Image Editing from multiple photos to one single best

Erik Johan Sundin
(Student id: 10407030)
Supervisor:
dr.Ing Jan van Gemert

Faculty of Science

University of Amsterdam

This thesis is submitted for the degree of

Master of Science Programme in Artificial Intelligence
Track Intelligent Systems

May 2015

Abstract

This thesis is exploring multiple images taken from a similar view point. It contributes with a editing tool that makes it easy to recompose a photo from a burst of images after the images is photographed. With the editing tool it is simple to hand pick one single photo from the burst of images in a practical way. Our method handles multiple images taken with a hand held camera were an efficient alignment procedure makes up for small camera motions during the capture. We find differences within the dynamics of a scene. These differences enables the feature of recompose the photo by moving different regions from one photograph to another in a graphical user interface. Our pipeline is implemented with client-side web-technologies. A user study with questionnaire and interviews shows that our method is practical and it works well with a large real world example.

Source code and demo available at *http://josundin.github.io/magcut/*.

Acknowledgements

Thanks to the Flickr users *badkleinkirchheim* for making the sledge-set and the bike2-set, *dmourati* for the skate-set and for putting these images under the Creative Common licence.

Thanks to Morris Franken for providing useful feedback in the designing of the user interface.

Thanks to Jan van Gemert who was really supportive during the creation of this master thesis and gave good constructive feedback.

Lastly a thank you to all the respondents who took the time to answer the survey and the interview respondents who volunteered.

Table of contents

Chapter 1

Introduction

A lot of cameras contains Continuous Shooting mode. This is a feature that allows multiple shots to be captured in a quick moment by holding down the shutter button. More generally this mode is called burst. This feature was created to capture motion and to allow a selection of the best image within the burst.

The limitation has been in the computational power of the camera. This mode is something that depend mostly on memory buffer size and in the DSLRs the mechanism of the mirror flip is a key factor. Recently the computational power of smart phones, DSLRS and compact cameras have been rising. For example iPhone 6 burst mode takes 10 photos per second. Nowadays people tend to use burst mode greatly to take several photos of the same scene.

Today almost all devices and PCs have a web browser pre-installed and all web browsers come with a javascript-engine. This is why javascript is the client-side scripting language of web technologies.

In this thesis we present a plug-in free web browser image editing tool. We chose to use javascript and HTML5 to implement our proposal as the tool in this way becomes platform independent and users can run it in the browser on any operating system. The advantages are that we can in a flexible way reach out to a large variety of users to test our tool. Thereby we can get feedback easily.

1.1 Objectives

The goal of this thesis is to create an interactive tool that makes it easier for users to get the single best picture from multiple images taken within the same scene. This involves both picking the best preferred image and to do image editing on the photograph.

The aim of this thesis is also to investigate how independent users use our proposal. We conducted a user study with a questionnaire and interviews to determine the usability and the end-user experience of the image editing tool. With the opportunities to recompose and to simple hand pick an appropriate image we believe that users will be more aware of how to photograph a scene. In the end we expect that the awareness will lead users to become more accomplished photographers.

1.2 Burst of Images

Burst mode is a good alternative when it comes to photographing moving subjects. One example is taking pictures of sports. Sports-photography can be hard because there is a lot of activity going on and the athlete(s) often move very fast and the fields are big. With the camera in burst mode we can hold down the shutter and pan the camera along with the athletes movements while the camera captures the images in a sequence.

You often only get one chance to capture a good snapshot properly. To do this in a situation where many things are happening at ones, to freeze the moment of a dynamic scene, when everything looks right can be a harsh task. Putting the camera settings to burst mode is good in other situations as well. Wildlife photography, animals and children are similar to athletes in the way that they move fast.

The success rate is usually low and therefore photographers need good software to edit down the number of candidates and to do post-processing. It is here our proposal comes into play. Our proposal is an image editor that let the user rearrange the photographed scene in order to maximize the visibility or to change the composition after the photos were taken. This can be done in several ways. Either by adding and moving objects or by removing objects within the scene.

Our application is suited for dynamic scenes where objects are moving. It also allows the camera to move slightly within a scene. We assume that the burst of images provided by the user contains some differences between each single image.

Chapter 2

The goal of the Image Editing Tool

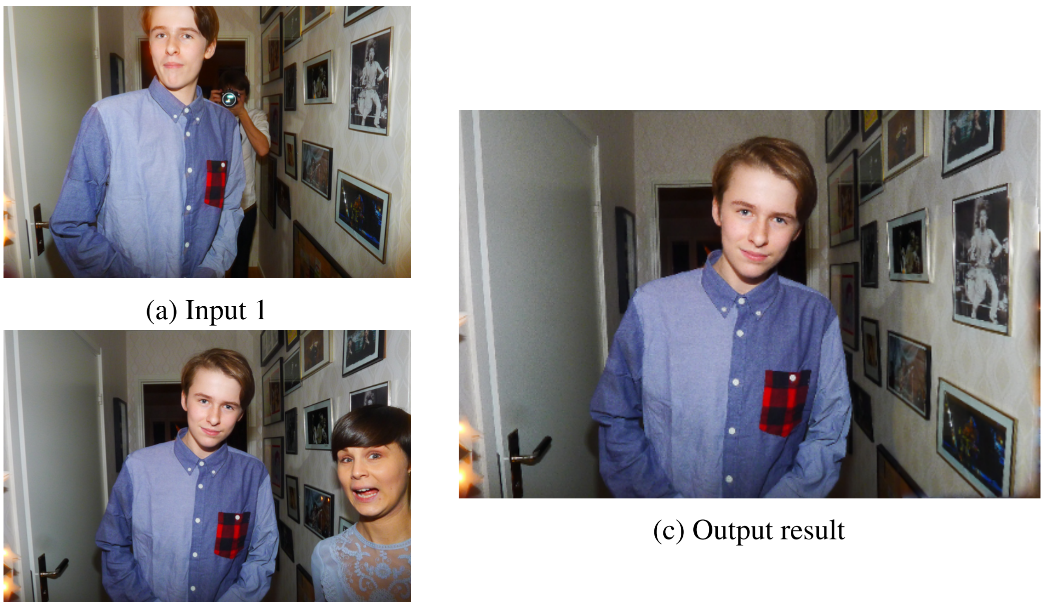
2.1 Use Cases Examples

Lets now take a look at what our proposed Image Editing Tool is capable of doing by going through a number of use cases of different examples. These examples scenarios are all relevant for usage in our application.

To demonstrate some breadth of the proposed Image Editing application we here present some pre-defined goals that are dealt with.

2.1.1 Example 1 Remove Content

We have two photographs taken from a sequence of a two person portrait in a corridor seen in figure 2.1 to the left. We decide that we only want a portrait of the person in the blue shirt. Ideally we want to combine the exposure *b* (down-left) where we have that person looking more relaxed then on exposure *a* (top-left), there are also other details like the photographer behind our protagonist who accidentally ended up in frame *a*. So in other words we want to remove the girl from image frame *b*. To do this we combine the two exposures and replace the girl to the right in frame *b* with the background wall from frame *a* to produce the result seen in figure 2.1 *c*.



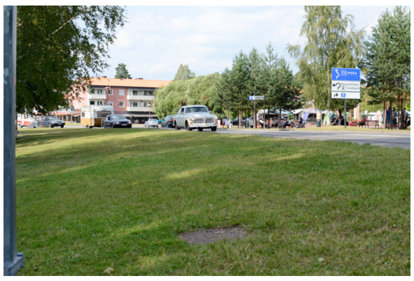
(a) Input 1

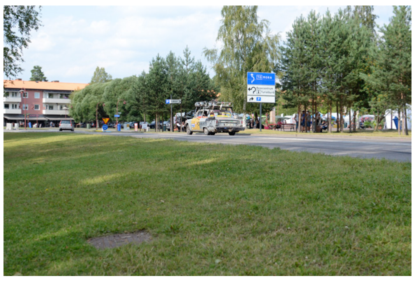
(b) Input 2

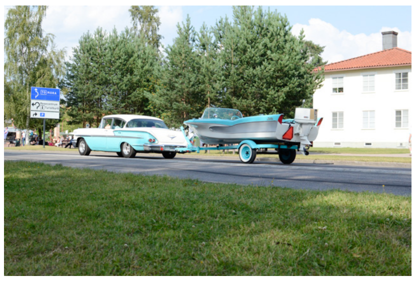
(c) Output result

Fig. 2.1 Example Remove Content

2.1.2 Example 2 Add content


(a) Input 1


(b) Input 2, Reference image


(c) Input 3

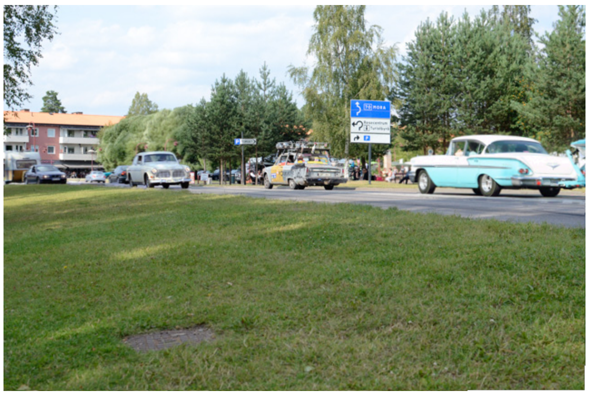
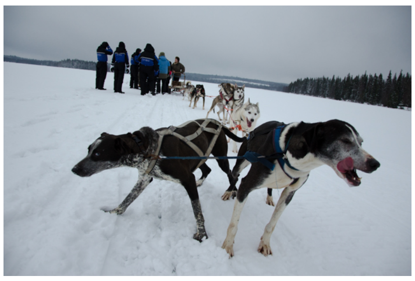

(d) Output Result

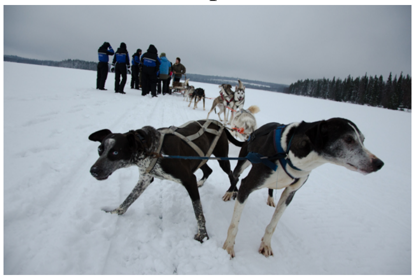
Fig. 2.2 Example Add Content

We have three photos taken from the same spot at slightly different times which essentially resulted in us capturing multiple cars. We have photographies of cars but in several different frames. The preference could be one single photo where as many cars as possible appear visible.

Our goal is to put together details from multiple images into one single frame. Here we are adding details from our pictures into one reference image. We select one reference view that will include an area of the multiple cars, this is the middle frame which also contains a car in the middle. By doing so we can now combine the left cars from the left frame and the right car from the right frame to create the concise photo we are looking for, see figure 2.2d.

2.1.3 Example 3 Swap Content


(a) Input 1


(b) Input 2

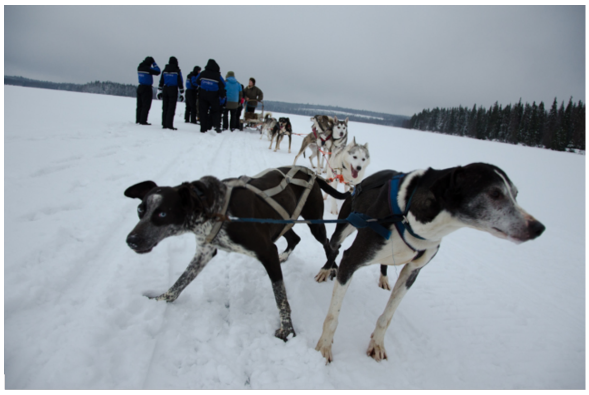

(c) Output result

Fig. 2.3 Example Swap Content

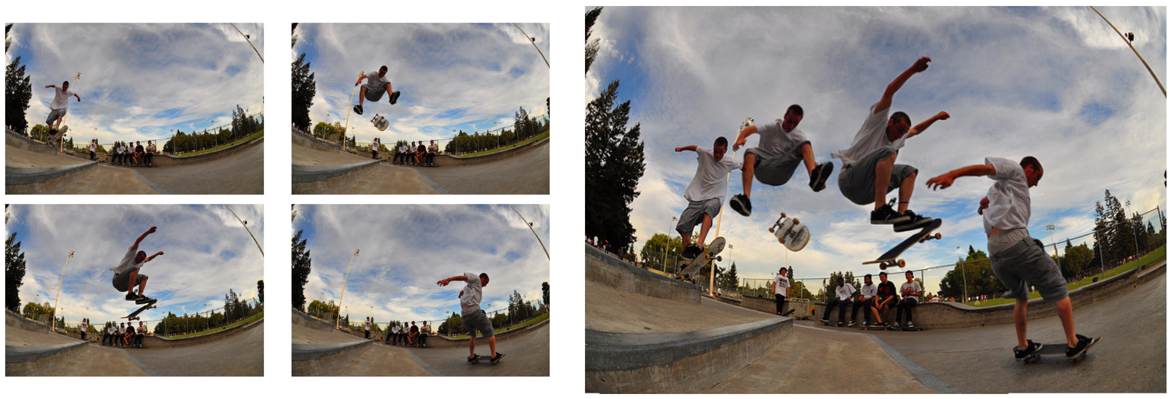
Here are two photographs from a sequence of shoots taken of a group of husky dogs. These animals are hyperactive and their movements can sometimes be very rapid. It is a difficult task to capture the moment when all the dogs look towards the camera and at the same time look engaged. To do this with only one single frame is even more challenging.

In this example we decided to combine the exposure to the top-left where the dogs in the middle are looking towards the camera with the down-left exposure where the two dogs in the front look more focused and more serious than on the top-left frame. So here we are replacing the faces of the two front dogs from image *a* with the faces of the dogs from image *b* while we simultaneously maintain the middle dogs looking towards us. The result is seen in figure 2.3 *c*.

When we replace the dogs faces from frame *a* with the frame *b* faces, the swapped result must match up in a color-wise symbiosis. This is where the two exposures becomes one in a seamless blend. We will come back to this in a later chapter.

2.1.4 Example 4 Duplicate Content

Here we want to capture the motion of two objects namely the skater and the skateboard in one single frame. The process is very similar to the previous example we are adding up



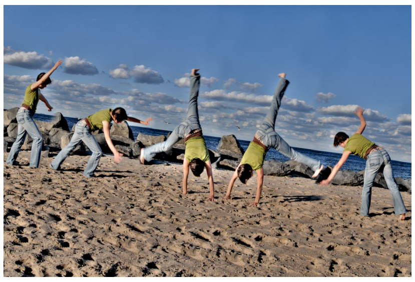
(a) Input images (b) Output Result

Fig. 2.4 Example Duplicating Content

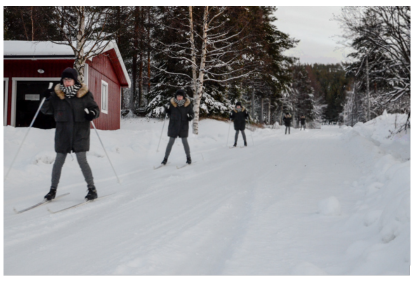
details from several different pictures into one single frame. The difference is that it is the same objects we are adding. Thus in this example we duplicate the two objects from our set of frames into one single frame.

With this technique one can use several images to, for example “clone” people within a scene and thereby visualize their path of motion in one single-frame. This cloning technique is not something new in the scope of Image Editing. It has been explored and exploited by photographers [3], but our application opens up a new more simple employment for the technique which is available to users under a less demanding approach. It is much easier to achieve results with our tool than with, for example commercialized software like photoshop. To be aware, the conditions are that we have appropriate data suited to the purpose in form of multiple photographies of the same scene and also taken from more or less the same viewpoint. And preferably, the scene in question should contain some form of dynamics.

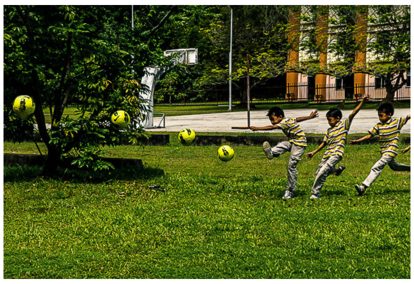
There are a lot of scenes that are of potential use for our application figures 2.5 shows just a few example results that can be obtained.



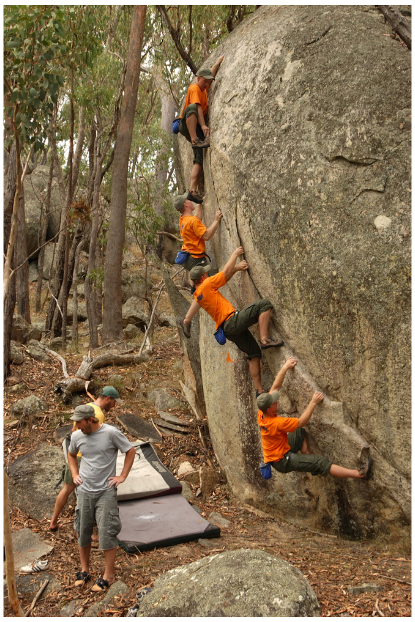
(a) Handstand



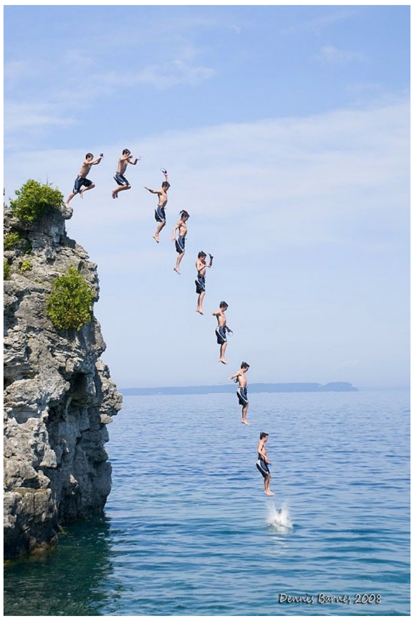
(b) Skiing



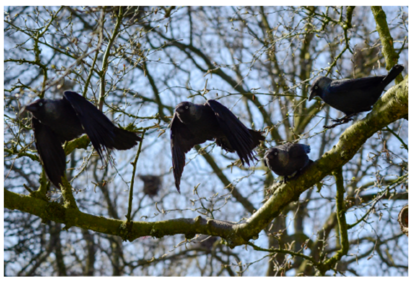
(c) Football



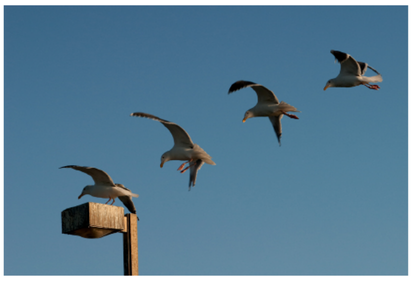
(d) Climbing



(e) Jumping



(f) Bird take off



(g) Bird landing

Fig. 2.5 Example sequences of duplicate Content

2.2 Related work

Previous works have shown why user interaction is important and how the arisen results of research in the related fields can be used alongside user interaction. In the paper of Barnes, et al. [6] they clarify some scenarios where user interaction is required to obtain the best result, as for example in their case retargeting algorithms.

Several methods currently used by photographers provide a control over the outcome where the user is trying to optimize a set of goals that are known to the user and not to the specific algorithm. In Barnes et al.'s paper [6], when it comes to Image editing, it is often the case that the user has difficulties to in advance exactly specify his/her objectives. In some cases this may even be impossible. Instead the artistic process of creating the desired image is to use trial and error methods. This is because the user aims to optimize the results with respect to personal criteria that are specific to the images under consideration. So authors in booth Barnes [6] and Bai [5] argues that human interaction is absolutely necessary in order to have a framework for Image Editing Tools. The reason is that it is through the user input the best results can be achieved with current algorithms. In this way users have a target which helps to lead the algorithm in the right direction. Furthermore they also argue that the photo editing tools to be created must be sufficiently flexible to allow users to carry out various goals. At the same time the tools must be effective enough to be able to show the results immediately so that the user can make the changes as needed.

Burst mode has been studied well in different areas of computer vision. It has been successfully exploited in the area of Image Denoising, the paper of Lui et.al [16] were they introduce a homography flow representation to image alignment that led to the mobile application Blink.

Early ideas in the direction of Spotting the Differences were found in the paper of Tasli et.al [27] were they introduce what we in this thesis call Mosaic view. Further more the paper proposes a user-friendly tool for exploring a personal photo gallery and recompositions with the help of patches from different photos.

««<

Chapter 3

Methods and Approach

This chapter provides an overview regarding the implementation of the Image Editing Tool. Here we present the algorithms and explain the main steps of the workflow that are used in the thesis.

3.1 Outline

A summary of our application pipeline consist of three main parts:

1. **Model the camera motion** by aligning the images. This is done with the help of perspective projection to compensate for small camera motions.
2. **Model the scene motion** by finding differences between the consecutive images.
3. **Inpainting** that combines the users selected layers from different images in to one final image.

The rest of this section will cover these main parts on a more comprehensive way and the following three chapters will go more in depth with each part.

3.2 Perspective Projection

3.2.1 Homogeneous coordinates

Homogeneous coordinates are used to estimate camera motion, meaning the rotation and translation between two images. Homogeneous coordinates have the quality to represent points that lay at infinite with coordinates that have finite values and can represent projective transformation. The transformation can be done with matrix multiplication.

3.2.2 Homography

A transformation between two images from the same projective plane can be calculated with the help of a homography. The homography H describes planar projective transformations, and it is invertible. In the case of this thesis we are dealing with 2D to 2D transformation. The relation of two corresponding points in homogeneous coordinates are:

$$\tilde{x}_1 = H\tilde{x}_0 \tag{3.1}$$

The homography H is represented by a 3x3 matrix with 8 degrees of freedom

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.2}$$

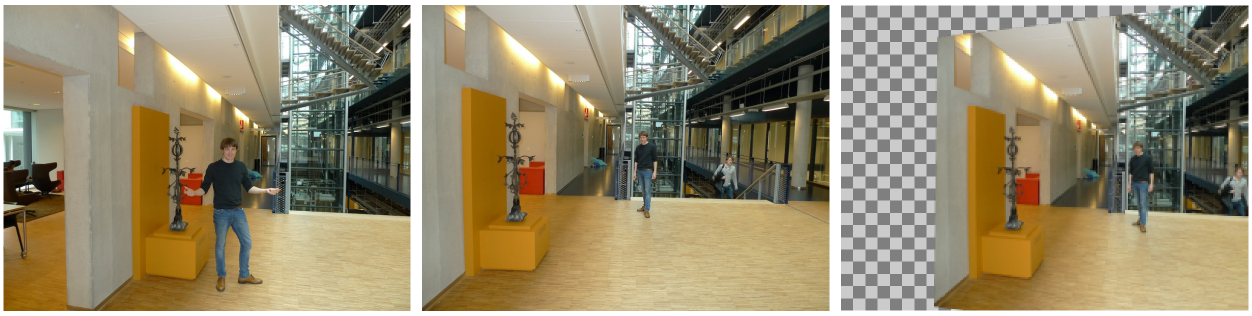
By aligning and adding several images together we create a mosaic as seen in Figure 3.1d. This is done by taking an input image(Figure 3.1b) and compute the perspective transformation relative to a base image (Figure 3.1a).

3.3 Finding the differences

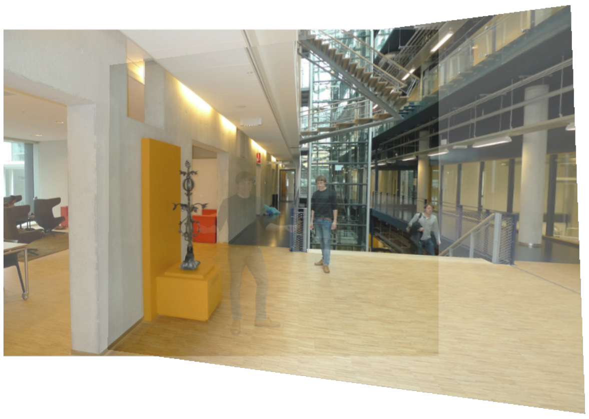
When we have aligned the images with respect to one base image we will determine the overlap by cropping the perspective rectified images to fit the overlap region of the base image. See Figure 3.1c. It is with the overlapping images that we address the algorithm of finding differences.

3.4 Inpainting

The final step of our pipeline is blending the user selected region into the reference image. We use the poisson image matting technique [20]. Blending is sometimes used along with image stitching as a composition technique to reduce artifacts and other miss alignments. Here we are using blending in a similar manner as to get rid of the visual seam that occur when inserting a region from one image into another image. The aim of the inpainting matting technique is to seamlessly blend the foreground, usually an object to the background, in our case the reference image.



(a) Input 1, the base image (b) Input 2 (c) Input 2 Rectified and cropped



(d) Mosaic

Fig. 3.1 Creation of mosaic

Chapter 4

Camera Motion

The creation of the mosaic is divided in: **1** Localizing points of interest referred to as interest points or sometimes keypoints. **2** Describe the points of interest with local Descriptors. **3** Find point correspondence by match the Descriptors, estimate a Homography with the Direct Linear Transformation (DLT), remove outliers with RANSAC and last reestimate a Homography with all inliers. **4** Warp the separate images and stitch them together. The stitching procedure handles multiple images from the same scene. To achieve a robust alignment we do isotropic normalization [13] on the found points.

4.1 Image Alignment

The image alignment is done to determine the overlap between the reference image and the transformed image. The first task of our pipeline is to find a homography H that can transform our input image to overlap the reference image again as seen in Figure 3.1c. A summary of the implementation is described in figure 4.1. These steps are implemented as described previously in this chapter.

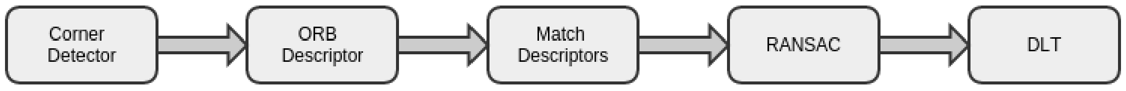


Fig. 4.1 Image alignment

4.2 Keypoint Detection

In order to extract local information of the images we first start with computing the interest points of the image. This is done with a blob detector. When designing a keypoint detector there is three main characteristics that we are keen to include according to Lepetit et.al [15]:

1. Fast to compute;
2. Should be stable in practice;
3. We should be able to attribute an orientation to the keypoint;

Our choice of keypoint detector is a Laplacian minimum eigenvalue based feature detector developed by [15].

As shown in figure 4.2 each pixel $\hat{x}$, is centered with a surrounding discretized circle and we consider the intensities of pixels around the circle. We then eliminate pixels whose value is close to or almost the same as the opposed two pixels in the circle. From the remaining pixels we approximate the Laplacian by using the value differences between pixels on the circle. Mathematically this approximation is described by first scan to reject keypoint positions:

$$\begin{array}{lll} if & |\tilde{I}(\hat{x}) - \tilde{I}(\hat{x} + dR_{\alpha})| \geq \tau & and \\ if & |\tilde{I}(\hat{x}) - \tilde{I}(\hat{x} - dR_{\alpha})| \geq \tau & then\ \hat{x}\ is\ not\ a\ keypoint \end{array} \tag{4.1}$$

Then we approximate the Laplacian of Gaussian, Here the Laplacian of Gaussian is approximated as following:

$$LoG(\hat{x}) \approx \sum_{\alpha \in [0:\pi]} \tilde{I}(\hat{x} - dR_{\alpha}) - \tilde{I}(\hat{x}) + \tilde{I}(\hat{x} + dR_{\alpha}) \tag{4.2}$$

Where $\hat{x}$ is the x and y position of a point, τ is a threshold value $\tilde{I}$ is the image smoothed with a Gaussian filter and $dR_{\alpha} = (Rcos_{\alpha}; Rsin_{\alpha})$ where R is the radius and α varying in the range $[0:\pi]$. Thereafter we threshold the minimum eigenvalue of the hessian matrix:

$$min(\lambda_1, \lambda_2) > \lambda \tag{4.3}$$

Where λ is the minimum eigenvalue threshold and λ_1 and λ_2 are the minimum eigenvalues of the Hessian. When the ratio of the eigenvalues is too high then the patch is considered as edge like and the keypoint is rejected.

The stability of the Laplacian and min eigenvalue detector is visualized in figure 4.3, illustrating that the points are retained during rotation. Our keypoint detector has a low computational complexity. It easily runs in real-time inside of a web browser on a standard PC. To keep the computation time low we have a fixed number of features points per image. Instead of sort points by non-maximal suppression radius as described by the authors of [7], we sort the points by their score and perform a maximum suppression after detecting the keypoints. In our case the score is the minimum eigenvalues of the Hessian matrix.

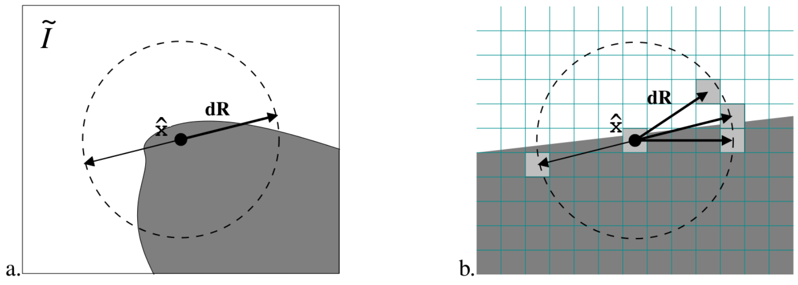


Fig. 4.2 a. Keypoint detection: if $\tilde{I}(\hat{x})$, $\tilde{I}(\hat{x}+dR)$ and $\tilde{I}(\hat{x}-dR)$ are similar then $\hat{x}$ is not a keypoint. b. We also look at the neighbor pixels to avoid responses near edges.

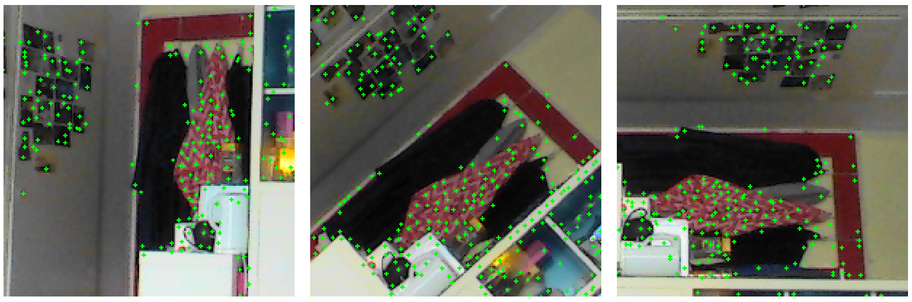


Fig. 4.3 Example of keypoint detection in a rotation scene

4.3 Oriented and Rotated BRIEF

To be able to represent a local image patch around the keypoints we compute a descriptor in the region of the selected points. For the development of the application in this thesis we have to process each photo in the web browser so we choose the Oriented and Rotated BRIEF [25] from now on referred to as ORB. Because ORB are fast to compute, it has the characteristics needed for a real-time application. And ORB are reliable, the features are rotation and illumination invariant and also partially scale invariant and robust to occlusions. ORB is a detector and a binary descriptor of local image features the main steps are:

1. Keypoint localization
2. Accurate orientation component of the keypoints
3. Keypoint descriptor by adding rotation invariance to BRIEF
4. Use the efficient hamming distance to compare features

4.3.1 Keypoint localization

In the original paper [25] the authors propose keypoint detection by FAST [24]. In opposition to the original proposal we are using the Laplacian and min eigenvalue based feature detector explained in previous section. This is due to the fact that the Laplacian keypoint detector works better with the way we are using image pyramids to detect interest points at different scale levels to imitate a scale invariant feature descriptor.

4.3.2 Accurate orientation component of the keypoints

We identify the characteristic direction of each keypoints by using image moments to find the dominant orientation of a point. First we follow the definition of moments of a patch as:

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y) \tag{4.4}$$

Then we find the centroids of the moments to determine an orientation to each patch as described by Rosin [23]:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{4.5}$$

And in the last step we conclude that the orientation of the patch is:

$$\theta = atan2(m_{01}, m_{10}) \tag{4.6}$$

As the angle measures are consistent regardless of the corner type this will also work for our laplacian detector.

4.3.3 BRIEF

Binary descriptors works by comparison of pairs of intensity values. One of the basic binary descriptor is the Local Binary Patterns (LBP). The LBP compares gray level value of one specific pixel with values of its neighbors to create one descriptor. BRIEF: Binary Robust Independent Elementary Features [9], is a compact binary descriptor. It randomly select pairs of intensity values with a fixed sampling pattern of ether 128, 256 or 512 pairs. The comparison of intensity values is a binary test τ on a patch p :

$$\tau(p;x;y) := \begin{cases} 1 & \text{if } p(x) < p(y) \\ 0 & \text{otherwise} \end{cases} \tag{4.7}$$

here $p(x)$ is the intensity of p at point x, smoothed. The BRIEF descriptor is build up by a n-dimensional bit string of these tests:

$$f_n(p) := \sum_{1 \le i \le n} 2^{i-1} \tau(p;x_i;y_i) \tag{4.8}$$

Where n is the number of sampled pairs. In the original ORB paper [25] they choose the patch size to be a 31 × 31 pixel window and show that this yields near optimal results for resistant to noise. Our descriptor is of the same size this means that one descriptor vector consists of 256 tests and are 32 bytes wide.

4.3.4 ORB

Since the BRIEF is sensitive to in-plane rotation ORB fixes this by adding rotation invariance to BRIEF by rectifying the patches with the help of the keypoints orientation . ORB does this by using the orientation θ to rectify each patch with its corresponding rotation. In this way ORB minimize the correlation under various orientation changes.

The similarity between descriptors is measured with the Hamming distance between the corresponding binary vectors. This is very efficient because Hamming distance can be computed with a bitwise XOR operation followed by a bit count. This yields fast comparison computation compared to (L2) Squared Eucledian distance or L1-distance between the feature vectors

4.4 Matching

For matcher we use a brute force matcher and the Hamming distance to find corresponding feature points in two images. That is we compare all features in one image against all features in the other image.

So for all the feature descriptors in image one, we loop through all feature descriptors from image two and compute a distance between them to find the first nearest neighbor (1NN) and the second nearest neighbor (2NN). We use the Hamming distance as a measure between the feature vectors. If they pass the ratio test 1NN/2NN < than a threshold, then we consider these two points as a corresponding pair. This criterion is also termed as Lowe's criterion [17].

4.5 The DLT algorithm

To estimate a homography H between point correspondence we use the Direct Linear Transformation (DLT) outlined by Hartley and Zisserman [14]. The DLT is derived from the vector crossproduct of equation 3.1: $\tilde{x}_1 \times H\tilde{x}_0 = 0$. The DLT is composed of a matrix A which has the relation to the homograph as: $Ah = 0$ where h is the homography in vector format and A is a matrix composed of linear equations:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1x'_1 & -y_1x'_1 & -x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1y'_1 & -y_1y'_1 & -y'_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n & y_n & 1 & 0 & 0 & 0 & -x_nx'_n & -y_nx'_n & -x'_n \\ 0 & 0 & 0 & x_n & y_n & 1 & -x_ny'_n & -y_1y'_n & -y'_n \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ \vdots \\ \vdots \end{bmatrix} \tag{4.9}$$

A, The left hand side matrix is a $2N \times 9$ matrix. Where x_n and y_n is the coordinates of a sampled point-correspondence and N is the number of sampled points. The homography h is a vector. The right hand side is a $2N$ zero vector. To estimate a homography of a pair of images we need to have four or more point correspondences. This set of linear equations can be solved with standard methods and in this thesis we are using the convenient and stable

Singular Value Decomposition (SVD).

$$A = USV^T \tag{4.10}$$

The solution to h is the singular vector corresponding to the smallest singular value. When the singular values in S are sorted in descending order then this is the last column of V.

4.6 Isotropic point Normalization

In general normalization of point coordinates yields better numerical stability [13], and in this thesis we do get a more robust estimate of H when all data points are normalized before estimating H with the DLT.

The normalization is performed in two steps, first the points are translated so that their centroid is at the origin, then the points are scaled isotropically so that the average distance from the origin is equal to $\sqrt{2}$. This normalization has its roots in [13] where the theory are more fully explained.

Still normalization is performed separately for the points in the first image x, and the points in the second image x'. By centering the points' centroid at the origin and scale each point so that their average distance from origin is equal to $\sqrt{2}$. To apply the normalization we construct a normalization matrix T by the following steps:

$$m_x = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.11}$$

$$m_y = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{4.12}$$

$$d = \frac{1}{n}\sum_{i=1}^{n} \sqrt{(x_i - m_x)^2 + (y_i - m_y)^2} \tag{4.13}$$

$$T = \begin{bmatrix} \sqrt{2/d} & 0 & -m_x\sqrt{2/d} \\ 0 & \sqrt{2/d} & -m_y\sqrt{2/d} \\ 0 & 0 & 1 \end{bmatrix} \tag{4.14}$$

Where x_i and y_i are the coordinates from either the point set x or x'. So here T is the similarity transformation which gives us normalized points from image one as:

$$\hat{x} = Tx \tag{4.15}$$

And normalized points from image two as:

$$\hat{x}' = T'x' \tag{4.16}$$

4.6.1 Finding a Homography

To determine a homography we need a set of point correspondence $n \geq 4$ from two images to supply the DLT algorithm from section 4.5 with. But now all data points are normalized so what we obtain from the normalized points is $\hat{H}$, then H is:

$$H = T'^{-1}\hat{H}T \tag{4.17}$$

This is the denormalization step of the normalized DLT.

4.7 RANSAC

To reject incorrectly matched features and to get a robust estimate of the homography we apply RANSAC. RANdom SAmple Consensus [12] is used to iteratively improve the estimated model parameters by removing outliers from the data. For our problem the mathematical model to fit is the Homography mapping the two input images and the outliers/inliers are the matched images points. The RANSAC algorithm is outlined by:

1. Randomly select a subset of the matched image points and estimates a homography from this subset. Here the estimation is done with the normalized DLT.

2. The points from the entire dataset thats fits the model with an error less then a threshold are inliers and they are placed in a consensus set. As an error measurement function we use the Symetric transfer error:

$$\sum_i d(x_i, H^{-1}x')^2 + d(x_i', Hx_i)^2 \tag{4.18}$$

3. When the maximum number of iterations has been reached then the homography is recomputed, but this time from the points in the consensus set and now we also need to denormalize the homography.

4.8 Perspective Warp

To align one image with another image we do make a transformation as described in 3.2. That is we apply the 3x3 homograpy matrix H to the image we want to transform. To do this we use Bilinear interpolation that considers the closest 2x2 neighborhood pixels. Considering Figure 4.4 where P is our new "warped" pixel point at (x,y) and $Q_{11}, Q_{12}, Q_{21}, Q_{22}$ is the neighborhood pixels. The interpolant can be written as:

$$P \approx \frac{(x_2 - x)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} Q_{11} + \frac{(x - x_1)(y_2 - y)}{(x_2 - x_1)(y_2 - y_1)} Q_{21} + \\ \frac{(x_2 - x)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} Q_{12} \frac{(x - x_1)(y - y_1)}{(x_2 - x_1)(y_2 - y_1)} Q_{22}$$

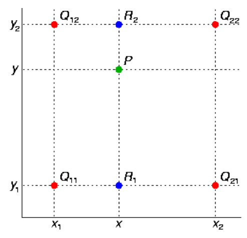


Fig. 4.4 Example of bilinear interpolation

To avoid holes and other artifacts we use inverse warping Algorithm. That is the mapping of pixels is done in reverse order, from destination to the source. For each pixel of the destination image the functions compute coordinates of the corresponding pixel in the source image.

Chapter 5

Scene Motion

After aligning the individual images and determine their overlap with respect to a single reference image we will now address the problem of finding the differences. We are creating a difference image in which the differences are converted into layers that users can interact with. These layers are for example highlighted with a color to visualize a difference between the two images.

One naive approach of finding differences would be to subtract the values of the two images at pixel level. Although this approach would work in an ideal environment in which two images are perfectly aligned, this is almost never the case in real-world scenarios where even the small exposure divergences will be registered as differences. We would however have as result, a difference-image that is cluttered with a huge number of small differences, as in figure 5.1.

The intention behind our approach is to compare regions inside of the two images rather than looking at individual single pixel differences. These regions can be seen as blobs. For each region, we create a separate layer to make them intractable and in that way users can click and choose which layered blob region they would like to include in the final result.

This means that each separate "blob"-layer is assigned an unique id so that our underlying architecture can keep track of the users interaction.

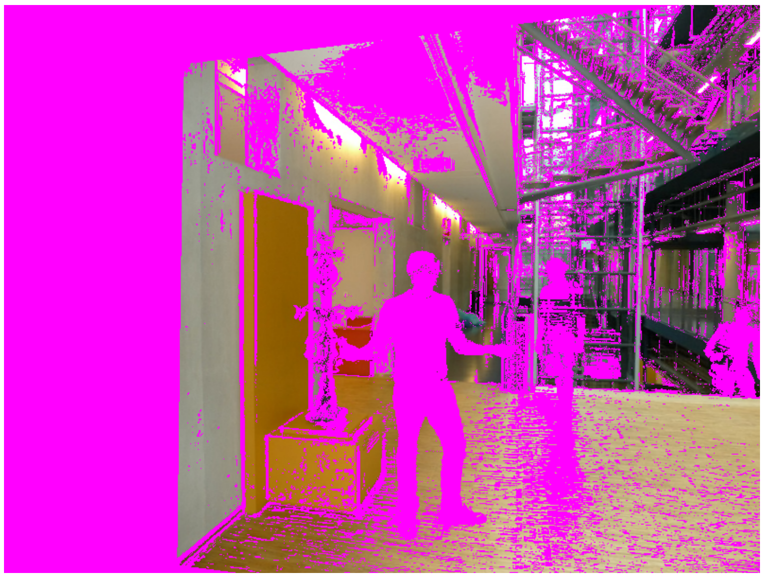


Fig. 5.1 The Pixel Differences between input 1 (fig 3.1a) and input 2 (fig 3.1c). In the Difference-image we can se that static objects are registered as differences, like the sculpture on the left side and in the roof. this is not something we want to achieve.

5.1 Level Set Method

Our goal is to segment the difference region between the multiple images, as in figure 5.2. Image segmentation is the process of isolating objects in the image. Most common is to separate the objects from the background by splitting partitions of the image into disjoint regions. In our case we wish to segment out regions were there is differences and we want to keep the regions homogeneous with respect to the content inside so that a blob-layer cover whole objects.

Level Set Methods have for a time been used for image segmentation [19] partly with energy minimization based methods as the Mumford–Shah Model [18], and with Geodesic Active Contours, Caselles et.al [10]. Resent advances in this field have shown state of the art results by using a level set tree structures, Dubrovina et.al[11].

5.1.1 Blob

The intractable difference region that we termed blob can be described as similar to a curve that is defined on our pixel grid. For convenience we call the grid Ω and the blobs C. We represent our blob by all the points inside of it in an implicit way with the function ϕ.

We can implicitly represent boundaries directly as the zero level of the function ϕ. One fundamental of the *Level Set Method* is that it represent the zero level of the function ϕ. This function is sometimes called the embedding function in the literature because it embeds the boarder as the zero level set of a signed distance function. Our blob C is the zero level set of the embedding function:

$$C = \{\mathbf{x} \in \Omega \mid \phi(\mathbf{x}) = 0\} \tag{5.1}$$

We represent a blob by all the points that holds a condition. In other words we separate the pixel domain of Ω into three different subdomains namely Ω^- the inside portion of the blob, Ω^+ the outside portion of the blob and $\partial\Omega$ the border between the inside and the outside. The border is also the zero level set defined as:

$$C = \{\mathbf{x} \in \Omega \mid \phi = 0\} \Rightarrow \quad \partial\Omega \tag{5.2}$$

The external region is defined as :

$$C = \{\mathbf{x} \in \Omega \mid \phi > 0\} \Rightarrow \quad \Omega^+ \tag{5.3}$$

The internal region is defined as :

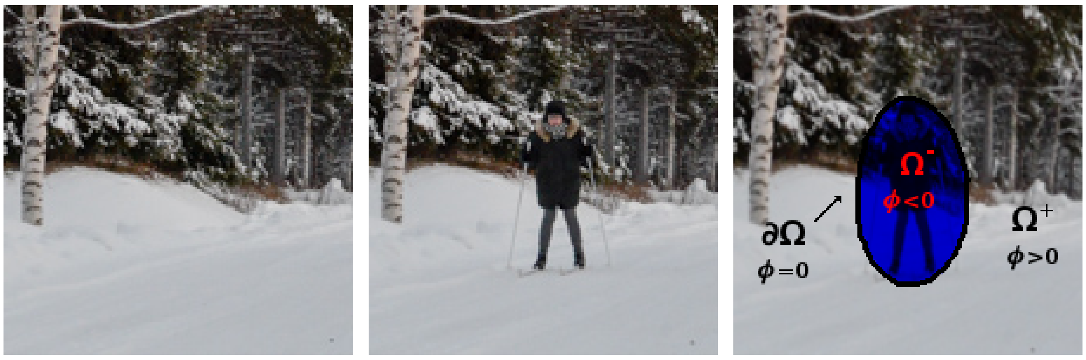
$$C = \{\mathbf{x} \in \Omega \mid \phi < 0\} \Rightarrow \quad \Omega^- \tag{5.4}$$

5.1.2 Signed Distance function

As seen in equation 5.1 and 5.2 the embedding function is exactly zero at the boundary this leads to the signed distance function that is the distance from any point $\mathbf{x}$ in Ω to the blob C:

$$\phi(\mathbf{x}) = \pm dist(\mathbf{x}, C) \tag{5.5}$$

This gives the distance to a blob and in our case we defined that the sign is negative inside a blob and positive outside a blob.



(a) Input 1 (b) Input 2 (c) Segment of output blob

Fig. 5.2 The differences between a patch from two images.

5.1.3 Level Set Equation

Every point on the boarder of the blob in figure 5.2c has zero distance to the blob, every point inside the blob has negative distance and every point outside the blob has positive distance. Blobs can be evolve by thresholding the level. What change the evolution of a blob is the motion in the normal direction. That is the outer normal which is the vector:

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|} \tag{5.6}$$

The speed of the movement is given by F, here the variable F is the speed in the outward normal direction. The motion comes from the difference of two images smoothed with a Gaussian kernel. *The Level Set Equation* is defined as:

$$\phi_t + F|\nabla\phi| = 0 \tag{5.7}$$

and can be reformulated as $\frac{\partial\phi}{\partial t} = F|\nabla\phi|$. This is a Partial Differential Equation (PDE) with two variables; x in space and the threshold t. We are checking at any threshold t where we have a sign change, and by mark that will give us a blob. Blobs are evolved by iterating this function. This way of representing gives the advantage that blobs can undergo splitting and merging through changes of the threshold as seen in figure 5.3, during all kinds of splitting or merging of blobs we can read out the value of ϕ at any threshold t.

The Level Set Equation requires that at a blob location, the embedding function ϕ must be zero for all thresholds [19] [26]. When we evaluate the embedding function at threshold t

then a blob location should be zero:

$$\phi(C(t),t) = 0 \quad \forall t \tag{5.8}$$

Here we have a two dependence on t. One, we consider the embedding function at different thresholds. The other dependency of t is the spatial location where we locate a blob at the threshold t.

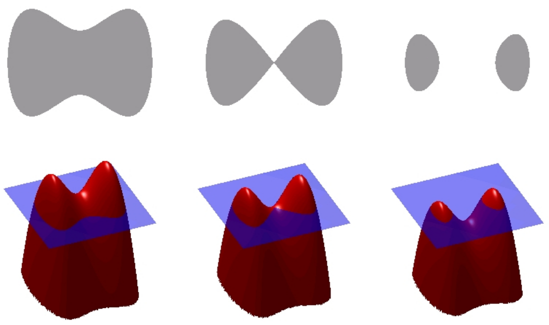


Fig. 5.3 The threshold value is changed by scroll the mouse wheel. This causes splinting and merging of blobs and these are handled with the level set method

5.2 Fast Marching Method

The level set method usually describes a problem where we track the zero level set $\phi = 0$ and we do not care about ϕ away from the boarder [26].

When the speed function $F > 0$, then we can formulate the evolution with an arrival function T, where $T(\mathbf{x})$ is giving the time for the boarder to reach a point (x,y) on the grid away from its initial location. The Fast Marching Method described as a Boundary Value Formulation it assumes that the boundary front evolves in only one direction that is along the normal direction, seen in figure 5.4. The motion of the boarder is described by the *Eikonal equation*:

$$|\nabla T|F = 1, \quad T = 0 \text{ on } \Gamma \tag{5.9}$$

Here Γ is a blobs initial location where the methods start and work outwards. From Γ, the pixels that we start from are said to have a frozen condition and we compute distances at their neighbors. Thorough implementation details are explained in [4] but in essential we are

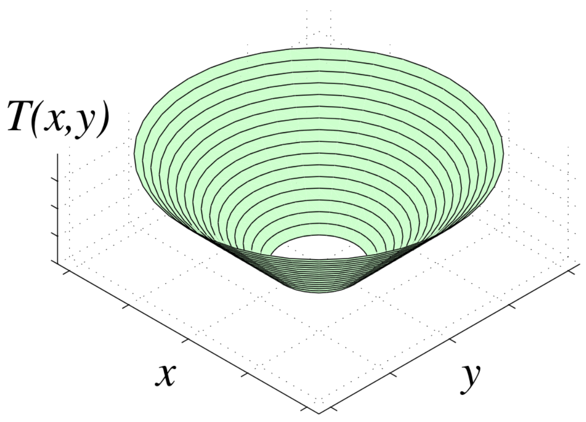


Fig. 5.4 Boundary value formulation [1]

solving the Eikonal equation by:

$$|\nabla T| = \frac{1}{F} \tag{5.10}$$

Where again T is the arrival time of the front and F is the speed of the evolution. We are following Sethian's approach to compute distances on our pixel grid. The initial values of the boundary are known, and the nodes neighboring these can be updated by the square length of the gradient:

$$|\nabla T|^2 = \begin{cases} \max(V_A - V_B, V_A - V_C)^2 \quad + \\ \max(V_A - V_D, V_A - V_E)^2 \end{cases} \tag{5.11}$$

Where V_A is the unknown distance value and the rest is the neighboring values as in figure 5.5. Here we are only using known values to compute the distance value and if some of the neighbors are not known then these terms are dropped out of the equation.

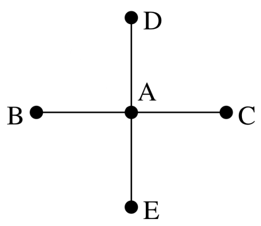


Fig. 5.5 Four-Connectivity

5.3 Geodesic Distance

For data lying on a nonlinear manifold, the true distance between two points is the geodesic distance on the manifold, i.e. the distance along the surface of the manifold, rather than the straight-line Euclidean distance. As show in the following figure 5.6. In the next section we will show how we can approximate the geodesic distance with the help of the fast marching algorithm

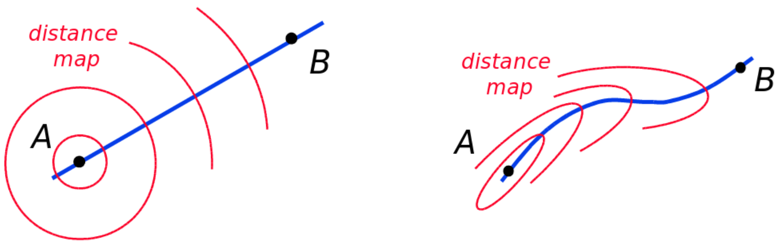


Fig. 5.6 Left; Euclidean distance, Right; Geodesic Distance on a manifold

5.4 Image Difference

5.4.1 Gaussian kernel

Our method consider differences in the three RGB color channels individually after which these values are summed, and scaled to values between 0 and 255. The core of our approach is to smooth in two steps. First the individual channels are smoothed with a pre-sigma and after subtracting we do an additional smoothing this time with a larger post-sigma. Our kernel is an approximation of the Gaussian function:

$$G(u,v) = \frac{1}{2\pi\sigma^2} e^{\frac{u^2+v^2}{\sigma^2}} \tag{5.12}$$

5.4.2 Difference map with Gaussian kernel

We will now explain how to build our difference map ϕ. First we filter out the three RGB channels of the images then we smooth each individual channel with a pre-defined sigma to remove fine grained noise. Next step is to differentiate the reference image from the transformed image, this is also done for each individual channel. The differentiation is then normalized by the three channels. The last step is to prevent the differences to be over cluttered with small irrelevant parts, so, for that reason we do a second smoothing this time with a larger post sigma. This procedure is outlined in algorithm 1 and this is done for all input images with respect to the reference image.

```
input  : RGBA channels of $img1$ of size $w \times h$
input  : RGBA channels of $img2$ of size $w \times h$
output : The Differences from the two images

$a$ and $diff$ are matrices of size $w \times h$ ;
for each RGB channel do
    $img1GaussBlur = gaussianBlur(img1[\textbf{channel}], kernelSizePre, sigmaPre)$;
    $img2GaussBlur = gaussianBlur(img2[\textbf{channel}], kernelSizePre, sigmaPre)$;
    $a+ = abs(img1GaussBlur - img2GaussBlur)$;
end
$diff = \frac{a}{3}$;
// do not consider non Overlap, by looking at the alpha channel
$i = w \times h$;
while $--i \geq 0$ do
    if img2[3] == 0 then
        diff[i] = 0;
    end
end
diff = gaussianBlur(diff, kernelSizePost, sigmaPost);
```

Algorithm 1: Find the Differences in two images

The result obtain by computing the difference of image 3.1a and image 3.1c with our method is visualized in figure 5.7 top right, this is our difference image. The difference image is a real valued height map represented in such way where high values mean that we have a large difference between the two images and small values stands for small to non difference at all at a specific point. It is from this algorithm that we create our ϕ function as described in section 5.1.3.

5.4.3 Label Layers

After obtaining the difference image we want to be able to interact with the differences by deciding what regions are desired differences and which are to be left unconsidered. This is done on two steps. First we are propagating the level set to a desired level by evaluate the height values at the boundary of different levels. The second step is to label each individual separated layer, as for example in figure 5.7 top left we have three separated layers at a specific threshold level.

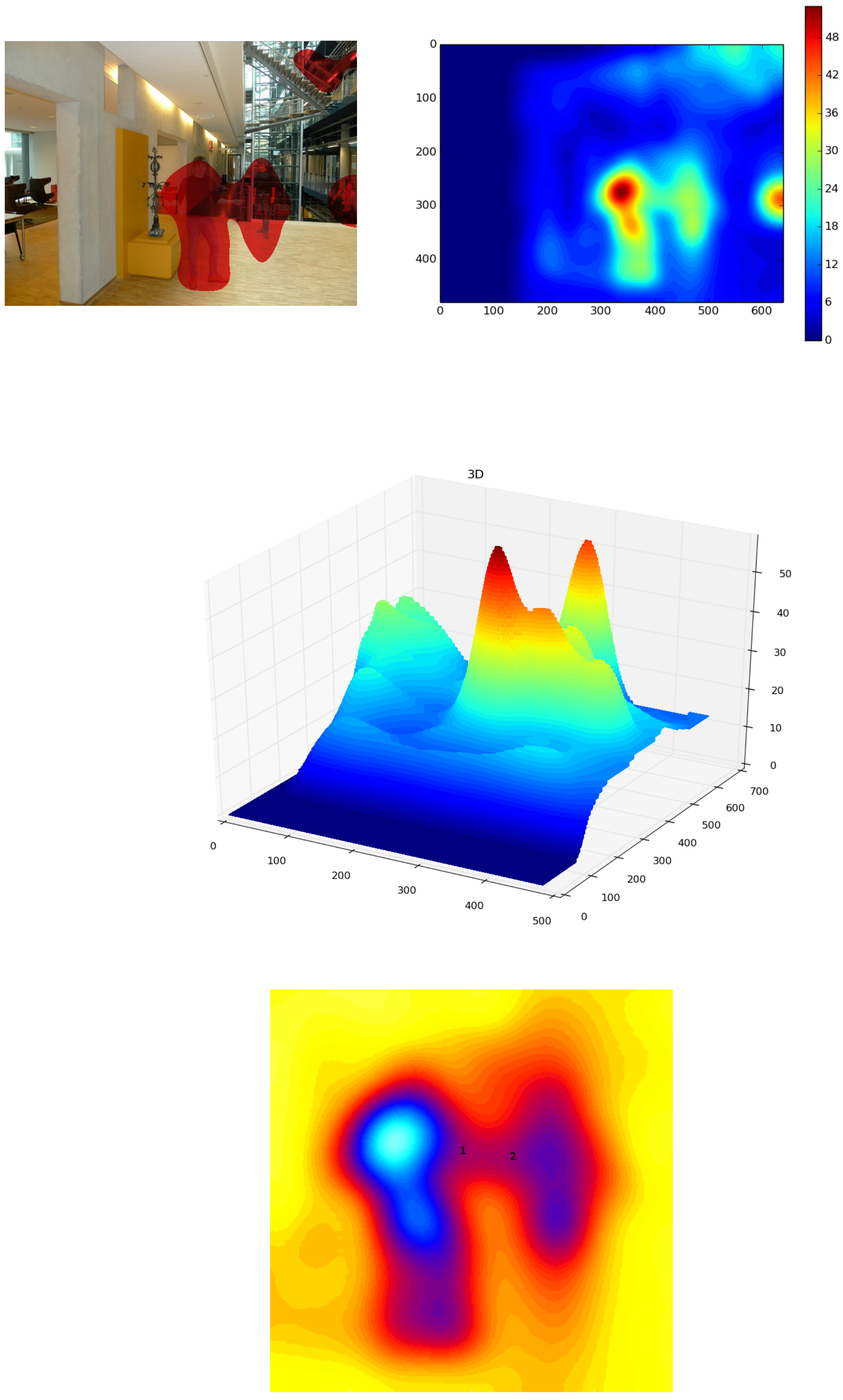


Fig. 5.7 (Top Left) The labeled blobs found in the (Top Right) Contours of the Difference image ϕ. (Middle) The difference image ϕ plotted in 3D, (Bottom) The inverse of the Difference image labeled at a higher level than the top left

In other words we are segmenting out regions lower than a threshold, this is visualized in Figure 5.7 where we go from the difference image figure 5.7 right to labeled layers figure 5.7 left.

This is done with our blob extraction algorithm which start by thresholding the difference image. So if a point *P*'s difference value is below a specific threshold *th* it will be set to zero in a layer map otherwise we start labeling *P* as a new label. The labeling is done by check if the pixel *P* has 8-connectivity with another pixel label, however if any of *P*'s 8-connecting pixel neighbors (see Figure 5.8) are already labeled then *P* will be part of that label. Setting the threshold to lower values will yield bigger blobs. This is outlined in Algorithm 2. As seen in the bottom of figure 5.7 we can label the same persons appearing two times differently, here with label 1 and 2.

```
input  : The Differences from the two images
output : Labeled blobs
for each pixel do
    if pixel > threshold then
        if neighbor(s) are labelled then
            label this pixel the same as it's neighbor;
        else
            create a new label for this pixel;
        end
    end
end
```

Algorithm 2: Outline of blob labeling

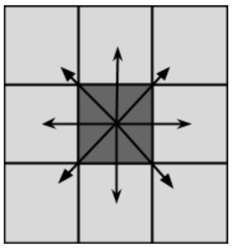


Fig. 5.8 8-Connectivity

5.5 Local adjustments

When the user selects a "blob"layer we want to be able to locally adjust that specific region without changing all other blobs. Now we drop the global signed distance function constraint and approach the local region slightly different.

This is best described with figure 5.9 where we compute the arrival of a propagating front. When an object is placed in the way of the propagating front (figure 5.9 middle) then the geodesic path will be longer in the points of the object (figure 5.9 right).

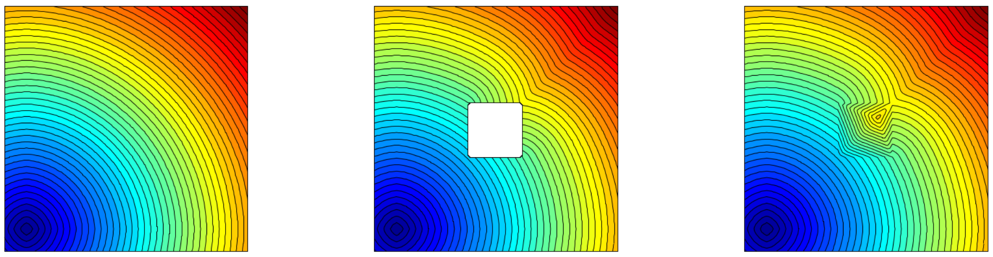


Fig. 5.9 First arrivals and shortest geodesic paths [1]

For all selected layers we compute a relative distance map (figure 5.9 right) by solving the Eikonal equation 5.10. This gives the users the ability to adjust each individual layer separately with a relative threshold value.

5.6 Optimization

As stated in section 5.2 the values of the boundary is our initial curve Γ and these values are known values, the neighboring pixels are updated and inserted into a priority que. The pixel with the smallest distance are then removed and marked as known. The computation of a heap based priority queue normally requires O(n log n) operations for n nodes. There is however a more effective solution presented by Yatziv et.al [28] it the paper *O(N) implementation of the fast marching algorithm* they termed the solution Untidy Priority Queue. This queue make use of the fact that any key insertion is always greater than or equal to the key item we just popped off. By maintaining a circular array of lists, items with the lowest key is always popped first.

Chapter 6

Inpainting

6.1 Blending

The last step of our pipeline is to combine the selected blob elements with the reference image and to manipulate these multiple sources into one convincing final image. This is obtained by blending the user selected blob into the reference image. Blending is sometimes used together with image stitching as a composition technique for correcting the difference of exposure in multiple shoots, and it is also used to reduce other miss alignments and artifacts that can occur when stitching images (see [20] and [8]).

We are using blending in mainly two ways. One way is to add an object from a source image into the reference image as seen in figures 6.1 a - c. The other way is to remove an existing object in the reference image with a background patch from a source image as seen in figures 6.1 d - e.

6.1.1 Blending and matting

To cut a foreground object out from one scene and put it in top of a different background is a common feature in a large number of image editing applications. The process of extracting the object from the image is often referred to as matting and the process of inserting the object into another image is sometimes referred to as compositing.

One form of blending is the closed form matting [21]. The closed form matting equation: $I = \alpha F + (1-\alpha)B$ is used to form a new image from a background (BG) and a foreground(FG) layer where these two layers can come from different sources. In the case of $\alpha(x,y) = \begin{cases} 1, & \text{in } FG \\ 0, & \text{in } BG \end{cases}$. Will give the resulting image $I(x,y) = \begin{cases} F(x,y), & \text{where } \alpha = 1 \\ B(x,y), & \text{where } \alpha = 0 \end{cases}$.

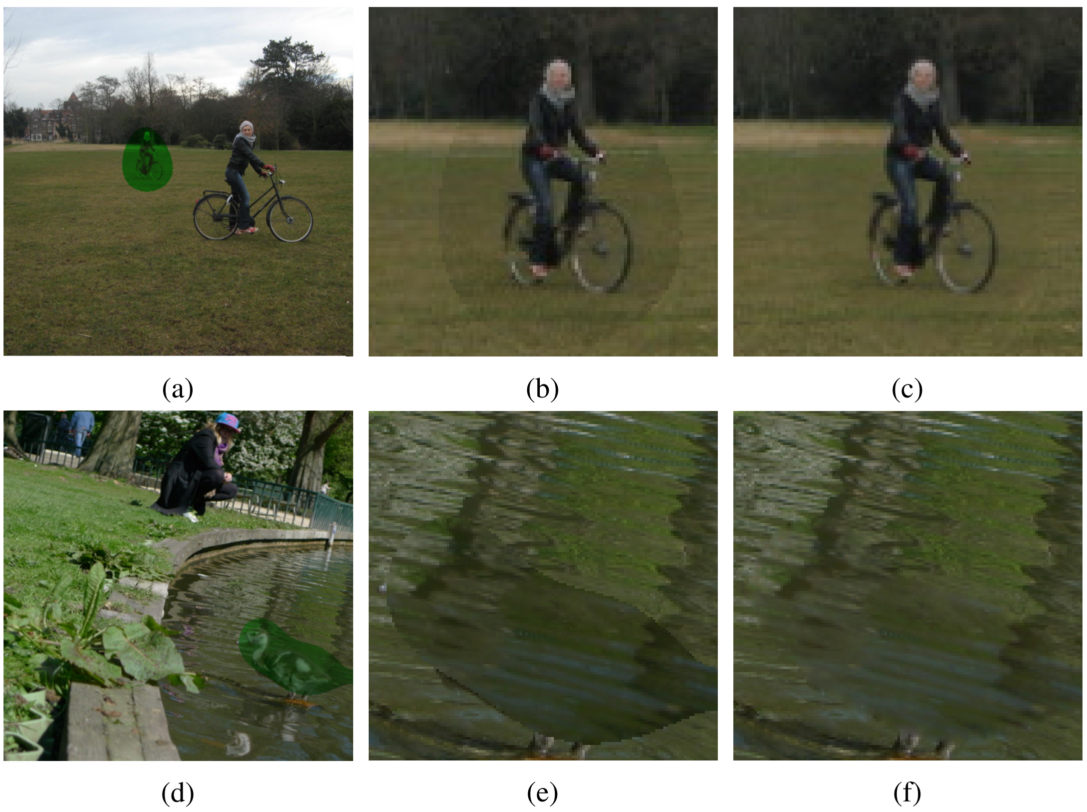


Fig. 6.1 In the left column we see the masked region marked as a green transparent layer. In the middle column we see the masked out region from the source image pasted into the reference image. In the right column we see the masked out region from the source image blended into the reference image. The mask and the result were created by our application.

This process is similar as cutting out a region from a foreground layer and putting it on the background image. In the resulting image we would see the edges and/ or a seam of the different layers. So instead of using the matting equation we will use a different terminology where we have three layers: A source image, A target image and a mask Figure 6.2.

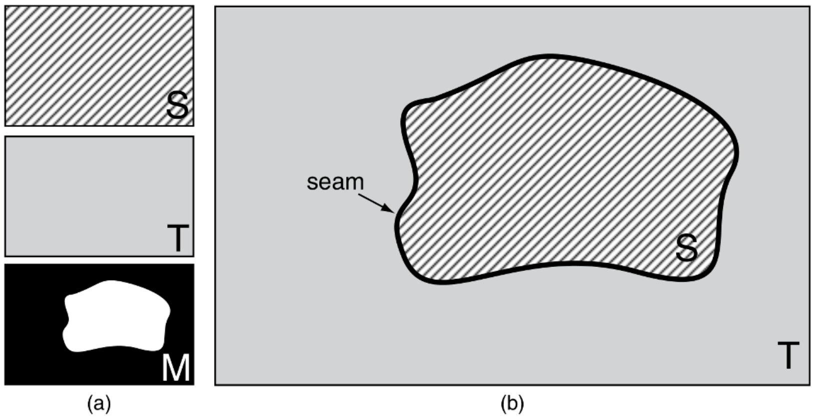


Fig. 6.2 Left: Source, Target and Mask. Right Visual Seam. Image credit [22]

6.1.2 Poisson blending

We use the gradient based Poisson image matting technique to seamlessly blend the selected region into the reference image. In the paper of Peréz et al ([20]) they present the Poisson Image editing. To describe the process Peréz et al use instead of a binary mask M and a source image S, they describe the source region as defined over a closed region Ω, the boundary of the region is denoted $\partial\Omega$ and $\mathbf{v}$ is a vector field defined over the source region Ω, see figure 6.3.

Now the objective is to make the edges inside Ω to look like the edges of the source S and to make the colors on the boundary $\partial\Omega$ to match the targets colors. This is formulated as an optimization problem in the continuous domain as:

$$\min_{f} \iint_{\Omega} |\nabla f - \mathbf{v}|^2 \text{ with} f|_{\partial\Omega = f^*}|_{\partial\Omega} \tag{6.1}$$

Here f is the intensity value at pixel p which also can be described as $I(x,y)$. $\mathbf{v}$ is the gradient field of the source $\nabla S(x,y)$. And f^* is the known destination function. Ω is the inside of our mask region where we are optimizing the image color of our composite. In other words we are minimizing the difference of the gradient of the image and the gradient of the source by making it as small as possible. So we can rewrite the equation as:

$$\min_{I(x,y)} \iint_{\Omega} |\nabla I(x,y) - \nabla S(x,y)|^2 dxdy \tag{6.2}$$

and from the equation 6.1 we have a constraint such that on the boundary the composite colors have to match the targets colors:

$$\text{s.t.} \quad I(x,y) = T(x,y) \quad \text{on} \quad \partial\Omega \tag{6.3}$$

This is to get rid of the visual seam while we are making the edges look as good as possible. In continuous domain this is a variational calculus problem that have the solution [22]:

$$\begin{aligned} I(x,y) &= \nabla^2 s(x,y) \quad \text{in} \quad \Omega \\ \text{s.t.} \quad I(x,y) &= T(x,y) \quad \text{on} \quad \partial\Omega \end{aligned} \tag{6.4}$$

The result can be seen by comparing figure 6.1b with figure 6.1c and figure 6.1e with figure 6.1e. Where we see that the edges of the region has been smeared out into the background and the colors inside the mask now match up.

6.2 Solve the discrete equation

We solve the discretize equation with the help of Gauss–Seidel method [2] as explained in the paper [20]. Gauss–Seidel Method is a sequential solver of linear system of equations it uses previously computed results. Thanks to that it to only have to allocate one instance for the size of the number of unknowns which in this case is dependent of pixels inside that mask area. In essential, rearranged from the paper [20] what we are solving is f_p visualized

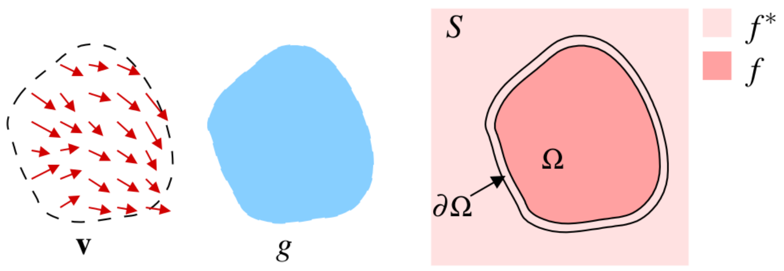


Fig. 6.3 poisson. Image credit [20]

in figure 6.3.

$$f_p = \frac{\sum\limits_{q \in N_p \cap \Omega} f_q + \sum\limits_{q \in N_p \cap \partial\Omega} f_q^* + \sum\limits_{q \in N_p} v_{pq}}{|N_p|} \tag{6.5}$$

Here f_p is the value of f at p, N_p is the set of p's 4-connecting neighbors, $q \in N_p$, v_{pq} is the projection of $\mathbf{v}(\frac{p+q}{2})$ on the oriented edge $[p,q]$. This is a quadratic optimization problem that we simultaneous solve in three steps. We are solving a system of linear equations for each of the RGB color of the pixel. The system is solved with the Gauss–Seidel method which iteratively solves a linear system of equations. The steps are divided in.

1. $\sum\limits_{q \in N_p \cap \Omega} f_q$
2. $\sum\limits_{q \in N_p \cap \partial\Omega} f_q^*$
3. $\sum\limits_{q \in N_p} v_{pq}$

The algorithm is found in Algorithm 3 where the *base_pixels* corresponds to f^* in figure 6.3, the *src_pixels* corresponds to the cut out region and the *result_pixels* corresponds to our blended image.

```
do
    for all pixels do
        if pixel p is inside the mask then
            // solve the system of linear equations
            for each RGB channel do
                var $sum\_f_q = 0$;  // step (1)
                var $sum\_\partial\Omega = 0$; // step (2)
                var $sum\_v_{pq} = 0$; // step (3)
                for all q do
                    if q is inside the maks then
                        $sum\_f_q += result\_pixel[q]$; // step (1)
                    else
                        $sum\_\partial\Omega += base\_pixel[q]$; // step (2)
                    end
                    $sum\_v_{pq} += src\_pixel[q]$; // step (3)
                end
                // Store the Estimeted RGB value of p
            end
        end
    end
    check if converged
while not converged;
```

Algorithm 3: Solve Poisson with Gauss–Seidel method

Chapter 7

The User Interface

Our whole proposal is implemented in javascript although this is not the fastest or the most supported (in terms of "of the shelf" library's) language, the reasons for using this technology are that our experiments is designed to be very dependent on human interaction, as motivated in Chapter 2. So by have the experiment live inside a webpage we can allow easy access for users to test and interact with it.

7.1 Overview

Our proposal is an interactive web tool that make it easy to get the single best picture from multiple exposures taken within the same scene. With the tool it is possible to recompose a photo from a burst of images after the image is photographed. This is done by remove, add, swap or combine objects in each individual photograph into one reference photo, as seen in figure 7.1. Supported web browsers are Chrome, Firefox and Safari although Chrome is highly recommended. To Explain the user interface we will take a step by step walk through in the next sections.

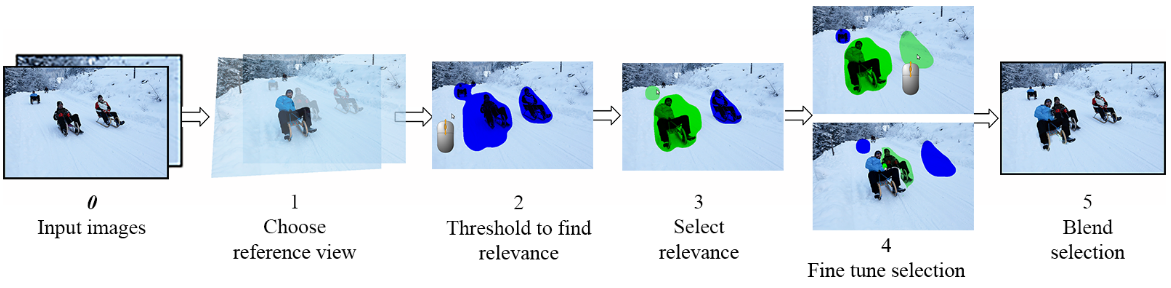


Fig. 7.1 From input images to output result

Step 1 Select reference view

The user selects their choice of reference image by scrolling the mouse wheel inside the "mosaic view" in this way they can traverse multiple images. To select one base reference image is simply done by click in the highlighted image.

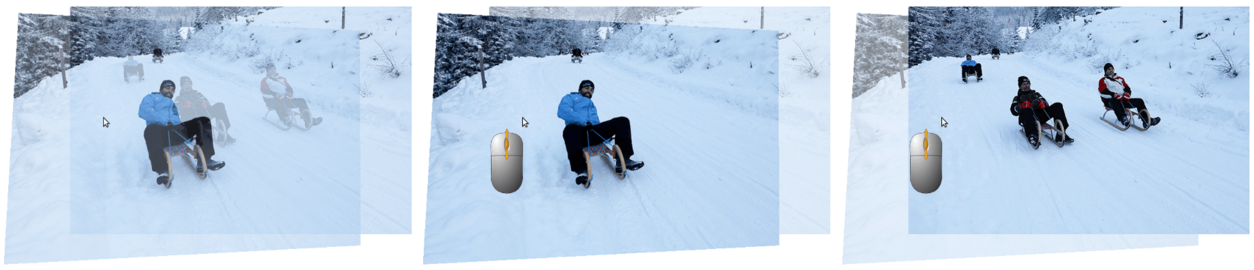

(a) Default view presented initial to the users. Mosaic view (b) Mouse wheel scroll in one direction (c) Mouse wheel scroll in the other direction

Fig. 7.2 Select reference view in the Mosaic view

Step 2 Global thresholding

The difference between the reference image and all other images are computed and highlighted, these highlighted areas are called layers. This allows the users to select content to add or remove. Adjusting the Global threshold is done by scroll the mouse wheel in anywhere except in the green layer, this is done to split or merge the layers with each other. The green is explained in next step.

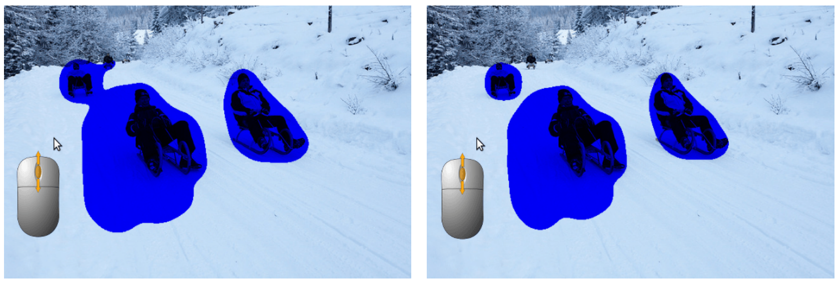

(a) Mouse wheel scroll in one direction to increase the global threshold (b) Mouse wheel scroll in one direction to decrease the global threshold

Fig. 7.3 Mouse wheel scroll outside any green layer

Step 3 Select relevance

When some content that the user want to change is present they can click in that layer to make changes in the image. The user can toggle the content of that specific layer by click in it. The green layer is just an indication that the layer is selected. The content inside the green layer are coming from an image other than the reference image. The content inside all other colored layers then the green mean that that content are coming from the reference image. Different colors means that in that area there is differences with respect to the reference image. The reference image in this example only have one other image then the reference image so there is only one blue color indicating differences between these two. To choose a green layer means that when the user clicks the blend button then the application will automatically blend the content we see inside the green layer into the reference image.

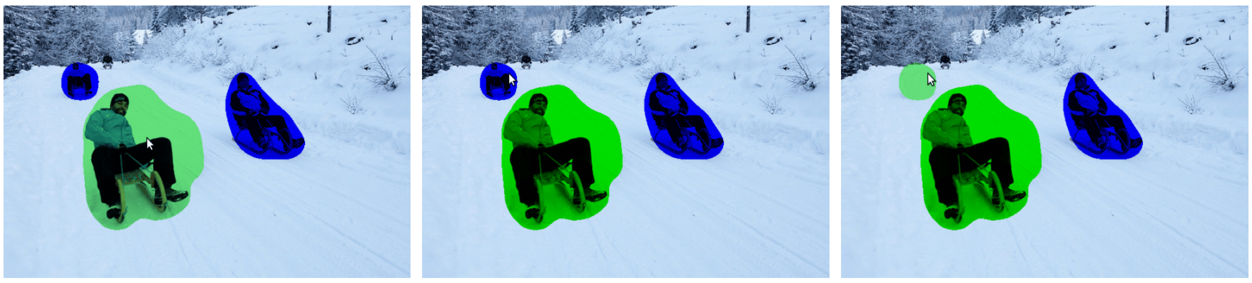

(a) (b) (c)

Fig. 7.4 Toggle by click inside layers to Select relevance

Step 4a Fine tune Relative threshold

Adjust the Relative threshold by scroll the mouse wheel inside any green layer, the layer you are currently hovering are highlighted by a light-green color to indicate which region you are adjusting.

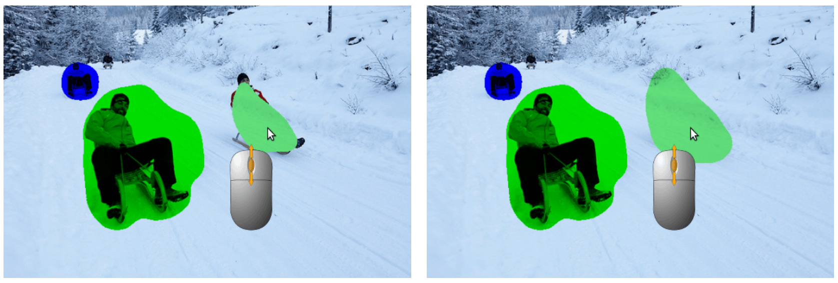

(a) Mouse wheel scroll in one direction to increase the global threshold (b) Mouse wheel scroll in one direction to decrease the global threshold

Fig. 7.5 Mouse wheel scroll inside any green layer

Step 4b Fine tune Paint/Erase

If the user are not satisfied with what the layer regions are covering they can manually paint or erase the layers by selecting these options in the check-box. Simply paint inside a green region to add layer. Erase parts of the layer by first select the eraser to the left.

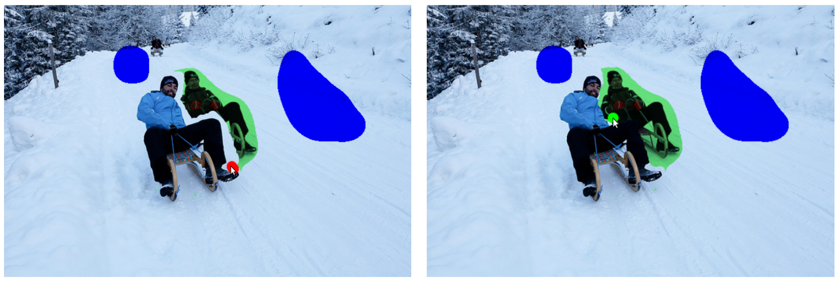

(a) Erase in a layer (b) Paint in a layer

Fig. 7.6 Paint/Erase tool

Step 5 Blend Selection

When the user is satisfied with the ruff result presented they just click the blend button to let the application blend in all selected layers in the reference image.

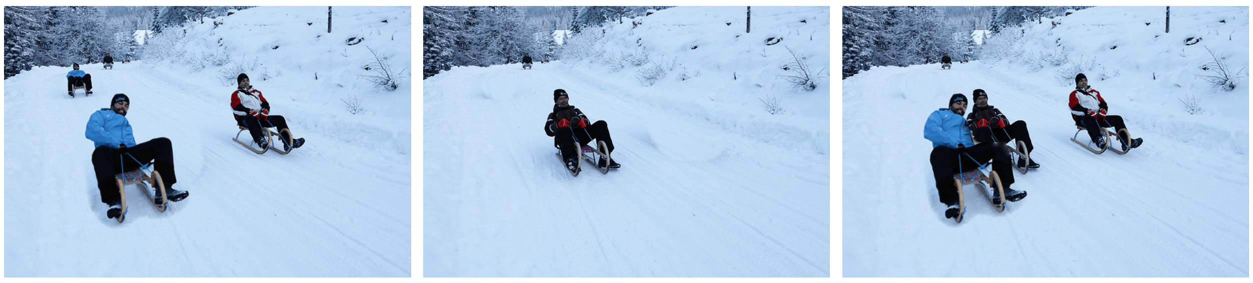

(a) (b) (c)

Fig. 7.7 Different results obtained of the input images

For a comprehensive view over Step 2 - 4a please see the flowchart in Appendix A.

Chapter 8

Evaluation and Results

In this chapter we describe the data collection approach of our research study. The methods we have chosen and why we choose these methods are explained. We also evolve why we based our study on a questionnaire and interviews.

8.1 Evaluation Method

To connect back to the hypothesis in Chapter 1 the goal with the project is to create a tool that makes it easier for users to end up with the best picture from multiple shoots from the same scene and to make the users more aware of how to photograph a scene.

We are evaluating the end-user experience of the Image Editing Tool with a two section questionnaire and by interviewing subjects in a semi structured way. The first section of the questioner is more general about the usage of the Image Editing Tool. The second section of the questioner is targeted to evaluate the usage and the usability of our tool. The second section of the survey is done by asking users concerning the five steps covered in the previous chapter. The first section of the questioner was published first and a few days later in association with the interviews the second section was published.

8.2 Quantitative Survey

The quantitative survey was represented by a questionnaire. The purpose of conducting a questionnaire was that we could in a flexible way reach out to a larger variety of individuals and thus get a broad collection of data on the use of the Image Editing Tool. We made the questionnaire with the help of the online tool designed especially for web surveys *Google Forms*, this is a service included in the Web-based Google Drive 2015.

The reason why we chose a web-based survey was partly because it can easily be spread rapidly to various online forums but also because we can easily compile the results with the provided survey tools.

After gathering the data the material were compiled in various charts in order to make the analysis of the data easier and transparent. The survey includes a total of 9 questions (see Next Section). These questions were of different types. We used most multiple choice questions, which meant that the respondent could choose one or more possible answers to a question. We provided one open question, made possible for respondents to write down what they thought. Non of the questions where made as a *Required question*. These questions were then used as the basis for subsequent interviews.

Before publishing the surveys we conducted a small pilot test for a small number of people to detect any flaws in the formulations of the questions and in the answer-options. The pilot test helped us to see the questionnaire from a different perspective and it also helped alter some questions before we published and sent out the questionnaire.

8.3 Qualitative Interviews

The purpose of the qualitative data collection was to acquire a greater and deeper understanding of the use of the Image Editing Tool. The main purpose with the interviews was to get answers to the questions that are difficult to answer via our survey. In this way we can also get a deeper understanding of answers to the questions that were asked in the survey. We started out from the questions in the questioner. This meant that we had a number of questions prepared for the interviews that we wanted to ask, but we could still add new questions during the interviews. During the process we were able to observe the users using the tool and we could gain knowledge of how the users adapted and perceived the user interface.

We chose to interview four individuals that had different experiences in photography and Image Editing.

8.3.1 Interview Method

The interviews took place in such a way that we first demonstrated the usage of the Image Editing Tool, similar to the step by step instructions in Chapter 7. We showed an example of how to recompose an image from one image-set here we used the sledge image set to change the man dressed in black with the beard man with the blue jacket while at the same time keep the surrounding people in that photo see figure 7.7a. We then asked the user to first do the same thing as we showed, and then we encouraged them to try and composed that set in a different way. Afterwards we asked the user to use approximately 10 minutes to go through our set of example image (see Appendix A.2) meanwhile we took notes of the comments they gave during interaction with the Tool.

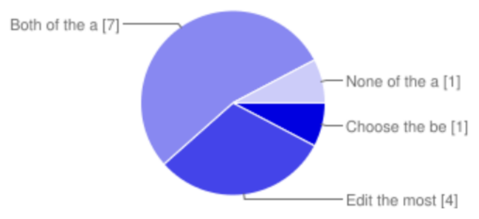
8.4 Results

In this section we are presenting the result and analysis from the quantitative survey, and the four qualitative semi-structured interviews. Through the survey, we received a total of 14 responses, of which 7 of the respondents did fill in the second section of the survey.

8.4.1 Results of the quantitative survey

When asked in which way the user would use the Image Editing Tool, it was found that almost all would use it in some way. More than half would use the Tool to both Choose the best picture from a set of multiple exposures and to Edit the most appropriate image from a set of images. 30% would use it to just edit the images.

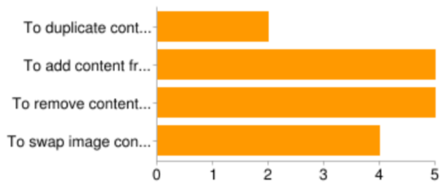
Would you use this application to:



Choose the best picture from a set of multiple exposures	**1**	7.7%
Edit the most appropriate image from a set of images	**4**	30.8%
Both of the above answers	**7**	53.8%
None of the above answers	**1**	7.7%

There was a shared view on how to use the tool, just as many said they would remove content as said they would add content from different images. One third had the opinion that they would use the tool to swap content, and only two of the respondent would use it for duplication of content.

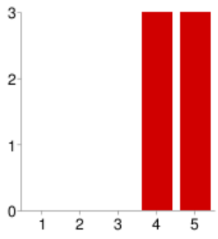
How would you use the Image Editing Tool?



To duplicate content from different images	**2**	16.7%
To add content from different images	**5**	41.7%
To remove content from a single image	**5**	41.7%
To swap image content	**4**	33.3%

We asked about how the tool might affect the awareness of photographer within a scene. Here we let respondents rate their opinion from 1-5 where 1 represented no influence and 5 represented a lot of influence. These responses indicate that the tool may have impact on how a photographer interacts and perceives the scenery surrounding them.

Would you be more aware of the scene you were about to photograph if you knew you could Edit the images in this application?



It would probably not affect me: 1	**0**	0%
2	**0**	0%
3	**0**	0%
4	**3**	25%
I would be more aware of the scene and the situation: 5	**3**	25%

Summary of the Results of Evaluating Step 1 - 5

Here we present the results from the second section of the questioner. These questions cover the steps of figure 7.1. The graphs to the corresponding steps are found on next page.

Step 1 As seen in graph *a* the majority found that the Mosaic view were helping them select a reference image. In graph *b* we see that the majority of the respondents would prefer to choose a reference image as presented in the Demo and only two said that they would prefer to look through each image one by one.

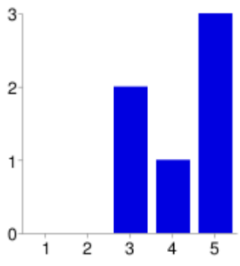
Step 2 In graph *c* we see that the respondents were able to perceive what the global threshold does, except for one single individual.

Step 3 Where not covered in the survey.

Step 4 Graph *d* tells us that the users were able to fine tune the selected layers, but this does not tell us whether they tried to adjust with the relative threshold or with the Paint/Erase tool.

Step 5 The graph in *e* tells that the respondents rate their opinion of the outcome of the resulting image as positive.

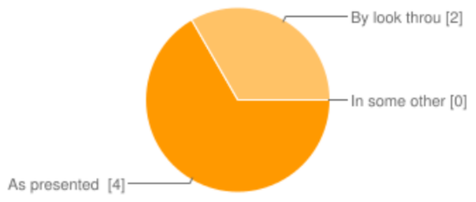
Does the mosaic view help to select the reference image?



Not really: 1	**0**	0%
2	**0**	0%
3	**2**	33.3%
4	**1**	16.7%
It does help in choosing a view: 5	**3**	50%

(a) Step 1

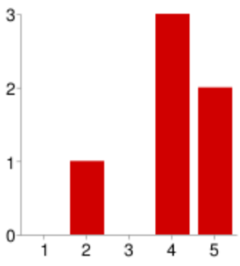
How would you prefer to select the reference image?



As presented in the Demo	**4**	66.7%
By look through each image one by one	**2**	33.3%
In some other way	**0**	0%

(b) Step 1

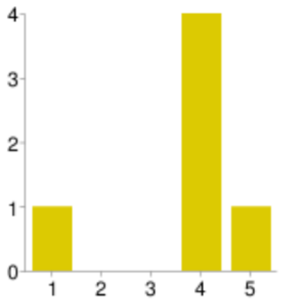
Is it clear from the user interface what the global threshold does, when you scroll the mouse wheel?



It is not clear: 1	**0**	0%
2	**1**	16.7%
3	**0**	0%
4	**3**	50%
It is clear: 5	**2**	33.3%

(c) Step 2

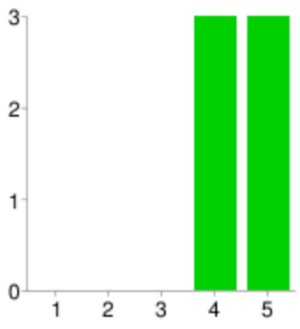
Where you able to Fine tune adjust the selected layers?



No it was hard: 1	**1**	16.7%
2	**0**	0%
3	**0**	0%
4	**4**	66.7%
Yes it was possible: 5	**1**	16.7%

(d) Step 4

What do you think of the resulting image?



Not satisfying : 1	**0**	0%
2	**0**	0%
3	**0**	0%
4	**3**	50%
Satisfying : 5	**3**	50%

(e) Step 5

8.4.2 Results of the Qualitative Interviews

Based on the four interviews, we chose to address and highlight the parts of the interview that provided interesting information for us to look closer at. The analysis results are presented and described below.

Observing the Interaction

Here we present the results of the part of the interviews when the user commented on how they used and interacted with the tool.

Fine tune adjustment with *Paint Layer*. There were several comments of how to improve the *Paint Layer* feature. The first was to only paint in one layer that the user self choose. Because it is a problem when there is multiple layers selected that are next to each other or multiple selected layers that are overlapping. When the user paints in one layer our tool chooses the layer that the pointer starts to paint in but it. This situation were perceived as the users lost control of the painting.

Fine tune adjustment with *Erase Layer*. There turned out to be a similar situation when *Erase Layer* is selected. When multiple selected layers next to each other or selected overlapping layers, then the user can not control in which layer to Erase but instead the erase tool will erase all the layers underneath the eraser.

Choose which layer to Paint/Erase. Suggestions where made that the user should be able to choose which layer to Paint or Erase in.

Erase without selecting. On the contrary to previews comment suggestions from other users where that it should not be necessary to first select a layer when Erasing in it. Instead they thought it would be better to erase any layer when the Eraser tool is selected.

Ctrl-z. Other suggestions where made that when the user have Painted or Erased in a layer they should be able to regret their last commands by pressing *Ctrl-z* on the key board to go back.

Zoom In. A desired feature when doing Fine Tune adjustment is to be able to zoom in on a region to be able to do even more fine grained adjustments.

Display threshold value. When doing adjustments to a threshold value it could help to display that value.

Display the options. Another suggestion was to show our layer options separately on a smaller window next to the main canvas.

Comments after the User tried the Tool

- The Instructions could be even more clearly and perhaps it would help to have interactive instructions. That is instructions that shows a small popup window on the different tool options.
- With the help of the mosaic view some user experienced that they gained knowledge about 3d perspective and the relation to 2d plane.
- Users would probably use the tool in a group portrait photo to make everybody look good.
- One user questioned the authenticity of the resulting image.
- The static differences does not effect the users experience, acording to the respondents when they have such control over what blobs to include or not include.

Chapter 9

Conclusions

We have presented and implemented a web based image editing tool that help photographer go from multiple images to one best preferred. Our approach finds the differences with the help of Level Set Methods. On an optimal way we do local adjustments of segments with Fast Marching Methods. The pipeline runs in a web browser without an external server, only a web server storing the necessary HTML5 and javascript files.

A user study validates that users could work and get satisfying results from multiple images of the same scene in a easy way by interacting with our proposed method. The empirical data gathered from the survey and the four interviews has been the basis of our analysis. The results clearly show that the tool is useful it also suggest that there is a lot to improve when it comes to the user friendliness. As we saw in most comments and suggestion they were about the fine tune Paint/Erase tool. This was unexpected because the main focus was on developing a good and fast enough algorithm for finding differences, the manual fine tuning was not our greatest priority but our study shows that this area is more important than expected.

The user study shows that the tool is useful in multiple areas as adding, removing, swapping and duplication of content. Division in how the respondents thought about how they would use the tool was relatively unpredictable, this result were surprising as we would expected that most user would choose to use the tool for duplicate content.

Our results also confirms the hypothesis that the Image editing tool leads to better pictures and that it can help that users to be more aware of how to photograph a scene.

References

[1] (2009). Mit 18.336 numerical methods for partial differential equations, spring 2009. (Accessed 25 March, 2015) [online] http://ocw.mit.edu/courses/mathematics/18-336-numerical-methods-for-partial-differential-equations-spring-2009/.

[2] (2012). Black, noel and moore, shirley. "gauss-seidel method. from mathworld–a wolfram web resource. (Accessed 28 March, 2015) [online] http://mathworld.wolfram.com/Gauss-SeidelMethod.html.

[3] (2012). mathieu bernard-reymond intervalles series. (Accessed 28 March, 2015) [online] http://matbr.com/intervalles/.

[4] Bærentzen, J. A. (2001). On the implementation of fast marching methods for 3D lattices.

[5] Bai, X. and Sapiro, G. (2007). A geodesic framework for fast interactive image and video segmentation and matting. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8.

[6] Barnes, C., Shechtman, E., Finkelstein, A., and Goldman, D. B. (2009). Patchmatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24:1–24:11.

[7] Brown, M., Szeliski, R., and Winder, S. (2005). Multi-image matching using multi-scale oriented patches. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 510–517, Washington, DC, USA. IEEE Computer Society.

[8] Burt, P. J. and Adelson, E. H. (1983). A multiresolution spline with application to image mosaics. *ACM Trans. Graph.*, 2(4):217–236.

[9] Calonder, M., Lepetit, V., Ozuysal, M., Trzcinski, T., Strecha, C., and Fua, P. (2012). BRIEF: Computing a Local Binary Descriptor Very Fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298.

[10] Caselles, V., Kimmel, R., and Sapiro, G. (1995). Geodesic active contours. *International Journal of Computer Vision*, 22:61–79.

[11] Dubrovina, A., Hershkovitz, R., and Kimmel, R. (2014). Image editing using level set trees. In *Image Processing (ICIP), 2014 IEEE International Conference on*, pages 4442–4446.

[12] Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. In *Comm. of the ACM*, volume 24, pages 381–395. CVIU.

[13] Hartley, R. I. (1997). In defense of the eight-point algorithm. *IEEE Trans. Pattern Anal. Mach. Intell.*, 19(6):580–593.

[14] Hartley, R. I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second edition.

[15] Lepetit, V. and Fua, P. (2004). Towards Recognizing Feature Points using Classification Trees. Technical Report IC/2004/74, EPFL.

[16] Liu, Z., Yuan, L., Tang, X., Uyttendaele, M., and Sun, J. (2014). Fast burst images denoising. *ACM Transactions on Graphics (TOG)*, 33(6).

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

[18] Malladi, R., Sethian, J. A., and Vemuri, B. C. (1995). Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:158–175.

[19] Osher, S. and Fedkiw, R. P. (2003). *Level set methods and dynamic implicit surfaces*. Applied mathematical science. Springer, New York, N.Y.

[20] Pérez, P., Gangnet, M., and Blake, A. (2003). Poisson image editing. In *ACM SIGGRAPH 2003 Papers*, SIGGRAPH '03, pages 313–318, New York, NY, USA. ACM.

[21] Porter, T. and Duff, T. (1984). Compositing digital images. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '84, pages 253–259, New York, NY, USA. ACM.

[22] Radke, R. J. (2012). *Computer Vision for Visual Effects*. Cambridge University Press, New York, NY, USA.

[23] Rosin, P. L. (1999). Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291–307.

[24] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443.

[25] Rublee, E., Rabaud, V., Konolige, K., and Bradski, G. (2011). Orb: An efficient alternative to sift or surf. In *Proceedings of the 2011 International Conference on Computer Vision*, ICCV '11, pages 2564–2571, Washington, DC, USA. IEEE Computer Society.

[26] Sethian, J. A. (1999). *Level set methods and fast marching methods : evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*. Cambridge monographs on applied and computational mathematics. Cambridge University Press, Cambridge, GB. Première édition parue sous le titre : Level set methods, 1996.

[27] Tasli, H. E., van Gemert, J. C., and Gevers, T. (2013). Spot the differences: From a photograph burst to the single best picture. In *ACM International Conference on Multimedia*.

[28] Yatziv, L., Bartesaghi, A., and Sapiro, G. (2006). O(n) implementation of the fast marching algorithm. *Journal of Computational Physics*, 212(2):393 – 399.

Appendix A

Flowchart over Step 1 - 4b

For a comprehensive view and to catch up with Step 2 - 4a, we here present flowchart. These include handling of mouse scroll wheel, mouse point and mouse click when the check box Select Layer is selected.

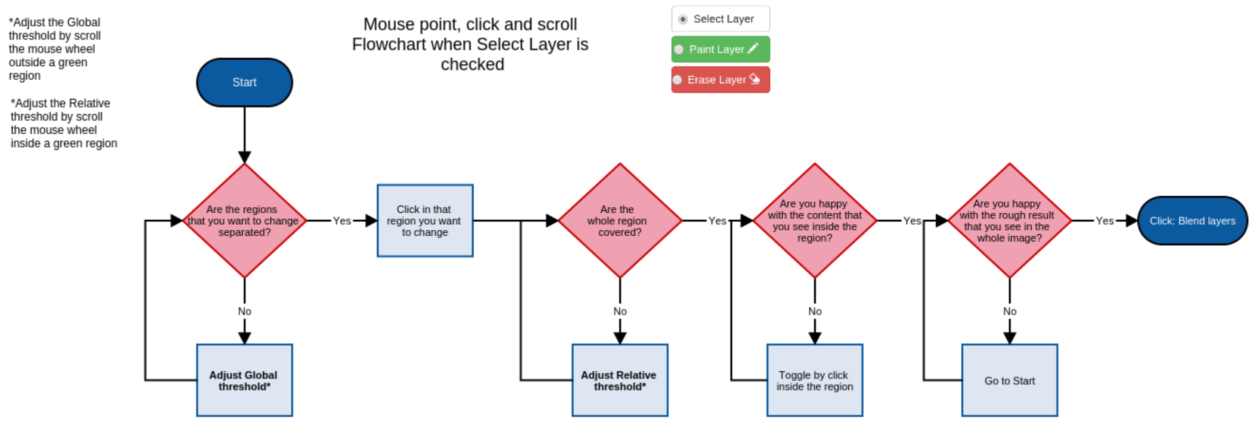
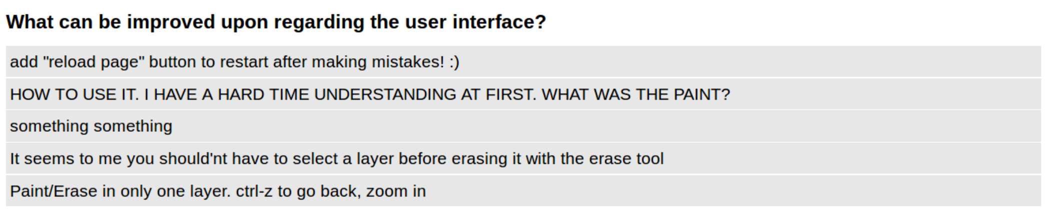

*Adjust the Global threshold by scroll the mouse wheel outside a green region
*Adjust the Relative threshold by scroll the mouse wheel inside a green region
Mouse point, click and scroll Flowchart when Select Layer is checked
Select Layer
Paint Layer
Erase Layer
Start
Are the regions that you want to change separated?
Yes
No
Adjust Global threshold*
Click in that region you want to change
Are the whole region covered?
Yes
No
Adjust Relative threshold*
Are you happy with the content that you see inside the region?
Yes
No
Toggle by click inside the region
Are you happy with the rough result that you see in the whole image?
Yes
No
Go to Start
Click: Blend layers

Fig. A.1 Flowchart

A.1 Question 4 From the Results

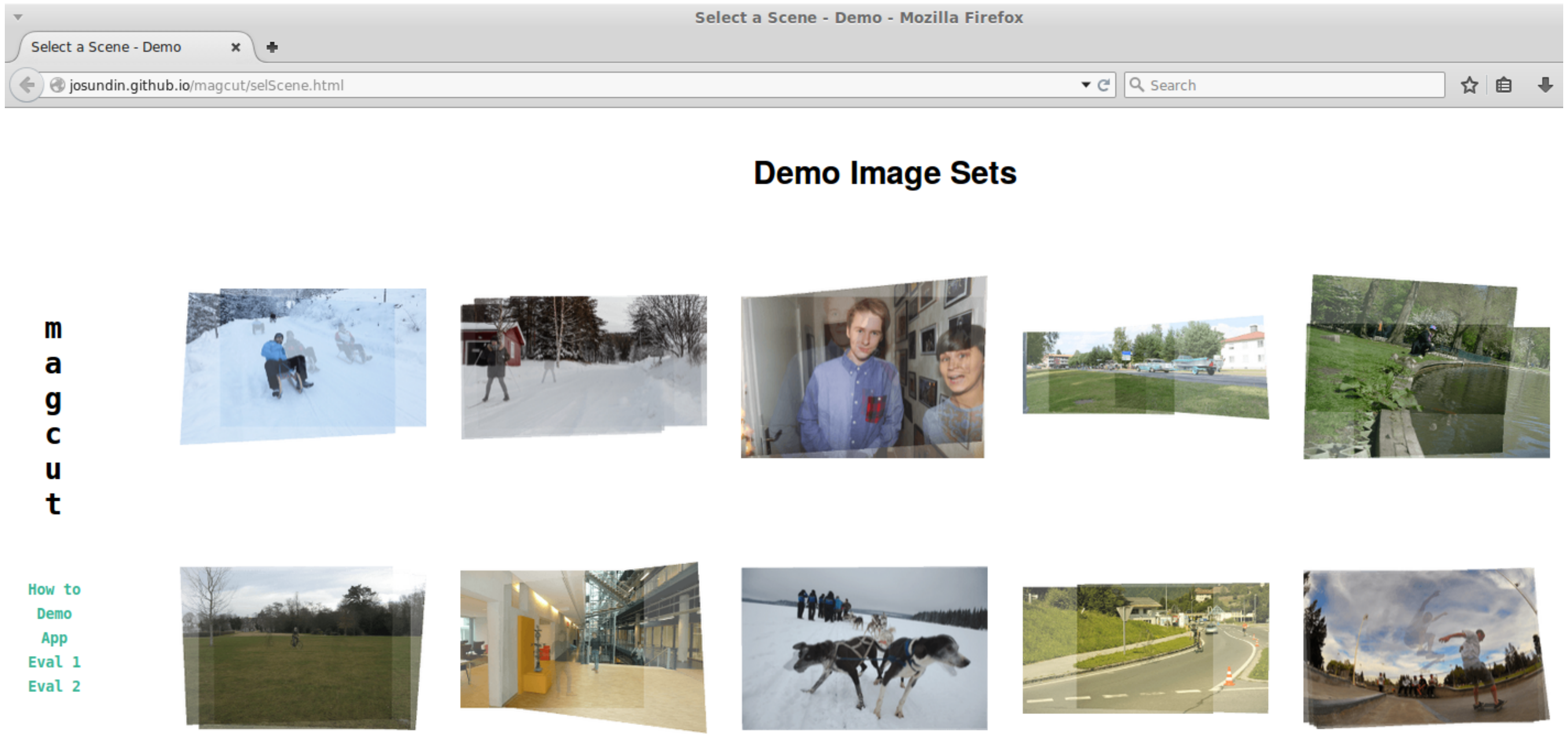


What can be improved upon regarding the user interface?

add "reload page" button to restart after making mistakes! :)
HOW TO USE IT. I HAVE A HARD TIME UNDERSTANDING AT FIRST. WHAT WAS THE PAINT?
something something
It seems to me you should'nt have to select a layer before erasing it with the erase tool
Paint/Erase in only one layer. ctrl-z to go back, zoom in

Fig. A.2 Question 4

A.2 Demo image sets


Select a Scene - Demo - Mozilla Firefox
Select a Scene - Demo
josundin.github.io/magcut/selScene.html
Search
Demo Image Sets
magcut
How to
Demo
App
Eval 1
Eval 2