. The possibility of estimating the entropy of variable only from some big enough sample is a consequence of this "idea behind" and the fact that entropy is an expectation.

The samples will be drawn again when using Parzen estimator. Hence it would be wise to denote them somehow. We will model the sample as a multi-set of pixel values, denote this multi-set with small letters (like a or b), the size of sample we will denote with N with the multi-set letter as an upper index (like N^a or N^b). Multi-set is capable of remembering with every its element its count how many times is this element present in the multi-set. We want this property mainly when writing $\sum_{u\in a}$ where we do expect that each addend will be present in the sum the correct count times depending on the count held for that element. For every multi-set a we will also denote with a' the *set* which can be constructed from multi-set a. The cardinality of a will be again denoted with $N^{a'}$.

This way we can rewrite entropies with respect to the samples which help to determine the values of entropies. The sample from overlap of the reference image we will denote a, the sample from overlap of the registered image we will denote b, it must hold $N^a = N^b$. Entropies are then

$$H_a(U) = -\frac{1}{N^a}\sum_{u\in a}\log(P_U[U=u]), \tag{3.62}$$

$$H_b(V) = -\frac{1}{N^b}\sum_{v\in b}\log(P_V[V=v]), \tag{3.63}$$

$$H_{a,b}(U,V)=H_{a,b}(W) = -\frac{1}{N^a}\sum_{u\in a,v\in b}\log(P_W[U=u\wedge V=v]). \tag{3.64}$$

The next pair of samples is used for the Parzen-window estimation technique. The probability density function from samples c and d ($N^c = N^d$) drawn from $A(x,y)$ and $B(x,y)$ is defined

$$P_{U,c}[U=u] = \frac{1}{N^c}\sum_{u'\in c} R_1(u-u'), \tag{3.65}$$

$$P_{V,d}[V = v] = \frac{1}{N^d} \sum_{v' \in d} R_1(v - v'), \tag{3.66}$$

$$P_{W,c,d}[U = u \wedge V = v] = \frac{1}{N^c} \sum_{u' \in c, v' \in d} R_2(u - u', v - v'). \tag{3.67}$$

This is the Parzen-window probability estimate technique. The R_1 is unary and the R_2 is binary smoothing function. It is generally a function with one (positive) peak from which this function symmetrically falls to zero. It is widely used the Gauss function which perfectly fulfils the requirement. The Parzen estimator can be though as of estimating the probability of given pixel value from weighted histogram. There are counted in all pixel values and the weight should lower the significance of pixel values further from the given value. The probability density function estimation can be done in several ways, advantage of Parzen estimate is its accuracy while being the non-parametric representative. Sometimes the mixture of smoothing functions is used. The second advantage is that this method do not require the whole overlap in our case (the whole trial record) instead some usually small sample is enough.

The Gauss function is controlled with two parameters, the mean value and the variance. The mean for this purpose is equal to 0, in [8] is stated that the theory of radial basis functions allows the non-localised functions to be used too. The second parameter is the variance, this controls the span of "mexican hat" (the graph of Gauss function). There exists an algorithm for estimating an optimal value for variance, the idea is to compute several entropies using different values for variance while keeping the same samples and choose the value at which the entropy was the lowest (see [18] on page 47). If we create the log-plot (logarithm will be along the x axis, the axis representing the input variance) than the shape will resemble a basin with flat floor plate. This means that the difference in variance in term of orders (!) doesn't affect much the final entropy. The range for example can be the values from 2 to 25. This allows us to have in our implementation pre-computed the Gauss function of one and two arguments into a table, both are with the standard deviation at value 2, thus the variance is 4. We won't lose accuracy much and we will gain some speed.

The definition of Gauss function in general is

$$g_{\psi_1}(u - \mu) = \frac{1}{\sqrt{2\pi\psi_1}} \exp\left(-\frac{1}{2}\frac{(u-\mu)^2}{\psi_1}\right), \tag{3.68}$$

$$g_{\psi_2}(w - \mu') = \frac{1}{2\pi\sqrt{|\psi_2|}} \exp\left(-\frac{1}{2}(w-\mu')\psi_2^{-1}(w-\mu')^T\right) \tag{3.69}$$

where the $w = (u, v)$ is the row vector, μ is the mean value, μ' is the two-element row vector of mean values, ψ_1 is the variance and ψ_2 is the covariance matrix

$$\psi_2 = \begin{pmatrix} \sigma_{1,1} & \sigma_{1,2} \\ \sigma_{2,1} & \sigma_{2,2} \end{pmatrix}.$$

We change the notation into

$$\sigma_{1,1} = \sigma_1^2, \qquad \sigma_{2,2} = \sigma_2^2, \qquad \sigma_{1,2} = \sigma_{2,1} = \rho\sigma_1\sigma_2.$$

The σ_1 and σ_2 are the standard deviations, ρ is the correlation coefficient. Now after some rearrangement

$$\begin{aligned} \psi_2 &= \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix}, & \psi_2^{-1} &= \tfrac{1}{|\psi_2|}\begin{pmatrix} \sigma_2^2 & -\rho\sigma_1\sigma_2 \\ -\rho\sigma_1\sigma_2 & \sigma_1^2 \end{pmatrix}, \\ |\psi_2| &= \sigma_1^2\sigma_2^2(1-\rho^2), & \sqrt{|\psi_2|} &= \sigma_1\sigma_2\sqrt{1-\rho^2}, \\ & & w\psi_2^{-1}w^T &= \tfrac{1}{1-\rho^2}\left(\tfrac{u^2}{\sigma_1^2} + \tfrac{v^2}{\sigma_2^2} - 2\rho\tfrac{uv}{\sigma_1\sigma_2}\right). \end{aligned}$$

In the implementation for Bohunice we can pretty safely set $\rho = 0, \sigma_1 = 2, \sigma_2 = 2$ and also $\psi_1 = 2$, then because of Parzen estimator $\mu = 0$ and $\mu' = (0,0)$. Altogether, the Parzen probability functions and the help equations while leaving the variance still in general form (non-assigned) we get

$$P_{U,c}[U=u] = \frac{1}{N^c}\sum_{u'\in c}\frac{1}{\sqrt{2\pi\psi_1}}\exp\left(-\frac{1}{2}\frac{(u-u')^2}{\psi_1}\right), \tag{3.70}$$

$$P_{V,d}[V=v] = \frac{1}{N^d}\sum_{v'\in d}\frac{1}{\sqrt{2\pi\psi_1}}\exp\left(-\frac{1}{2}\frac{(v-v')^2}{\psi_1}\right), \tag{3.71}$$

$$\begin{aligned} P_{W,c,d}[U=u\wedge V=v] = \frac{1}{N^c}\sum_{u'\in c,v'\in d}\Bigg(&\frac{1}{2\pi\sigma_1\sigma_2}\cdot \\ &\cdot\exp\left(-\frac{1}{2}\frac{(u-u')^2}{\sigma_1^2} - \frac{1}{2}\frac{(v-v')^2}{\sigma_2^2}\right)\Bigg). \end{aligned} \tag{3.72}$$

Now we'll present the final definition and hints how to compute the MI evaluation of given alignment (c_x, c_y) of the reference image $A(x,y)$ and the registered image $B(x,y)$.

First we are supposed to draw four sample multi-sets, two (a and c, $a\cap c = \emptyset$) from the reference image and two (b and d, $b\cap d = \emptyset$, $N^a = N^b$ and $N^c = N^d$) from the registered image. In [18] on page 48 there is a hint how to compute all entropies using only two samples (one for each image). It is called the cross-validation. It splits sample into two distinct samples and this way we get four again. Entropy for the random variable U can be expressed using cross-validation as

$$H_a(U) = -\frac{1}{N^a}\sum_{u\in a}\log(P_{U,a-\{u\}}[U=u]). \tag{3.73}$$

For random variable V and the random vector W is the expression similar.

We had tested this approximation of entropy (according to 3.73) but sadly to say we had no success. Matching behaved completely chaotic regardless of sample sizes and the way samples were drawn. There had to be some mistake in the implementation or we must had failed to assure the conditions that should be held when using the Parzen-window technique (one can find them in [18]). But the latter didn't seem to be true because we had tried the Parzen-window probability estimation aside and it worked, see figure 3.1, the situation was exactly the same as in the program from what we conclude that the problem was not (or at least was not only) in the Parzen estimator.

From figure 3.1, although it doesn't seem that the Parzen estimator is working absolutely properly, we believe that this is not the core of the problem. For the MI evaluation

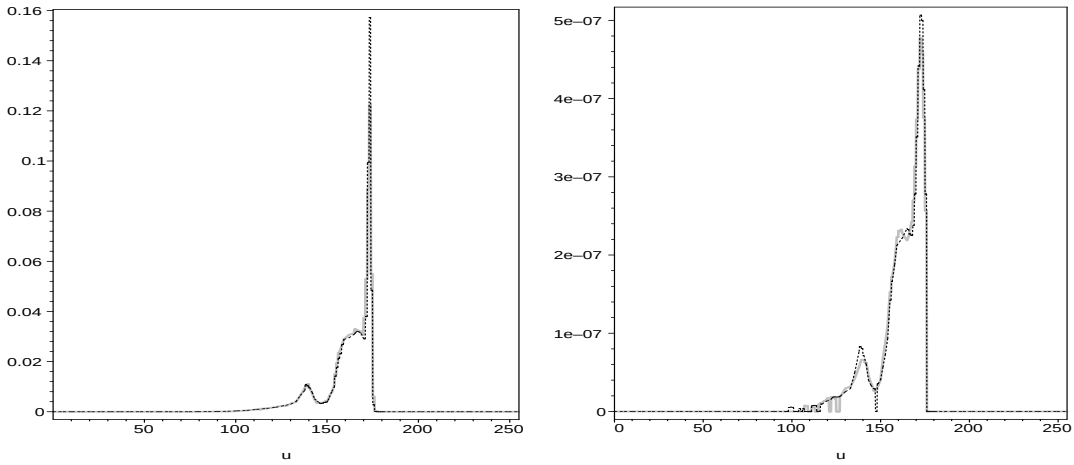


Figure 3.1: The probability density functions of the same overlap. The left picture is the probability density function estimated from histogram, total number of pixels with the given value divided by the total count of all pixels. The right one is the probability density function computed using the Parzen-window technique with the sample size of 750 pixels, the sample was drawn uniformly from the entire overlap, the smoothing function was the Gauss function with zero mean and standard deviation of 2. The shape is roughly preserved but values are shifted and linearly scaled a little. Nevertheless the entropy computed from these probability density functions is for "black dashed" function 3.481 and 3.488, for "thicker gray solid" function 3.540 and 3.469' in the left and right picture.

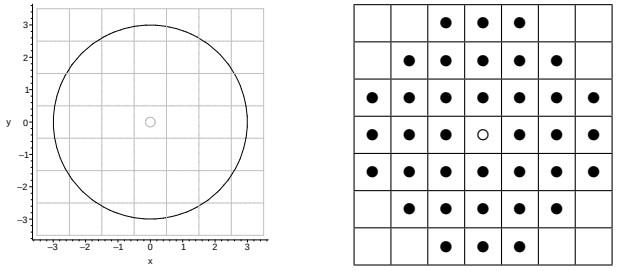


Figure 3.2: In the left picture we have outlined the surroundings for pixel pair, i.e. for estimating the joint entropy H_W. In both pictures the given pixel pair has two pixel values, the first value is in the centre of the x axis (horizontal), the second value is in the centre of y (vertical). The pixel pair values which are supposed to be in the surrounding can be easily determined from the position in the picture from their coordinates. In the left picture the surround of given pixel value pair contains all value pairs inside the circle, each value pair is represented with a single square. The right table then summarises the left picture exactly, i.e. pixels marked with • are accepted to be in the surround of given value pair marked with ∘. For entropies of U and V the surround for pixel value z is supposed to be the pixel values in range $z \pm 3$.

measure is important the value of entropy and in that figure we have shown that the entropy for single random variable is roughly the same. The situation is more confusing as we will see later in the section with tests that the problem might be just in the Parzen-window estimation technique, precisely in the settings of this technique.

Also from time to time in image registration the fluctuations are present during the matching process. Furthermore we have observed that these entropies of single random variables behave like a constant during the matching process and the responsibility of decision which alignment is the optimal one is purely on the joint entropy. And in fact the parameters of this valuations method were estimated so that the joint entropy is approximated well.

Finally we have established a functional solution. The samples are drawn uniformly from the entire overlap, one sample per image. The smaller sample size the bigger fluctuations during the image matching process and the more easier the method had trend to return bad alignment. For image compositions from Bohunice the value $N^a = N^b = 750$ seems to be mostly alright.

The entropy was computed for all pixel values from the sample including their surround values, see figure 3.2 for surround description. Each pixel value was computed in the final entropy sum maximally once, i.e. no pixel value was counted in more then once. For sample a we will denote with $\bar{a}$ the *set* which contains the whole a' and the surrounds for every pixel value from a'. The set from its properties will ensure that there are no two same values which is important in order to what we've stated about the entropy computation. The second deviation is the classical form of entropy computation in spite of the fact that we are estimating the entropy from a sample. The final forms for entropies of the mutual information evaluation are therefore

$$H_a(U) = -\sum_{u\in\bar{a}} P_{U,a}[U=u]\log(P_{U,a}[U=u]), \tag{3.74}$$

$$H_b(V) = -\sum_{v\in\bar{b}} P_{V,b}[V=v]\log(P_{V,b}[V=v]), \tag{3.75}$$

$$H_{a,b}(U,V) = -\sum_{u\in\bar{a},v\in\bar{b}} P_{W,a,b}[U=u\wedge V=v]\log(P_{W,a,b}[U=u\wedge V=v]), \tag{3.76}$$

$$MI(c_x,c_y) = 100(H_a(U)+H_b(V)-H_{a,b}(U,V)). \tag{3.77}$$

The probabilities are computed using equations 3.70–3.72, the variables u, u' respective v, v' are substituted with $A(x,y)$, $A(x',y')$ respective $B(x+c_x,y+c_y)$, $B(x'+c_x,y'+c_y)$ with appropriate coordinates. The coordinates and their notation are very dependent on the implementation and so unlike previous methods we will stop at this point with further formulation of MI. We believe the established notation with two random variables and one random vector will not confuse the reader in such way he won't be able to reproduce the MI method into a computer program.

The evaluation is smaller-square pass computation. This needs to be explained. When trying to keep the "rough complexity" estimation fashion (which is explaining in words instead of exact mathematical formulas the asymptotic time complexity of implementation), we had to find some expression for the square time complexity when the argument is actually a portion of the correct input. The correct input is the size of the overlap. The portion of it represents the sample size which is used for estimating the entropy. This

sample size is then processed in a square time. From the asymptotic point of view it is of course unimportant because the portion is really just a constant times the input size variable and we normally drop off constants in such cases. In fact that's what we were doing with estimation of time complexity by previous evaluation methods. But in practical implementations (mainly when the time consumption is crucial factor) often these multiplicative and additive constants play significant role.

Previous methods including the MI with probability from histogram are "almost almost" about the same speed, this implementation with a sample size for which the evaluation behaves satisfactorily is significantly slower even when the computation goes through a small portion of the entire overlap. This is a result that needs to be emphasised and the explanation for expression smaller-square pass computation.

The "extended" sample $\bar{a}$ in contrast to the sample a does not change anything, the constant will rise a little but still the portion it represents will remain really a little portion of the entire overlap volume. For instance in Bohunice we have found out that usually $card(a') \leq 150$. Remember a' is a set of *pixel values* and so always $card(a') \leq 255$ when colour depth is 8 bits, then see typical histogram in figure 3.1. In spite of the fact that surround is defined rather large (36 surrounding values for joint case and 6 for marginal cases) the volume of extended sample does not become enormous, it holds $card(\overline{a \times b}) \leq 4000$. There are $N^a (= N^b)$ sample pixel values drawn from the given overlap and thus also N^a pixel value pairs. There are lot of values or pairs of values repeating.

For every element we append its surround, the size of surround in marginal cases is unimportant in comparison to the size of the joint case (entropy computation) because we must go through all extended sets to compute their corresponding entropies. It is therefore enough to reason only about the joint case which is the major contributor into the final cycle count in implementation. But we have seen that this surround volume will not affect the cycle count tremendously while the overlap volume is typically in Bohunice's situation still from 10000 to 70000 depending on the actual evaluated alignment.

The time consumption suddenly becomes pretty clear when the certain values are given. The "pseudo-asymptotic" complexity is $card(a) \cdot card(\overline{a \times b})$ which is far more then the maximum overlap volume. It is asymptotic because this estimation doesn't take care of implementation constants while it is non-asymptotic because it takes care about the portion volume as explained in some previous paragraph. That's a disappointment because we thought that the portion of data that needs to be processed will greatly save time.

From practical results we suggest the proper use when the registered data are far larger, for example when two entire images are overlaid and some registration has to be done. In such case we assume that any sample or surround size changes will not be necessary. Hence the speed ratio will become more lucrative for the MI using Parzen estimator. For the given present situation in Bohunice this variant of MI evaluation is a few times slower as will be seen later.

The estimation of the range of possible evaluations is little complicated too. The maximal evaluation is reached when the given alignment is optimal, i.e. in terms from theory of information we would say: The reference image explains the registered image well. So at optimal alignment that information measure $I(X|Y)$ is high, the random variable Y has enough information about (knows better) the random variable X and hence Y can determine from its value the value of X. And vice versa.

The information measure is defined

$$I(X|Y) = H(X) - H(X|Y). \tag{3.78}$$

This is equal to the equation 3.52 and more appropriate for further explanation. When the optimal alignment is reached then the conditional entropy $H(X|Y)$ will be exactly 0. The worse can Y estimate X the higher is the entropy, entropy is often called as the uncertainty level. The single entropy $H(X)$ is roughly constant during the matching process, it was noted on page 35. This concludes that the maximum value of mutual information is the maximum that can be reached with single entropy.

For entropy it holds

$$H(X) = \sum_{x=i}^{j} P_X[X = x] \log_a(P_X[X = x]) \leq \log_a(j - i + 1). \tag{3.79}$$

The indices of sum are arbitrarily selected. The maximum value of entropy is bounded by the logarithm of maximal count of possible values of measured variable. In our implementation we have used the natural logarithm, the typical number of entropy computed values is the typical value of $card(\bar{a})$. Hence the (mostly) maximal value of MI evaluation method is estimated to be $100 \cdot \ln(150) \approx 500$. It is multiplied by 100 because of equation 3.77. Minimal value is 0.

The MI evaluation works fine, notably when the probability density functions are estimated from histogram. It is rather brightness and contrast tolerant/resistive. It is so perhaps because of the information theory background. The approach of information extracting from the data is probably more robust than simple colour values comparison like in SSC or SAVD. This approach will probably handle registration of images which are roughly identical, i.e. basic objects/features are present that allow humans to recognise with high level of certainty the most optimal alignment.

The fact we are computing only statistic from probabilities allow us to use CR and MI for multi-modal image composition. The uni-modal composition is a special case of multi-modal composition, thus these methods can be of course used in uni-modal composition. The probabilities are estimated from histogram simply as a ratio of the count of pixel with given value against the total count of all pixels or using special technique called Parzen estimate. There is no trace of colour matching (as in SSC or SAVD) in the probability estimation. Thus allows us to apply these methods on some two images registration where the same objects are pictured in different colours (pictured using different colours in the texture but the same relief). The important is that the volume which takes this object in image and also its shape is roughly the same so the probabilities of pixel values (of colours) in both images will be roughly equal.

When multi-modal image composition is performed then the need for intensity conversion function I may occur anyway. If not yet when image registration then for sure when image stitching. The composition will be very notable (if not unpleasant) when for example some object's half is in the reference image and the rest is in the registered image, object is pictured half in one colour and the rest in another colour.

This effect may not be undesired when doing image composition the "other way" which is not implicitly assumed in this paper. One can need to create a final image from two

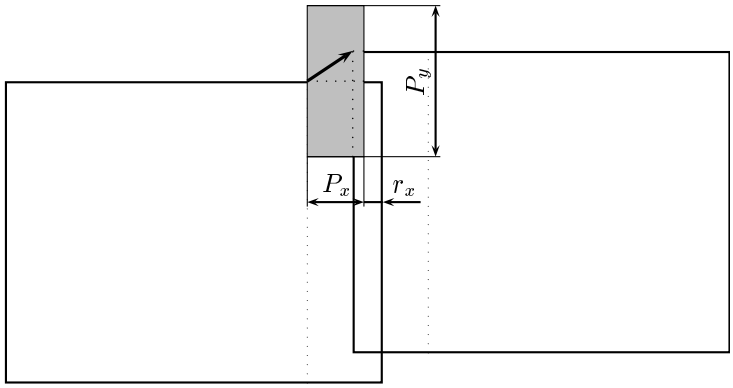


Figure 3.3: The figure outlines the image registering process when some certain alignment is tested. The alignment is given by vector (c_x, c_y) pictured as thick arrow, this vector ranges from the gray area which visualises the parameter space $P' = P_x \times P_y$ of all possible alignments. The situation displays the way it was implemented, i.e. not really all possible alignments are tested. The narrow track of restricted alignments with strength of $r_x = 10$ is shown here too. Please note that the alignment $(0, 0)$ corresponds to situation when maximum overlap is achieved, this will be useful for next graphs to understand.

images by overlaying them. For example when the same part of human body is scanned once with x-ray (the first image contains mainly bones) and once with MR (the second image contains mainly organs).

3.4 The comparison of matching evaluation functions

3.4.1 The registering process visualising tool

For the purpose of speed up of registration process we had to develop some tool. The purpose of this tool had to be in the visualisation of the registration process. That tool had to help us with decision which optimisation technique we would have to use.

At the beginning of this paper we explained the approach that we would implement. The idea was searching the parameter space to find the best solution for equation 3.1 (on page 9). The problem is partially in the decision which parameters are better in term of optimality of alignment. And partially in the decision how to search that space while it is usually quite large. Some solutions for the first decision process we had described in the previous section.

We have called the methods introduced there as the alignment (e)valuation methods, see the very beginning of section 3.3, property 3.18 (on page 20). Sometimes we used the term functions. And indeed these are a real functions. Their input domain is the parameter space P', in our (Bohunice's) case it is the Cartesian product $P_x \times P_y$. The sets P_x and P_y are outlined for example for horizontal image registration in the figure 3.3. The range of possible evaluation values is dependent on the evaluation method and was defined for each method in the previous section. We will often refer to the visualising tool as to a visualising function which is in fact the same as the evaluation function. Now the registering process visualising tool is nothing else than a simple 3D graph of given

evaluation function.

The search for optimal alignment is then nothing else than the search for global maximum of this function. The most desired property of each evaluation function is then probably its smoothness. Despite the fact we cannot directly derive such functions. Because if we had been able to than we could have used more results from theory of numerical computations. The second desired property is that the corresponding graph will have only one peak with broad fundament and thin steeple. The broad fundament is good for beginning of optimisation technique which will immediately get the correct direction toward the optimal alignment. Then as the evaluation gets steeper the alignment search will eventually converge faster to its optimum. But this needs to be seen (visualised) first.

Martin Čapek in [5] introduced very similar concept of visualising the registration process. In fact there were used exactly the same functions with two following exceptions. Firstly, the optimal alignment is reached in global minimum and also the entire registration problem is defined similarly but for minimum. Secondly, from the entire visualising tools are pictured only those points which can be reached from the optimum point while moving only "up to the hill." Put another way, there are pictured only such points for which a marble can slip into the optimal point. Čapek calls this ZAF, in original Zóna Atrakce Funkce and translated into English ZAF, it means the Attraction Zone of Function.

3.4.2 Visualising the registration process

We will show six images at one page in every following subsection, each image will be the registration process visualising tool, i.e. the graph of visualising function. Each set will be picturing the registration of the same two images. The left image will be the reference image and will remain constant during all tests, one can found this image in Appendix as a figure A.1 (on page 68). The right image will be the registered image and will be changing during tests, figure A.2. In each figure there will be a reference into the appendix section were the corresponding registered image can be found.

All images in tests had the same dimension 1232×972 pixels and were registered with maximum overlay set to 7%. Thus the entire parameter space is defined as

$$0 \leq c_x \leq 75, \qquad -68 \leq c_y \leq 68. \tag{3.80}$$

The possibility of c_y to hold some negative value means that the direction of this registration was horizontal, as outlined in figure 3.3. The name of the evaluation method which matching process is visualised in each picture can be found on the left from the graph. The optimal alignment was at $(27, 0)$.

Please, see the figure 3.4 for reference how does look like a graph of each evaluation method when registering the original data, i.e. no adjustments at all. Next section will try to simulate what might happen in real praxis. We will always try to artificially adjust the reference image more than it would be acceptable in praxis to demonstrate that selected methods are capable of more than it will be usually expected from them.

Notice that some evaluation methods produce a graph with one strong peak, this has a broad base and slowly gets steeper. On the other hand some methods evaluates every alignment as strictly bad if the alignment is not very nearby the optimal one. As the alignment search approaches close neighbourhood the evaluation value is emerged closer

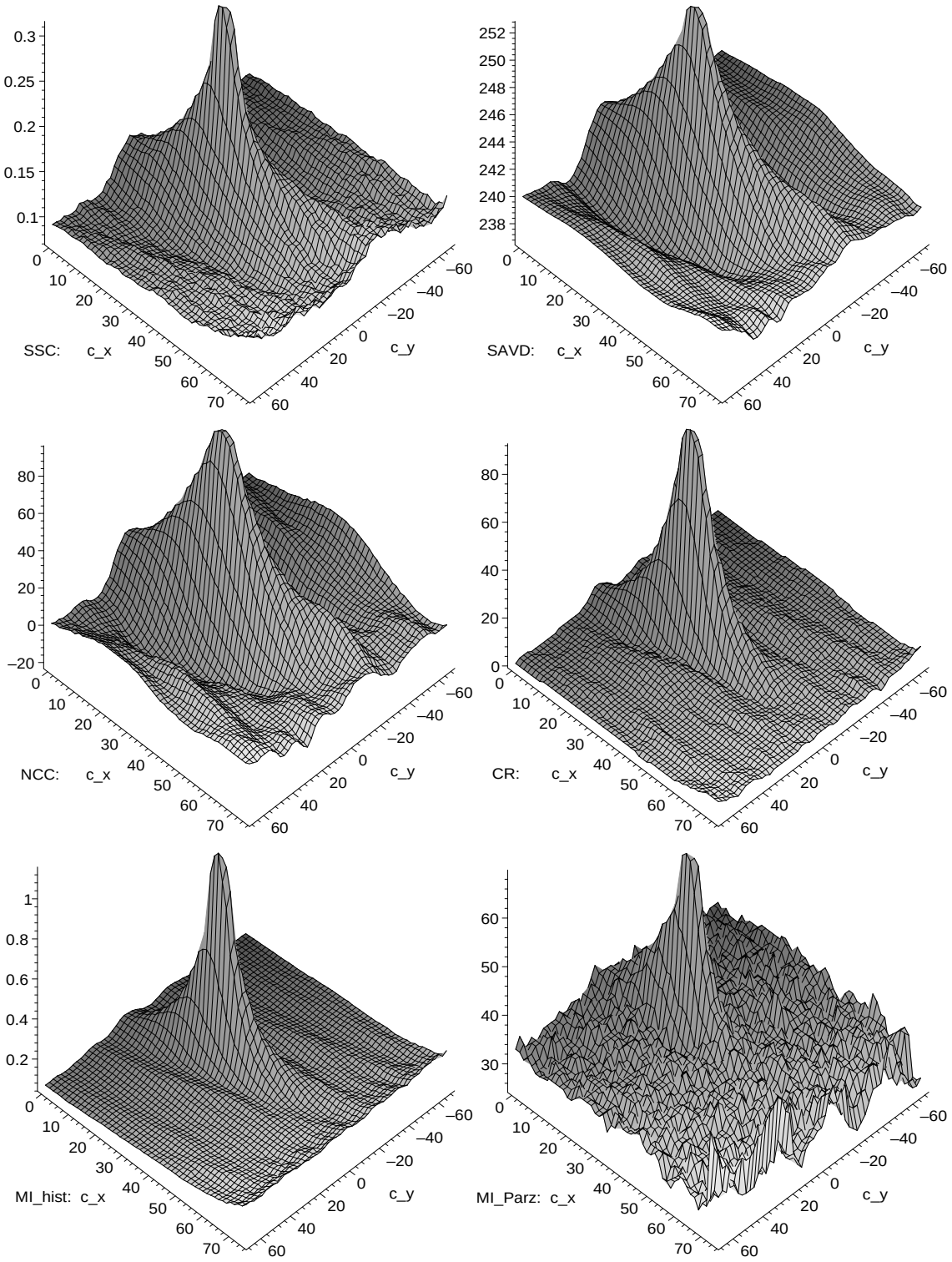


Figure 3.4: Images of the registration visualising tool for each evaluation method. The registered image wasn't adjusted at all.

to the value at optimal alignment. Also notice some similarity between SAVD and SSC in this and all following tests.

All methods seems to have no difficulties for such type of images to find the optimal alignment except the Mutual information computed using Parzen estimator. There are notable fluctuations which in this case/test cause no problems.

Test with different brightness

In this test the registered image was adjusted to be more brighter than the reference image. We have used the "Brightness-contrast" function in the GNU GIMP program and we set the contrast scale button in that tool to the value 80. The resulting image can be found in Appendix as a figure A.3.

The first result, SSC and SAVD got fooled by the histogram shift. Even when the SAVD seems to find the optimal alignment. The normalisation part of evaluation helped to rise the evaluation of alignments defining smaller overlaps (the definition 3.27 is on page 23). Please notice the evaluations of the optimal alignment, it is under the "basic" values from the previous test. While the differences in pixel values probably were bigger. We don't know the reason for this to happen. The NCC got lower too.

The multi-modal methods CR and MI doesn't seem to have any difficulties at all. This is what they were created for, same objects pictured in different colours in each image.

Test with different contrast

We had simulated the contrast adjustment with the "Spread" function in GNU GIMP program. We set the parameter of this function to the value of 15. The effect is that the registered image becomes fuzzy, objects in image lost their edges. Edges were broken into lots of small particles and these were randomly dropped around their initial positions. Original objects are still recognisable, they retain their shapes, colours and positions. The image has shaky feel, see figure A.4.

In this test all methods worked satisfactorily. There are no problems unlike in the test with different colours for the same objects. This test is rather to test the fuzziness of evaluations methods as noticed in the introductory section.

Fairly no problems except again the MI with Parzen estimator. While the MI with probabilities computed from histogram is nicely smooth, its counterpart in each test gets more and more interfered. These two versions of MI follows the same concept which MI from histogram proved to be functional, hence the problem is in the Parzen estimator which should be adjusted to fit this registration more better.

Test with noise

The image was artificially adjusted to become very noisy. We used the "Noisify" function from GNU GIMP program and we set the "Independent check box" to enabled, than scaled all three colours red, green, and blue to value of 0.40. This way we had received very noisy image, the noise is equally spread across the entire image and is colourful too. To have a better idea, please, refer to Appendix to see the image in the figure A.5.

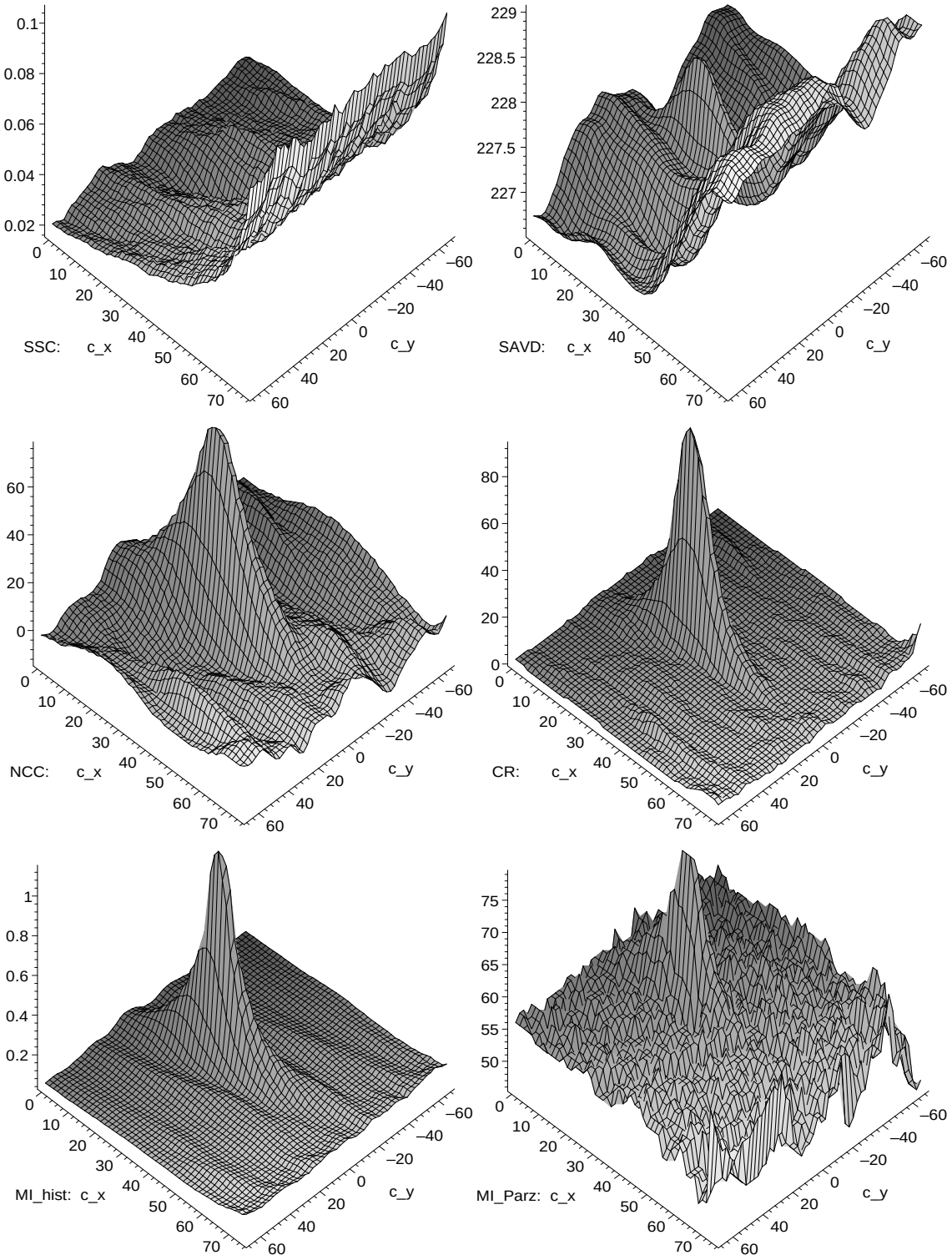


Figure 3.5: Images of the registration visualising tool for each evaluation method. The registered image was made more brighter than the original.

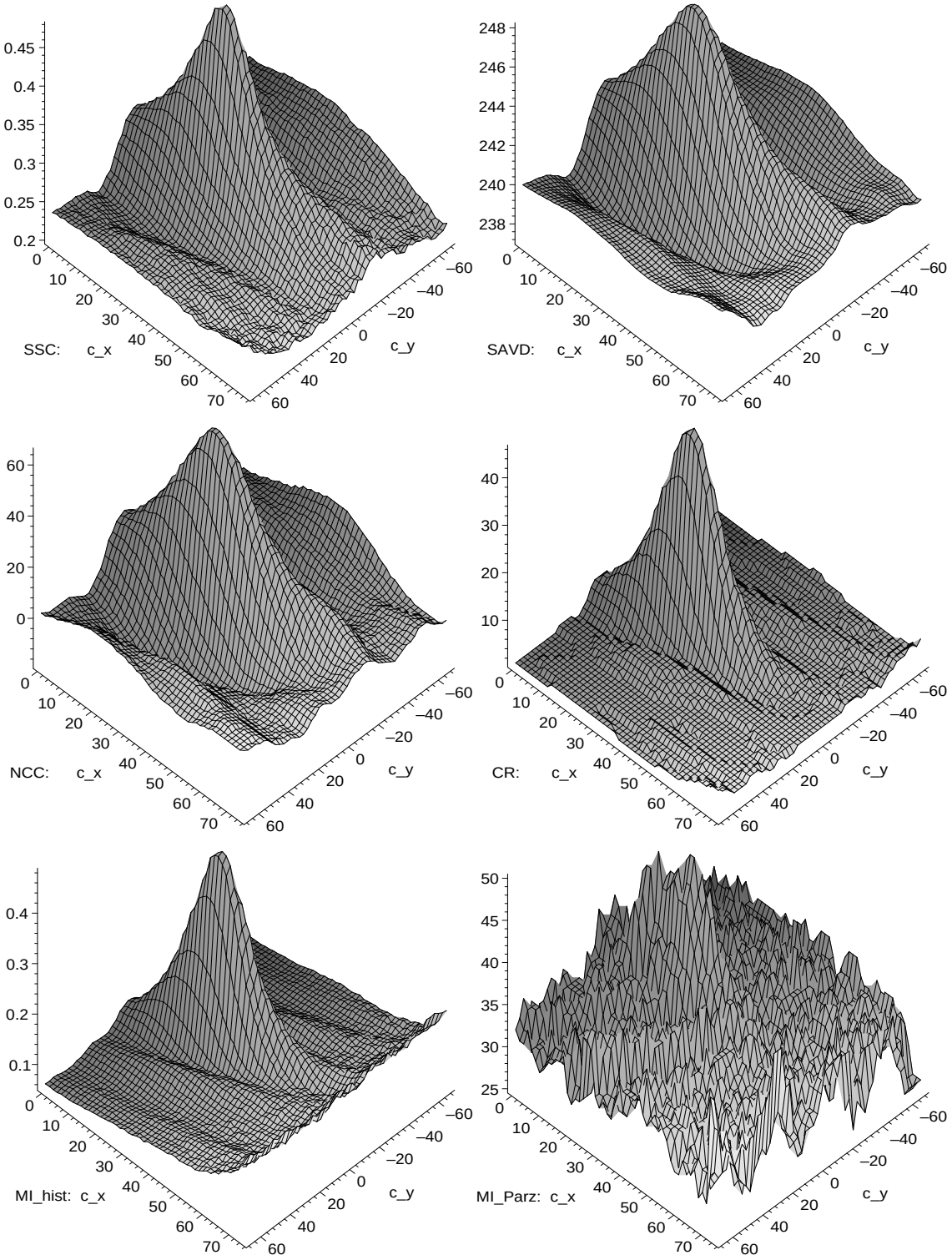


Figure 3.6: Images of the registration visualising tool for each evaluation method. The registered image was jittered, the bigger objects are retained, edges lost their sharpness.

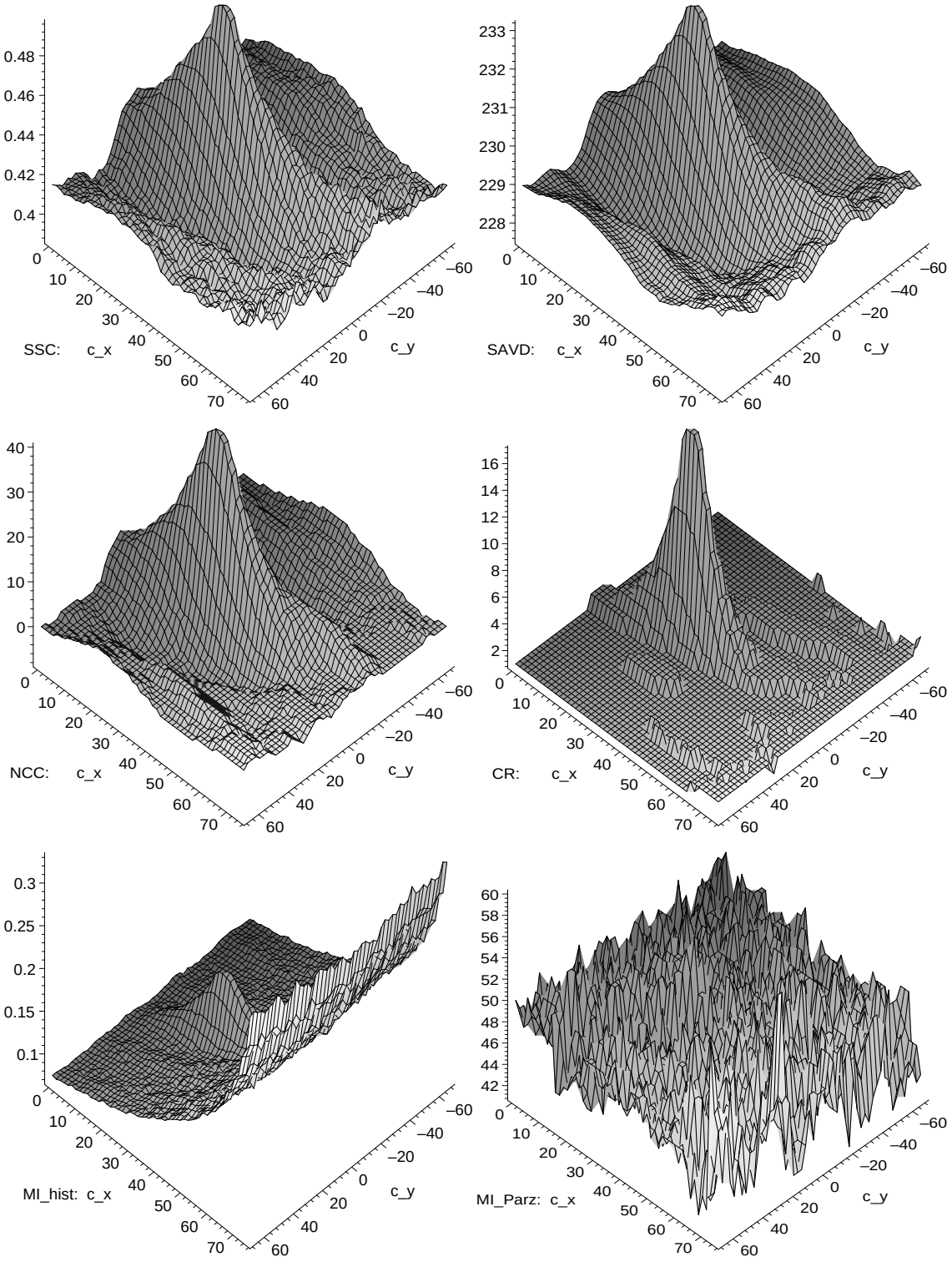


Figure 3.7: Images of the registration visualising tool for each evaluation method. The registered image was covered with random, equally spread noise. The original objects are still to be recognised.

In this test we wanted to show how much can be the registered image "over-noised" while the reference image remains untouched. All methods except the both MIs work so far. It seems that MI in such application has some difficulties.

These are very similar to problems that arised when normalising the SSC, the smaller overlaps are advantageous. We had therefore narrowed a little (by 10 pixels) the space of all possible alignments to eliminate alignments which made the SSC and SAVD unusable. This was caused by the very same effect. The remainder can still be found, look for evaluation at higher x coordinate in the SAVD visualising graph, SAVD evaluation starts to rise there suddenly.

The explanation for MI to work poorly may be in the underlying entropies. The entropy measures uncertainty, the more random the higher entropy. We have stated previously that from our investigation it seems that the entropies for single variables (the "marginal" entropies) variate very little in given registration problem, they behave like they are independent on the overlap from given image. The MI is then estimated mainly from the joint entropy computed from both overlaps at given alignment.

When there is a noise present in only one image, then it becomes harder to predict how does look like the overlap of registered image for the overlap of reference image. But, the smaller overlap the more probably we can hit the correct values because the smaller overlap then the less space for the randomness of noise to show its variety. Therefore we may be able to "explain" the overlap from registered image better and so the uncertainty (entropy) gets lower. The joint entropy contributes to the mutual information with negative sign and hence it in fact "highers" the evaluation. The smaller overlap the higher evaluation and that's what is also visualised.

3.4.3 Time consumption comparison

In this subsection we will present a table (3.1) which aims to compare selected evaluation methods by the time that is required to find the optimal alignment.

The table contains data which were obtained under the same conditions, i.e. the same maximum overlap size, the same images, no optimisation techniques enabled. Each time shown in both columns is the time that is required only for pure computation of evaluations of all alignments possible, specially there is not included any input readings, printing results, stitching images, e.t.c. The program was compiled without any compiler optimisations.

We have decided to present a total time required for the entire optimal alignment search instead of presenting the time for one single evaluation of some given alignment. We believe this approach has at least two advantages. One can make a better idea how much time does one registration take, the time measuring is not so influenced by time measuring error. Both are important.

Firstly, the required time automatically involves the variance of overlap size which is a crucial factor in time consumption. Otherwise (when the time represents a single alignment) we would have to state how big the certain overlap was while there is probably no exact way to determine from this time the time that would be required to compute single evaluation of some other alignment respective overlap of different size. The idea how fast/slow is the given method wouldn't be then any good. And secondly, the measuring error is mainly important when further calculations are performed.

	7%	12%
SSC	13.066s	75.245s
SAVD	10.293s	62.225s
NCC	15.124s	87.584s
CR	18.654s	103.274s
MI, histogram	41.348s	190.774s
MI, Parzen	642.135s	1986.840s

Table 3.1: Time comparison. The entire space of all possible alignments is searched, the same images, the same maximum overlap defined, nothing else involved in the final time. Registered original images can be found in Appendix, figures A.1 and A.2. The data in the column label "7%" describe registration speed with the maximum overlap set to 7%, the second column labelled "12%" describes the time required when the maximum overlap was set to 12%. The parameters of the hardware on which this values were obtain can be found in the caption to the table 4.1 (on page 53).

Another time comparison after some optimisation enabled is in table 4.1 in section 4.2. In caption of that table one can also find the parameters of personal computer on which the data in both tables were obtained.

Chapter 4

Refinements

4.1 Colour depth of matching data

The first observation after first processed test data was the fact that it doesn't actually much matter for what colour component the matching is performed. One could notice in section 3.3 that every evaluation method copes with the image data as with a single value for each pixel.

These days virtually every system is in colours. These are usually represented in the Red, Green, and Blue components and so is the output from libraries which perform the image file formats reading. By the way we have used [9, 15]. We had computed the evaluations for these three colour components and also for the gray-scale value which we obtain using equation

$$\text{Gray-scale} = 0.299 \cdot \text{Red} + 0.587 \cdot \text{Green} + 0.114 \cdot \text{Blue}. \tag{4.1}$$

The coefficients are well known and should represent the way how do humans see in the night when the light intensity is not enough for colour view. But even then humans still can fairly recognise colours, notably the green and then red. Finally it is also important to say that our gray-scale values were stored using 8 bits as were also each colour components.

The matching process were thus computed four times, for every colour component and also for the gray-scale data. It was mostly returning similar coordinates (in pixels) as the optimal alignment. All four coordinates were variating a little but never more than 5 pixels for the given image pair. Sometimes it occurred that one resulting coordinate pair (translation vector) for some colour component was completely outside the given range, i.e. differing more than 10 pixels in both coordinates. But it has never happened during our tests that outside the given range was the result from gray-scale data, it has also never happened that there were two "colour" results outside the range and the rest was good. Always the majority in colour results claimed the optimal solution.

We had therefore decided to convert each image into gray-scale values when reading into memory. This is a common solution which helped us to answer several questions and also brought us great memory savings. The program after that still works satisfactorily. The answers were not exactly answers, correctly speaking the data conversion allowed us to skip answering to following questions as these suddenly weren't actual.

If there had been no conversion than we would have to decide what to do with three results. Shall we pickup one from them but how to determine which one? Or shall we

average them and shall we average them equally or weighted the way as we convert into gray-scale? Shall we ignore the outside value when averaging and what should be the tolerance for determining the outside results?

There is probably no easy answer for the first question. The picking up one from three values algorithm cannot be sensibly constructed notably when quite general system should be created. In such case we don't know the area of expected solutions in advance and therefore we have no clue which solution is the good one. And if we knew the area (that is partially true in Bohunice, we'll see later) what to do when in the expected area are two different (notable distance) solutions? Perhaps the area of expected solutions should be small enough, that would answer this question and also introduced a new one: Is there a need to compute the alignment when we know pretty accurate with high probability the small enough area of optimal alignment? So this is evidently not the way.

The second and the third questions are more reasonable. It seems that we can safely place confidence on the majority scheme. The situations where the third value was notably different happened rarely and so in this case we presume we can compute the coordinates of optimal alignment by simply averaging the "major" values. The tolerance we would suggest should be such that the sutura camouflage algorithms (see section 4.5) should "cover" it, i.e. they should be able to hide the difference in image composition/stitch resulting from alignments from that tolerance range in term that human will hardly notice it.

It must be also noted that we had tested data only from Bohunice's system, i.e. the data from *only one* system. So we can only guess what to do on someone's data.

The used memory after data/colour conversion has shrinked into one fourth of the previous usage. One fourth because each pixel was stored in four bytes, three for colour components and one was for alignment of each pixel into the memory cell which should improve all the data transmissions between processor and its memory on today personal computers. Now we had four pixels in one memory cell instead of only one. It has also introduced some speed up, because the physical amount of memory that have to be processed (read) is four times smaller.

4.2 The speed, optimisation techniques

4.2.1 The need for speed up

The speed in this matching approach when all possible alignments are tested is important factor. On average personal computers which are heavily used in Czech Republic such type of computation is rather slow.

The main slow down factor is the speed of memory. As we saw in section 3.3 the computation does not contain lots of time-consuming mathematical operations. Well, there are some square roots in NCC but these are computed once per alignment. Then natural logarithms in MI which are a problem.

Speeding up the MI evaluation method

Interesting question is: Will there be any help when Taylor series of logarithms were used instead of "real" logarithms? We haven't tested that and so we don't know the exact

answer.

Note, the logarithms are only in the variant of MI where the Parzen estimator is used. This variant has several problems in the current implementation. The first and worse one is that this methods fails to find the optimal solution of registration from time to time. Generally it can be said that whenever this evaluation method fails when matching certain two images from some image composition then it will fail for any other matching of arbitrary image pairs from that image composition. And vice versa, whenever it works then it will work for the rest of given composition. The nature of images from given composition is usually of the same kind. Thus we conclude that the problem is in the estimation of parameters rather than in the luck of sample draw when alignment evaluating. But still after re-setting the Parzen estimator (so it begins to be functional again) this method has lots of small fluctuations (see subsection 3.4.2). The fluctuations are the reason that makes us believe that lower precision but faster approximation of logarithms won't help. The program will then faster fail (but that also might be considered as a partial help). But this is still a guess and the problem is open to further investigation which might prove that the fluctuations come from something more complicated than only the randomness of sample draw.

Another solution which is often used in computer graphics are the look-up tables in which the precomputed values for "slow" operations are stored. But this is possible only when the amount of input values that can be requested to be computed is not overwhelming because of memory consumption.

In the MI computation we probably cannot say which values (arguments of logarithm) will be requested. The reason is the variance of matching images. These can be very different and from this point of view we can expect the probability density functions to be very different too. This doesn't involve only the shape of such functions but also their values. In figure 3.1 (on page 34; from subsection 3.3.6 regarding the MI evaluation function) we saw that even the level or the distance from zero base (from x axis) can be different. So we kept logarithms the way they are.

Instead we used the look-up table for Gauss function. This gained some notable speed up. We could have afforded such table because the arguments to Gauss function can be expected from known range. These would be the natural numbers representing all possible pixel values. So the look-up table when 8-bit colour depth is used is not so large and can be easily stored with good precision in memory. For instance our table for Gauss function of two arguments (for estimating the joint probability) took exactly 512KB of memory. That is much in compare to typical program structures but represents nothing in comparison with the total program usage (overlapping regions of images are stored in memory, typical image composition then takes about 200MB of memory).

4.2.2 Gradient descend and n-step technique

So the need for optimal alignment searching is now pretty obvious (mainly from section 3.4.3). In our implementation we have used two optimisation techniques. Both are general techniques when given space needs to be searched for global extreme value of some "cost function." The first technique we call the n-step technique, the second one is classical gradient descend.

The n-step optimisation technique

The so-called n-step optimisation technique can be explained for example this way. It first shrinks the entire space of possible alignments into one n-th of the original size by picking up only each n-th alignment, it can be though of as creating a grid/net with square cells with edge size of n. This smaller space is searched for global maximum, say it is found in alignment given by $(c_x^{(n)}, c_y^{(n)})$. Then next step will be $\lfloor n/2 \rfloor$-step optimisation technique which will be ran over the parameter space given by square with the edge size of $2n$ pixels and the centre of square will be that $(c_x^{(n)}, c_y^{(n)})$. Given space is searched again more precisely with the grid of cell size $\lfloor n/2 \rfloor$ pixels. A new maximum is found in $(c_x^{(\lfloor n/2 \rfloor)}, c_y^{(\lfloor n/2 \rfloor)})$ and the optimisation restarts with a new parameter $\lfloor \lfloor n/2 \rfloor /2 \rfloor$ and also a new square space is defined, e.t.c. Finally after $\lfloor log_2(n) \rfloor + 1$ (including the initial grid search) steps we will eventually find the optimum in $(c_x^{(1)}, c_y^{(1)})$. The step size is always divided by number 2 and whenever a new square search space is defined it is always checked that any part of the square is not out of bounds of the space of all possible alignments. If it is out of bounds than the square changes simply into rectangle by clipping the outer parts.

This technique is very simple and efficient, on the other hand it can easily be fooled with local maximums. If the shape of the visualising function has a few notable local maximums mostly with the similar base size as the peak containing optimal alignment, it can easily create a square after initial search which would be located over some of that local maximums. Thus the method will fail to find the global maximum. Fortunately the shapes of selected methods are very kind to this optimisation technique as they are "single peaked" with absence of notable local maximums. It is partially due to the nature of processed data and due to the behaviour of selected evaluation methods/functions. This technique has no initial starting point, it simply needs the evaluation function and its parameter space to find the solution.

We have observed that really a speed up can be gained when yet $n = 4$. Higher values work, 6 or 8 for n is fine and $n = 12$ or $n = 16$ work mostly too. But higher values won't bring any principal speed up, see table 4.1 (on page 53). Higher values also rise the possibility of misregistration even when the visualising function has only one peak. When the initial step is to big it can occasionally happen that the corresponding grid of tested alignments will simply overdraw this peak, in other words the peak base is too narrow so it whole fits inside one cell. After that this technique will start a new search with half search step over a square space with some alignment at its centre. This square will be rather small in compare to the initial space and it may easily happen that the peak will be out of bounds of that square. In this case it is impossible to find the optimal alignment when following given scheme of n-step technique. The search position will always remain in the first defined square.

To avoid a search in the local maximum, lower values for n are suggested. This way the method will have better overview of the entire initial search space and hence can better decide where to put the first square and so where to begin with more detailed search. This suggestion is also supported by the fact that even small numbers can gain big speed up. Therefore we stayed with $n = 4$ even when higher values are working.

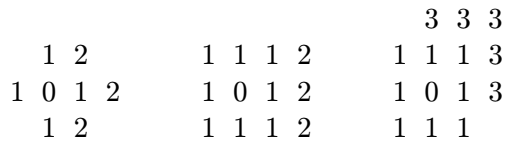


Figure 4.1: Both pictures are showing the new alignments that must be computed in order to have all information needed for correct next movement decision. The already computed ones are labelled with numbers 0 and 1. New alignments for movements along axes (to the east in our example) are denoted with number 2 and are shown in the left picture for the 4-directional version and in the middle picture for the 8-directional version. In the right picture the alignments that need to be evaluated when moving the traverse directions (to the north-east in our example) are labelled with number 3.

The gradient ascend optimisation technique

The second optimisation technique is the gradient descend technique which we use the opposite direction, i.e. we are going "up to the hill" instead of going "down the hill." This the reason why we call it gradient *ascend*. The technique requires an initial starting point, evaluation function and its parameter space. Unlike the previous technique there is one requirement more, the starting point.

The principle of gradient *ascend* is from given point in the searched space (at the very beginning this is the starting point) estimate the derivation of evaluation function in all possible directions and move to the adjacent point in that space in the direction in which the derivation is the highest. So the actual point (in our case the alignment) is travelling through the search space (all possible alignments) until it reaches the position from which is no better way (the optimal alignment). No better way means that there is no direction in which the movement will rise the value of evaluation. The derivations of evaluation functions are approximated in our implementation with simply the difference between the given point and its adjacent alignments.

We are using the 8-directional search but there exists a 4-directional search too. In both cases this means the number of possible movements that can be done from the given point. The latter possibility allows us to move either in the y coordinate (to the north or to the south) or in the x coordinate (to the west or to the east). That are 4 possibilities, the 8-directional movement adds the possibilities to move in both coordinates at once (to the north-east, south-east, south-west and to the north-west).

What is the difference? Please look at the figure 4.1, the difference is in the number of evaluations that needs to be computed in order to make a decision for next movement. Each evaluation costs time and so we are trying to minimise the total count of evaluations needed for finding the optimal solution, in fact that's the definition of optimisation itself. When moving along axes it doesn't matter what directional approach we are using. We must always evaluate three new alignments (labelled with number 2 in figure 4.1). But in the 4-directional approach the traverse movements needs to be combined from two "basic" movements. That means we must first evaluate three new alignments and then again new three for the second part of movement, together for traverse movement we must compute evaluations of six alignments. But when the 8-directional approach is in use then in the given situation we need to compute only five new alignments (all labelled with number 3

in figure 4.1). So we have saved us one alignment evaluation.

Well, when some cache for evaluations of computed alignments is introduced the difference will become eliminated, because in that traverse movement one alignment is computed twice in 4-directional approach without cache. Even cache for last few (less than ten) alignments will help and won't be even to hard to implement while still very fast for the search through.

We haven't implemented cache but instead the 8-directional approach. The difference when programming is not terrible and the savings are always fine. The disadvantage at the initialisation when four more alignments must be computed is quickly defeated.

This technique is rather easy to implement and very efficient when it is started anywhere in the ZAF part of visualising function. It also supports the requirement on the shape of the visualising function (announced in the previous section on page 39). The broad fundament/base of peak gives better chances to let the technique convert into the optimal alignment while at the end the search becomes quicklier and quicklier as the surface of peak becomes more steeper.

4.2.3 Two way optimisation

In section 3.2 (on page 18) we have noted about accuracy of the scanning table Märzhäuser 2D. It helped us very much indeed. During the tests we have observed that the optimal alignments (respective the parameters of translation of coordinates that represent each alignment) are nearby. Every image composition (the set of images with their relative positions given, a table is typical representation) seems to have its own optimal alignments set. In this set all optimal alignments found during whole image composition are stored.

The set is very small, its cardinality is usually less then 15. The count becomes even smaller when we divide this set (elements are vectors (c_x, c_y)) into two sets, the first holding values of c_x and the second holding values of c_y. In such event each set has less or equal than 5 elements. This is a situation in Bohunice, the scanning table is accurate and the quality and nature of pictures in one image composition is balanced, i.e. all images have equal properties regarding the registration process. We have used this fact for two things.

The first one is the optimisation techniques combination. During the whole image composition we were calculating the average from coordinates of each optimal alignment found. In fact there were two such calculation, one for horizontal registering and one for vertical registering. We call these averages from optimal alignments the default alignments.

The two way optimisation is then nothing else than for first image registration in given direction (horizontal or vertical) compute the optimal alignment and make it also the default alignment. From now for every other registration in a direction which has its default alignment defined we will use the gradient ascend technique with the default alignment as its starting point. There is a condition requiring to have the default alignment defined in order to use the gradient ascend. That's obvious but we have still written it there to let reader realize that switch back to n-step optimisation technique is possible during the whole image composition process. Just imagine the first five registrations are all in horizontal direction, the sixth one is vertical. There will be no default alignment for the sixth registration because it is in other direction. Without the condition there it won't be possible to return back to n-step technique and so we have decided to emphasise

	Entire search	4-step technique	8-step technique	16-step technique	Gradient ascend
SSC	55.323s	3.372s	1.027s	0.465s	0.031s
	17.956s	1.171s	0.281s	0.124s	0.011s
SAVD	44.746s	2.701s	0.821s	0.369s	0.036s
	15.851s	0.964s	0.248s	0.111s	0.288s
NCC	63.568s	3.995s	1.215s	0.546s	0.050s
	22.988s	1.746s	0.411s	0.195s	0.101s
CR	74.594s	4.746s	1.441s	0.648s	0.060s
	36.696s	2.261s	0.679s	0.305s	0.110s
MI, histogram	121.567s	7.835s	2.296s	0.990s	0.085s
	67.329s	4.367s	1.272s	0.532s	0.229s
MI, Parzen	873.925s	56.460s	15.468s	5.384s	0.352s
	486.021s	30.657s	8.450s	2.939s	0.589s

Table 4.1: The optimisation techniques speed up compare. Each cell in the table above has two numbers. The upper one is the time required for one registration process (optimal alignment search only, no stitching) from a program compiled without any compiler optimisations. The lower one in each cell is the time required for the same operation when program was compiled with compiler optimisations. The column "Entire search" has times when no optimisation was enabled, i.e. all possible alignments were tested. The middle columns has times for the same registration (two same images) when the n-step optimisation technique is enabled with $n = 4, 8, 16$, the default alignments were obtained this way. The column "Gradient ascend" contains times required from the 8-directional version of this technique starting from previously searched default alignments, the data to be registered were two images from the same image composition but not those two from previous two tests. It should not be those two because at such situation the time required for gradient ascend optimisation technique will have have no meaning, the optimisation will start at the optimal alignment and therefore finish extremely quickly. Both images had dimension 1232×972 with maximum possible overlap set to 11%, thus the space of all possible alignments was defined by equations $0 \leq c_x < 125$ and $-106 < c_y < 106$. Notice the upper bound for x coordinate, it counts with restricted region of strength 10 pixels ($125 + 10 = 135 \approx 0.11 \cdot 1232$). The data were obtained on personal computer with Intel(R) Celeron(TM) processor running at 1500MHz with 512MB SDRAM running at 125MHz, 3014.65 bogomips, Linux OS with kernel 2.4.20. Important note, while the compiler (GNU `g++` in version 3.0.4) non-optimised program successfully found optimal alignments in each situation, the compiler optimised program didn't. In the table holds that whenever the time of optimised program was worse than time for non-optimised then the optimal alignment wasn't found. Strange is that this seems to happen only with gradient ascend technique.

it this way.

When the accuracy of scanning table is high, so these two sets for values of c_x and c_y of all found optimal alignments are rather small, then we have another notable speed up. The table 4.1 will clarify this when certain numbers are seen.

The sum of evaluations saved

Just one more note about the number of evaluations needed for one registration. Both techniques represent always a reduction of evaluations. Both are also dependent on the size of the space of all possible alignments which we will for this while denote by $C_x \times C_y$. Both sets C_x, C_y are ranges in this context, they should be "discretely continuous" in the term mentioned once in this paper, i.e. for example in range of natural numbers from 1 to 10 there are no numbers missing (no holes), the sequence is complete.

The n-step optimisation is clear, its work is strictly prescribed and there are no possibilities to function other way. Hence the count of evaluations required for finding the optimal solution can be precisely formulated as

$$S_n \leq \lfloor |C_x|/n \rfloor \cdot \lfloor |C_y|/n \rfloor + \sum_{i=1}^{\lfloor \log_2(n) \rfloor} \left\lfloor \frac{2 \cdot \lfloor n/2^{i-1} \rfloor}{\lfloor \frac{\lfloor n/2^{i-1} \rfloor}{2} \rfloor} \right\rfloor^2 . \tag{4.2}$$

The equality holds whenever the first square established fits whole in the space of all possible alignments.

The first summand represents the initial grid search, the $\lfloor$ and $\rfloor$ are alternative. There should be some kind of rounding to natural numbers as the count naturally is a natural number. But the choice how to round is arbitrary, it depends on the implementation. If we decide always to test the boundary of possible alignments space then the $\lceil$ and $\rceil$ should be used. We have implemented the initial search as starting the grid search at alignment $(\min(C_x), \min(C_y))$ and test every n-th alignment in both directions (along axes) as long as we are inside the searched space. If the size is not divisible by n than alignments lying on half of boundary aren't evaluated in this initial grid search.

The second summand represents the count of evaluations performed until the step size is equal to 1, at this last level optimal alignment is determined. There are $\lfloor \log_2(n) \rfloor$ levels, searched squares are always decreased to one fourth of the previous size (edges are shorten to one half of previous length). After the initial search when first square is established and from this moment on it holds that the edge size is $2j$ and the corresponding square is searched with step $\lfloor j/2 \rfloor$. Hence the number of performed evaluations is $\lfloor 2j/\lfloor j/2 \rfloor \rfloor^2$. The determination of edge size in i-th level of optimisation relies on the statement that both sequences $a_{k+1} = \lfloor a_k/2 \rfloor$ and $b_k = \lfloor m/2^{k-1} \rfloor$ starting with $a_1 = b_1 = m$ are the same.

An example from Bohunice. We use $n = 4$, then for situation from table 4.1 we have $C_x = \langle 0, 124 \rangle$, $C_y = \langle -105, 105 \rangle$. The S_n can be therefore rewritten as

$$S_4 = 31 \cdot 211 + \left(\frac{2 \cdot 4}{4/2} \right)^2 + \left(\frac{2 \cdot 2}{2/2} \right)^2 = 6573. \tag{4.3}$$

Compare it to $|C_x| \cdot |C_y| = 26735$ needed for the entire space search.

Don't be mistaken that the ratio of expected evaluations and the ratio of required time is not the same. The reason for this is that the size of the actual overlap given by certain

right-now-evaluated alignment is variating. The bigger overlap the more time it takes to process all of it but not proportionally, the size ratio of big versus small overlaps do not exactly follow the ratio of time spent when evaluating alignments which define given overlaps.

The gradient ascend optimisation technique cannot be predicted as accurate as the n-step optimisation. In our situation we are expecting the shape of the space of all possible alignments to be a rectangle. The worst case estimation can be seen from next example.

To let the technique travel the longest possible way it is enough to imagine that the evaluation of alignments in the space is arranged in the following manner. Thus we can imagine that the starting point will be in bottom left corner. The evaluation of alignments will be "arranged" in horizontal lines. The first, the fifth, the ninth (and so on) lines will contain evaluations growing from left to right, the third, the seventh, the eleventh (and so on) lines will contain evaluations growing the opposite direction. Even lines will be zero except for some their endings. Right endings of the second, of the sixth (and so on) lines will contain evaluation value between their upper and lower neighbourhood. Similarly for the fourth, the eighth (and so on) lines with their left endings. Now we believe the reader has got the idea. The gradient ascend technique will move on the "evaluation ranges" across the entire space, odd lines are the ranges and the endings on even lines act as bridges to over-pass the valley. Bridges must be up to the hill to prevent the algorithm of this technique from stopping at this point. This ranges will be defined as long as they will be part of the searched space, the alignment with highest evaluation will on the end of the path.

The worst count required to find the maximum value in such artificial example is also the general worst count S_{grad}. Hence it holds

$$S_{grad} \leq \frac{1}{2}|C_x| \cdot |C_y|. \tag{4.4}$$

In real application similar artificial evaluations of parameter space is hardly to expect. We do not expect such specialities as the even lines in the given example which task is just to prevent the algorithm from going straight to the optimum. Instead smooth visualising functions are to be found.

The expected worst case we believe is therefore equal to the count of evaluations required to travel relatively straight across the space, thus the value of $\sqrt{|C_x|^2 + |C_y|^2}$ can be (we believe) safely considered to be the expected worst case.

In Bohunice we have gained great speed up with this technique. Due to the behaviour of the imaging system we can predict where the most probably will be the next optimal alignment. The time required for registration is really a minimum. From tests we found out that to reach the optimal alignment less than 10 evaluations must be computed.

4.3 Empty fusion

During the tests we have also observed that there are problems when registering pictures which have in their common overlap mostly or solely the background. This is such kind of images where even human cannot for sure say what is the correct alignment. The evaluation methods behave differently but always decide some bad alignment to be the

optimal one. The idea how to solve this problem was surprisingly simple. We have used in such cases the default alignments. So this is the second thing for which the accuracy of scanning table helped us very much. This concept is pretty nice but requires a few non-trivial decisions to make.

We must first somehow find out that currently we are registering two images which contain mostly the background. And when we find such images we must decide what to do with them. The answer for the latter is already answered, we use the default alignments. But this introduces a new question: How do we find the default alignment; what shall we do, when the very first registered images in the given image composition will be those with majority of background in their overlap (so we cannot use the default alignment)? The answers will be in order in which the implemented program solves such problem.

4.3.1 The measure of suitability for registering

When images are read into the memory (well, only overlaps from them), they are converted into gray-scale. After that they are measured. The measure should be fast and should describe the suitability for image registration, the higher value the better and as we will see later the better to start registration with this image. We will use the information from this measure to establish the order in which will be registered images from the whole image composition. The goal is not to point out images with background but conversely point out the image with the best suitability because this image (and its neighbourhoods) will define the defaults alignments.

The measure we have implemented is simply a sum over the entire overlap of given image, the sum is counting all differences between values of neighbourhooding pixels, every neighbourhooding pair only once of course. The idea behind is that the difference in pixel intensities is the key that helps human brain to establish the optimal alignment.

Imagine following situation, we have two white sheets and we want them to match somehow. When both or even one sheet is absolutely empty then we have no chance to decide the optimal alignment. When we draw a single dot on each sheet we learn something. We can align these sheets in the manner in which the first dot is above the second one. But we still don't know the rotation of sheets (in general, in Bohunice we don't have to take care). If we put another dot pair on both sheets we can establish the rotation. So it seems the more changes (and dot is a change of values) in that white space on the sheet the more we know about the alignment, except the situations in which these dot pairs don't align and then the second dot pair confused us instead of helping us. So let's presume every object pair will be matching for next reading. Let's continue, imagine we'll write a line on both sheets, will it be more help? Sure, dots can easily be overlooked or the scanning system can create a new one (the dark noise present in CCD cameras for example, but obviously in different situation, not the white sheet). The lines theoretically don't necessary bring us more information for determining the alignment but still it can help when some noise is present around. Then we can write another line or begin with some more complicated objects. For human brain, and that term human brain is important to emphasise, these added objects help to improve the capability to find optimal alignments. Now try to use our measure on all that intermediate levels of drawing on two white sheets. As we were trying to explain the more perceptible objects are on the sheets the better it is for registration. The measure will give us the same result because

each object represents on its boundary some change in pixel values, the more objects on the sheet the more differences in values of pixels representing that sheet.

Challenging question is whether the higher certainty for humans when registering will also mean the higher probability for evaluation methods to "point out" the optimal alignment. We don't know the answer unfortunately. We can only say from our results in Bohunice that it seems so, the reader is encouraged to see the figure 4.2.

4.3.2 The order for registering in image composition

So now, after measuring all overlaps of each image present in the given image composition, these (overlaps) have assigned some number. The very first image that will be used for the first registration and also for the definition of the default alignments will be that one which has the highest sum from assigned numbers of its overlaps.

The rest of order is determined using the following algorithm which is almost the Prim algorithm for determining the minimal spanning tree. That is to say we can imagine the images as a nodes of a graph with valuation of its edges, edges "will be" overlaps of images. Edges will not be oriented and will exist only and only if the given two nodes (images) are adjacent in the horizontal or vertical direction. The valuation of edge is simply the number assigned to one side of overlap plus the number assigned to the counterpart side of overlap. The difference in the following algorithm is in the fact that it is not trying to create a minimum (well, if we had to create then maximum) spanning tree but we are interested in the order in which we would add nodes into the maximum spanning tree.

The notation will be Im for the set of all images, for $i \in Im$ the $Pr[i]$ is the array which determines whether given image i would be in the spanning tree. And $Val[i]$ is the number which is assigned to every overlap, it is the already best number "under" which one would get into this node from the spanning tree at that moment. The values of edges between the nodes i and j will be denoted by $Mes[i \leftrightarrow j]$, for j it holds and also introduces new notation $j \in Sur[i]$ where $Sur[i]$ is the set of all images which are to the right, down, left or up from the i, i.e. the set of images which must be registrated with i.

1. Initiation: for every $i \in Im : Pr[i] := 0$ and $Val[i] := -\infty$
2. Starting point: $i' := \max_{i \in Im}(\sum_{j \in Sur[i]} Mes[i \leftrightarrow j])$
3. Iteration starts: while $\exists i \in Im : Pr[i] = 0$ do
4. forall $j \in Sur[i'] : Pr[j] = 0 \wedge Val[j] < Mes[i \leftrightarrow j]$ do $Val[j] := Mes[i \leftrightarrow j]$
5. $Pr[i'] := 1$, print(i')
6. $i' := \max_{i \in Im : Pr[i]=0}(Val[i])$
7. Iteration ends: end while

The final order in which are images processed in the given image composition is the order in which images are printed in step 5.

Now we know how do we create the order in which we register the images. We have learnt that for this purpose we had developed some measure. We also know that when

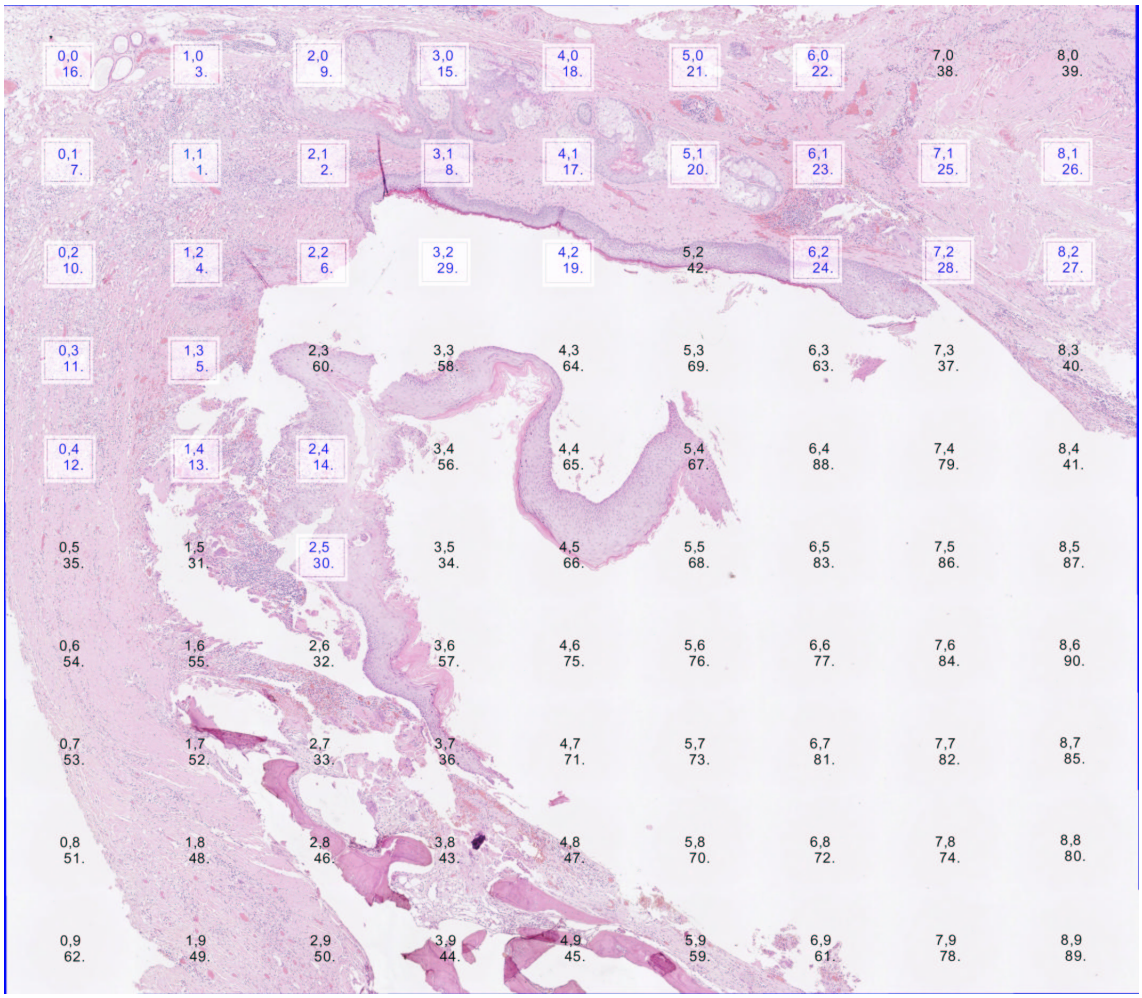


Figure 4.2: This is some image composition of 90 images. Each image is labelled with its position and the order number. The image number 1 is at position $[1, 1]$. This figure should show the reader how does work the measurement combined with algorithm for creating order for registration. The first one third of all images in established order are designated with their labels in lighten frames. Notice the "empty fusions" have the highest numbers, thus these are "embedded" inside at the end of registering process like filling the gap (using the default alignments).

we detect an image with the majority of background in its overlap we will align it using the default alignment. Finally we know the whole answer for question at the beginning of this section and that is from where we decide the values for default alignment(s) — horizontal and vertical. Each default alignment comes from the very first registration result in the whole image composition in the given direction. We know that previous measure in combination with the modified Prim algorithm ensure the property that the very first registration is perhaps the most suitable (we didn't prove that) and hence (we hope) the least inclinable to misalignment. During the following image registrations we still administrate the default alignments so it holds that default alignment is the average from all previously computed alignments all in given direction.

Final debt is how do we recognise the image with "empty fusion." We find such image when the computed alignment is outside the given threshold. When the default alignment is (C_x, C_y) and the right-now computed alignment (c_x, c_y) than it is considered to be misalignment when $|C_x - c_x| > T$ or $|C_y - c_y| > T$. We make use of the benefit of the accuracy of scanning table and hence we use that table as the detector of images with background, in Bohunice is the $T = 10$ more than enough. We actually solve the problem of potential misalignment after the misalignment occurs unlike trying to avoid it. For example avoid it by detecting the nature of image and according to the detection result, then make decision what algorithm will be used.

4.4 The order for establishing final image positions

The modified Prim algorithm from previous subsection ensures a property which is very useful for efficiency of implemented program. Before that we will allow ourselves small technical note.

The easiest approach to keep the program design as simple as possible is to perform the image composition in three steps. In the first step one has to decide the parameters of transformation of coordinates, that's obvious. In the preceding text we have seen a few hints how to do that. The second step is to determine the final position of top left corner of every image in the composition using the pre-computed parameters of coordinates transformations. In the third step we make use of positions and somehow compose the final big image. When we notice the property of the mentioned order creating algorithm, we can perform the first and the second step at once. As stated it is the modified Prim algorithm for creating the minimal spanning tree of given connected graph. The modification is in the fact that actually were not building any real spanning tree when performing the algorithm.

The graph consist of images from composition in our case. The idea of Prim algorithm is to begin with some node, it doesn't matter which one it will be because in the spanning tree must be every node present. We are constructing a set of nodes which holds all nodes that are *already* in the rising spanning tree, at the beginning there is only one point. The iteration process adds one point in every cycle, it adds such point which is *reachable from that set* with the lowest valuated edge. The iteration stops when the set has all nodes from given graph. The order in which we are computing the registration is the order in which images (nodes) are added into that set during the iteration process.

Two facts were emphasised in the previous paragraph. Both are saying the same

thing, for every image but the first one in our registration order it holds that it must have at least one its neighbourhood somewhere before in the order. We start the modified Prim algorithm with the most appropriate image (the step 2) and and that's it. The very first image in the registration order can receive arbitrary position, we have set it the centre of coordinates $[0, 0]$. As the registration process is performed each image after the registrations with all its neighbours can receive its final position. Because of there must exists at least one of its neighbourhoods that have the final position already determined, then one has to simply use the information from found alignment.

4.5 Image stitching

4.5.1 Possible problems when image stitching

We have finally got in this paper to the second part of every image composition, we finally got to the image stitching. Let's consider the horizontal stitch of two images during this section.

There can be perhaps lot of different approaches, the most simple is to overlay one image over the another one. Depending on the situation and demands but mostly this solution won't be acceptable because the overlay will be noticeable. It will be noticeable because of image incompatibility which is present in virtually every imaging system. The incompatibility is mostly the displacement of same objects even at the optimal alignment. Well, optimal in term that there isn't any better alignment. Sometimes the colour tint can happen.

When colour adjustment should be performed, we may use the intensity conversion function I introduced in 3.1 (on page 9). It will simply repair the colours by mapping the correspondence between them and applying this information on one image. The I function must be beforehand determined manually and our implementation didn't make use of this function.

When misplacement of objects occurs there is no easy help. The registered image can be for example broken down into a few smaller pictures and the whole image composition process can be applied on the reference image and each part of registered image separately. After all parts are processed the final composition will take place. But sadly even when we've used this solution as an example, we don't predict it good results. It is rather complex, and it becomes yet more complex when fully automatics is demanded. There are also some experiments when morfing images but we haven't studied that.

Instead we had concentrated on the image data conversion, i.e. we are not trying to move the data so the overlap will be less noticeable but we are trying to over-pass from the first image data onto the second image data. We call that with the term sutura camouflaging because of the fact that we are actually trying to hide the conversion of data.

4.5.2 The situation in Bohunice

In Bohunice the misplacement occurs. When objects are perfectly aligned in the upper part of overlap, the objects from lower part need to be shifted by few pixels — usually less then 10 pixels.

After the final positions are established, we know perfectly how broad will be the overlap and where it will be positioned. Overlaps are very broad (usually around 50 pixels), this can be also seen from the visualising tool. In horizontal case it is the distance in x coordinate from the top of the peak (from the optimal alignment) to the maximum value of c_x of every possible alignment. And plus 10 pixels because this is the narrow strip of alignments that excluded from searching. So we have plenty of room to convert from one data onto another one.

4.5.3 General converting function

We have implemented the simplest method, when stitching we are trying to smoothly traverse from data from the first image onto the data from the second image. This is performed for every pixel in common overlap using the weighted sum of corresponding pixel pair values. The weights are controlled by the position in the overlap. This way we have a very general framework for implementing the stitch process because in fact the weight functions can be arbitrary.

Suppose for given two images we have their final positions and so we have their common overlap defined as a rectangle with its width $W = card(X_A)$ and height $H = card(Y_A)$. Recapitulate X_A and Y_A will be the coordinates from overlap for the reference image $A(x, y)$, X_B and Y_B will be the coordinates from overlap for the registered image $B(x, y)$, $T'(x, y)$ will be the optimal alignment transformation of coordinates. Finally $C(x, y)$ will be the new merged overlap which will be in the end embedded between the rests of the reference and registered images, the definition will be

$$\forall x \in X_A, y \in Y_A : C(x, y) := w_1(x, y) \cdot A(x, y) + w_2(x, y) \cdot B(T'(x, y)). \tag{4.5}$$

The property of weights should be met, i.e. $\forall x, y : w_1(x, y) + w_2(x, y) = 1$. We will call the $A(x, y)$ and $B(x, y)$ the original pixel values, the $C(x, y)$ will be simply the computed or resulting values.

Often approach when performing the stitch in horizontal direction (we have left and right images) is to fix the weight function at each y coordinate, hence it holds $\forall x, y : w_1(x, y) = weight(x) = 1 - w_2(x, y)$. The weights can be than defined only through the $weight(x)$ function which defines the value for the whole column y. This function describes how to merge one pixel row from the overlap by defining both general functions for every row at once. In this text we will refer to the $weight(x)$ function as to the one-row weight function.

4.5.4 Sutura camouflage

For the purpose of sutura camouflage we had set more requirements on the general weight function. These also should be held for one-row weight function. It might be better first to overview the figure 4.3 before reading on.

Firstly, the function should be smooth. We are trying to overcome from one overlap data to the other overlap data. If both overlaps are identical than it doesn't matter how we manage that. The property for weights will ensure that the resulting pixel value will be the same as both original pixel values from which we compute given one. When images are not identical, the changes will occur. These will be changes from both original pixel

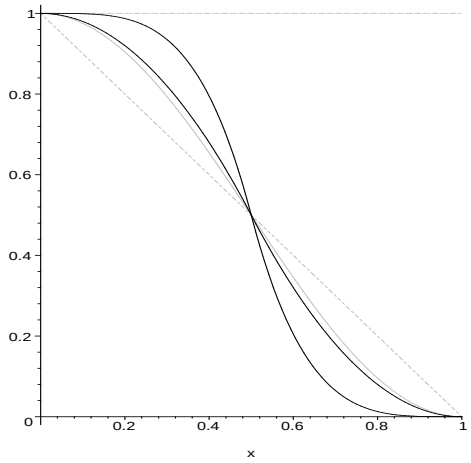


Figure 4.3: Several one-row weights functions $weight(x)$. The horizontal dashed line in gray on top is the representative for overlay of one image over the another one. The gray dotted is a linear, the gray solid is cosine function $0.5 + (cos(\pi \cdot x)/2)$. The thin black solid line is the 2nd order polynomial while the thick is the 4th order polynomial. Both polynomials are made up from two parts. The first part is actually $1 - p_n(x)$ where the $p_n(x)$ function is polynomial $1/2 \cdot (2x)^n$, $x = 0 \ldots 0.5$. The second part is the first part rotated by 180 degrees around the point $[1/2, 1/2]$.

values, these might be noticeable. The smoothness aims against the rapid changes. The weight should slowly changing its value so the eye will get used to the resulting changes. When the appropriate length in which the weight changes its values is set the over-pass will be less noticeable.

Secondly and not surprisingly, we demand that the weight function $w_1(x, y)$ at fixed y coordinate will start with value of 1 and converge into 0. Hence the second weight $w_2(x, y)$ at the same y coordinate will behave exactly conversely.

These two requirement are the core of our merging/sutura camouflaging algorithm. The figure 4.3 shows functions $weight(x)$ that were tested. Not mentioning the weight representing overlaying one image above the other one, there are linear, cosine and polynomials (of 2nd and 4th order) curves. The interval in which the weight gets from its initial value to its final value is exactly 1. Hence allowing us to stretch this interval into an arbitrary length. Let's denote some one-row weight function from figure 4.3 as $Weight(x)$ where $x \in \langle 0, 1 \rangle$. The overlap can be then defined as

$$\begin{aligned} \forall x \in X_A, y \in Y_A \quad : \quad C(x,y) := Weight\left(\frac{x - \min(X_A)}{card(X_A)}\right) \cdot A(x,y) \\ + \left(1 - Weight\left(\frac{x - \min(X_A)}{card(X_A)}\right)\right) \cdot B(T'(x,y)). \end{aligned} \tag{4.6}$$

This solution is fast and acceptable. The disadvantage is its dependency on the width of overlap which makes it harder to find the optimal curve for weight function. The slope of weight function is important because it defines the distance in which we will over-pass from the first data onto the second data.

When the slope is too slow than the distance gets longer leaving us large space where the over-pass was performed. In large space it is more likely to notice something, usually the "double effect" occurs. The merit of this effect is the fact that for pixels which defines values of $x \approx 0.5$ it holds $Weight(x) \approx 1 - Weight(x)$, the weight function for data from the reference image and the weight function for data from the registered image are roughly equal. Result is that when same objects in such area of overlap are not exactly aligned than they are pictured twice nearby with half intensity, they look like they are vibrating.

On the other hand when the slope is too fast than the distance will be short and notable changes will occur. If the area splits some object in the image and the left part and right part of this object aren't aligned very tight, then for example the edges of this object as we follow them from left part to right part will suddenly "turn" to meet its counterpart in the area where the weight function for the registered image is in full strength, i.e. is equal or almost the 1. This will almost look like a step in the edge which we will immediately notice.

We believe there exists always some distance in which the over-pass will be the least noticeable. If data are not too incompatible than it is possible to perform an over-pass which will be hardly noticeable. In our test we have found that the curve constructed from two 2nd order polynomials works fine for smaller overlaps, typically given by 6% or less. For bigger overlaps the same weight function but created from 4th order polynomial is used instead. See figure 4.4 for their comparison.

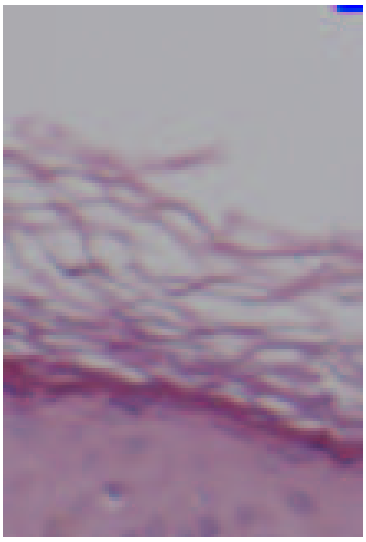
Notice the behaviour of each weight function in that figure. The overlay image demonstrates the incompatibility at given alignment. Linear weights demonstrates the "double effect" as this weight is changing its values very slowly while the overlap was pretty wide. The 2nd order polynomial weight seems to have some difficulties with this kind of data, such effect happened only in this part of entire overlap. The rest of overlap was the interior of pictured cell and the stitch was there quite good unlike in the shown part. The 4th order polynomial weight exhibited pretty good sutura camouflage. The length in which the weight function changes its values is notably short than in the linear weight function. Nevertheless the double effect is still noticeable as long as the image is zoomed in. The optimal distance is still shorter as we will see in the next paragraph.

The "zig-zag" method

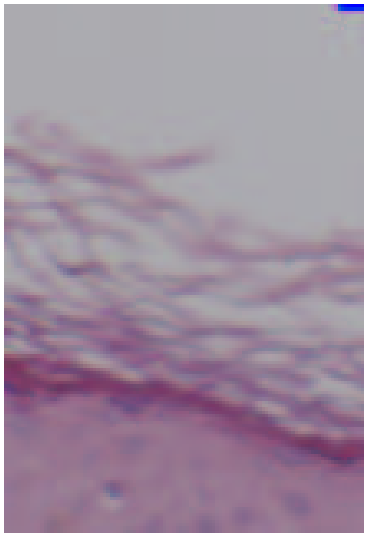
We had firstly used the approach of one-row weight function, later after some remarks we made the decision to use the method suggested by MUDr. Feit. We have implemented the "zig-zag" camouflaging method which is the general weight function. This method has its roots in the 2nd order polynomial curve. The curve remained but the distance in which this curve is computed became constant, for all tested images the most appropriate distance seemed to us to be the distance of 20 pixels.

This improves the feel from the stitch, from the camouflage. Better results still can be yield. What does help us to recognise the stitch? When we use the one-row weigh function, even in the narrow strip of 20 pixels, we still have over-pass in each row aligned one above another one. For example we wouldn't normally notice an over-pass in some certain row if there weren't similar row with more noticeable over-pass. This second over-pass might not be good, it might attract our attention and after that we might also notice the first row. The idea of "zig-zag" method is now clear, we will simply prevent neighbourhooding

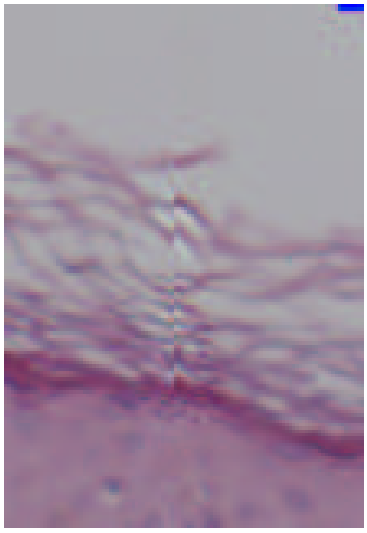
Overlay:



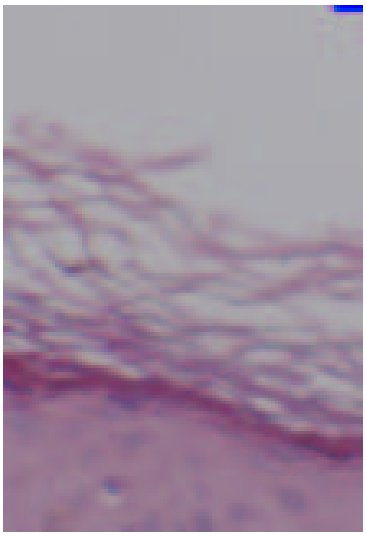
Linear:



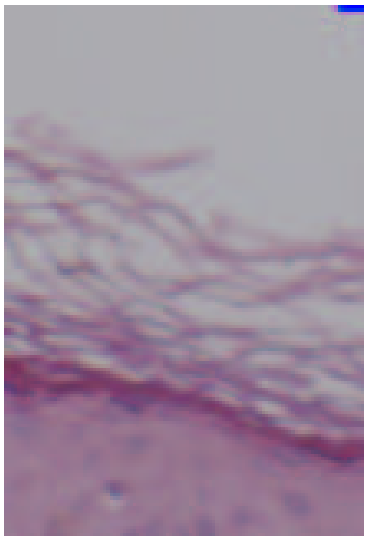
2nd order polynomial:



4th order polynomial:



"Zig-zag":



"Zig-zag" visualised:

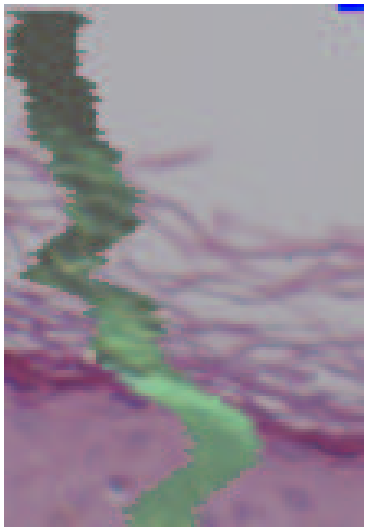


Figure 4.4: The figure shows six different parts of two images composition. All examples are part of two images stitched at always the same alignment. In each case different weight functions were used, see the text above each picture. The "Zig-zag visualised" is the ordinary zig-zag method with the exception that each pixel in the area where weight functions are strictly less than 1 is the inverted resulting pixel value, i.e. we are using 8-bits per one pixel and so when $p \in \langle 0, 255 \rangle$ than p inverted is $255 - p$. This way the bottom right image demonstrates sample pass of the zig-zag method, it emphasise only such pixels where the camouflage was actually performed. The overlap was 101 pixels wide, each image is a copy of constant rectangle from constant position from that overlap.

over-passes to be aligned into a column as they would normally be. One request arisen, the zig-zag line along which the over-passes are performed should be continuous/smooth. The reason for that is to avoid performing over-pass on the left edge of overlap while the next over-pass will be in the right edge. The effect of hiding the place of over-pass will be lost as there will be suddenly change of data source (part of row from the reference image, next row at the same column from the registered image, next row again from the reference image) thanks to image incompatibility.

The formalism for this method. The 2nd order polynomial presented in figure 4.3 we will denote with $poly(i)$, $i \in \langle 0, 1 \rangle$. The zig-zag line will be represented with some continuous function $x = line(y)$, $x \in \langle \min(X_A), \max(X_A) - 20 \rangle$ and $y \in Y_A$. The counterpart weight $w_2(x, y)$ will be to meet the weight property as $w_2(x, y) = 1 - w_1(x, y)$. The general weight function w_1 simulating the "zig-zag" method is as

$$\forall x \in X_A, y \in Y_A : w_1(x, y) = \begin{cases} 1 & x < line(y) \\ poly(\frac{x - line(y)}{20}) & line(y) \leq x \leq line(y) + 19 \\ 0 & x > line(y) + 19 \end{cases} . \tag{4.7}$$

Chapter 5

Conclusion

This thesis was all about the image composition problem. It was written perhaps more as a manual which should help when implementing a solution regarding this problem. We believe that every creator of a system which purpose is to compose images will be able after reading this text to create a new system, with prior decision what matching methods to use and why to use them. He/she has also learnt some hints that may help him/her with development of fast, reliable, and accurate system in term of quality of final composed image.

We have tested major algorithms for evaluating the optimality of given alignment. These algorithms are representatives of the so-called statistic-based family, i.e. they compute various statistics directly on the raw image data instead of extracting features from images in order to discover the optimal alignment from these features. The image pre-processing can be of course performed in this case but our results prove that it is not necessary.

We have also shown that pre-processing is not required even when registered images are not exactly identical. The selected algorithms mostly did their job when the registered image was significantly brighter, the texture of registered image wasn't sharp or when one image was disturbed by equally spread noise. The correlation ratio algorithm for estimating the optimality of alignments using basic statistics have passed all tests and therefore we would suggest it as the most general algorithm for registration.

The time consumption of the entire image composition problem is closely related to the registration process. The registration is the time most demanding step in image composition. The reason for that is in the amount of data that must be processed during the evaluation of every alignment. We have published two tables in this thesis which are comparing selected matching methods by time. One can from these also imagine the amount of time that is required. The first table shows times for pure search of the entire space of all possible alignments, the second table shows times when optimisation techniques were enabled.

The second table brings encouraging news. Up to three orders of magnitude speed up can be achieved for the composition problem. This result makes use of special two level optimisation which in addition make use of some property of given image scanning system. Unfortunately this optimisation is probably not applicable in general because of that property. The property is high accuracy of scanning table which for example cannot be expected when composing arbitrary images without prior knowledge of their source.

Finally we have found a satisfactory solution for image stitching when these are almost identical. Our solution is fast and accurate in term that it preserve the information found in the data. That means for example that it preserves shapes and colours of objects. The innovation of this method is in its uniqueness where the stitching is actually performed rather than the way it is performed.

All results in this thesis come from a program that was developed for this purpose. Data were used from the Brno Faculty Hospital — Bohunice. The nature of images produced by the scanning system in Bohunice are described here, two rather typical images are also shown in the Appendix.

In the Appendix one can also find the documentation for this program which is describing its usage as well as its capabilities. It is programmed in C++ and it provides interfaces for further extensions of matching or sutura camouflaging methods. The matching methods do not necessary belong to the statistical-based family.

Image composition is a complex problem, mainly because it contains an image registration as its sub-problem. We have observed some difficulties with the mutual information matching method. Mainly with version which uses the Parzen-window technique for estimating probabilities.

This method is interesting in two things. Firstly it uses the theory of information which we believe is powerful tool. Secondly the Parzen estimator does not require the entire overlap data to estimate the evaluation of corresponding alignment. This will probably induce another speed up. The speed up probably won't happen in the application described in this thesis. It may occur when larger images are used or when the image composition becomes the image merging, in this application the overlaps will enlarge to the whole size of images.

When image merging or comparing should take place, from results of this thesis we presume that concern should be directed on some parameters estimation for the mutual information evaluation or some more efficient optimisation technique which will handle very general shapes of visualising function.

Appendix A

Sample images

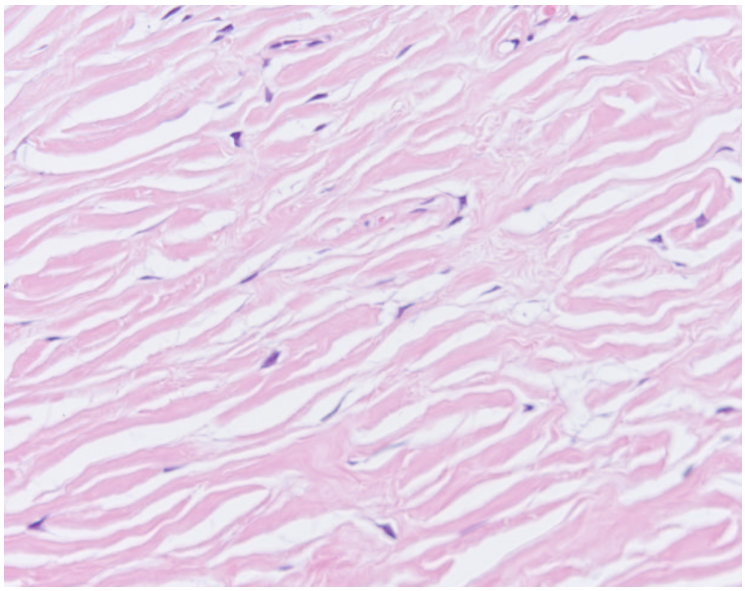


Figure A.1: This is the reference (the left) image which was used in all the tests found in subsection 3.4.2 starting at page 39. The stripe on the right side/edge of width of 85 pixels coming from top to bottom will be the maximum overlap in the registration.

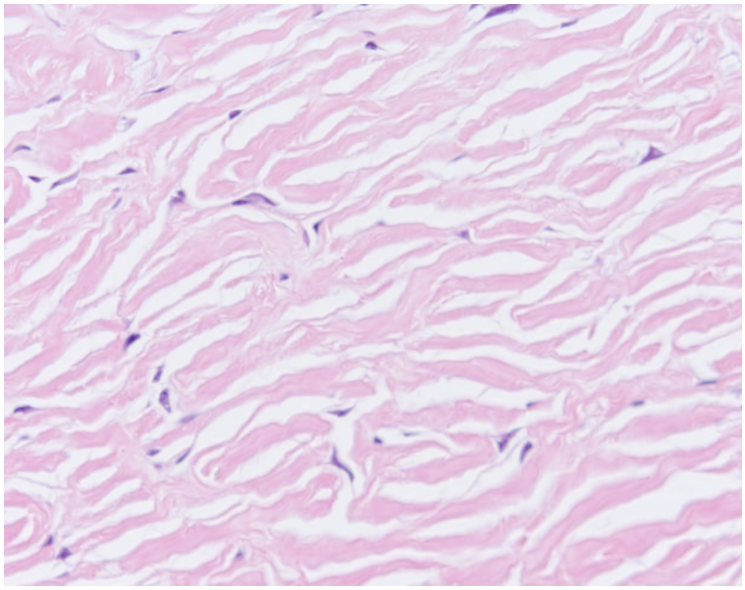


Figure A.2: This is the registered (the right) image without any adjustments. In all tests this image was adjusted somehow before the registration. The overlap is located at the left edge. Its width is again 85 pixels. Notice the nature of image, contrast and brightness are pretty the same as in the reference image. For discussion and figure with visualising tool, please refer to subsection 3.4.2 (on page 39).

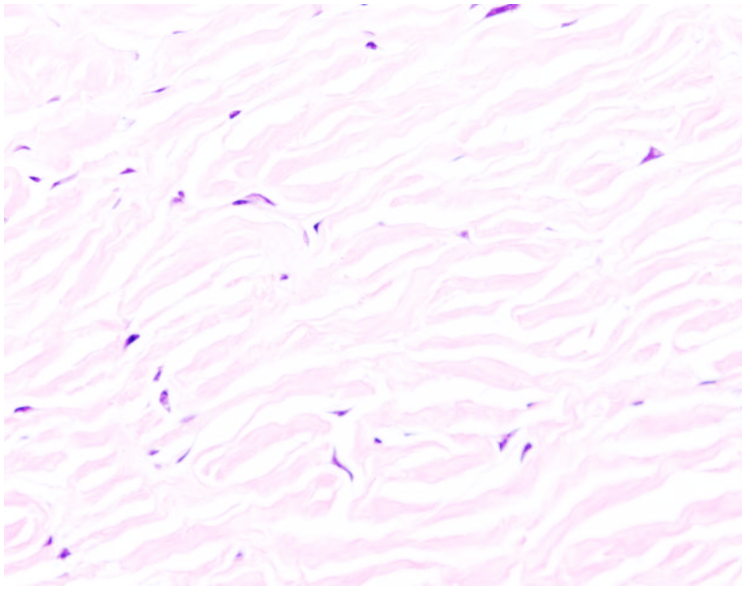


Figure A.3: This is the registered image, adjusted to become more brighter then the original. No artificial noise added. For discussion and figure with visualising tool, please refer to subsection 3.4.2 (on page 41).

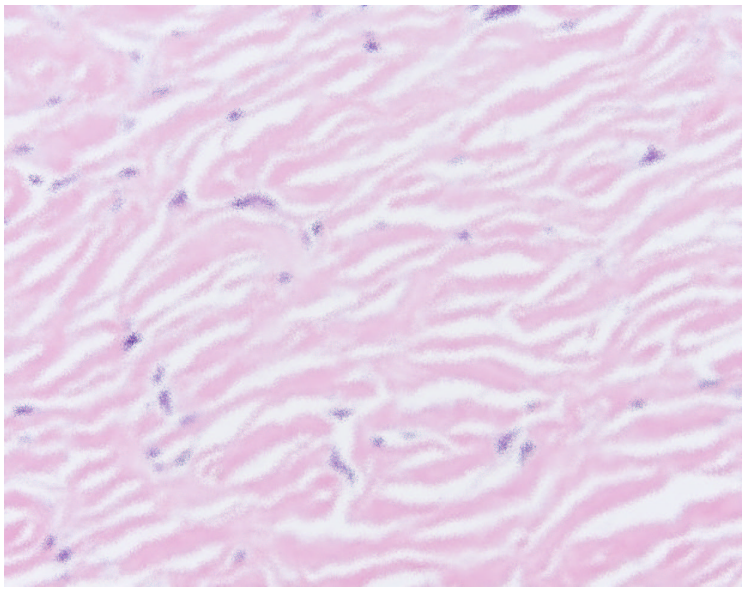


Figure A.4: This is the registered image, adjusted to lost its sharpness. No artificial noise added. For discussion and figure with visualising tool, please refer to subsection 3.4.2 (on page 41).

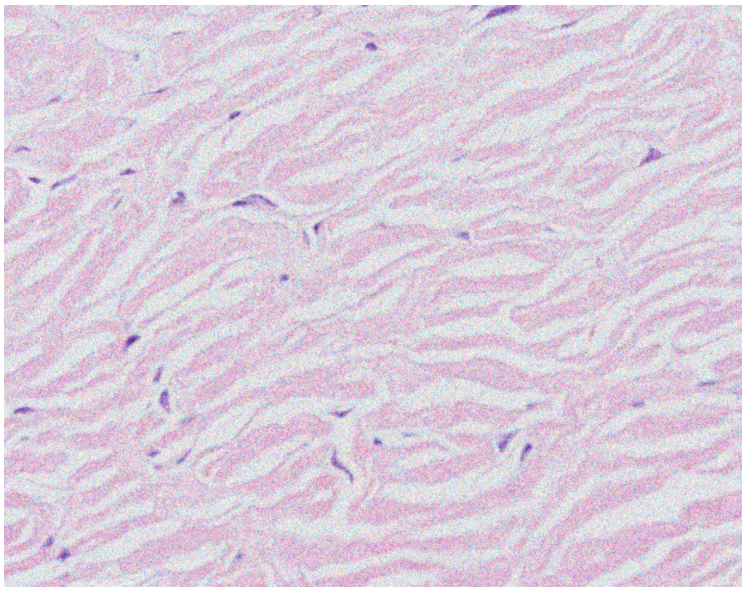


Figure A.5: This is the registered image, the original image with random noise. No brightness or sharpness adjustments. For discussion and figure with visualising tool, please refer to subsection 3.4.2 (on page 41).

Appendix B

Program documentation

B.1 Legal notes

This program uses two file formats libraries. The first one ([15]) is used for reading and writing `TIFF` files, it is a part of the implemented program and therefore we are obligated to copy here the following two lines.

The second library ([9]) is used for reading `JFIF JPEG` file formats. It is also a part of implemented program and therefore we must write here following two lines.

B.2 Input

The input to the program is a configuration file in which the images are specified as well as parameters to the program. Parameters can also be specified in the command line (at the prompt line). The configuration file has two parts.

The first part represents a table in which each cell is exactly one image that should be composed into the final whole image composition. The size of the table is restricted not to exceed the total count of 900 images in composition while there can be maximally 30 rows and maximally 30 columns. The dimension of images is not restricted anyhow, i.e. images can be as big as demanded by user, they also don't have to have the same dimensions. Each row is represented with one line in configuration file or with few immediately following lines each except the last one is ended with whitespace and backslash symbol in this order. Images in a row are represented with their corresponding file names, path can be included, white spaces in names or paths are not allowed. There are no delimiters between files in one row except at least one whitespace. Left images in a row are closer to the start of a table row definition. Upper rows in a table have their definitions closer to the beginning of the configuration file.

The second part of configuration file is the parameter definition part. We will also parallel describe the parameters accepted in command line, these will always begin with - (the minus sign). Parameters in configuration file are recognised only at the beginning of each line. After the parameter is parsed in the given line, the rest of line is ignored. Parameters in command line are given the usual way. There is no order of parameters specified. If the same parameter is given both in the configuration file and in the command line, the value from command line will be used — it will override the configuration file setting.

`-h, --help`
: Whenever this parameter is found, all other parameters are ignored, program will show help and exit. No composition is performed.

`-c, --config`
: Requires the path/name to the configuration file. If omitted than it is searched for the configuration file `zdroje.cfg` in the current directory. There is a note below regarding this parameter.

`-l, --log, LOG=`
: Requires the path/name to the file which will be written with information regarding the image composition process during this run. If this file exists, it will be first truncated to zero length (erased) and than the information written. We call this file the log file. See note at the end of this section about the default names behaviour.

`-a, --pripsat`
: This option will preserve the content of log file if such file exists, new information will be appended.

`-o, --output, TIF=`
: Requires the path/name to the file which will (re)written with final image composition. This file will in `TIFF` file format. See note at the end of this section about the default names behaviour.

`-p, --prekryti, PREKRYTI=`
: Requires (only) number. It is the maximum overlap possible given in percents. The percent mark (%) should be not appended to the number.

`-k, --krok, KROK=`
: Requires number. This is the value for n in n-step optimisation technique, see subsection 4.2.2 (on page 50).

`-v, --vyhledavani, VYHLEDAVANI=`
: Requires number. This way the evaluation function is selected. There are accepted: 0 for SSC, 1 for SAVD, 2 for NCC, 3 for CR, 4 for MI (the version depends on the special compiler parameter value), 5 for NONE. The NONE method simply defines the optimal alignment to be the one right in the middle of the space of all possible alignments.

`-r, --korekce, LIMIT_KOREKCE=`
: Requires number. This number defines the threshold for determining the "empty fusion," see the end of the section 4.3 for explanation of the T parameter (on page 59).

`-s, --prolinani, PROLINANI=`
: Requires number. The number defines which weight functions pair will be used when sutura camouflaging. Accepted values are: 0 for overlay, 1 for linear, 2 for cosine, 3 for 2nd order polynomial, 4 for 4th oder polynomial weight functions, 5 for "zig-zag" method. See section 4.5 for explanation.

Please note, under Microsoft(R) environment are accepted all parameters in configuration file without any change. Parameters on the command line are limited only to the `-h` and `-c`. In Unix, Linux environments all described parameters (both in configuration file and in command line) are supported.

There are several default values which can be summarised using the command line style as `-p 11 -k 4 -v 1 -r 10 -s 5`.

The default name behaviour. If no parameters specified than it is expected to read the configuration file in the current directory under the name `zdroje.cfg`, to write (and first erase if such exists) the log file in the current directory under the name `spojeni.log` and to write (and first erase if such exists) the final whole composition image into the file `spojeni.tif`. If the `-c` or `--config` is specified, than the default names of log and final image files are changed to be in the same directory where the configuration file is to be found, the file names are the same as the configuration file name but the extension `.cfg` is substituted with `.log` and `.tif`. This behaviour can be overridden using the `LOG=` respectively `TIF=` parameter in the configuration file or yet more when using the `-l` or `--log` respectively the `-o` or `--output` in the command line.

The first part is always preceding the second part. The pass from the first part into the second part is denoted with a line beginning with the @ (the at-sign) symbol. The rest of this line is ignored.

In both parts the whitespace is considered to be the space of exactly one symbol (usually produced by space bar key) or the tabulator symbol. Commentaries are allowed, commentary is every line beginning with `#` (the hash sign) until its end. Empty lines are allowed.

B.3 Output

Program has in fact three outputs. It is the whole composition image, the log file and the text produced on the screen (on terminal in Unix terminology).

The final image composition file stores the rectangular image. In rectangle the area where is no input image the blue background is inserted.

After reading this thesis the log file content and also the text produced on terminal will be self explanatory. On the terminal minimal information is presented, there is only the current state of image composition process maintained so the user can guess how long will it take until the composition is done. There are also some information regarding the "empty fusion" checking, for every registration the parameters of transformation of coordinates

are printed as well as the time required to find these parameters, also the position in the order for registering is shown. This we believe is enough to help experienced user to discover any potential misalignments when good knowledge of scanning system and the nature of data is provided, such user need not to wait until the final composition is created to discover that something went wrong.

The log file contains the same information as is printed on the screen, the formatting is also preserved. Depending on compilation time parameters something more can be found in the log file. But still it holds that this thesis will help to explain information found in the log file. Next section will show tiny example, that will make it all more clear.

B.4 Limitations

There are limitation mainly of the hardware kind but as always some software ones will be there too.

The program can run on every system on which somebody managed to compile it. Except the function for retrieving current time and the file formats libraries all is written in C++ and thus well supported under different environments.

The hardware limitation is only the memory and disk space, fortunately both can be predicted. The memory requirements have two major contributors. Firstly all overlaps stored in gray-scale from all images present in composition are read into the memory. Therefore when the ordinary image dimension is say $I_x \times I_y$ and the maximum overlap is set to o percent, the composition will be table with T_r rows and T_c columns then the first memory requirement is

$$Usage_1 < 4 \cdot T_c \cdot T_r \cdot \frac{o \cdot I_x \cdot I_y}{100}\text{B}. \tag{B.1}$$

Because of the overlap will be always $o \cdot I_x \cdot I_y/100$ (regardless horizontal or vertical direction), each registration (image pair) has two overlaps. Secondly when image stitching there are always maximally two entire rows of table read into memory at given moment. We are using 4B for one pixel then

$$Usage_2 \approx 4 \cdot 2T_c \cdot I_x \cdot I_y\text{B}. \tag{B.2}$$

It is not estimated exactly because it depends on the alignments of images, for example if one row produces after alignment the stair effect of images then bounding box must be allocated in memory. Be also aware of the memory consumption from the file format library which is usually less than 100MB.

The disk space is limitation for two reasons. When not enough memory, swap file managed by operating system may be used. The images are usually large. Hence the final composition will be large too, this file must stored somewhere. The swap should be therefore of the same size as computed in the previous paragraph if not enough memory is accessible. The disk space left should be maximally the size that is required for all images in the composition together.

Bibliography

[1] R. Bednář. Latex. Web pages. reference manual, URL `http://www.cstug.cz/latex/lm/frames.html` (January 2004).

[2] P. Bourke. Cross correlation. Web pages, September 1996. Document can be found at URL `http://astronomy.swin.edu.au/~pbourke/analysis/correlate/index.html` (March 2003).

[3] L. G. Brown. A survey of image registration techniques. Technical report, Columbia University, January 1992.

[4] M. Budíková, Š. Mikoláš, and P. Osecký. *Teorie pravděpodobnosti a matematická statistika*. Masarykova univerzita v Brně, 1998. Druhé, přepracované vydání, ISBN 80-210-1832-1.

[5] M. Čapek. *Registrace snímků z konfokálního mikroskopu*. Disertační práce, České vysoké učení technické v Praze, 1999.

[6] J. Feit, V. Ulman, W. Kempf, and H. Jedličková. Pořizování obrazů o velmi vysokém rozlišení metodou skládání. *Česko-Slovenská patologie a soudní lékařství*, 40/49(1), 2004.

[7] A. Fusiello. Tutorial on rectification of stereo images. Technical report, Dipartimento di Matematica e Informatica, Università di Udine, 1998. Document can be found at URL `http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/FUSIELLO/tutorial.html` (December 2003).

[8] S. Gilles. Description and experimentation of image matching using mutual information. Technical report, Oxford University, 1996.

[9] E. Hamilton. *JPEG File Interchange Format*, September 1992. Version 1.02, URL `http://www.ijg.org/files`.

[10] A. Junghanns. The matrix and quaternions faq. Web presentation. Document can be found at URL `http://www.j3d.org/matrix_faq/matrfaq_latest.html` (December 2003).

[11] J. Kučera. PB161 Programování v jazyku C++. Web pages. Supplementary web to course PB161 held at FI MU, URL `http://www.fi.muni.cz/usr/jkucera/pb161/`.

[12] A. Roche, G. Malandain, X. Pennec, and N. Ayache. The correlation ratio as a new similarity measure for multimodal image registration. In *Proceedings MICCAI'98*, volume 1496 of LNCS. Springer Verlag, 1998.

[13] A. Roche, G. Malandain, X. Pennec, and N. Ayache. Multimodal image registration by maximization of the correlation ratio. Technical report, INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, 1998. URL `ftp://ftp.inria.fr/INRIA/publication/RR/RR-3378.ps.gz` (November 2003).

[14] Electronic statistics textbook. Web presentation. URL `http://www.statsoft.com/textbook/stathome.html` (April 2003).

[15] *TIFF 6.0 Specification*, June 1992. URL `http://www.libtiff.org`.

[16] P. Viola and W. M. Wells III. Alignment by maximization of mutual information. *International Journal of Computer Vision*, pages 137–154, 1997. Document can be found at URL `http://www.ai.mit.edu/people/viola/research/publications/IJCV-97.ps.gz` (March 2003).

[17] P. Viola, W. M. Wells III, H. Atsumi, S. Nakajima, and R. Kikinis. Multimodal volume registration by maximization of mutual information. In *Medical Image Analysis*. Oxford University Press, 1995. URL `http://www.ai.mit.edu/people/viola/research/publications/MIA-95.ps.gz` (March 2003).

[18] P. A. Viola. *Alignment by Maximization Of Mutual Information*. PhD thesis, Massachuttes Institute of Technology, June 1995. URL `http://www.ai.mit.edu/people/viola/research/publications/PHD-thesis.pdf` (March 2003).

[19] P. A. Viola and W. M. Wells III. Alignment by maximization of mutual information. In *International Conference on Computer Vision 1995*, 1995.

[20] M. Zaffalon and M. Hutter. Robust feature selection by mutual information distributions. Technical report, IDSIA Switzerland, June 2002.

[21] A. Zisserman. Geometric framework for vision i: Single view and two-view geometry. Technical report, University of Oxford, 1997. URL `http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/EPSRC_SSAZ/epsrc_ssaz.html` (December 2003).