User-guided Image Editing using Radial Basis Functions

by

Evgeny Patrikeev

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Master of Science
in Computer Science.

Victoria University of Wellington
2015

Abstract

Good image editing tools that modify colors of specified image regions or deform the depicted objects have always been an important part of graphics editors. Manual approaches to this task are too time-consuming, while fully automatic methods are not robust enough. Thus, the ideal editing method should include a combination of manual and automated components. This thesis shows that radial basis functions provide a suitable "engine" for two common image editing problems, where interactivity requires both reasonable performance and fast training.

There are many freeform image deformation methods to be used, each having advantages and disadvantages. This thesis explores the use of radial basis functions for freeform image deformation and compares it to a standard approach that uses B-spline warping.

Edit propagation is a promising user-guided color editing technique, which, instead of requiring precise selection of the region being edited, accepts color edits as a few brush strokes over an image region and then propagates these edits to the regions with similar appearance. This thesis focuses on an approach to edit propagation, which considers user input as an incomplete set of values of an intended edit function. The approach interpolates between the user input values using radial basis functions to find the edit function for the whole image.

While the existing approach applies the user-specified edits to all the regions with similar colors, this thesis presents an extension that propagates the edits more selectively. In addition to color information of each image point, it also takes the surrounding texture into account and better distinguishes different objects, giving the algorithm more information

about the user-specified region and making the edit propagation more precise.

Contents

1 Introduction 1

1.1 Problem statement 1

1.2 Background 2

1.3 Research goals 3

1.4 Thesis outline 5

2 Literature Survey 7

2.1 Image deformation 7

2.1.1 Manual deformation 8

2.1.2 Automatic registration 10

2.2 Edit propagation 16

2.3 Gabor filters 18

2.4 Random projection 19

3 User-guided Image Deformation using RBF 21

3.1 Basic image deformation 22

3.1.1 Baseline method: B-spline warping 22

3.1.2 Application: automatic image registration 26

3.2 Interpolation by radial basis functions 33

3.2.1 Image warping using RBF interpolation 35

3.3 Results and discussion 36

4 Texture-aware Edit Propagation using RBF 41
4.1 Edit propagation using RBF interpolation 41
4.2 Gabor wavelets and filters 45
4.3 Edit propagation using Gabor filters 48
4.4 Random projection 52
4.5 Results and discussion 56

5 Conclusion and Future Work 65

List of Figures

2.1 A joint histogram. 13

3.1 An image with an example of desired deformation result (cartoon shaped eyes). 22

3.2 Four control points x_1, x_2, x_3, x_4 with their displacements and a sample point x with an unknown displacement value. 23

3.3 Basis functions $\beta_1(u)$, $\beta_2(u)$, $\beta_3(u)$ and $\beta_4(u)$. 24

3.4 Result of the deformation after displacing two (yellow) of the control points. 25

3.5 Nose deformations with a 4x4 grid (left) and a 10x10 grid (right) with outer points omitted. 25

3.6 Source, target and intensity differences. 26

3.7 Warping with optimized control points and the target image. 29

3.8 New target image. 29

3.9 Run of the algorithm on low resolution versions of the image. Sample deformation results (5 selected steps) with altered control points (green) and the target image (bottom right). 31

3.10 Result of registration and the target image (full resolution). 33

3.11 Gaussian-based RBF 34

3.12 From left to right: initial image of G.W. Bush, deformed image with control points, final result of attempt to enlarge the smile. 37
3.13 Deformation using a denser grid of control points. 38
3.14 From left to right: new initial positions of control points, deformed image with these control points, final result. . . . 38
3.15 Initial conditions and the result of RBF warping. 39

4.1 (a) Image with user-specified edits (strokes). (b) Edits interpolated in coordinate space only. 42
4.2 A result of edit interpolation in $XYRGB$ space. 43
4.3 Image with user strokes of different colors (red and green) and the result of edit interpolation. 45
4.4 Real and imaginary parts of a Gabor wavelet. 46
4.5 Two Gabor kernels. 46
4.6 An image of a brick pavement and its convolution with two Gabor kernels of different directions. 47
4.7 An image of a windsurfer on the water surface and its convolution with two Gabor kernels of different frequencies. . 48
4.8 (a) Image with specified edits, RAW image courtesy of Elio Ausili . (b) Result of applying an existing edit propagation approach. 49
4.9 (a) Pixels that have strong affinity with the user-specified ones. (b) Result of applying Gabor filter to the original image. 49
4.10 Results of selected 24 Gabor filters with frequencies 0.1, 0.25, 0.4, 0.55 applied to an image. 51
4.11 Results of edit propagation, performed as an interpolation of edits in selected high-dimensional space with 24 Gabor filters. 53

4.12 Result of edit propagation, performed as an interpolation of edits using 400 Gabor filters. 54
4.13 Two results of edit propagation with random projection method, applied to the space with 24 dimensions (Gabor values): despite the same initial parameters, right image receives bigger amount of edit. 55
4.14 Edits of green color, applied after RBF interpolation with Gabor features (right) and without Gabor features (left). Zero edits are denoted as gray. 57
4.15 Results of convolving the sample image with kernels of frequencies 0.1, 0.175 and 0.25 (from left to right), cropped for convenience. 57
4.16 Results of convolving the sample image with kernels of direction $\frac{2}{3}\pi$ and frequencies 0.4, 0.5, 0.6 (from left to right). 58
4.17 Results of convolving the sample image with kernels of frequency 0.4 and directions $0, \frac{\pi}{12}, \frac{\pi}{6}$ (from left to right). 59
4.18 Results of edit propagation that use different weights for Gabor features during interpolation. 59
4.19 Results of edit propagation that use different width of Gaussian interpolation kernel (0.1, 0.2 and 0.3). 60
4.20 Two more images with user-specified strokes of yellow and red colors. 61
4.21 Edit propagation results for yellow strokes, applied to the lawn. 62
4.22 Results for red strokes, propagated to the bricks of the wall. RAW image courtesy of Edward Musiak, https://flic.kr/ps/FRqCG 63

Chapter 1

Introduction

1.1 Problem statement

With the increase in the use of digital images over the last decades, convenient ways to edit or deform such images are of importance not only for professionals, but also for the average users. However, for large images and complex or subtle editing tasks, manual approaches are too time-consuming while fully automatic methods are not robust enough to provide visually acceptable results.

The ideal editing method includes a combination of manual and automated components. Such user-guided editing tools should contain elements of automation that require only a few intuitive guidelines from the user and provide high-quality results. The tool aims at successfully interpreting the given instructions and applying specified edits to the image, while relieving the user from doing numerous similar routines.

A mathematical method, known as function interpolation, also infers a complete result from an incomplete input. It takes a few function values at predefined points as input and then processes this data to be able to estimate the value at any point, based on the user-defined set of points. Considering user editing instructions as values of the sought function, it is easy to pose and solve several common image editing operations using function interpolation.

This thesis explores some of the advanced applications of user-guided editing and shows that interpolation approaches, based on Radial Basis Functions (RBF), can handle several problems while easily accepting user input. In particular, we will show that editing of object shape and colors can both be formulated using this approach.

1.2 Background

Digital images, such as photographs, are acquired by photosensors of the camera that detect intensity of light, reflected from the scene. However, inappropriate settings or inaccuracies in the chosen camera position may result in an undesired perspective, shape or size of image components. For images or photographs, editing by deformation may be necessary to edit or eliminate various kinds of optical distortions. Artists or even average users also might want to deform images arbitrarily for creative purposes, varying deformation parameters to achieve miscellaneous goals; from enlarging objects of the scene to scaling face features and creating caricatures, comic or monster faces. In all of these cases, techniques that move, scale, rotate and deform various parts of image are useful.

Another important application of image deformation is aligning two similar images together by deforming one of them. The process is known as image registration and is used to find correspondence and aligning transformation in numerous areas like medical imaging, remote sensing or face recognition. For example, medical scans taken at different times are aligned to identify changes in patients' conditions, while scans of different modalities are aligned to combine their information on a single scan. Therefore, finding an aligning deformation is an essential part of the registration process.

On the other hand, colour editing may also be required for a digital photograph, where slightly incorrect camera settings have resulted in dull or inexpressive colours. But even in an image with colours as realistic as possible, there might still be a need to emphasize the key objects, for example, by editing some of the colors or making them more deep or saturated.

The majority of such manipulations need to be applied to a particular region of the image and normally require the user to specify such region with a selection tool or by creating a layer mask. Manual selection of the region boundary is a tedious and time-consuming task, which gets even harder for regions whose actual boundaries are complex and lie between the pixels. This results in a series of pixels with intermediate colours, such as the ones on the boundaries of fuzzy objects like hair or grass. Such boundaries, if creating the mask, must be carefully feathered to produce a seamless and artifact-free result.

One of the ways to avoid such selections would be to let the user manually specify the edits (change of color, luminance, saturation, etc.) for a small set of pixels, then hand these to the program that will find the regions with similar appearance and apply further edits to those regions. The problem of processing user-specified edits and applying them to the right regions of an image is called edit propagation.

1.3 Research goals

Radial Basis Function interpolation is in the main focus of this thesis. RBF interpolation is shown as a useful theoretic tool for user-guided image editing, with successful applications to image deformation and color editing.

Existing methods of image deformation and shape editing are numerous, each having different types of parameters, structure and applications. For example, some methods [41] use regular grids of control points whose movements influence deformations of regions between them, while others [4, 3] allow arbitrary number of scattered control points, giving more user guidance. At the same time, some deformation methods apply better in fully-automatic regime but may fail in the complex cases, while some produce more impressive results but require strict user guidance.

This thesis considers various user-guided approaches and cases when they are easily applicable. In addition to an overview of such approaches, details of implementation are given for the two of them (B-spline warping and RBF warping) along with a comparison of advantages and potential applications.

To edit the color of a specific image region, the user has to specify a color change and a particular region. This can either be done manually or by edit propagation, which is a semi-automatic user-guided process. The problem of edit propagation is relatively new and was first formulated as an optimization problem of finding the best edit function subject to optimization constraints, given the user input.

However, a couple of novel approaches, such as RBF interpolation [24] and Locally Linear Embedding [7], were introduced recently. The thesis focuses on the interpolation-driven approach since it uses radial basis functions, which are in the central focus of our work. For a small set of pixels with user-specified edits, the approach uses color information to find pixels with similar appearance and to propagate edits to those pixels. But, as described in the section 4.3, such approach can apply changes to other objects that the user did not mean to edit, if their color just happens to be similar to the specified areas.

While looking for pixels with similar appearance, we are not limiting ourselves with just the color information, but also considering the change of texture and its patterns. The resulting improved algorithm, therefore, performs more accurate and selective propagation of user-specified edits.

Following are the key contributions, covered by the thesis:

- An overview of manual, automatic and user-guided methods of deformation, with preference given to user-guided ones;

- A detailed comparison of B-spline warping and radial basis function warping, along with some of their sample results and implementation details;
- A theoretical outline of edit propagation using RBF interpolation, followed by sample results and implementation details;
- An extension to the existing edit propagation approach, which makes the propagation more selective and texture-aware without requiring explicit segmentation of the image.

1.4 Thesis outline

The rest of the work is organised as follows. Chapter 2 provides a literature survey on the whole range of topics that our research touches, to give an outline of the current state of art. Chapters 3 and 4, the core of this thesis, explain the main ideas and theory behind them, presenting a full description of the performed work along with experimental results and discussion. Finally, Chapter 5 provides the conclusion of this thesis.

Chapter 2

Literature Survey

Digital images are organized as a table of color values, known as pixels. Color of each pixel is represented by a single value for grayscale pixels, three values for the majority of color pixels, or more. Image *width* is the number n of its pixel columns, *height* m is the number of pixel rows, while both these numbers denote the image resolution $n \times m$. In this thesis, pixels are also referred to simply as *points*. Instead of using sequential integer numbering ($x_{int} = 1, 2, ..., width$ and $y_{int} = 1, ..., height$), floating point coordinates are used for the convenience of mathematical calculations:

$$x_{float} = \frac{x_{int}}{width}, \quad x_{float} \in [0, 1]$$
$$y_{float} = \frac{y_{int}}{height}, \quad y_{float} \in [0, 1]$$

2.1 Image deformation

Image warping is a general term that denotes an arbitrary image distortion. A warping is a function W: $\mathbb{R}^2 \rightarrow \mathbb{R}^2$, which maps every position (x, y) on the initial image to a position (u, v) on the resulting image:

$$(u, v) = W(x, y)$$

A more detailed survey on image warping can be found in [15]. Intuitively, warping is a deformation function that specifies where each pixel of the image is shifted. But it is almost impossible to define such a function for each pixel manually. Therefore, warping functions are usually defined by a set of parameters, which are used as constraints to derive the function over the whole image plane, through a warping algorithm.

Having those methods available, generation of a warp with known parameters is a simple task. However, depending on the specific application of the sought warping, search strategies for the warping parameters may vary. Examples of such applications are:

- The removal of optical distortions caused by a camera or viewing perspective;
- Metamorphosis or morphing between two images;
- Registration or alignment of two or more images.

Some of these require the user to set the parameters manually, while others are semi- or fully-automatic. Since our work is focused on user-guided editing, we will use this division in the following sections to briefly outline various warping methods and elements of user guidance in each of them.

2.1.1 Manual deformation

A warping function usually deforms the image without changing the colors and aims to distort it. Another technique, called *morphing*, aims at creating a smooth transition between two images by a combination of warping deformation and color blending.

In his survey on basic image morphing methods, Wolberg [41] provides an overview of D. Smythe's approach that uses *mesh warping* for morphing between two faces. Two meshes denote two grids of control points located at their corresponding features (nose tip, eyebrows, etc.) on both the *source* (initial) image and the *target* (final) image. A smooth transition is then created between the control points of two grids, while their displacements are interpolated to be applied to all of the pixels. Spline interpolation, used in the approach, is explained in detail in Chapter 3 of this thesis. Finally, adding the transition between colors completes the morphing process.

Beier and Neely [4] propose another approach, *field morphing*. As a variant of Shepard's interpolation [2], it uses a simpler interface where the user needs to specify just two or more lines on the source image, and their new positions and lengths on the target image. While correspondences between the points on the pairs of lines are given straight away, mapping for the rest of the points is defined as a weighted sum that depends on the distance from each of the lines. The approach simplifies the specification of features by allowing to place them on arbitrary positions without any grid structure. Overall, the approach is more user-friendly and gives users more freedom, but also requires careful handling in order to avoid possible deformation artifacts.

Arad et al. [3] apply radial basis functions to image warping. Instead of using lines, as in Beier's approach, the user is required to specify a set of scattered control points, as well as their locations on the final warped image. Displacement values of each control point are then used for interpolation using radial basis functions. Similarly to Beier's approach, warping for each point is computed as a weighted sum of terms that depend on the radial distance from each of the control points. The approach also has fewer possible artifacts, as it is easier for the user to avoid overlaps of its single control points' movements, whereas long feature lines contain more points, are rather cumbersome and more likely to intersect with each other.

So far we can observe a general distinction between existing methods: some types of warping are obliged to have control points in a hierarchy or structure and some simply use scattered control points with arbitrary user-specified positions. The latter provides more user interaction, but may also increase complexity and the time for warp computation.

2.1.2 Automatic registration

In the previous section, we considered warping methods that let users manually set warping parameters, such as the positions of control points. However, we also mentioned an application of image warping that requires an automatic selection of its parameters: this is necessary when we need to find a deformation that aligns two or more similar objects. This problem, known as *image registration*, has numerous applications, including:

- *Medicine*, when comparing MRI scans acquired from subjects at different times or aligning images with different modalities (e.g. MR and CT);
- *Remote sensing*, when putting several satellite images into one coordinate system;
- *Modelling*, when building a statistical model out of a series of images, such as human faces;
- *Recognition*, when comparing an input image with images in database to find closest match.

The process of registration usually involves two sets of data (pixels, points, contours, etc.) that are referred to as *source* A and *target* B. Then, while transforming the source set, *similarity measures* (or cost functions) are used to deduce a shape of the source that is most similar to the target. In the case of a *rigid* image registration, for example, one image is usually aligned to another by a Euclidean transformation. Images get aligned, when the similarity measure of the deformed source and the target is either maximized or minimized – depending on the kind of measure that is used.

Definition 1 *To register one set of data S (source) to another set T (target) is to find a transformation W, such that the value of chosen similarity measure M(W(S), T) is optimal.*

An effective registration algorithm should be able to register similar objects on different images through the analysis and identification of their features and characteristics, rather than just using some predefined knowledge about these objects. Although there are a huge number of registration approaches, whose strategy depends on degrees of freedom for image deformation, class of aligned objects or the actual applications, medical image registration is by now one of most well-researched branches that also gives a good introduction into the registration problem in general [18]. Moreover, three abstract components of registration can be outlined:

- *Similarity measure* or a cost function, which quantifies the difference between two objects (source and target), using any chosen parameters, such as pixel intensity. It is the tool for the algorithm to determine if changes in the source object are desirable.
- *Set of transformations* to be used in order to make the changes. For the above-mentioned rigid case, they are: translation and rotation of the whole object - without changing its structure. Restrictions may also be required, as if we need only smooth deformations.
- *Optimization* or the logic by which the transformations are applied - in order to perform more suitable transformations, decrease the resulting number of iterations and calculate the similarity measure efficiently.

These components are important to specify when implementing a registration. Therefore, we will consider the most common types of image registration approaches, based on their similarity measures, aligning functions and other noticeable features and principles that were used. However, we will not cover the choice of optimization techniques, as it falls into the category of numerical methods which can be found in collections like [31].

The first type to consider is *intensity-based* image registration. As mentioned previously, images are commonly represented as tables of pixels, where each cell stores its color information in the form of a single grayscale intensity value or several values of color channels. In the case of medical imaging (MRI, CT, etc.), grayscale voxel intensity usually means different qualities of the matter, depending on how it resonates with an electromagnetic field. A sequence of such flat medical images of a subject can be assembled into a volume image, represented as grids of voxels (volume pixels). But regardless of the particular information that images carry, the purpose of an intensity-based registration approach is to align source and target images, based on the comparison of their intensity or color values.

One of the simplest similarity measures that compares intensity values of source and target images is the sum of squared differences.

Definition 2 *Sum of squared intensity differences (SSD) of two images S and T represents a mean squared error between their intensities and is expressed by the formula:*

$$SSD = \sum_{(x,y)\in[1,w]\times[1,h]} |I_S(x,y) - I_T(x,y)|^2$$

where $I_S(x,y), I_T(x,y)$ are the intensity values of source and target at point (x,y) and w, h are the width and height of the images respectively.

SSD is used in registration of images, MRI scans, volume images [17, 34] and others. However, the measure is not invariant to different lighting conditions or image modalities.

Another well-known measure is mutual information (MI), which originated from information theory. It was first used for registration as a measure that better handles multimodality images – such as MR, CT or PET

[8, 29, 38]. It relies on intensity distributions rather than intensity values and is based on *joint entropy* $H(S,T)$ – a measure of information in the two images combined [29, 18] – which, in turn, is based on the concept of *joint probability*.

Definition 3 *Suppose our source and target images (*S*,* T*) have* $w \times h$ *pixels with corresponding intensity values* $I_S(x,y), I_T(x,y) \in [0,255]$ $\forall x = 1,...,w$ *and* $y = 1,...,h$*. Then, for two arbitrary intensity values* i *and* $j \in [0,255]$*, the joint probability* $p(i,j)$ *is the ratio between the number of pixel positions* (x,y) *such that* $I_S(x,y) = i, I_T(x,y) = j$ *and the overall number of pixels:*

$$p(i,j) = \frac{\#\{(x,y) \mid I_S(x,y) = i, I_T(x,y) = j\}}{w \cdot h}$$

Since there are only 256^2 pairs of intensity values, we can plot probabilities of all the intensity pairs on a 256×256 graph, known as the *joint histogram*. The range of $p(i,j)$ is $[0,1]$, but if we multiply all these values by 255, we can represent the histogram as a square grayscale image (Figure 2.1), where the brightest points correspond to most probable intensity pairs. For two copies of the same image, the intensities are equal for all the positions of pixels and therefore its histogram is visually represented as points on the diagonal. On the contrary, a dispersed joint histogram with lots of non-zero values outside the diagonal, indicates the discrepancy of the intensities and a degree of misalignment between two images.

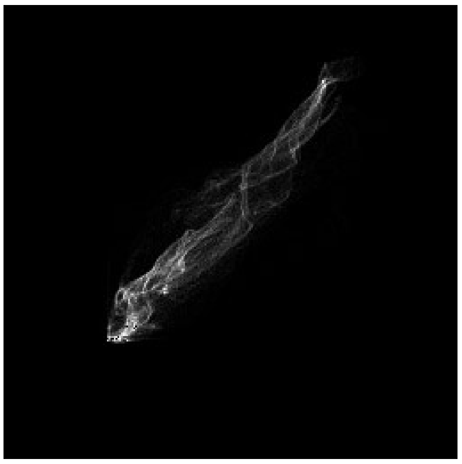


Figure 2.1: A joint histogram.

All joint probability values are used to calculate the joint entropy:

$$H(S,T) = -\sum_s \sum_t p(s,t) \log p(s,t)$$

Finally, mutual information of source and target images is given by:

$$MI(S,T) = H(S) + H(T) - H(S,T)$$

where $H(S)$ and $H(T)$ are the individual entropies, calculated using individual probabilities:

$$H(S) = -\sum_{s} p(s) \log p(s); \; p(i) = \frac{\#\{(x,y) \mid I_S(x,y) = i\}}{w \cdot h}$$

$$H(T) = -\sum_{t} p(t) \log p(t); \; p(i) = \frac{\#\{(x,y) \mid I_T(x,y) = i\}}{w \cdot h}$$

If S and T are totally unrelated, their mutual information will be equal to 0. Their joint entropy, in this case, will be equal to the sum of the individual entropies $H(S)$ and $H(T)$, otherwise it will be less than that. This way, to register two images we have to minimize $H(S,T)$ in order to maximize the information they share.

The above-mentioned SSD and MI were used in our experiments (outlined in the next chapter) as the most common similarity measures. But implementing a registration approach, it is also necessary to specify the set of aligning transformations. Earlier, we defined rigid registration, which uses only Euclidean transformations (translation and rotation), which are used in the papers referred to above [17, 8, 29, 38]. But there is also a *non-rigid* kind of registration – any registration that allows changes of the object's inner structure during alignment.

Such types of deformations as RBF-based warping [14, 32] or spline-based warping [35, 34], surveyed in previous sections, have been used to perform non-rigid registration. Applications include aligning scans of the subject at different times to identify changes in the patient's body [14] (such as tumour growth) or different poses of the subject [34] (to make a deformable model), where non-rigid deformations are required. This thesis also presents an implementation of non-rigid registration, as one of the image editing examples.

However, there is another significant group of approaches that is impossible not to mention in this survey: *feature-based* registration. Feature-based methods find correspondence between image features such as landmarks, lines, contours, etc. It is rather easy when the nature or type of object is known: in the case of face tracking, for example, feature points can be detected according to the intensity patterns and relative locations of the face features – a well-known application of the belief propagation algorithm [33, 21]. After features are detected on both source and target faces, registration is easy to perform.

But there are algorithms that work without prior knowledge about the depicted objects – those are based on the distribution of intensity gradients in local parts of an image. One of these feature descriptors is the histogram of oriented gradients (or HOG [9]).

Probably the most famous feature-based algorithm, known as scale-invariant feature transform (SIFT [26]), computes gradient orientation based descriptors (similar to HOG) for automatically identified keypoints. Those descriptors are invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes. Detection of SIFT keypoints starts from constructing a scale space by repeatedly convolving (blurring) the initial image intensity values with Gaussians to produce a set of Gaussian images. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images and the process is repeated for Gaussian images of different scale, down-sampled by a factor of 2. Then, maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel to its neighbors in regions at the current and adjacent scales. Those optima are then filtered by the algorithm by a threshold on minimum contrast and on the ratio of principal curvatures to achieve the resulting keypoints.

To make the descriptors of the achieved keypoints invariant to rotation, they are assigned one or more orientations based on local image gradient directions. Keypoint descriptors are built by computing the gradient magnitude and orientation at each image sample point around the keypoint and forming a set of orientation histograms. An additional Gaussian weighting function is applied to give less importance to gradients farther away from the keypoint centre. Finally, the descriptor is produced as a 128 element vector from the values of these histograms.

Keypoint-based registration is then performed by aligning keypoint pairs with minimum Euclidean distance for the corresponding descriptor vectors. Zaharescu et al. [43] later extended techniques, based on histogram of oriented gradients, to the case of 3D meshes with the MeshDOG feature point detector that seeks extrema of the Laplacian of a scale-space representation of any scalar function defined on a discrete manifold (generalization of the image intensity function) and a MeshHOG descriptor which is a generalization of the histogram of oriented gradients.

2.2 Edit propagation

Edit propagation allows the user to adjust the color of an image by making a few isolated examples of the desired changes in the form of brush strokes. Edit propagation is generally implemented as an optimization that prefers color and spatial proximity, subject to interpolating the constraints specified by the user.

Lischinski et al. [25] proposed an interactive tool that applies tonal adjustments to images from a few user strokes. The adjustments are made to the stroke pixels, and then propagated to the whole region with similar luminance using an edge-preserving energy minimization method. This method, however, may fail for parts of images with complex texture or pattern. An et al. [1] follows the work of Lischinski and applies edit not only to regions with similar luminance, but to any region with similar appearance. To identify such regions, appearance distance is introduced as a tool to assess the similarity of two pixels through their feature vectors. The assessed features include pixel color as well as the average and standard deviation of the color in a fixed-size neighborhood around the pixel to account for texture variations.

Xu et al. [42] continue the work on affinity-based edit propagation, using an optimization problem formulation, similar to [1]. However, to reduce the complexity of this optimization, authors group individual pixels into clusters and compute affinity among cluster centers. Thus clustering pixels by K-d tree partitioning is an efficient alternative to the computation of affinity between all pixel pairs.

Chen et al. [7] apply Locally Linear Embedding to image pixels, which represents each pixel as a linear combination of its neighbors and builds a lower dimensional manifold that preserves this relationship. Instead of just requiring similar pixels to receive similar amounts of edits, the method tends to maintain the manifold structure formed by the pixels in their feature space (e.g. colour space). One of the positive outcomes is the maintained relationship between ambiguous pixels with blended colours and their neighbours, which helps to edit such pixels more precisely and to avoid artifacts that usually occur at object boundaries.

Li et al. [24] considered edit propagation as a scattered data interpolation problem, rather than an optimization problem. From several user strokes of white color for areas to edit and black strokes to constrain the colors, the authors constructed a weight function with given values of 0 and 1 for user-specified set of points. The function is then interpolated by radial basis functions (RBF) in 5-dimensional $XYRGB$ space (2 coordinates and 3 color channels) to also take into account affinity between colors. The space, therefore, is denoted as the *affinity space*. After the editing weights are interpolated for all of the pixels, arbitrary edits can be applied to the image and to take maximal effect on the pixels with big weights (close to 1), while fading closer to the boundary of the edited region.

One of the noticeable features of the interpolation-based edit propagation approach, stated in [24], is that since negative interpolation coefficients (the ones that each basis function is multiplied by) cause unwanted edits in distant parts of the image, those are eliminated by a non-negative constraint. However, to enforce that constraint, non-negative least squares optimization is required, making the approach still include elements of the optimization.

Our extension of the interpolation-based edit propagation approach, explained in the section 4.3 of this thesis, uses extended $XYRGB+$ affinity space where interpolations take place. Extended coordinates of such space characterize each point more precisely, taking into account not only colors, but also texture variations. But unlike the work of [1], where only the two texture features (average and standard deviation) were used for texture identification, a series of spatial and frequency characteristics, received by applying Gabor filters to the image, are considered and applied. Additional innovations of our method are described in section 4.3.

2.3 Gabor filters

Campbell et al. [6] were among the first scientists to conduct research on simple cells of the mammalian visual cortex and measure their responses to the frequency of impulses or the contrast sensitivity. After that, mathematical models of image representation in visual cortex followed, based on both spatial and spatial frequency variables.

The first of such models, presented by Marčelja [27] in one-dimensional form and by Daugman [10] in two-dimensional form, used Gabor functions to describe the receptive fields of simple cells in the visual cortex, due to their simultaneous maximal localization in space and frequency domains.

One of the later works by Daugman [11] provided a more complete method to represent 2D signals (digital images) using Gabor functions as a basis. The compact representation stores three layers of coefficients that correspond to each specific Gabor function of the basis. Daugman also demonstrates reconstruction of the image from these values of coefficients without noticeable loss of information, along with applications of the technique in image compression. A significant theoretical advance was done later by Lee [23], in his proof that under several proposed conditions, a set of continuous 2D Gabor wavelets provides a complete representation of any image.

Jain and Farrokhnia [19] proposed a texture segmentation approach that uses Gabor filters. They use sets of values, obtained by applying various Gabor filters to an image, as texture features of the single points to better distinguish them. Given those features, the corresponding pixels are then clustered by an unsupervised clustering algorithm to produce a segmentation of the image parts with different texture. This application of Gabor functions and corresponding filters received sufficient attention and was used in several further texture segmentation approaches, extending it to the color domain [28] or adding rotation invariance [44].

However, a cornerstone in such segmentation methods is the possibility of selecting an arbitrary number of filters to get a number of texture features. It creates a trade-off between precision, achieved by computing and comparing a large number of those features, and the actual computational efficiency. Thus, several works have focused on designing the optimal set of Gabor filters [16, 13, 37] ever since their first usage in texture analysis.

2.4 Random projection

Since we use values from Gabor filters as features to extend $XYRGB$ affinity space of image pixels, values achieved as a result of every filter lie in an extra dimension. A large number of such features impacts the speed of the algorithm, as well as raise other non-intuitive aspects of data in high dimensions, known as the curse of dimensionality. One of these aspects is that the amount of training data needed for regression is exponential in the number of dimensions.

There are many approaches to reducing the dimensionality of the space:

Principal Component Analysis [40], various manifold learning approaches (e.g. Isomap [39] and Locally Linear Embedding [36]), etc. However, this thesis explores the random projection approach, inspired by Johnson–Lindenstrauss lemma [20], which states that N points in a high-dimensional space can be projected into a space of about $\log N$ dimension, with inter-point distances approximately preserved. The projection can then be accomplished by simply multiplying the original data for every pixel with an appropriately constructed random matrix [5]. The approach is explained in more detail in section 4.4.

Chapter 3

User-guided Image Deformation using RBF

This thesis investigates user assisted image editing with radial basis functions (RBF) in two case studies:

- *Image deformation* – comparison of RBF-based approach with automatic registration using B-splines shows advantages and disadvantages of the approaches.
- *Edit propagation* – RBF interpolation is applied to this task, and a new algorithm that outperforms previous work in the area is introduced.

This chapter starts with an outline of the most simple and easy-to-understand approaches to image deformation and continue with the more advanced ones. One type of method, explained later in this chapter, applies better to automatic deformations and works better with algorithmic manipulation of its parameters. It is presented in contrast to the methods with more user-guidance to easily demonstrate the benefits of providing the user with a more intuitive editing interface as well as more freedom in handling it.

3.1 Basic image deformation

Suppose we need to change the shape of some objects depicted in the image or its segments: for example, to make a face look like a caricature, cartoon or superhero character (Figure 3.1).

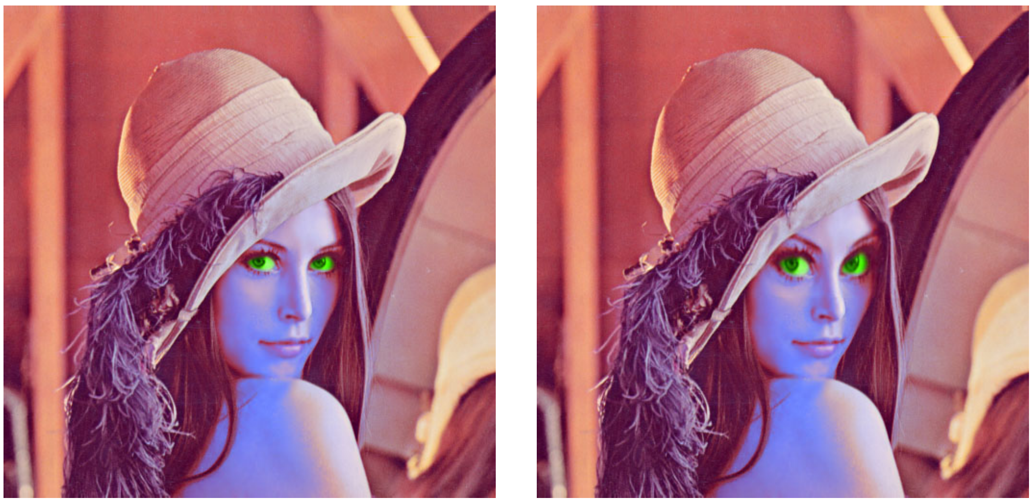


Figure 3.1: An image with an example of desired deformation result (cartoon shaped eyes).

Obviously, due to the number of pixels, this task is intractable to perform only by moving single pixels manually. The next sections give an overview of several deformation tools and the theory behind them, as well as automated and semi-automated approaches for different cases of image deformation.

3.1.1 Baseline method: B-spline warping

Image warping is a method of free-form image deformation. Although there are several approaches to implement it, the most easy-to-understand is to subdivide the image using a uniform grid and to manipulate the vertices of this grid as control points. As a result, movement of a control point will also move the pixels in the adjacent rectangles.

In the case of cubic B-spline warping, displacement values for every point of the image are calculated as a weighted sum of the displacements of the 16 (4x4) control points that surround that point. The weighting principle is easier to explain in a one-dimensional case, as shown in Figure 3.2. Suppose for the point x in 1D space we know the displacements p_1, p_2, p_3, p_4 for its 4 neighboring control points, such that: $x_1 < x_2 < x < x_3 < x_4$.

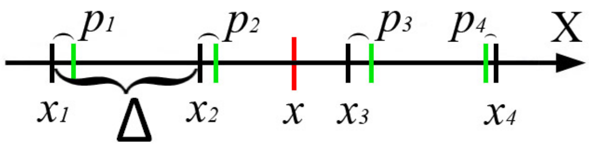


Figure 3.2: Four control points x_1, x_2, x_3, x_4 with their displacements and a sample point x with an unknown displacement value.

The resulting displacement of the point x depends on where it is located between x_2 and x_3. More specifically, a Δ-length interval between x_2 and x_3 is scaled down to $[0, 1]$ by a function $u(x) = (x - x_2)/\Delta$, providing local coordinates on this interval. Obviously, $u(x_2) = 0$ and $u(x_3) = 1$. Then, displacement function $d(u(x))$ is a sum of control point displacements p_i weighted by basis cubic polynomials $\beta_i(u(x))$ (Figure 3.3):

$$d(u) = \sum_{i=1}^{4} p_i \beta_i(u)$$

where

$$\beta_1(u) = \frac{(1-u)^3}{6}$$
$$\beta_2(u) = \frac{(3u^3 - 6u^2 + 4)^3}{6}$$
$$\beta_3(u) = \frac{-3u^3 + 3u^2 + 3u + 1}{6}$$
$$\beta_4(u) = \frac{u^3}{6}$$

This way, the function gives highest weights to the displacements of the control points in immediate proximity to x (i.e. x_2 and x_3) and smaller weights to the two further control points (x_1 and x_4).

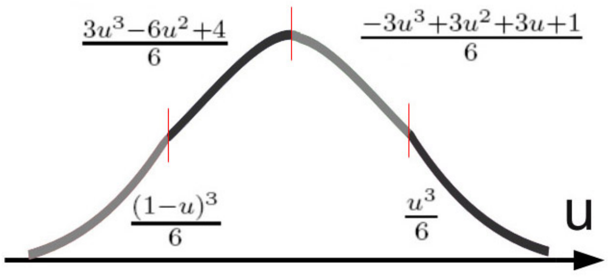


Figure 3.3: Basis functions $\beta_1(u)$, $\beta_2(u)$, $\beta_3(u)$ and $\beta_4(u)$.

For the 2D case, the sum is a bit more complex, since we need to take into account the displacements $px_{i,j}, py_{i,j}$ $(1 \leq i, j \leq 4)$ of 16 neighboring control points along both x and y. Then, the displacement $d(x, y) = (d_x(x, y), d_y(x, y))$ is calculated as:

$$d_x(x,y) = [\beta_1(u)\ \beta_2(u)\ \beta_3(u)\ \beta_4(u)] \begin{bmatrix} px_{1,1} & px_{1,2} & px_{1,3} & px_{1,4} \\ px_{2,1} & px_{2,2} & px_{2,3} & px_{2,4} \\ px_{3,1} & px_{3,2} & px_{3,3} & px_{3,4} \\ px_{4,1} & px_{4,2} & px_{4,3} & px_{4,4} \end{bmatrix} \begin{bmatrix} \beta_1(v) \\ \beta_2(v) \\ \beta_3(v) \\ \beta_4(v) \end{bmatrix}$$

$$d_y(x,y) = [\beta_1(v)\ \beta_2(v)\ \beta_3(v)\ \beta_4(v)] \begin{bmatrix} py_{1,1} & py_{1,2} & py_{1,3} & py_{1,4} \\ py_{2,1} & py_{2,2} & py_{2,3} & py_{2,4} \\ py_{3,1} & py_{3,2} & py_{3,3} & py_{3,4} \\ py_{4,1} & py_{4,2} & py_{4,3} & py_{4,4} \end{bmatrix} \begin{bmatrix} \beta_1(u) \\ \beta_2(u) \\ \beta_3(u) \\ \beta_4(u) \end{bmatrix}$$

where the local coordinates of the $\Delta_x \times \Delta_y$ frame between four central control points are given by $u(x) = (x - x_{2,2})/\Delta_x$ and $v(y) = (y - y_{2,2})/\Delta_y$.

An example of a spline warping result is presented in Figure 3.4. Displaced control points just influence a local, relatively small region around them and the rest of the points serve as a constraint. Therefore, the more detailed or intricate we want or deformation to be, the denser grid of points we should use. For example, if we want to make the nose smaller on the image, we can use different grids of points to deform it and achieve different results (Figures 3.5).

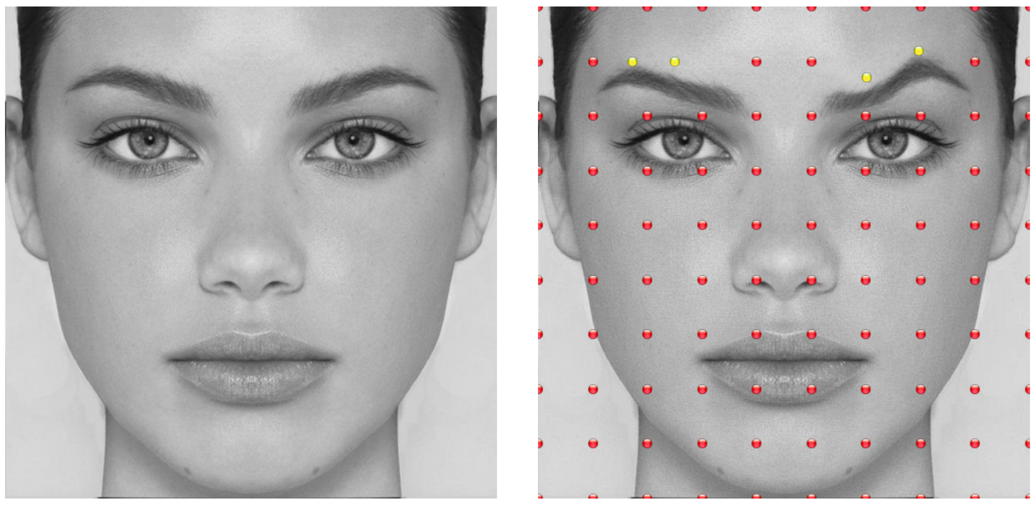


Figure 3.4: Result of the deformation after displacing two (yellow) of the control points.

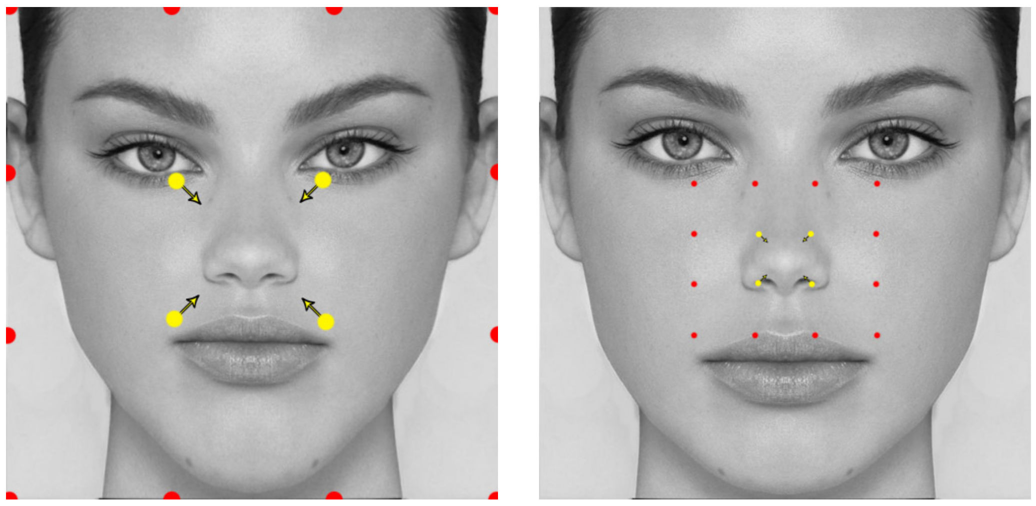


Figure 3.5: Nose deformations with a 4x4 grid (left) and a 10x10 grid (right) with outer points omitted.

The user must, therefore, decide which area should be influenced by the deformation and adjust the control points of the grid of corresponding size to deform it, followed by smaller deformations using denser grids. However, given several grids of various sizes, setting positions for all the control points may turn out to be a laborious task that needs automation.

The next section considers one of the ways to automate such a warping using image registration.

3.1.2 Application: automatic image registration

As explained in the literature survey, we need at least two images – source and target – to generate a warping deformation that aligns them. As a realistic case with enough complexity, slightly altered grayscale photographs of the same face were used for our registration (Figure 3.6). Registration in this case is called *local* as it is enough to register only a smaller part of the image.

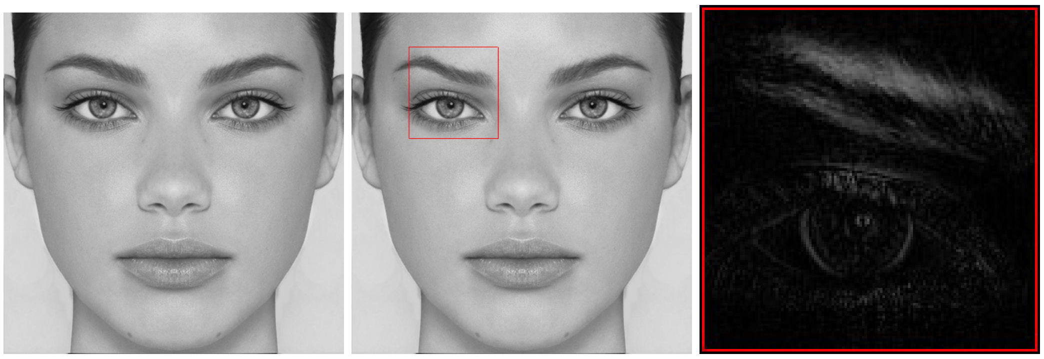


Figure 3.6: Source, target and intensity differences.

Next, we are going to specify our registration problem.
Given:

- set Ω of all the grayscale images with $n \times n$ pixels;
- source and target images $S, T \in \Omega$ with intensity values at every pixel:
$$I_S(x_i, y_j), I_T(x_i, y_j);\ i, j = 1, ..., n$$
where i and j are the horizontal and vertical positions of a single image pixel, and $(x_i, y_j) \in [0,1]^2$ are their corresponding real coordinates respectively ($x_i = i/n, y_j = j/n$);

- sum of squared differences between source and target intensities as similarity measure:

$$SSD(S,T) = \sum_{i,j=1}^{n} |I_S(x_i, y_j) - I_T(x_i, y_j)|^2$$

- a set CP of m initial positions of control points that parameterize the warping function:

$$CP = \{cp_1, ..., cp_m\}$$

  where $cp_k \in [0,1]^2$. Points are initially placed in a regular grid and provide an identity warping when supplied to the warping function.

- B-spline warping function $W : (\Omega, \mathbb{R}^{2m}) \to \Omega$ that for a given image $p \in \Omega$ and a set of control point positions $CP \in \mathbb{R}^{2m}$ returns a warped image $W(p, CP) \in \Omega$.

Sought: a set of parameters for the warping function – in particular, a set of control point positions:

$$CP' = \{cp'_1, ..., cp'_m\}$$

that provides a deformation of the source image S:

$$S_{dfm} = W(S, CP')$$

such that the deformed source S_{dfm} is as similar to the target T as possible, in terms of SSD:

$$SSD(W(S,CP'),T) \leq SSD(W(S,CP),T) \;\; \forall \, CP \in [0,1]^{2m}$$

Next, we are going to give a high-level explanation of how the registration is performed, outline the important features used in our implementation and present a scheme of the algorithm in the form of pseudocode.

We have already specified such important instruments as the similarity measure (sum of squared intensity differences) and means of deformation (B-spline warping) for our algorithm. Then, we also choose BOBYQA

(Bound Optimization BY Quadratic Approximation [30]) as an *optimization method* in our implementation: BOBYQA is a derivative-free solver of constrained optimization problems, which is handy in our case as we are looking for bounded displacements for a set of control points.

In general, we need to hand the cost function $f_{cost}(\cdot)$ along with starting arguments $\{cp_1, ..., cp_m\}$ to the optimizer and to receive the resulting optimal points $\{cp'_1, ..., cp'_m\}$. However, given $2m$ variables that need adjustment, the optimization can be exponential in the number of variables in the worst case. Therefore, to increase speed and efficiency, we have to cut some unnecessary elements of deformation and cost functions:

- *Number of control points.* There is no need to optimize a big number of control points at once, especially those that are far from each other and whose areas of influence do not overlap. Hence, we identify points $\{cp_k\}$ whose neighborhoods on source and target have the biggest sum of squared intensity differences – by a localized SSD:

$$SSD_{loc}(S, T, cp_k) = \sum_{\|cp_k-(x_i,y_j)\|<r} |I_S(x_i, y_j) - I_T(x_i, y_j)|^2$$

  where r is a preset value of neighborhood radius. Then, only this reduced set $\left\{\bar{cp}_1, ..., \bar{cp}_q\right\}$ with $q \ll m$ is optimized by BOBYQA: although the optimizer has access to all the control points from CP to perform trial warpings, most of them are fixed during optimization and do not affect the efficiency.

- *Evaluation of the cost function.* We do not need to create separate images for the source, warped by every trial combination of control point positions – this will result in unnecessary memory operations. Since we just need to compare it with the target, for each pixel of the source we can simply calculate its new warped position and add up intensity differences with the target pixels at the corresponding positions:

$$f_{cost}(S,T,CP) = \sum_{i,j=1}^{n} |I_S(x_i, y_j) - I_T(Eval(x_i, y_j, CP))|^2$$

where $Eval(x_i, y_i, CP) = (\bar{x}_i, \bar{y}_i)$ returns the coordinates of a pixel after the warping, defined by the set of control points CP. While SSD_{loc} is used to find points whose neighborhoods have the biggest SSD, f_{cost} is used to compare SSD values before and after the particular warping is applied – to see if it makes the source more similar to the target.

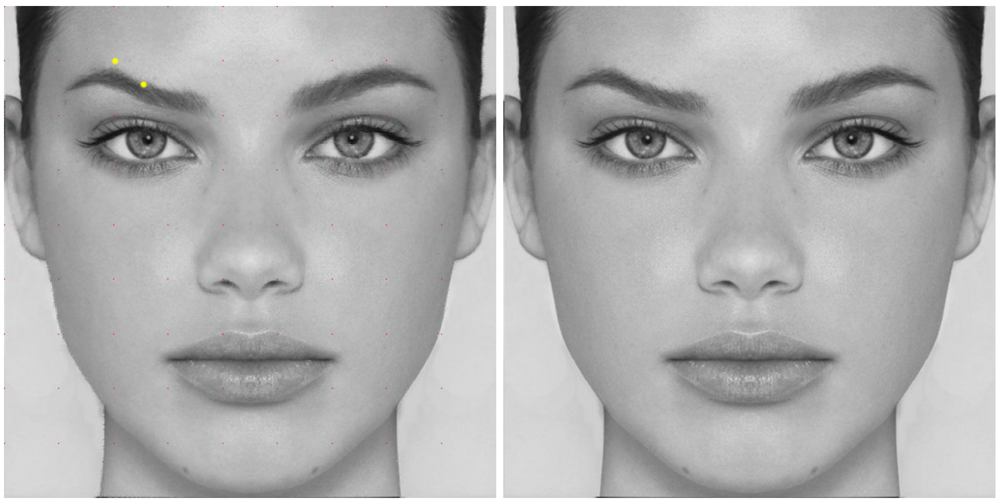


Figure 3.7: Warping with optimized control points and the target image.

Following these principles for the two images, cost function and control points of a warping are given to $BOBYQA(S, T, \{\bar{cp}_1, ..., \bar{cp}_q\}, f_{cost})$ and modified gradually on each iteration to achieve the warped result of the registration (Figure 3.7).

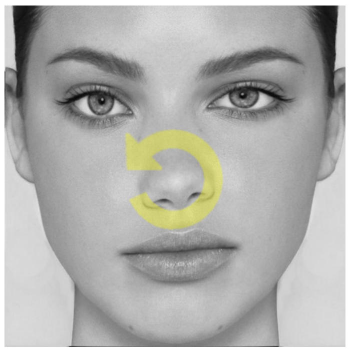


Figure 3.8: New target image.

After we have succeeded with local registration, the next goal is to register two images that differ not just at a single neighborhood, but at almost every point. In this experiment, the target image was created by the whirlpool effect in Photoshop, applied to the source (Figure 3.8).

Since such deformation is global, there are too many points whose neighborhoods has sufficient values of SSD between source and target. Therefore, the approach is slightly different:

- *Iterating over the most problematic neighborhoods.* We split the optimization on iterations, where on each iteration we take a single control point cp_k whose neighborhood has the maximal SSD and hand it to the optimizer, along with a set of its neighboring control points $\{\hat{cp}_1, ..., \hat{cp}_q\}$. Knowing that the optimizer will only modify control points that affect this neighborhood, we can also localize the trial warping and omit the computations of differences outside it in our cost function:

  $$f_{cost}(S, T, CP, cp_k) = \sum_{\|cp_k - (x_i, y_j)\| < r} |I_s(x_i, y_j) - I_t(Eval(x_i, y_j))|^2$$

  where r is the preset radius of region, influenced by any movements of cp_k and its neighbors $\{\hat{cp}_1, ..., \hat{cp}_q\}$.

- *Single warping instead of a composition of warpings.* After every run of BOBYQA on the subset $\{\hat{cp}_1, ..., \hat{cp}_q\}$ of control point positions, we receive a more optimal set $\{\hat{cp'}_1, ..., \hat{cp'}_q\}$ of positions, which we reassign in CP and finalize the iteration. One of the ways to proceed would be to apply the warping $W(S, CP) = S'$ to the source image, then make S' the new source image and to store the resulting transformation as a composition of warpings found on each iteration. However, applying too many warpings will cause degradation of the image due to repeated sampling and increase SSD even in areas that were not supposed to be modified. Therefore, we will stick to the search of a single warping, defined by the positions CP of control points, which we will gradually modify.

- *Multiresolution approach.* To reduce the number of evaluations, we create a higher level of iteration over the image resolution. On the first iterations, we run the optimizer on low-resolution copies of images, reduce the number of pixels and computations on them (Figure 3.9). We then gradually increase resolution on every iteration to increase precision of the registration approach.

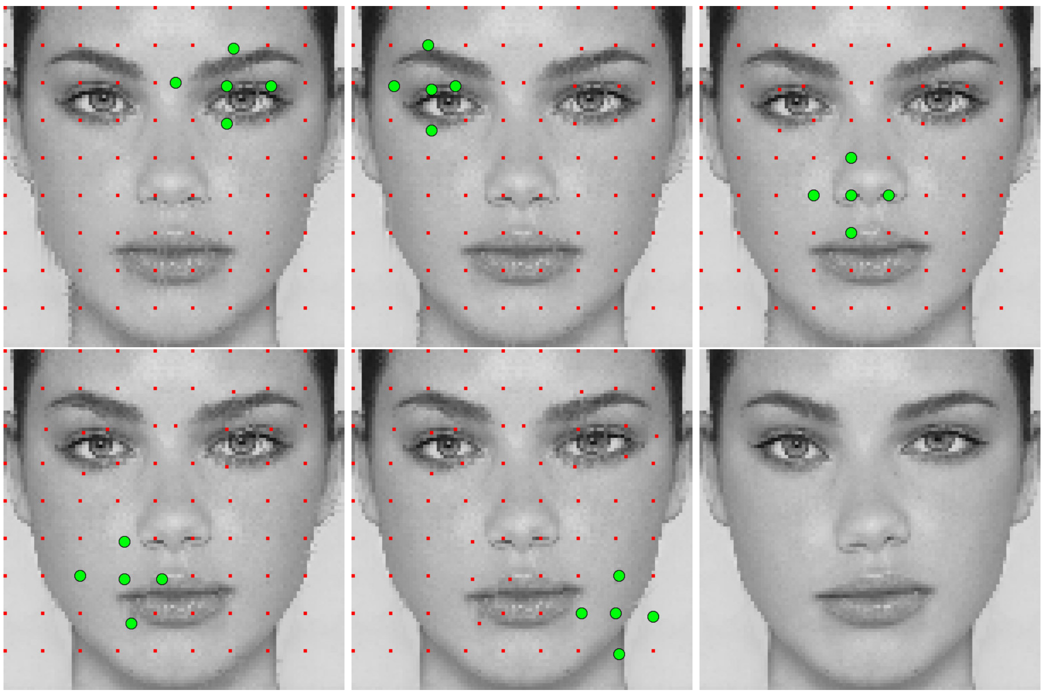


Figure 3.9: Run of the algorithm on low resolution versions of the image. Sample deformation results (5 selected steps) with altered control points (green) and the target image (bottom right).

A result in the actual resolution is shown in Figure 3.10. The algorithm concentrates more on the eyes, since that area has bigger intensity gaps and thus more information, while features like nostrils are just two gray lines. In order to align those features better we have to either add face feature extraction or use other cost functions that better characterize those features.

The presented implementation demonstrates the automated usage of image editing tools, such as B-spline warping: in the case when an image is too hard to warp manually, we can find another sample image that looks like the desired result of warping – and hand both of these images as source and target to the registration algorithm, which will adjust the deformation parameters automatically.

We finish our baseline experiment by providing the final pseudocode of our algorithm.

Algorithm 1 Global intensity-based registration

```
1: $CP = \{cp_1, ..., cp_m\}$ initial set of the control points in a regular grid
2: for $(i = 5;\ i \geq 1;\ i--)$ do
3:     decrease the resolution of source and target $i$ times: $S \leftarrow S_i$, $T \leftarrow T_i$
4:     $ssdVal = SSD_{glob}(W(S_i, CP), T_i)$ – value of the SSD on this step
5:     $ptsChecked = [\,]$ – empty list for points that have been modified
6:     while $(length(ptsChecked) < m)$ do
7:         find a point $cp_k \in CP$, whose neighborhood has the biggest SSD
               $cp_k = \underset{cp_k \notin ptsChecked}{\text{argmax}} SSD_{loc}(S_i, T_i, cp_k)$
8:         find the closest neighboring points for $cp_k$: $\{\bar{cp}_1, ..., \bar{cp}_q\}$
9:         run the optimizer and then modify the set of points:
               $CP \leftarrow BOBYQA(S_i, T_i, \{cp_k, \bar{cp}_1, ..., \bar{cp}_q\}, \bar{f}_{cost})$
10:        if $(ssdVal < SSD_{glob}(W(S_i, CP), T_i))$ then SSD not decreased
11:            set the modified points back $CP \leftarrow \{cp_k, \bar{cp}_1, ..., \bar{cp}_q\}$
12:            add $cp_k$ to the $ptsChecked$ array not to get back to it
13:        else $ssdVal \leftarrow SSD_{glob}(W(S_i, CP), T_i)$
14: return $CP$
```

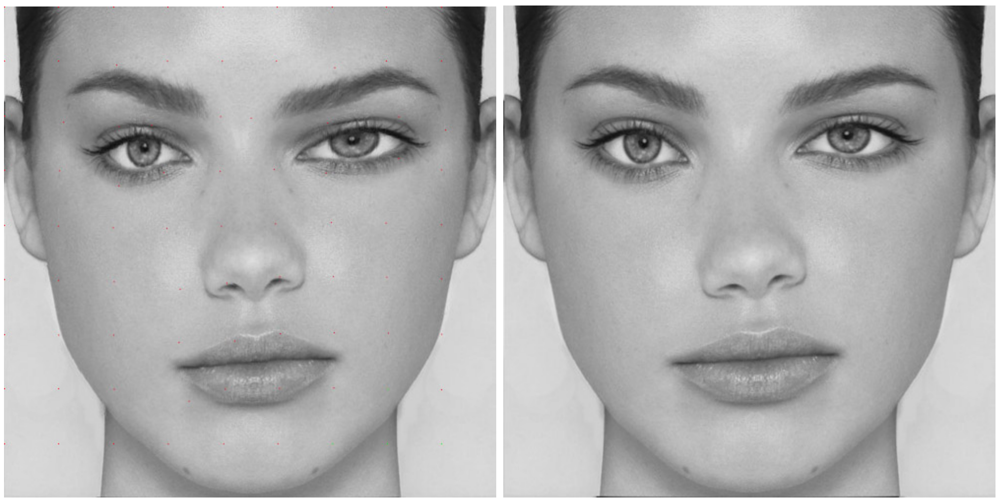


Figure 3.10: Result of registration and the target image (full resolution).

3.2 Interpolation by radial basis functions

Suppose we have a function $f(\mathbf{x})$ whose values are known for some $\mathbf{x}_1, \mathbf{x}_2$, ..., $\mathbf{x}_n$ values of the argument (also known as data points):

$$f(\mathbf{x}_i) = f_i, \text{ for } 1 \leq i \leq n.$$

Interpolation, in this case, is a method of constructing intermediate function values for the rest of the points. An example of interpolation, shown in Figure 3.11, was done using radial basis functions (RBF), which are one of the most commonly used scattered data interpolation techniques.

RBF interpolation is a weighted sum of radial basis functions. Each basis function is centered at a data point:

$$\tilde{f}(\mathbf{x}) = \sum_{k=1}^{n} w_k \phi(\|\mathbf{x} - \mathbf{x}_k\|), \tag{3.1}$$

where ϕ is a function from $[0, \infty)$ to $\mathbb{R}$ and $\{w_k\}$ is a set of weights. The function ϕ is called radial since its value depends only on the distance from the origin, so that $\phi(\mathbf{x}) = \phi(\|\mathbf{x}\|)$ and the influence of a single data point

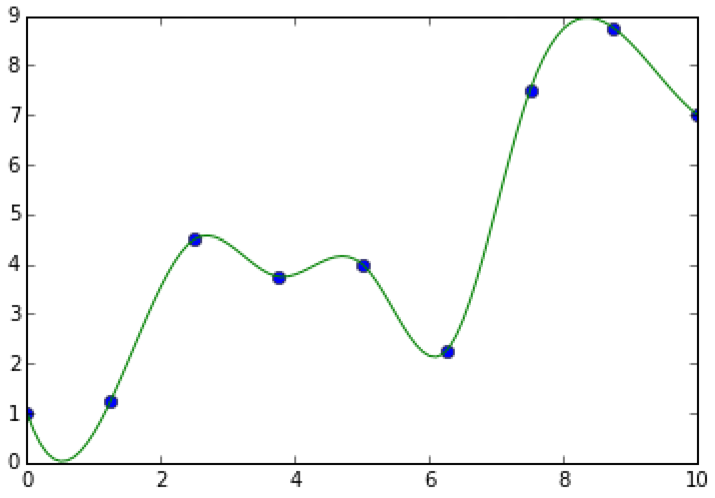


Figure 3.11: RBF interpolation with a Gaussian kernel $\phi(r) = \exp(-r^2/2\sigma^2)$

on any interpolated point is constant on a circle, centered at that point. An advantage is that regardless of the number of dimensions of the input space, ϕ just takes one argument.

The resulting interpolation also depends on the choice of RBF kernel, that is, the particular form of the function ϕ. There are numerous kernel functions, however one of the most commonly used is Gaussian kernel, as seen in Figure 3.11:

$$\phi_{Gauss}(r) = e^{-\frac{r^2}{2\sigma^2}} \tag{3.2}$$

After the kernel function is chosen, we need to calculate the RBF weights from interpolation conditions. Those are obtained from a linear system expressing that the function $\tilde{f}(\mathbf{x})$ should have specified values $\tilde{f}(\mathbf{x}_i) = f_i$ at our n given data points:

$$\tilde{f}(\mathbf{x_i}) = \sum_{k=1}^{n} w_k \phi(\|\mathbf{x}_i - \mathbf{x}_k\|) = f_i, \text{ for } 1 \leq i \leq n. \tag{3.3}$$

This is a linear system of equations where RBF weights w_k are the unknowns. For any pair of data points $(\mathbf{x}_i, \mathbf{x}_k)$, we can easily calculate

$\phi(\|\mathbf{x}_i - \mathbf{x}_k\|)$. Denoting $\phi_{i,k} \equiv \phi(\|\mathbf{x}_i - \mathbf{x}_k\|)$, it is possible to represent the interpolation conditions as:

$$\Phi \cdot w = \begin{bmatrix} \phi_{1,1} & \phi_{1,2} & \phi_{1,3} & \cdots & \\ \phi_{2,1} & \phi_{2,2} & \cdots & & \\ \phi_{3,1} & \cdots & & & \\ \vdots & & & & \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \end{bmatrix} = F \tag{3.4}$$

By solving this square system with as n equations and n unknowns, we can calculate weights and put them into the formula 3.1 to get the final interpolation function, that returns a result for any intermediate point x.

3.2.1 Image warping using RBF interpolation

Now suppose we have selected a set of control points $\{(x_i, y_i) \mid 0 \le i \le n\}$ on an image and have specified where each of them should go during the deformation. In the user interface, this can be done by two clicks for each point, however it will be stored simply as the displacement values for each of these points. Next, we can consider a function D: $\mathbb{R}^2 \to \mathbb{R}^2$ that for each point returns its displacement values:

$$D(x_i, y_i) = (dx_i, dy_i),\ \forall i : 0 \le i \le n$$

the D function is easier to interpolate as two separate functions:

$$D_x(x_i, y_i) = dx_i,\ D_y(x_i, y_i) = dy_i,\ \forall i : 0 \le i \le n$$

where dx_i and dy_i are values of displacement along x and y axes for the point (x_i, y_i). Important to notice, that for some points the user must specify zero displacement values ($dx_i = 0, dy_i = 0$): these are necessary restrictions for the points that need to stay on their places.

Interpolating both $D_x(\cdot)$ and $D_y(\cdot)$ will provide us with displacements for any point (x, y) of the image, depending on its distance to the points with specified displacements:

$$D_x(x, y) = \sum_k^N w_k \phi(\sqrt{(x - x_k)^2 + (y - y_k)^2}),$$

and the corresponding sum for $D_y(\cdot)$. The example results of this displacement interpolation are demonstrated in the next section, along with a comparison of RBF and B-spline warping approaches.

3.3 Results and discussion

One of the advantages of the spline interpolation is that it is local, i.e., an interpolated displacement value of a point depends only on displacements of control points around it and, this way, needs no computations of distances from the interpolated point to each of the control points, as in the case with the RBF interpolation. A fixed-size set of neighboring control points (such as 4x4 in cubic case) is enough to interpolate each point using a cubic spline basis.

On the other hand, RBF interpolation applies better for scattered data with no grid structure, but with arbitrary positions for control points, where each of those influences each interpolated displacement. This may be superior for user-guided editing. It relieves the user from the routine of selecting the size of the control point grid, modifying each control point over the affected regions and deciding which blocks of control points should be moved together.

For example, if we want to make the smile wider on the photograph (Figure 3.12) using spline warping, we have to move several points of a row up, and to move several points of another row down. However, since control points of a grid are located slightly away from the lips, the resulting deformation enlarges the whole area between the points, but not the actual smile. This can be solved by:

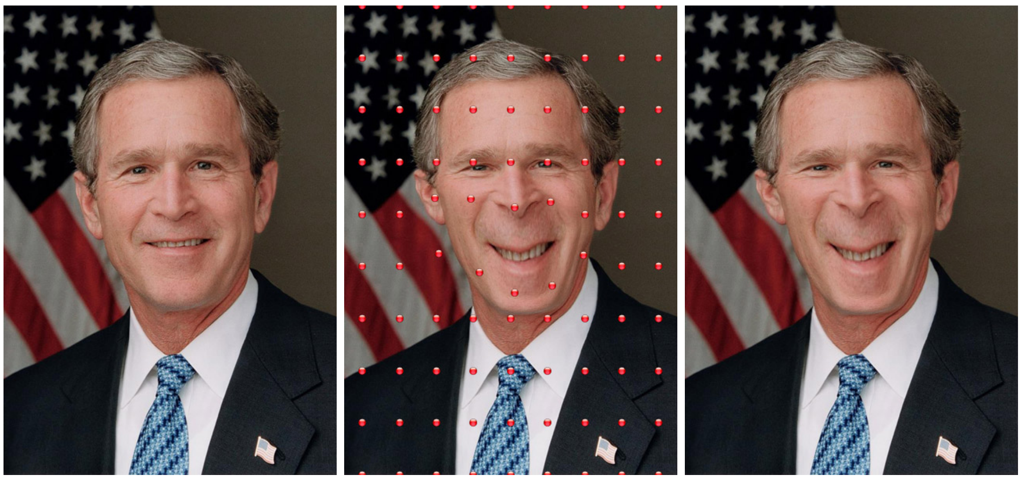


Figure 3.12: From left to right: initial image of G.W. Bush, deformed image with control points, final result of attempt to enlarge the smile.

- Making the grid of control points more dense – which results in a bigger set of control points that needs careful handling and therefore increases the possibility of artifacts. The warping demonstrated in Figure 3.13 involved only vertical control point displacements but even those were numerous. The warped smile was closer to the desired result, but the lower lip still turned out a bit asymmetric;
- Changing initial positions of control points (Figure 3.14) to coincide with the features. This requires a non-uniform grid of control points, which is only possible if control point positions are interpolated instead of displacements. It gives the user a chance to locate control points on any desired features. Grids of lower density, in turn, allow easier manipulation both vertically and horizontally – resulting smile in Figure 3.14 was stretched in both directions. However, uncareful handling still resulted in small artifacts like curvy upper lip, different sized ears or folds between nearby points.

Obviously, it is possible to master the manipulation of these points so as not to cause unwanted artifacts, but the above-mentioned difficulties result in an inevitable conclusion for spline warping: there is always a

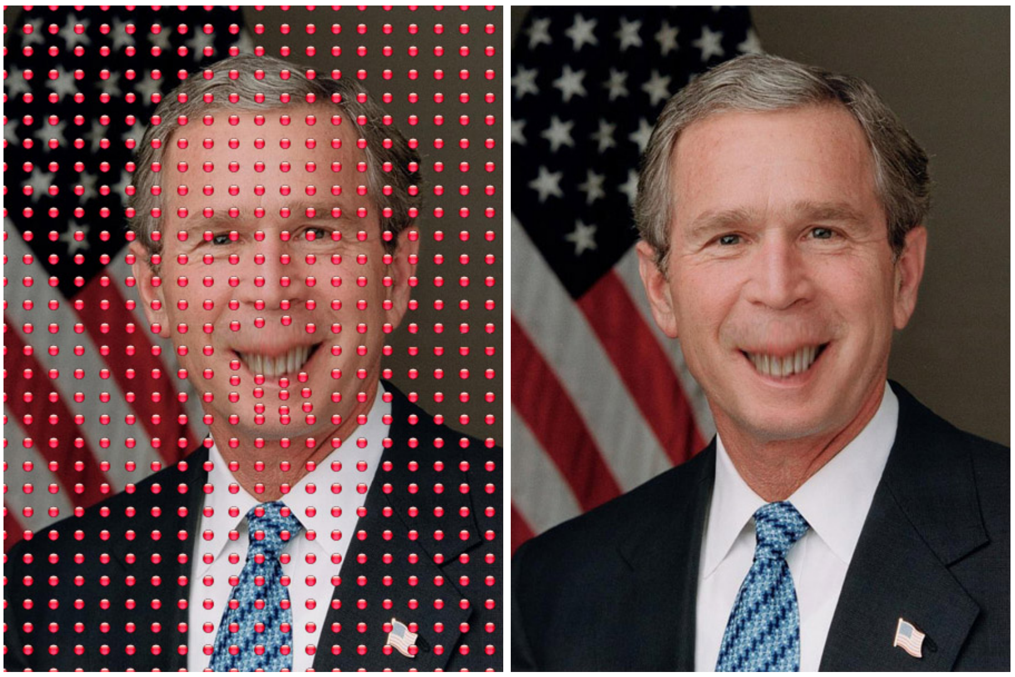


Figure 3.13: Deformation using a denser grid of control points.

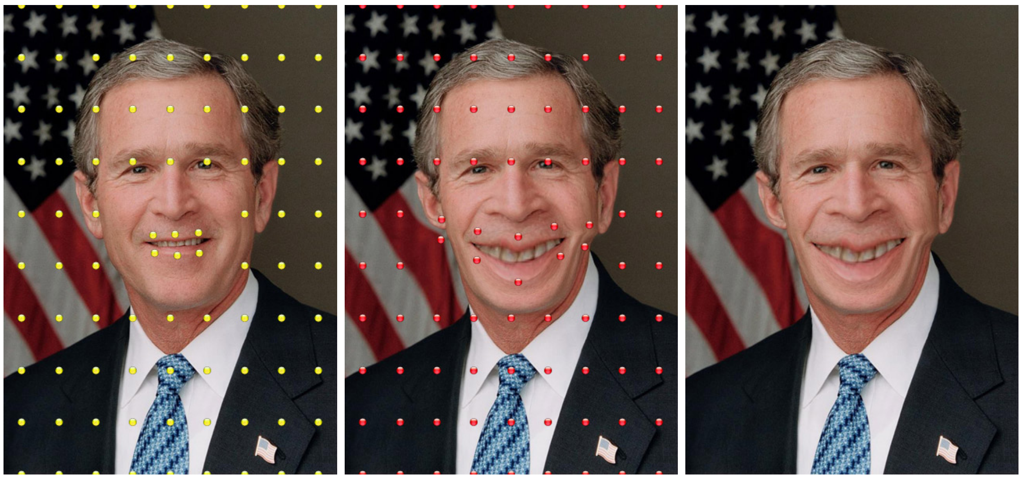


Figure 3.14: From left to right: new initial positions of control points, deformed image with these control points, final result.

trade-off between detailed deformation (needs more points in a grid) and easy control over the control point grid.

Unlike the spline warping, which required at least 10x10 control point grid, the RBF displacement interpolation demonstrated in Figure 3.15 used only 13 control points: where 8 of them are constraining "anchors" and 5 are specified displacements, denoted by red arrows. It is typical of the RBF approach: the smaller number of required control points saves user effort, while allowing rather intuitive instructions: "*these points stay on their places and these points should be moved here and here*".

Moreover, the result of the deformation is smooth and without noticeable artifacts – despite the length of the stretch, which is about two times bigger than in previous examples. This way, the result demonstrates the ability of the approach to handle highly elastic deformations.

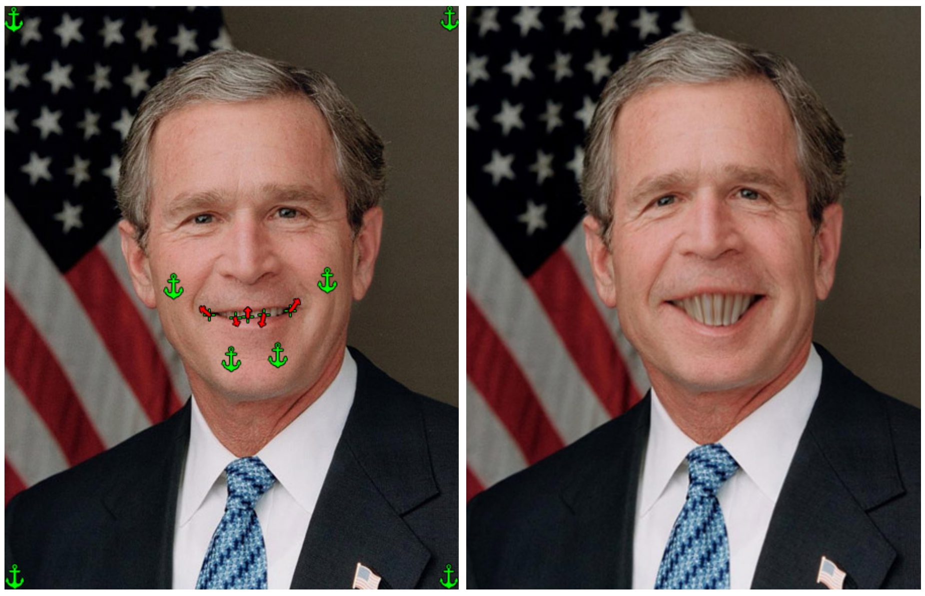


Figure 3.15: Initial conditions and the result of RBF warping.

On the other hand, RBF warping requires that the user is clear which points should be moved where. Therefore, if used for image registration, there is an extra problem of identifying positions of control points to be manipulated – this task is usually performed manually [14, 22].

Chapter 4

Texture-aware Edit Propagation using RBF

This chapter introduces an approach to process user-specified color edits and propagate them to the whole image. Such edits are made in the form of strokes that change the colour or hue of an initial set of pixels. Our algorithm then analyzes that kind of input, as well as the image itself, in order to apply the edits to a wider area of similar pixels that the user originally meant to edit. In other words, it evaluates the amount of alteration each pixel of the image receives, based on the known set of initial edits.

Then, the existing edit propagation method is extended by enabling it to distinguish parts of images not only with different colors, but also with different texture. As a result, such an edit propagation approach gives users the ability to apply edits more selectively – only to the objects or regions with similar appearance, in terms of both texture and colors.

4.1 Edit propagation using RBF interpolation

Apart from using radial basis functions to interpolate deformations of image points and apply them to the whole regions around those points, similar interpolation techniques can be used to edit colors of the image – by the interpolation of input edits for some of its points.

Suppose we have three edit functions $h_r, h_g, h_b : (x, y) \in \mathbb{R}^2 \to \mathbb{R}$ that for some set of n input pixels of an RGB image return the values of edits for its red, green and blue channels:

$$h_r(x_i, y_i) = red_i \ ; \ h_g(x_i, y_i) = green_i \ ; \ h_b(x_i, y_i) = blue_i \ , \quad 1 \leq i \leq n$$

where red_i, $green_i$ and $blue_i$ are the values that need to be added to the color channels of the pixel with coordinates x_i and y_i. Those values can be set by brush strokes of various kinds: for example, one that changes the hue is demonstrated in Figure 4.1a. For the regions that need no changes we apply a constraint indicated by setting edits at those pixels to zero $h_{r,g,b}(x, y) = (0, 0, 0)$ and marking them with black strokes.

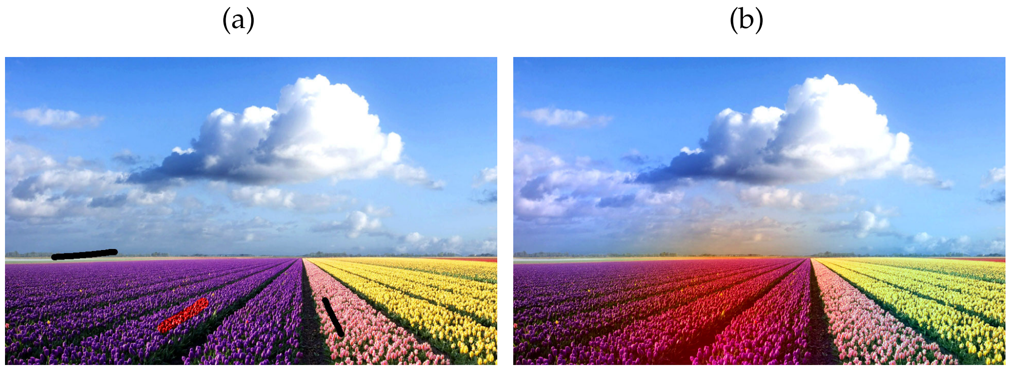


Figure 4.1: (a) Image with user-specified edits (strokes). (b) Edits interpolated in coordinate space only.

The next thing to do is to interpolate each of the three functions in order to get a smooth editing effect over the intuitively-marked region. However, it is obvious from Figure 4.1b that interpolated edits are just being propagated to nearby pixels, while fading closer to the locations of black strokes.

A way to propagate the edits to the pixels of similar colors would be to interpolate extended versions of the functions h_r, h_g and h_b. Instead of working in XY coordinate space, suppose we are looking for three functions $\tilde{h}_r, \tilde{h}_g, \tilde{h}_b : \mathbb{R}^5 \to \mathbb{R}$ with domain $XYRGB$, that is, the coordinate space united with the color space.

In this space, distance is small only for the points whose locations on the image are close and whose colors are similar. In this way, just a simple extension of the editing functions' domain results in a much more accurate color edit, where a field of purple flowers from Figure 4.1a is turned red in Figure 4.2.

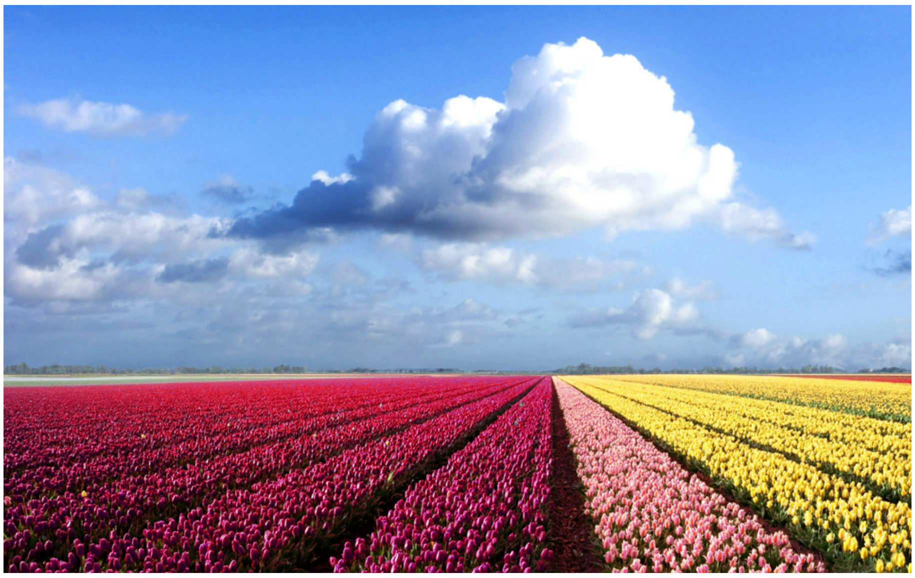


Figure 4.2: A result of edit interpolation in $XYRGB$ space.

A similar RBF interpolation-based approach was proposed by Li et al. [24], however it is slightly different from the scheme, described above. Instead of performing edit interpolation for all three color channels in their work, [24] use interpolation to weigh each pixel from 0 to 1, similarly to creating an image matte. Interpolation conditions for the weight (matting) function $M : \mathbb{R}^5 \to [0, 1]$ are set to 1 at chosen n pixels that need editing (marked with white strokes):

$$M(x_i, y_i, r_i, g_i, b_i) = M(\mathbf{x}_i) = 1 \qquad (1 \leq i \leq n)$$

while the areas that need to retain their color are marked with black strokes, and the function M there is set to zero.

After M is interpolated and the weights are known, arbitrary edits can be applied to the image and will have maximal effect on the pixels with big weights (close to 1), fading closer to the boundary. The color I' of the edited pixel with a feature vector $\mathbf{x} = (x, y, r, g, b)$ can be denoted as:

$$I'(\mathbf{x}) = M(\mathbf{x}) \cdot e + (1 - M(\mathbf{x}))\, I(\mathbf{x})$$

where $I(\mathbf{x})$ is the color of the pixel on the original image and e is the color applied in editing.

Another one of the algorithm's features is the constraint on the RBF weights w_i (equation 3.4, $\Phi w = F$) to be non-negative. The proposed constraint is based on the observation that negative weights cause edits in distant parts of the image to be unusually high, and is aimed at eliminating such coefficients. The weights $w_i \geq 0$ are found using non-negative least squares optimization. And although results of the optimization may differ from the exact solution, non-negative weights ensure that the edits diminish in distant portions of the image.

Since the matrix Φ (equation 3.4, $\Phi w = F$) with interpolation conditions is filled only with non-negative values, it is impossible to enforce the non-negative constraint on w, when vector F have negative values of the function that is being interpolated. It is not the case with the non-negative function M, but if we are interpolating edit functions like $(\tilde{h}_r, \tilde{h}_g, \tilde{h}_b)$ directly, we might need to interpolate their negative values as well. For example, if we need to decrease the value of the red channel, the value of edit $\tilde{h}_r(x_i, y_i, r_i, g_i, b_i)$ might be negative for some i.

Another disadvantage is that after M is interpolated, we can only use its values to apply a single edit e to the set of pixels with weights $M(\mathbf{x}) > 0$. It would be impossible to make several different edits to different regions as shown in Figure 4.3, where interpolation was done without the non-negative constraint.

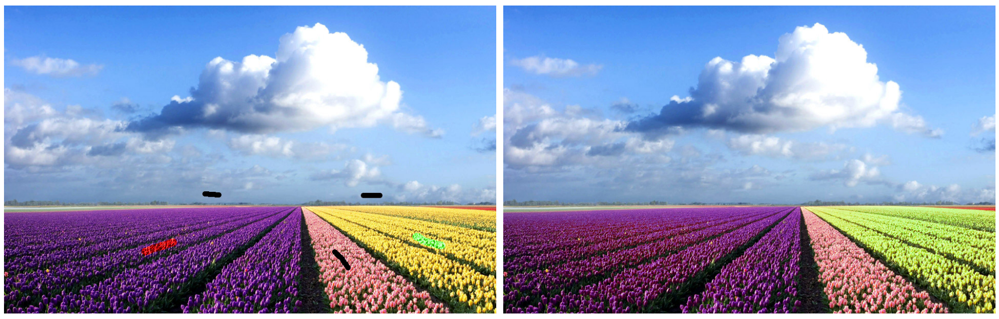


Figure 4.3: Image with user strokes of different colors (red and green) and the result of edit interpolation.

In our approach, we leave the non-negative constraint behind and aim to prevent the above-mentioned unwanted edits by other means, including adjusting σ of the kernel (eq. 3.2) or applying more interpolation constraints using black strokes.

We also extend the domain of our interpolated edit function by adding a series of extra dimensions to $XYRGB$ space. Extended coordinates of such an "$XYRGB+$" space provide more features for each pixel, making the pixel affinity more accurate and this way preventing the edits from being applied to the regions with different texture, despite the similarity in colors.

4.2 Gabor wavelets and filters

In a broad sense, texture of an image is a set of intensity variations that take place on each of its parts. On such images, each particular intensity variation can be considered as a fluctuation or oscillation of intensity values over some surrounding region. To be able to describe those oscillations in mathematical terms, we are going to reference *wavelets*: wave-like oscillations with an amplitude that diminishes to zero away from the centre (Figure 4.4). A wavelet is a prototypical case of an oscillation with known or preset parameters, while several wavelets with different parameters can be used as a basis to decompose and process more complex functions or signals. Images can be considered as two dimensional signals and processed using two dimensional wavelets. In the following sections we are going to use the *Gabor wavelets*, which have been already used in a number of works for extraction of texture features from images and texture segmentation.

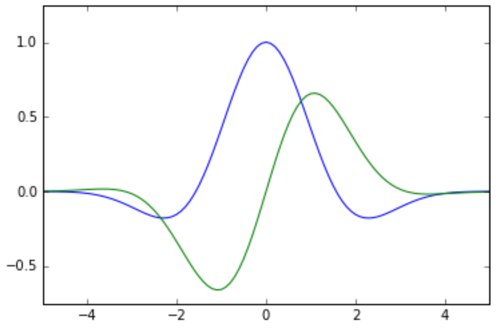


Figure 4.4: Real and imaginary parts of a Gabor wavelet.

While wavelets just serve as basis functions, the actual signal processing tool is a digital *filter* – a system that performs mathematical operations on signals. An image filter can be represented by its *kernel*: a matrix of pixels in which each pixel is assigned a value or a weight. Such matrices are usually much smaller than the images, for example, two of the Gabor kernels, represented in Figure 4.5, are only 35x35 pixels each.

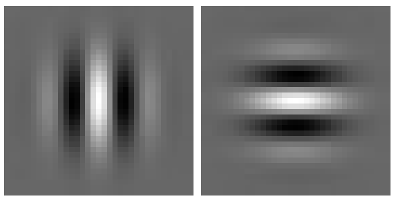


Figure 4.5: Two Gabor kernels.

To apply the filter, we *convolve* its kernel with the image, by sliding the center of the kernel over every pixel and multiplying each value in the kernel by the intensity value of the pixel directly underneath it. These values are then summed together, divided by the overall number of terms and stored as the resulting output intensity of the pixel that was directly under the center of the kernel.

Each Gabor kernel has two parameters: frequency λ and direction θ. When convolved with the image, the Gabor kernel reflects energy at that specific frequency and direction. By giving the highest response (returning the highest value) at the locations with corresponding intensity changes, the filter provides a localized frequency description and captures local features of the signal.

For example, the two Gabor kernels shown above have the same frequency but different direction: $\theta_1 = 0$ (vertical) and $\theta_2 = \pi/2$ (horizontal). The results of convolving those kernels with a simple image of a brick pavement (that just has vertical and horizontal lines) are illustrated in the Figure 4.6.

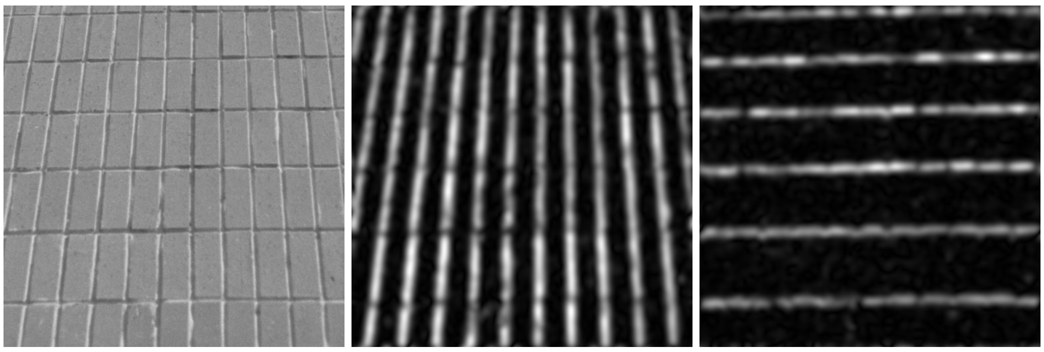


Figure 4.6: An image of a brick pavement and its convolution with two Gabor kernels of different directions.

It is easy to see from the examples that convolution with each particular kernel highlights only the intensity fluctuations of its particular orientation. Another one of the kernel parameters, frequency, is responsible for the size of the details it captures. In the Figure 4.7 (middle) we can see the result of convolving another image with a kernel: the waves on the water surface got highlighted. Then, convolving it with a kernel that has twice the frequency, results in (Figure 4.7, right) highlighting the smaller details. For more efficient processing of colored textures in further algorithms, we consider intensity values of corresponding grayscale textures of the sample images, rather than each color channel separately.

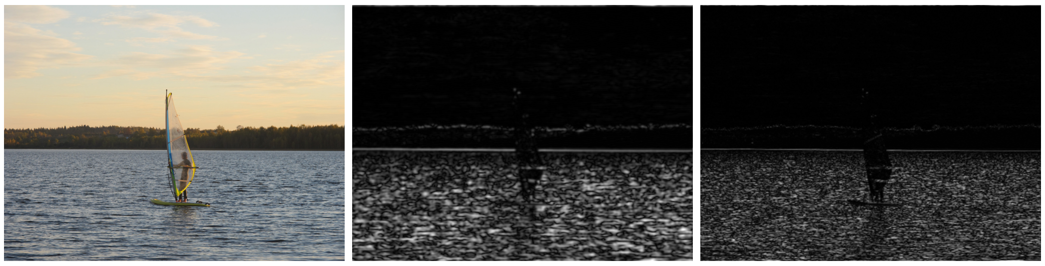


Figure 4.7: An image of a windsurfer on the water surface and its convolution with two Gabor kernels of different frequencies.

The ability to give a response to the details with certain properties makes the Gabor filters valuable in edge detection and texture segmentation. Moreover, simple cells in the visual cortex of the brain of mammal species are proven to be modelled by Gabor functions [6, 27], creating a hypothesis that the Gabor filters work similar to the part of perception process in the human visual system.

4.3 Edit propagation using Gabor filters

Now suppose we have an image with two different objects on it that have similar shades of color (e.g. house and field in Figure 4.8a), and we want to edit just one of them while constraining another one to be unchanged. However the approach, outlined earlier in this section, almost ignores the constraints that we put on the house of the same color (Figure 4.8b), making it greenish.

This happens because, given only the positions and colors of the user-specified pixels, the affinity is strong for all the pixels with the similar shades of color, shown in Figure 4.9a. Edits are therefore still propagated to the pixels of the similar color, including the part with the house. Moreover, the constraints put on the part of the same color produce negative weights in the interpolation, resulting in editing another part of the house with purple color, which is the opposite to green.

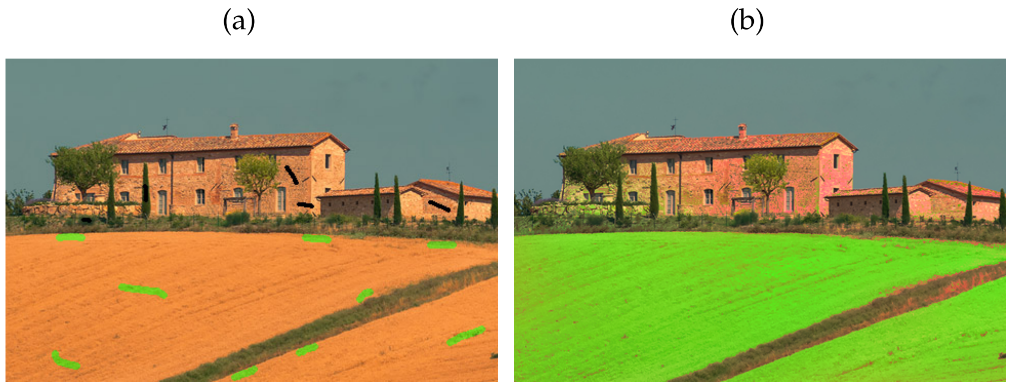


Figure 4.8: (a) Image with specified edits, RAW image courtesy of Elio Ausili . (b) Result of applying an existing edit propagation approach.

The way to distinguish a pixel of the house from a pixel of the field is obvious from the results of one of the Gabor filters (Figure 4.9b): the Gabor values for the house (light gray and white) are different from the ones for the field (dark gray and black). Thus, by the analysis of Gabor values, it is possible to differentiate between the two objects of similar color but different texture, we propose that those values can be used as extra coordinates of our points.

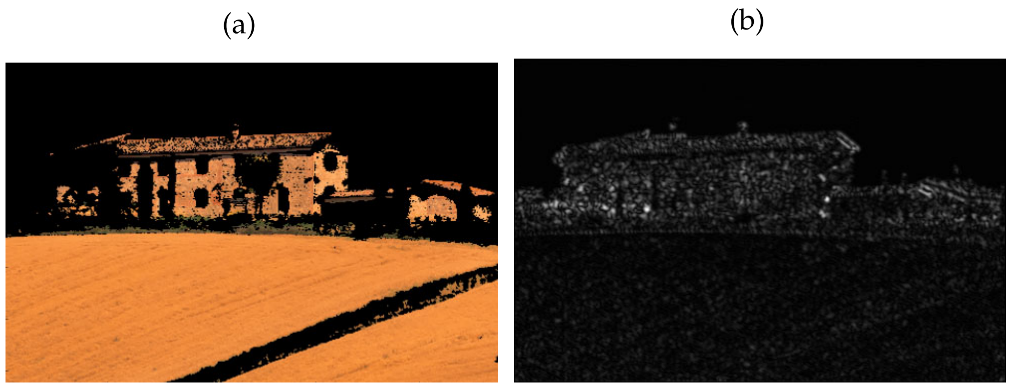


Figure 4.9: (a) Pixels that have strong affinity with the user-specified ones. (b) Result of applying Gabor filter to the original image.

Suppose we selected a set of filters, including m groups, each with its own frequency ($\lambda_1, ..., \lambda_m$). Then, each one of those groups also has 6 filters with different directions ($k\pi/6$, $k = 0, ..., 5$):

$$\theta_1 = 0,\ \ \theta_2 = \frac{\pi}{6},\ \ \theta_3 = \frac{\pi}{3},\ \ \theta_4 = \frac{\pi}{2},\ \ \theta_5 = \frac{2\pi}{3},\ \ \theta_6 = \frac{5\pi}{6}.$$

Even though we are free to use any arbitrary filter, the proposed set of $6m$ filters (with varying number and values of frequencies) was used in most of our experiments and is helpful to specify our notations unambiguously.

To extend our $XYRGB$ space, we first apply each of $6m$ filters to our original image (Figure 4.10) and get $6m$ resulting images with a Gabor value in each pixel. This allows us to consider each pixel $\mathbf{p}$ as a point in a $\mathbb{R}^{5+6m}$ space:

$$\mathbf{p} = (x, y, r, g, b, g_{11}, g_{21}, g_{31}, g_{41}, g_{51}, g_{61}, g_{12}, ..., g_{6m})$$

where g_{ij} is the Gabor value at a point of an image, convolved with the Gabor kernel of direction θ_i and frequency λ_j. Now, having the same edits at some points $\mathbf{p}_i$, as specified at Figure 4.9a, we need to interpolate three editing functions $\bar{h}_r, \bar{h}_g, \bar{h}_b : \mathbb{R}^{5+6m} \to \mathbb{R}$, for each of the three RGB color channels:

$$\bar{h}_r(\mathbf{p}_i) = red_i\ ;\ \bar{h}_g(\mathbf{p}_i) = green_i\ ;\ \bar{h}_b(\mathbf{p}_i) = blue_i\ ,\quad 1 \le i \le n$$

Here, as before, we put our interpolation conditions into the system $\Phi \cdot w = F$ (equation 3.4) and solve it for the weights w, to get the interpolated values of $(\bar{h}_r, \bar{h}_g, \bar{h}_b)$ at all of the points:

$$\bar{h}_r(\mathbf{p}) = \sum_{i=0}^{n} w_{ri}\phi(\|\mathbf{p} - \mathbf{p}_i\|),\ \bar{h}_g(\mathbf{p}) = \sum_{i=0}^{n} w_{gi}\phi(\|\mathbf{p} - \mathbf{p}_i\|), \qquad (4.1)$$

$$\bar{h}_b(\mathbf{p}) = \sum_{i=0}^{n} w_{bi}\phi(\|\mathbf{p} - \mathbf{p}_i\|)$$

However, another issue should be considered: even though the interpolation kernel ϕ only depends on the distance between $\mathbf{p}$ and $\mathbf{p}_i$, this

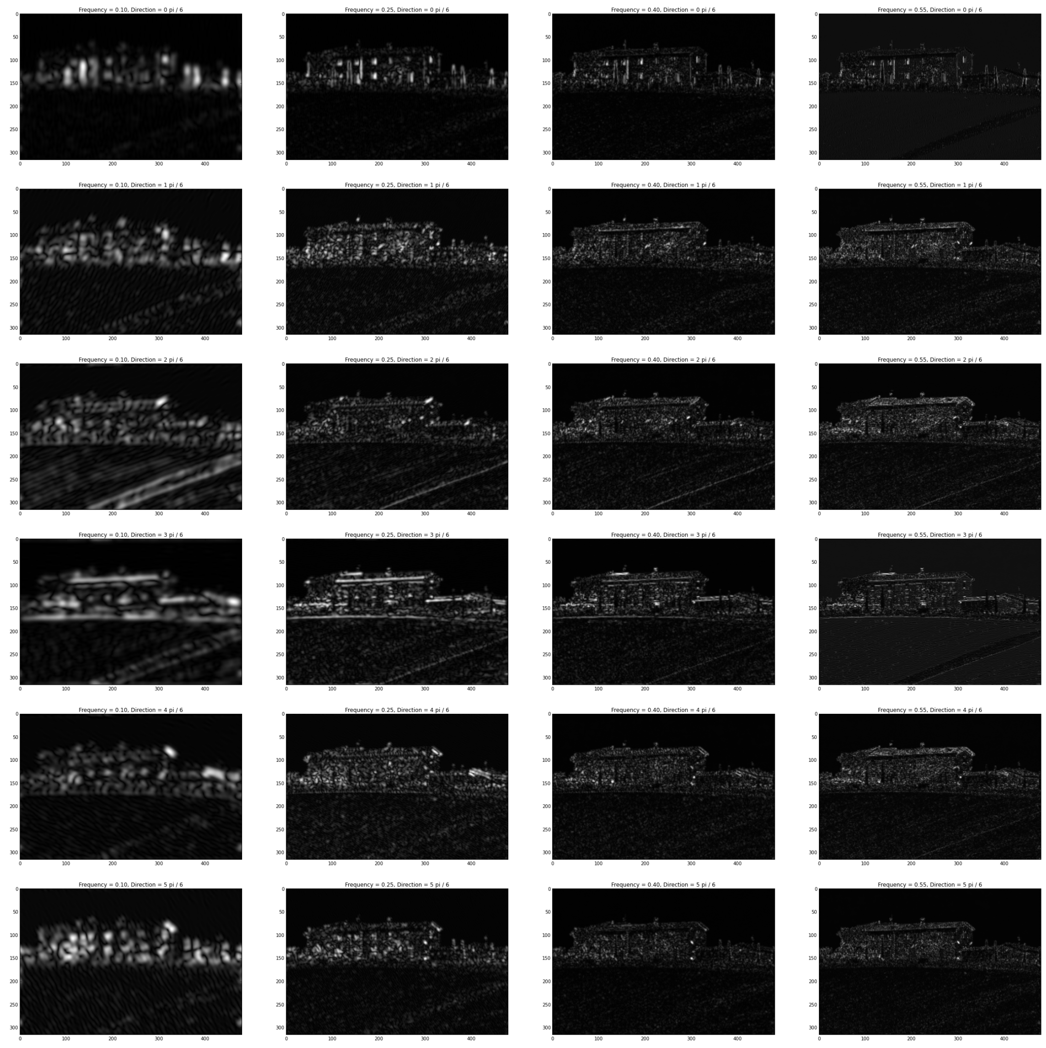


Figure 4.10: Results of selected 24 Gabor filters with frequencies 0.1, 0.25, 0.4, 0.55 applied to an image.

distance can be computed in various ways. The most common way is the Euclidean distance:

$$\|\mathbf{p}-\mathbf{q}\| = \sqrt{(\mathbf{p}-\mathbf{q})\cdot(\mathbf{p}-\mathbf{q})} = \sqrt{\sum_{i}^{5+6m}(p_i-q_i)^2} \tag{4.2}$$

where $\mathbf{p}, \mathbf{q} \in \mathbb{R}^{5+6m}$ are any two points in our affinity space, and p_i, q_i are the i-th coordinates of those points.

But in this sum of squared differences, we may also need to manually select weights ω_i to emphasize particular dimensions:

$$\|\mathbf{p}-\mathbf{q}\|_\Omega = \sqrt{\sum_{i}^{5+6m} \omega_i (p_i - q_i)^2} = \sqrt{(\mathbf{p}-q)\cdot\Omega\cdot(\mathbf{p}-\mathbf{q})} \tag{4.3}$$

where Ω is a diagonal matrix of all the weights:

$$\Omega = \begin{bmatrix} \omega_1 & 0 & 0 & \cdots & 0 \\ 0 & \omega_2 & \cdots & & 0 \\ 0 & \cdots & & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & 0 & & & \omega_{5+6m} \end{bmatrix} \tag{4.4}$$

Proper tuning of these weights in our distance function helps achieve desired interpolation results. For example, varying values of ω_i we can tune our edit propagation to be based on high-frequency details rather than low-frequency ones, or we can assign small weights to color differences, so edits will be propagated to a wider range of colors. A sample result of edit propagation, achieved after weight adjustment, is presented in Figure 4.11.

4.4 Random projection

Using Gabor values as extra components of pixels' feature vectors, we work in at least $(5 + q \cdot r)$-dimensional space, where q is the number of directions and r is the number of frequencies of the chosen Gabor filters. The large number $q \cdot r$ of filters adds computational cost. As well, a large number of features requires consideration of high-dimensional phenomena (collectively known as the curse of dimensionality [12]). For example, the amount of training data required for classification or regression may scale exponentially in the number of dimensions.

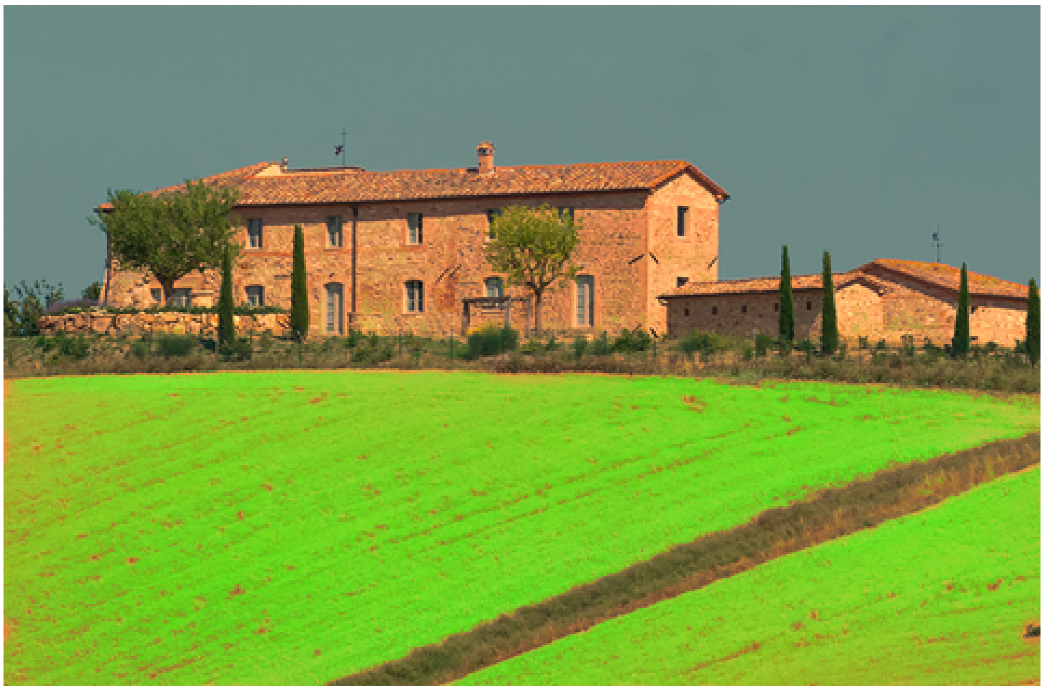


Figure 4.11: Results of edit propagation, performed as an interpolation of edits in selected high-dimensional space with 24 Gabor filters.

There are a number of approaches to dimensionality reduction, including several that were surveyed in Chapter 2. In this work we explore a *random projection* approach based on the Johnson Lindenstrauss lemma [19]. This lemma states that N points in a high-dimensional space can be projected into a space of about $\log N$ dimension, with inter-point distances approximately preserved. Suppose we have a set of $q \cdot r$ Gabor values for every pixel of the image:

$$\mathbf{g} = (g_{11}, ..., g_{qr})$$

The projection X^{RP} of such set of Gabor values can then be accomplished by simply multiplying the original data g for every pixel with an appropriately constructed random matrix R:

$$X^{RP}_{s\times 1} = R_{s\times qr}\mathbf{g}_{qr\times 1}$$

where $s < qr$ is the number of dimensions of the projected data and the matrix R is a randomly generated matrix with variance $1/s$.

The matrix R is generated only once and then multiplied with the vector of Gabor values for every pixel. However, after the dimensionality reduction is performed, it would be impossible to apply weight adjustment for the same set of dimensions, as done in the equation 4.3 by the matrix of weights Ω. For this reason, R can be initially multiplied with Ω:

$$X^{RP}_{s\times 1} = (R_{s\times qr}\Omega)\mathbf{g}_{qr\times 1}$$

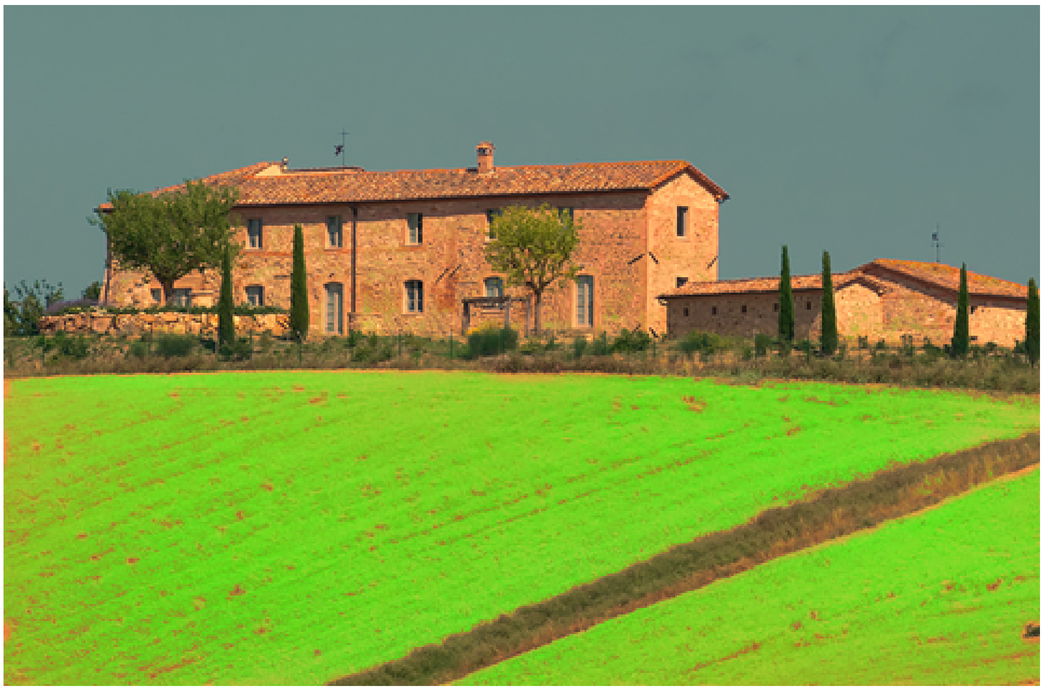


Figure 4.12: Result of edit propagation, performed as an interpolation of edits using 400 Gabor filters.

When the number of dimensions is big enough, the distribution of random values preserves distances between points well: looking at the result of edit propagation using 400 Gabor values (Figure 4.12), reduced to 40 by the random projection method, it is hard to tell the difference with the original result that did not use random projection (Figure 4.11). However, applying the random projection to a space with a relatively low number of dimensions may produce different results for different random matrices (Figure 4.13).

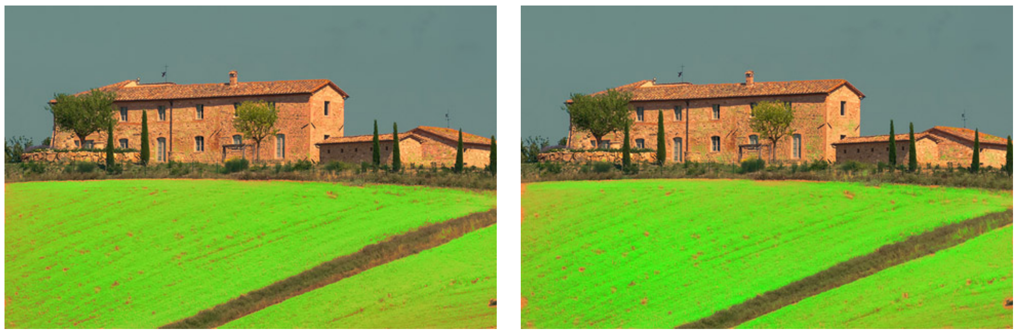


Figure 4.13: Two results of edit propagation with random projection method, applied to the space with 24 dimensions (Gabor values): despite the same initial parameters, right image receives bigger amount of edit.

Moreover, the complexity of the whole edit propagation procedure can be done by the analysis of how the numbers of dimensions (Gabor values) and RBF data points (equation 4.2) affect the overall number of basic operations performed (such as multiplications). At the first step of dimensionality reduction, the original data g for every pixel is multiplied with a constructed random matrix R, requiring $s \cdot qr$ multiplications times the number of image pixels. Then, computation of distances from every pixel to each of the data points requires $s \cdot n$ multiplications times the number of pixels, where n is the number of data points. Therefore, total amount of multiplications for every pixel during an interpolation with dimensionality reduction is: $s \cdot (qr + n)$.

For the interpolation without dimensionality reduction, computation of distances from every pixel to each of the data points (equation 4.3) requires $qr \cdot n$ multiplications times the number of pixels. Thus, the edit propagation approach with dimensionality reduction requires k times less multiplications for every pixel, where k is the ratio:

$$k = \frac{qr \cdot n}{s \cdot (qr + n)}$$

In both of the above-mentioned cases (with $qr = 24$ and $qr = 400$), we used about 100 data points ($n = 100$). However, in the relatively low-dimensional space of 24 Gabor values, it was possible to reduce the dimensionality to the minimum of 12, otherwise the precision was unacceptably low:

$$k_{24 \to 12} = \frac{24 \cdot 100}{12 \cdot (24 + 100)} = \frac{2400}{1488} \approx 1.6$$

But if using a big number of Gabor values, such as 400 in total, the complexity is reduced significantly:

$$k_{400 \to 40} = \frac{400 \cdot 100}{40 \cdot (400 + 100)} = \frac{40000}{5000} = 8.$$

Therefore, the given scheme reduces speed successfully, but only generates consistent results in spaces of unreasonably high dimension.

4.5 Results and discussion

In an example of edit propagation, presented in the Figure 4.8a, we used strokes of bright green color to make the applied edits more obvious. As the result (Figure 4.11), edits got propagated only to the region with the specified texture: Figure 4.14 demonstrates a comparison of edits, applied to the image with and without using Gabor values as image features.

It is also important to note that, despite the restriction on the texture, there is a chance for some undesired pixels to still have similar appearance to the user-specified ones (in terms of Gabor features) and to receive small amounts of edits. This shortcoming can be seen in the upper half of Figure 4.14 (right), however it is less obvious in our next examples, when using more natural colors.

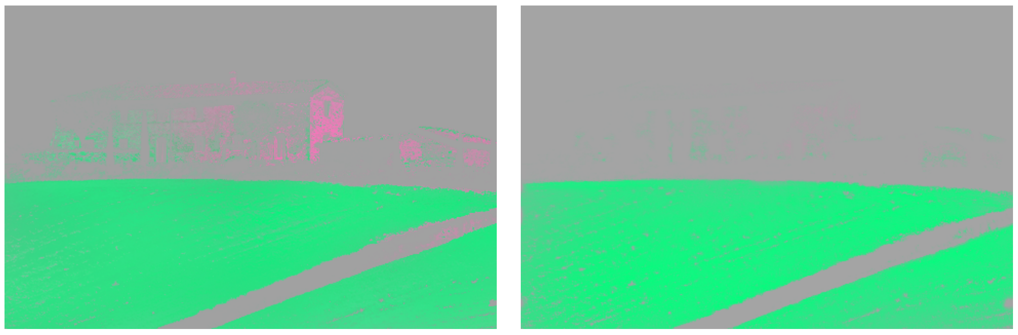


Figure 4.14: Edits of green color, applied after RBF interpolation with Gabor features (right) and without Gabor features (left). Zero edits are denoted as gray.

To get such results, however, proper tuning of a series of parameters is required. Appropriate values were selected experimentally for the following list:

- Set of *frequencies* and *directions* of the Gabor filters (section 4.2);
- Weights ω_i on the diagonal of matrix Ω (equation 4.3);
- σ coefficient of the Gaussian kernel (equation 3.2).

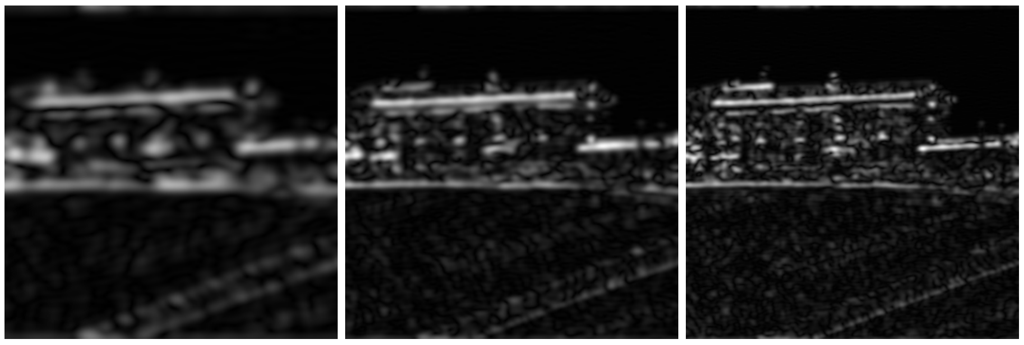


Figure 4.15: Results of convolving the sample image with kernels of frequencies 0.1, 0.175 and 0.25 (from left to right), cropped for convenience.

The selection started with a frequency $\lambda_1 = 0.1$. The corresponding kernel (Figure 4.5) is 35x35 pixels in size (convolving an image with a kernel of a lower frequency takes too long due to an even bigger size of such kernel). Then, from the results of convolutions with kernels of higher frequencies (Figure 4.15), it is clear that the middle result ($\lambda = 0.175$) is similar to the results on both left and right sides. Due to concerns about duplicated texture information and computational efficiency, $\lambda_2 = 0.25$ was selected as the next value of frequency.

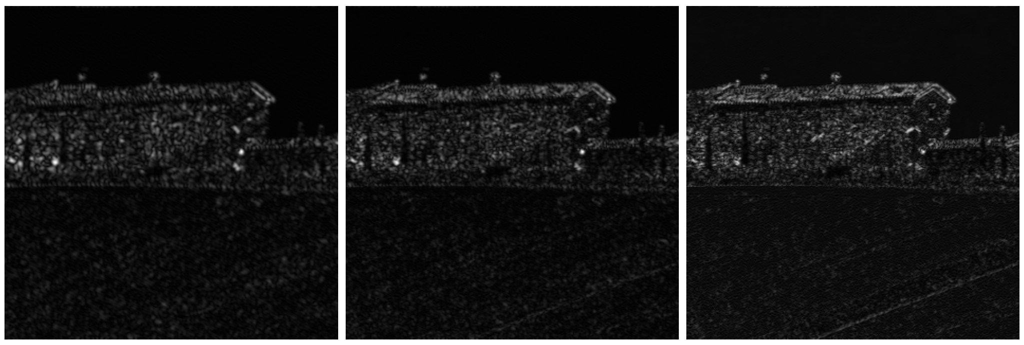


Figure 4.16: Results of convolving the sample image with kernels of direction $\frac{2}{3}\pi$ and frequencies 0.4, 0.5, 0.6 (from left to right).

On the other hand, Figure 4.16 shows that kernels of high frequencies capture similar details. Based on this observation, a frequency $\lambda_3 = 0.4$ was added to get a minimum set of three frequencies: one low, one middle and one high. For the case of very small details, one more frequency $\lambda_4 = 0.55$ was also put into the set optionally.

The results of filters with different directions are shown in Figure 4.17. The image in the middle ($\theta = \frac{\pi}{12}$) also appears to combine some information from both left and right results, leading us to the selection of a $\frac{\pi}{6}$-step between the directions of kernels. This concludes our choice of six directions ($k\pi/6$, $k = 0, ..., 5$) and gives us the total of 24 corresponding kernels and filters, mentioned previously in Figure 4.10.

Obviously, this selected set of filters is minimal, and can be extended

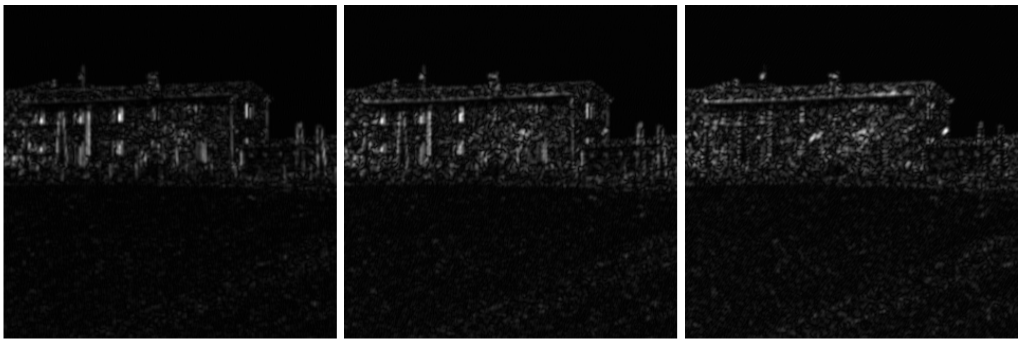


Figure 4.17: Results of convolving the sample image with kernels of frequency 0.4 and directions $0, \frac{\pi}{12}, \frac{\pi}{6}$ (from left to right).

if more details are required. The efficiency of using a bigger set of filters can also be increased by applying dimensionality reduction, explained in section 4.4 of the previous chapter.

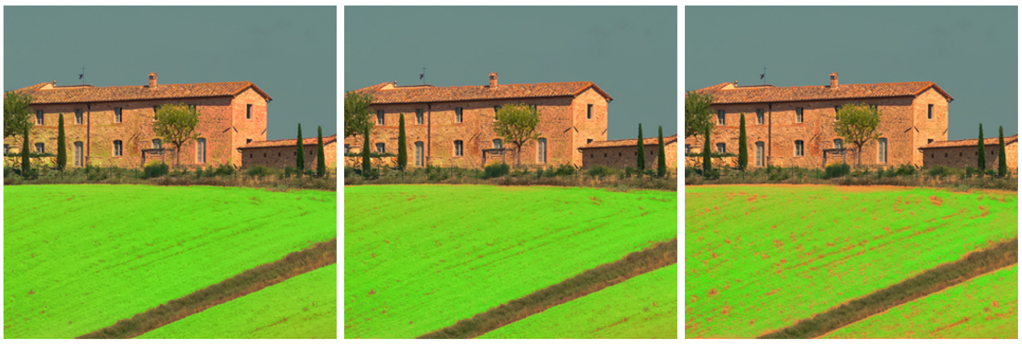


Figure 4.18: Results of edit propagation that use different weights for Gabor features during interpolation.

The edit propagation procedure was then tested on a series of parameter values to adjust the weights ω_i (equations 4.3 and 4.4) for all of the Gabor features. Figure 4.18 demonstrates three of the sample outputs that have insufficient, moderate and excessive edits. With all weights for Gabor features set about the same ($\omega = 1.0$), the result (left) is similar to the one that use no Gabor features at all: the edits are propagated to the building as well. But setting weights too high ($\omega = 5.0$, right) puts too much emphasis on the Gabor features and results in the lack of edits for some parts of texture. Selecting a weight in the middle ($\omega \approx 3.0$, middle) produces the desired result.

The final parameter to adjust is σ (Figure 4.19): a coefficient of the Gaussian kernel (equation 3.2), also known as the kernel width. If σ is set too low ($\sigma = 0.1$, left), the edits diminish sharply away from the user-specified strokes, while high kernel width results in edits applied to a much wider region ($\sigma = 0.3$, right). As before, we select a value in the middle ($\sigma \approx 0.19$, middle) that provides the best results.

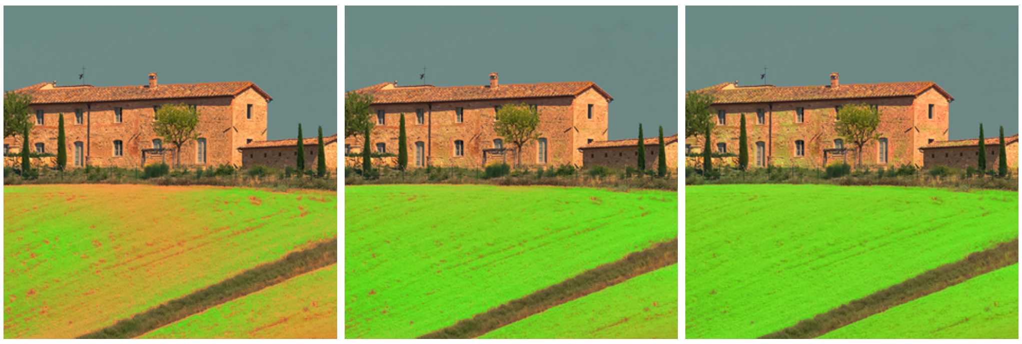


Figure 4.19: Results of edit propagation that use different width of Gaussian interpolation kernel (0.1, 0.2 and 0.3).

Two more images with user-specified strokes are presented in Figure 4.20 to demonstrate the success of our method. The results of edit propagation (both with and without Gabor features) are shown in detail in the following figures.

Figure 4.21 demonstrates the propagation of yellow-colored strokes to a green lawn. Without using Gabor features (Figure 4.21a), some of the green plants also received some edits (most obvious on their light parts). Shades of blue also appeared on the green leaves away from constraining strokes, as a consequence of negative interpolation weights. On the other hand, interpolation with Gabor features left most of the plants unaltered, only altering the lawn (Figure 4.21b).

Another example in Figure 4.22 shows the propagation of red-colored strokes to a brick wall. Since the dark brown color of bricks is similar to the color of the wooden door, it also receives edits if not using Gabor features. Moreover, negative interpolation coefficients slightly alter the white wall making it bluish. An approach that uses Gabor features (Figure 4.22b) prevents those edits, altering the colors of the desired areas only.

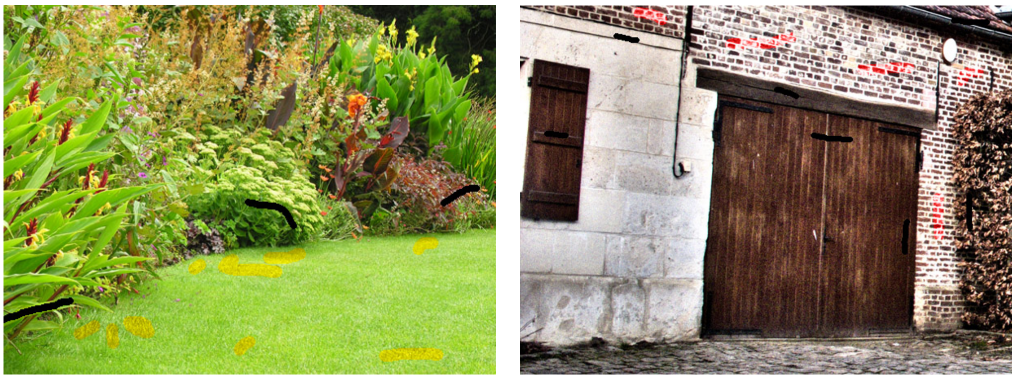
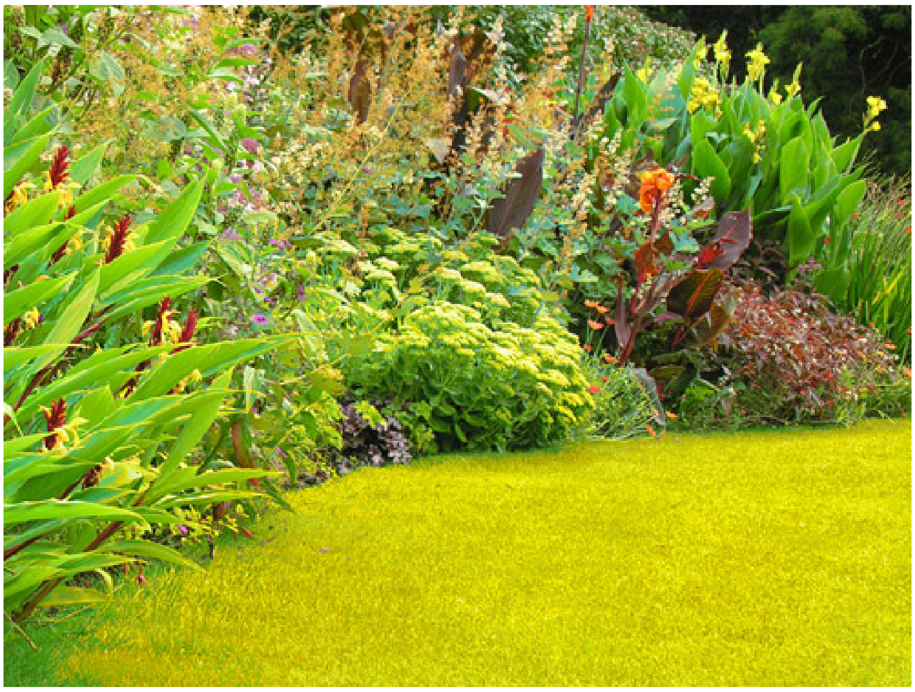


Figure 4.20: Two more images with user-specified strokes of yellow and red colors.

(a) RBF interpolation



(b) RBF+ Gabor

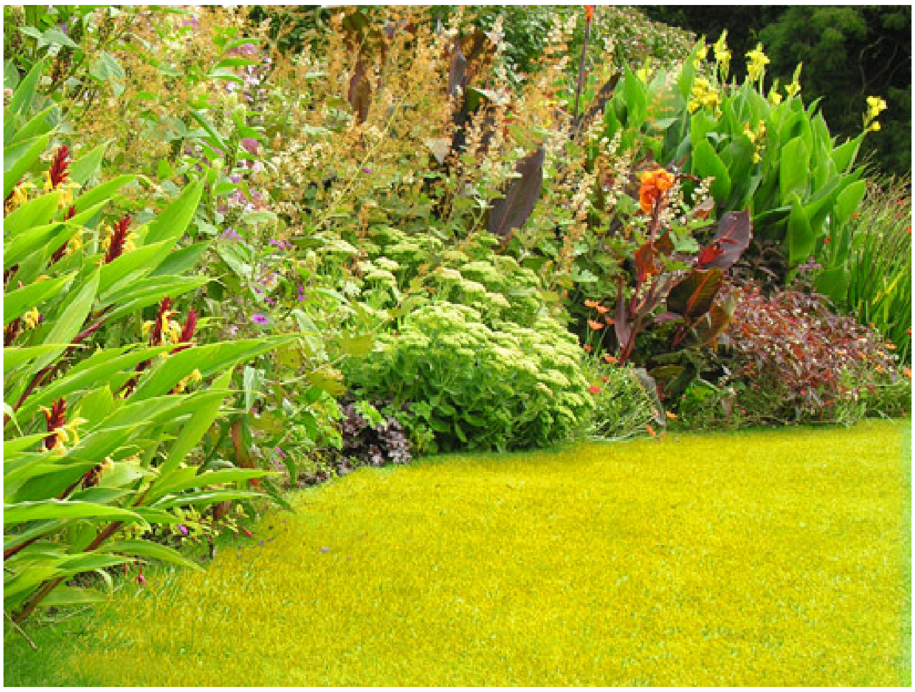
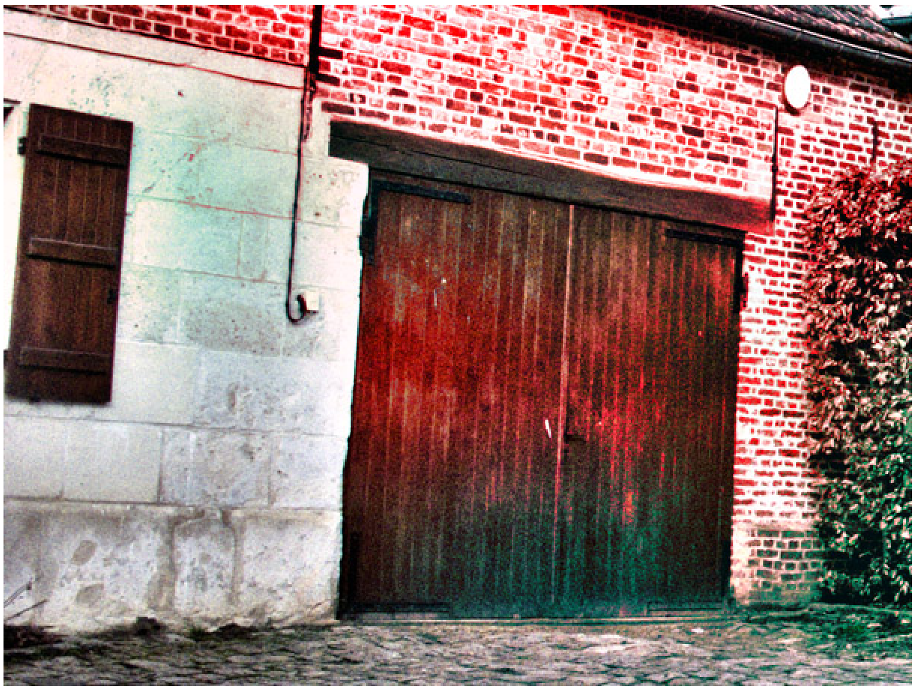


Figure 4.21: Edit propagation results for yellow strokes, applied to the lawn.

(a) RBF interpolation



(b) RBF + Gabor

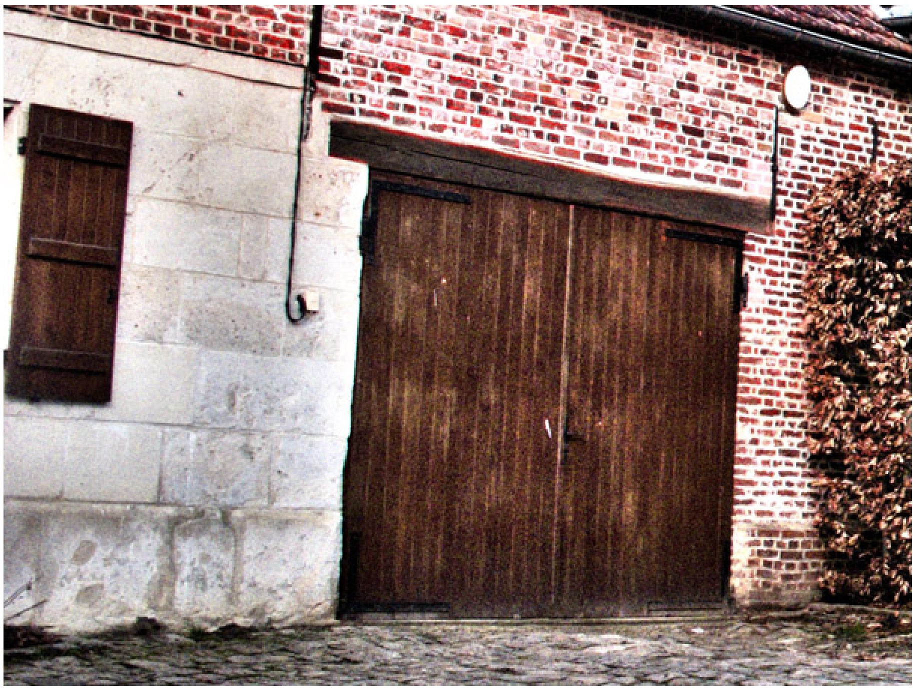


Figure 4.22: Results for red strokes, propagated to the bricks of the wall. RAW image courtesy of Edward Musiak, https://flic.kr/ps/FRqCG .

Chapter 5

Conclusion and Future Work

In this thesis, radial basis function regression was shown to be a useful tool for user-guided image editing, with successful applications to image deformation and color editing. Some enhancements were also made to the space of RBF interpolation of color edits, making the result of editing more precise.

The thesis compared B-spline and RBF interpolation approached to image deformation. In B-spline warping approaches, there is a notable trade-off between an easy control over the control point grid and a level of detail for the deformation: more detailed deformations need more points in the control point grid. Using B-splines for warping, the user usually has to either place control points on a grid that does not necessarily align with features of the images, or to adjust some of the redundant control points that are an essential part of the grid structure.

Warping that uses RBF, conversely, allows placing a suitable number of control points at desired, semantic locations (such as the corner of the mouth) along with a specification of where each control point should be moved. RBF allows the control points to be placed exactly where desired, and only at those places. Then, given the positions of the control points along with the desired movements, it provides smooth and artifact-free results of the deformation.

Next, this thesis presented a texture-aware extension to the edit propagation approach, based on RBF interpolation. Extending $XYRGB$ feature space with Gabor features makes the propagation more selective and allows edits to be applied according to the texture and colors of the regions rather than colors solely. It also provides a good alternative to the non-negative constraint on interpolation and does not require numerical optimization.

On the other hand, problems of high-dimensional spaces may arise if working with a large number of Gabor features. We found that the random projection technique (see section 4.4) applies only to the spaces with irrelevantly high numbers of dimensions. For the cases with a relatively small number of Gabor features (e.g. 18-24 in the presented examples), compact and accurate representation of the image texture features is still an open problem.

Minor faults of the method were also observed in the regions where Gabor values partly coincided with the values for edited regions, resulting in slight edits of undesired pixels (upper half of Figure 4.14). Small bits of texture that carry uniform color are especially liable for this defect. This way, the problem of finding an accurate affinity function for two sets of Gabor features also needs to be addressed in future work.

Still, results of experiments demonstrate the success of using Gabor features to better distinguish points with different textured appearances. These results were also achieved without requiring explicit segmentation of the image. The approach can be used for texture-aware user-guided edit propagation.

Bibliography

[1] AN, X., AND PELLACINI, F. Appprop: All-pairs appearance-space edit propagation. *ACM Trans. Graph. 27*, 3 (Aug. 2008), 40:1–40:9.

[2] ANJYO, K., LEWIS, J. P., AND PIGHIN, F. Scattered data interpolation for computer graphics. In *ACM SIGGRAPH 2014 Courses* (New York, NY, USA, 2014), SIGGRAPH '14, ACM, pp. 27:1–27:69.

[3] ARAD, N., DYN, N., REISFELD, D., AND YESHURUN, Y. Image warping by radial basis functions: Application to facial expressions. In *CVGIP: Graphical Models and Image Processing* (1994), pp. 161–172.

[4] BEIER, T., AND NEELY, S. Feature-based image metamorphosis. *SIGGRAPH Comput. Graph. 26*, 2 (July 1992), 35–42.

[5] BINGHAM, E., AND MANNILA, H. Random projection in dimensionality reduction: Applications to image and text data. In *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2001), KDD '01, ACM, pp. 245–250.

[6] CAMPBELL, F. W., COOPER, G. F., AND ENROTH-CUGELL, C. The spatial selectivity of the visual cells of the cat. *The Journal of Physiology 203*, 1 (1969), 223–235.

[7] CHEN, X., ZOU, D., ZHAO, Q., AND TAN, P. Manifold preserving edit propagation. *ACM Trans. Graph. 31*, 6 (Nov. 2012), 132:1–132:7.

[8] COLLIGNON, A., VANDERMEULEN, D., SUETENS, P., AND MARCHAL, G. 3d multi-modality medical image registration using feature space clustering. In *Proceedings of the First International Conference on Computer Vision, Virtual Reality and Robotics in Medicine* (London, UK, UK, 1995), CVRMed '95, Springer-Verlag, pp. 195–204.

[9] DALAL, N., AND TRIGGS, B. Histograms of oriented gradients for human detection. In *In CVPR* (2005), pp. 886–893.

[10] DAUGMAN, J. G. Two-dimensional spectral analysis of cortical receptive field profiles. *Vision Research 20*, 10 (1980), 847 – 856.

[11] DAUGMAN, J. G. Complete discrete 2-d Gabor transforms by neural networks for image analysis and compression, 1988.

[12] DUDA, R. O., AND HART, P. E. *Pattern Classification and Scene Analysis.* John Willey & Sons, New Yotk, 1973.

[13] DUNN, D., AND HIGGINS, W. Optimal Gabor filters for texture segmentation. *Image Processing, IEEE Transactions on 4*, 7 (Jul 1995), 947–964.

[14] FORNEFETT, M., ROHR, K., STIEHL, H. S., AND SYSTEME, A. K. Radial basis functions with compact support for elastic registration of medical images. *IVC 19* (1999), 1–2.

[15] GLASBEY, C. A., MARDIA, K. V., IOMATHEMATICS, B., AND SCOTL, S. A review of image-warping methods. *Journal of Applied Statistics* (1998).

[16] GRIGORESCU, S. E., PETKOV, N., AND KRUIZINGA, P. Comparison of texture features based on Gabor filters. *IEEE Trans. on Image Processing*, 2002.

[17] HAJNAL, J. B., SAEED, N., OATRIDGE, A., WILLIAMS, E. J., YOUNG, I. R., AND BYDDER, G. M. Detection of subtle brain changes using

subvoxel registration and subtraction of serial mr images. *J. Comput. Assist. Tomogr 19(5)* (1995), 677–691.

[18] HILL, D. L. G., BATCHELOR, P. G., HOLDEN, M., AND HAWKES, D. J. *Medical image registration*, 2nd. ed. IOP Publishing Ltd, UK, 2001.

[19] JAIN, A. K., AND FARROKHNIA, F. Unsupervised texture segmentation using Gabor filters. *Pattern Recogn. 24*, 12 (Dec. 1991), 1167–1186.

[20] JOHNSON, W., AND LINDENSTRAUSS, J. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, vol. 26 of *Contemporary Mathematics*. American Mathematical Society, 1984, pp. 189–206.

[21] KAI LIAO, W., AND COHEN, I. Belief propagation driven method for facial gestures recognition in presence of occlusions. In *Proc. of IEEE conference on Computer Vision and Pattern Recognition workshop, 2006 Page(s):158*, p. 158.

[22] LAPEER, R., SHAH, S., AND ROWLAND, R. An optimised radial basis function algorithm for fast non-rigid registration of medical images. *Computers in Biology and Medicine 40*, 1 (2010), 1 – 7.

[23] LEE, T. S. Image representation using 2d Gabor wavelets. *IEEE Trans. Pattern Anal. Mach. Intell. 18*, 10 (Oct. 1996), 959–971.

[24] LI, Y., JU, T., AND HU, S.-M. Instant propagation of sparse edits on images and videos. *Comput. Graph. Forum 29*, 7 (2010), 2049–2054.

[25] LISCHINSKI, D., FARBMAN, Z., UYTTENDAELE, M., AND SZELISKI, R. Interactive local adjustment of tonal values. *ACM Trans. Graph. 25*, 3 (July 2006), 646–653.

[26] LOWE, D. G. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision 60*, 2 (Nov. 2004), 91–110.

[27] MARČELJA, S. Mathematical description of the responses of simple cortical cells∗. *J. Opt. Soc. Am. 70*, 11 (Nov 1980), 1297–1300.

[28] PALM, C., AND LEHMANN, T. M. Classification of color textures by Gabor filtering. *MG&V 11*, 2/3 (Sept. 2002), 195–219.

[29] PLUIM, J. P. W., MAINTZ, J. B. A., AND VIERGEVER, M. A. Mutual information based registration of medical images: A survey. *IEEE Trans. Med. Imaging 22*, 8 (2003), 986–1004.

[30] POWELL, M. J. D. The BOBYQA algorithm for bound constrained optimization without derivatives.

[31] PRESS, W. H., TEUKOLSKY, S. A., VETTERLING, W. T., AND FLANNERY, B. P. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, 3 ed. Cambridge University Press, New York, NY, USA, 2007.

[32] RAJESWARI, R., AND IRUDHAYARAJ, A. Image registration using radial basis function. In *Machine Learning and Computing (ICMLC), 2010 Second International Conference on* (Feb 2010), pp. 210–212.

[33] RHEE, T., HWANG, Y., KIM, J. D., AND KIM, C. Real-time facial animation from live video tracking. In *Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (New York, NY, USA, 2011), SCA '11, ACM, pp. 215–224.

[34] RHEE, T., LEWIS, J., NEUMANN, U., AND NAYAK, K. S. Scan-based volume animation driven by locally adaptive articulated registrations. *IEEE Transactions on Visualization and Computer Graphics 17*, 3 (2011), 368–379.

[35] ROHR, K., STIEHL, H., SPRENGEL, R., BEIL, W., BUZUG, T., WEESE, J., AND KUHN, M. Point-based elastic registration of medical image data using approximating thin-plate splines, 1996.

[36] ROWEIS, S. T., AND SAUL, L. K. Nonlinear dimensionality reduction by locally linear embedding. *Science 290*, 5500 (2000), 2323–2326.

[37] SANDLER, R., AND LINDENBAUM, M. Gabor filter analysis for texture segmentation. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition Workshop* (Washington, DC, USA, 2006), CVPRW '06, IEEE Computer Society, pp. 178–.

[38] STUDHOLME, C., HILL, D. L. G., AND HAWKES, D. J. Multiresolution voxel similarity measures for MR-PET registration. In *Information Processing in Medical Imaging: Proc. 14th International Conference (IPMI '95)* (1995), pp. 287–298.

[39] TENENBAUM, J. B., SILVA, V. D., AND LANGFORD, J. C. A global geometric framework for nonlinear dimensionality reduction. *Science 290*, 5500 (2000), 2319–2323.

[40] TIPPING, M. E., AND BISHOP, C. M. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B 61* (1999), 611–622.

[41] WOLBERG, G. Recent advances in image morphing. In *In Proc. Computer Graphics Internat* (1996), pp. 64–71.

[42] XU, K., LI, Y., JU, T., HU, S.-M., AND LIU, T.-Q. Efficient affinity-based edit propagation using k-d tree. *ACM Trans. Graph. 28*, 5 (Dec. 2009), 118:1–118:6.

[43] ZAHARESCU, A., BOYER, E., VARANASI, K., AND HORAUD, R. Surface feature detection and description with applications to mesh matching. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on* (June 2009), pp. 373–380.

[44] ZHANG, J., TAN, T., AND MA, L. Invariant texture segmentation via circular Gabor filters. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on* (Aug 2002), vol. 2, pp. 901–904 vol.2.