

# Mobile Image Retargeting



Daniel Graf

Bachelor Thesis  
February 2013

*Supervisors:*  
Dr. Daniele Panozzo  
Prof. Dr. Olga Sorkine



# Abstract

We propose an algorithm for axis-aligned content-aware image retargeting specifically optimized for mobile devices and we show that interactive image retargeting is possible even with a low-power, mobile CPU.

Considering the limited screen space, we designed a novel user interface that allows the user to paint saliency maps directly onto the retargeted image while simultaneously updating the underlying deformation in real time.

Our algorithm incorporates automatic cropping into the image optimization to significantly improve the retargeting quality.

Finally, we apply our algorithm in a picture gallery to greatly improve the screen space utilization with respect to existing approaches.





# Zusammenfassung

In dieser Arbeit präsentieren wir einen Algorithmus zum Verändern des Seitenverhältnisses eines Bildes, engl. *Image Retargeting* genannt. Dabei berücksichtigen wir den Inhalt des Bildes und suchen eine achsentreue Abbildung, die wichtige Bildteile möglichst unverändert lässt. Unser Algorithmus ist speziell optimiert für mobile Geräte. Wir zeigen, wie wir die Bildverarbeitungspipeline so implementieren, dass wir auch auf Plattformen mit eingeschränkter Rechenkapazität interaktive Bildraten erreichen.

Um die begrenzte Bildschirmgrösse zu berücksichtigen, entwickelten wir eine neuartige Benutzeroberfläche, die es erlaubt, wichtige Bildteile direkt im veränderten Bild anzumalen. Dabei wird sowohl das veränderte Bild als auch eine Karte, die bezeichnet, welche Bildteile wichtig sind, fortlaufend aktualisiert.

Wir erweitern unseren Algorithmus um das Bild nicht nur zu verformen, sondern, falls nötig, auch automatisch zu beschneiden, um auf diese Weise zum gewünschten Seitenverhältnis zu kommen. Dies verbessert die Qualität des veränderten Bildes massgeblich.

Zum Schluss zeigen wir eine praktische Anwendung, eine einfache Bildergalerie, um die mobilen Einsatzmöglichkeiten von Image Retargeting aufzuzeigen.



## Bachelor Thesis

# Mobile Image Retargeting

### Introduction

Image retargeting allows to change the aspect ratio of a picture, without introducing noticeable distortion and thus allowing a more efficient use of the screen space. This is particularly important for mobile devices, but unfortunately these techniques are computationally expensive and so they are usually used before the content is sent to the mobile device. A novel retargeting algorithm developed at IGL allows retargeting of images in a few milliseconds when executed on a desktop CPU. The bachelor thesis will adapt this algorithm to run on the Apple iOS operating system.

### Task Description

The candidate will study how to develop mobile applications on iOS, and then implement on iOS the Axis-Aligned Retargeting algorithm starting from an existing C++ implementation. A touch-based user interface will be developed to allow the user to interactively paint the saliency map and see the result of the deformation in real-time. Depending on the skill and interests of the candidate, the integration of the algorithm inside the iOS web browser will allow the retargeting of web pages directly on the device.

### Detailed Task List:

**Learn to use the CVXGEN solver generator:** The retargeting algorithm requires to solve a quadratic programming problem. In this project, we will use the CVXGEN system to generate a customized solver that can be embedded directly into the binary deployed on the mobile device. The student will learn how to describe problems using the CVX syntax and then use it to generate the solver required by the retargeting algorithm.

**Experiment with the desktop implementation of Axis Aligned Retargeting:** The code is available on the IGL website.

**Learn to develop a basic iOS application:** A basic iOS application that renders a single image using opengl will be developed to familiarize with the iOS SDK.

**Compile the core algorithm on an iOS device:** The student will implement the algorithm and run it on an iOS device. No user-interface will be implemented in this stage, the objective is to evaluate the running time performances and make sure that the solver is able to run on the ARM architecture with the limited resources available on a iOS device. If necessary, different QP solvers will be tested until a suitable one is found. Precision of the solution and the running time will be compared with an equivalent implementation running on a normal workstation.

**Design of a touch-based UI:** An UI will be developed to allow interactive saliency painting and tweaking of the algorithms parameters.

**Optional extension 1, Cropping:** The original formulation of Axis Aligned Retargeting does not include a cropping factor in the optimization. The integration of cropping in the optimization process is an interesting research challenge that will greatly improve the practical impact of the mobile application developed in this thesis.

**Optional extension 2, Integration in a web browser:** The retargeting algorithm could be integrated in the safari web browser to retarget webpages to improve their layout on different screen sizes. The integration will be possible only if iOS provides enough control of the internal components of the web browser.

### Grading scale

The thesis will be graded using the following scale:

|     |  |
|-----|--|
| 6.0 | Extraordinary quality, the results are much higher than expected.                      |
| 5.5 | Thesis results are very good; student expanded on the original theme.                  |
| 5.0 | Thesis meets expectations.   |
| 4.5 | Thesis partially meets expectations and has minor deficits                             |
| 4.0 | Thesis meets minimum quality requirements; but has deficits and is below expectations. |

Dr. Daniele Panozzo

Prof. Dr. Olga Sorkine

# Acknowledgements

I want to thank Dr. Daniele Panozzo and Prof. Dr. Olga Sorkine for their support and for advising me on this thesis. I also want to thank Jacob Mattingley for the CVXGEN solver.



# Contents

|  |           |
|--|-----------|
| <b>1. Introduction</b>   | <b>1</b>  |
| 1.1. Goals . . . . .   | 3         |
| 1.2. Overview . . . . .  | 3         |
| <b>2. Related Work</b>   | <b>5</b>  |
| 2.1. Retargeting . . . . .                                       | 5         |
| 2.1.1. Problem Statement . . . . .                               | 5         |
| 2.1.2. General classification of retargeting operators . . . . . | 6         |
| 2.1.3. Discrete Approaches . . . . .                             | 7         |
| 2.1.4. Continuous Warping Methods . . . . .                      | 7         |
| 2.1.5. Comparison . . . . .                                      | 12        |
| 2.2. Saliency Evaluation . . . . .                               | 12        |
| 2.2.1. Low-Level Salient Region Detection . . . . .              | 13        |
| 2.2.2. High-Level Object Detection . . . . .                     | 13        |
| 2.3. Convex Optimization . . . . .                               | 14        |
| 2.3.1. Problem Formalization . . . . .                           | 14        |
| 2.3.2. Interior Point Method . . . . .                           | 14        |
| <b>3. Algorithm</b>  | <b>17</b> |
| 3.1. Retargeting Operator . . . . .                              | 17        |
| 3.1.1. Quadratic Convex Optimization Problem . . . . .           | 18        |
| 3.1.2. Energy Formulation . . . . .                              | 18        |
| 3.1.3. Boundary Constraints . . . . .                            | 21        |
| 3.1.4. Spline Interpolation . . . . .                            | 23        |
| 3.2. Cropping . . . . .  | 24        |
| 3.2.1. General Approaches . . . . .                              | 25        |

|           |  |           |
|-----------|--|-----------|
| 3.2.2.    | Threshold-Based Cropping . . . . .           | 26        |
| 3.2.3.    | Implementation . . . . .                     | 27        |
| 3.2.4.    | Crop-Treshold Range . . . . .                | 31        |
| 3.3.      | Saliency . . . . .                           | 32        |
| 3.3.1.    | Gradient-Based Saliency Detection . . . . .  | 33        |
| 3.3.2.    | Face Detection . . . . .                     | 33        |
| <b>4.</b> | <b>Implementation</b>                        | <b>35</b> |
| 4.1.      | Solving the Quadratic Program . . . . .      | 35        |
| 4.2.      | Interface . . . . .                          | 36        |
| 4.2.1.    | Fixed-Point Grid Stabilization . . . . .     | 37        |
| 4.2.2.    | Crop Preview . . . . .                       | 40        |
| 4.3.      | iOS technologies used . . . . .              | 41        |
| <b>5.</b> | <b>Results</b>                               | <b>43</b> |
| 5.1.      | App Performance . . . . .                    | 43        |
| 5.2.      | Retargeting quality using Cropping . . . . . | 44        |
| 5.3.      | RetargetMe Benchmark . . . . .               | 44        |
| 5.4.      | Applications . . . . .                       | 46        |
| 5.5.      | Limitations and Future Work . . . . .        | 46        |
| <b>6.</b> | <b>Conclusion</b>                            | <b>49</b> |
| <b>A.</b> | <b>Appendix</b>                              | <b>51</b> |
| A.1.      | CVXGEN Problem Statement . . . . .           | 51        |
|           | <b>Bibliography</b>                          | <b>53</b> |



# Introduction

Digital images are captured in various aspect ratios. In still photography, a ratio of 3:2 (1.5) emerged from the format of analog 35mm film. Since the raise of digital image sensors, the ratio of 4:3 (1.33) is widely-used, which in turn originates from the beginnings of the motion picture industry. Video content is usually produced with a specific target screen in mind. Television has recently switched from a 4:3 (1.66) to a 16:9 (1.77) ratio and cinema also uses widescreen formats as wide as 2.40:1.

Consumer devices, like computer monitors, tablets and smartphones, use various aspect ratios and are used to display a large variety of images. For compensating the mismatch in aspect ratio, the content often gets stretched or cropped, which leads to visual artifacts like unnatural proportions or invisibility of important parts of the image. The easiest way to avoid this problems is to proportionally downscale the image to fit the screen, which is called letterboxing due to the additional horizontal black bars on the screen.

To make full use of the available screen real estate, many methods for non-uniformly scaling the image to the desired target resolutions have been developed in recent years. These full- or semiautomatic solutions try to preserve important areas of the image while compensating in the non-relevant parts.

The goal of this thesis is to bring content-aware image retargeting to mobile devices. The limitations on computational power, battery life and screen space on handheld devices pose a challenging problem and new algorithmic solutions are necessary.

## 1. Introduction



**Figure 1.1.:** iOS user interface. Using this touch-based user interface, the user can paint the saliency map and can preview the image in the target aspect ratio also seeing the cropped image parts.

## 1.1. Goals

Based on the axis-aligned retargeting algorithm designed by [Panozzo et al. 2012], we develop an application for the iOS operating system. For this we need to solve a quadratic convex optimization problem and apply a piecewise bilinear warp to the image.

We present a novel touch-based user interface, that enables the user to paint the region of importance directly onto the retargeted image. This allows interactive refinement of the deformation on a full screen preview of the resulting image. To provide an interactive experience, we need our retargeting pipeline to achieve a real-time performance of 30 Hz.

In addition, we present and implement an extension of the axis-aligned deformation that also incorporates cropping into the process. This increases the flexibility of the optimization and the quality of the result without significant loss of performance.

Finally we compare the quality of our retargeting method with existing approaches and present possible applications of mobile image retargeting.

## 1.2. Overview

In Chapter 2, we give an overview of previous retargeting techniques and relevant work in the areas of salient region detection and convex optimization.

Chapter 3 describes our algorithm in detail and states the underlying optimization problem. We explain the influence of the saliency map as well as the many parameters of the optimization.

Chapter 4 points out the specifics of an implementation on a mobile device using the Apple iOS SDK. There we also present our new user interface for simultaneous painting and retargeting based on fixpoint stabilization at the painting finger’s position.

In Chapter 5, we test our solution on the RETARGETME benchmark. We discuss the quality of the cropping extension and discuss possible directions for future work. We also apply our algorithm to a photo gallery, where retargeting is used to fit pictures with various aspect ratios into a regular grid layout.

## *1. Introduction*

## Related Work

In this Chapter, we review existing content-aware image retargeting pipelines. All methods decouple the problem into two steps. First, we extract some saliency map from the image. The saliency map defines which features of the original image should be preserved and can be defined on a pixel- or region-level. Using this importance measure, the retargeting operator then tries to find an optimal way to generate a retargeted image. These two steps are usually independent. So any approach for salient region detection can be combined with any retargeting operator.

First, we focus on the retargeting process by specifying the problem we want to solve and the objectives we aim at. Then, we compare several discrete and continuous approaches and discuss their suitability for mobile applications.

In Section 2.2, we summarize previous work on salient region detection and argue why manual saliency drawing is important for high-quality image retargeting.

In Section 2.3, we give a short introduction to quadratic programming problems, which is the type of convex optimization we will use for the retargeting operator proposed in this thesis.

### 2.1. Retargeting

#### 2.1.1. Problem Statement

We look at a digital image  $I$  as a sampled representation of some continuous 2D function. If  $I$  is of width  $W$  and height  $H$ , then  $I$  is a set of point samples at the vertex positions of a rectangular regular grid of  $W$  columns and  $H$  rows. Each such point sample, called *pixel*, is a discretized

## 2. Related Work

representation of the sampled color. With  $I(x, y)$  we denote the sample in row  $x$  and column  $y$  of the grid. Usually a byte of information for each of the three color channels (red, blue and green) is stored.

The retargeting task consists in generating a new image  $I'$  of width  $W'$  and height  $H'$  that is perceptually as similar as possible to  $I$ . Usually,  $W' < W$  and/or  $H' < H$ . But some methods also support image enlarging, where  $W' > W$  or  $H' > H$ . Moreover, certain operators mainly take the aspect ratio of the retargeted image into account, i.e.  $W'/H'$ , and handle different resolutions of the same ratio just by down- respectively upsampling.

Since human visual perception is based on many different factors like symmetry, recognition of known objects or textures and noise, it is difficult to give an unambiguous quality measure for  $I'$ .

According to [Shamir and Sorkine 2009],  $I'$  should satisfy three main objectives:

- The important *content* of  $I$  should be preserved in  $I'$ .
- The important *structure* of  $I$  should be preserved in  $I'$ .
- $I'$  should be free of visual artifacts.

The definition of *important* features of  $I$  is based on a saliency measure  $S : I \rightarrow [0, 1]$  and possible further constraints.

### 2.1.2. General classification of retargeting operators

The simplest retargeting operator is *cropping* where  $I'$  is a rectangular region of  $I$  of size  $W' \times H'$ . This can be done manually, semiautomatically [Santella et al. 2006] or fully automatically [Fan et al. 2003] using a saliency measure  $S$ . Of course, this only provides satisfying results if all of the important content is already localized in such a subwindow.

Extension of an image to the case of  $W' > W$  or  $H' > H$  can be seen as the “inverse” of cropping. This is often called *letterboxing* since the original image is clamped between black stripes on two or even four sides. This is a very inefficient use of the available screen space, however, it perfectly satisfies all three objectives in [Shamir and Sorkine 2009]. Therefore, letterboxing is widely used, for example in the default image viewers and video players of iOS and Android and almost every TV-set.

An alternative approach is *uniform homogeneous scaling*, which corresponds to this mapping:

$$I'(x, y) = I(\lfloor x \cdot \frac{H}{H'} \rfloor, \lfloor y \cdot \frac{W}{W'} \rfloor)$$

This mapping might lead to aliasing artifacts as some pixels of the original image are either ignored or repeated. Using continuous interpolation, these artifacts are removed. However, homogeneous scaling does not preserve important structure as the whole image is shrunk or stretched.

More sophisticated operators are usually classified into *discrete approaches*, which operate on pixels, and *continuous warping methods*, which consider the image as a continuous function.

### 2.1.3. Discrete Approaches

#### Seam Carving

Without loss of generality, we assume that the task is to find  $I'$  with  $H' = H$  and  $W' < W$  with  $n = W - W'$ , meaning that we only want to decrease the width of the original image.

In general, discrete retargeting operators try to repeatedly remove pixels starting with the least salient ones. Without additional constraints, deleting single pixels would destroy the rectangular shape of the image. Removing exactly  $n$  pixels per row or the  $n$  least salient columns would give the desired size but also introduces visible discontinuities.

Seam carving by [Avidan and Shamir 2007] (SC) repeatedly removes the least salient seam of the image, where a seam is defined as a vertical 8-connected path that contains exactly one pixel per row. A dynamic programming algorithm allows finding such an optimal seam in time  $\mathcal{O}(W \cdot H)$ . The complexity to remove  $n$  seams is in  $\mathcal{O}(W \cdot H \cdot n)$ .

In many cases, SC generates very appealing results. However, discontinuities can still appear since chains of pixels are removed.

By calculating the order of seam removals and storing them in a so called *multi-size image*,  $I'$  can be quickly updated for varying  $W'$ . This only holds if the saliency map  $S$  is fixed; any change to  $S$  requires a full recomputation of the seam removal order. A faster way to find the  $n$  seams to remove is presented in [Huang et al. 2009] where a specialized minimum bipartite matching solver is used to find a good approximation to the  $n$  optimal seams. The computational effort is greatly reduced, but it is still too expensive for real-time, high resolution retargeting on mobile applications.

Seam carving can be extended for image enlarging or object removal and can also be used to retarget videos [Rubinstein et al. 2008].

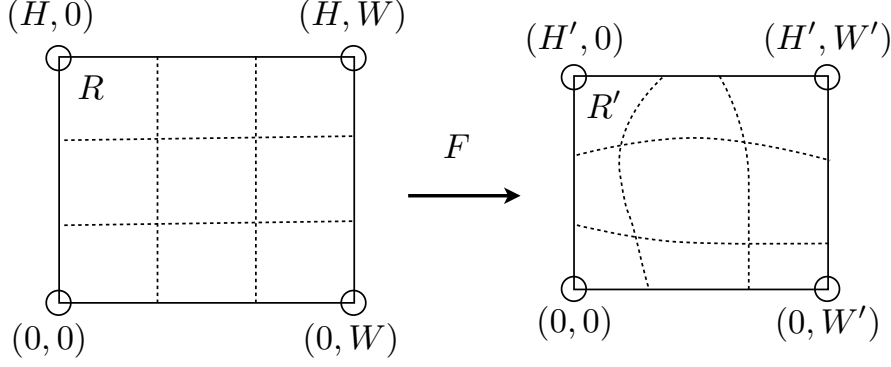
#### Multi-operator Retargeting

In [Rubinstein et al. 2009], a method for combining scaling, cropping and seam carving is presented. An optimal way to combine these operators is found out of exponentially many possibilities using an additional dynamic programming approach. With optimization times of several minutes, this method is not appropriate for mobile use, but the achieved quality was ranked significantly higher than pure seam carving and among the best operators in the user study of [Rubinstein et al. 2010].

### 2.1.4. Continuous Warping Methods

Warping methods look for a continuous mapping  $F$  from the initial rectangle  $R$  of size  $W \times H$  to the target rectangle  $R'$  of size  $W' \times H'$ . By abstracting from the discretized input image, we can create operators that express new pixel values as a superposition of pixels in the original image. The mapping is discretized using piecewise-elements to make the computation feasible. The final image  $I'$  is then computed by applying the map  $F$  to  $I$ .

## 2. Related Work



**Figure 2.1.:** Illustration of the mapping from grid  $G$  to grid  $G'$ .

We define the following formal setup following the notation given in [Shamir and Sorkine 2009], which is based on [Wang et al. 2008] and [Krähenbühl et al. 2009]:

The mapping  $F : \mathbb{R}^2 \rightarrow \mathbb{R}^2$  with  $F(x, y) = (F_x(x, y), F_y(x, y))$  should transform  $R$  into  $R'$  and therefore satisfy

$$F_x(0, \cdot) = 0, F_x(H, \cdot) = H', F_y(\cdot, 0) = 0, F_y(\cdot, W) = W',$$

which is visualized in Figure 2.1. Optimally  $F$  is a warp that keeps every part of the original image unchanged, which we can express locally by constraining that the Jacobian  $J_F(x, y)$  should be the identity

$$\forall (x, y) \in R : J_F(x, y) = \begin{pmatrix} \frac{\partial F}{\partial x} & \frac{\partial F}{\partial y} \end{pmatrix} = \begin{pmatrix} \frac{\partial F_x}{\partial x} & \frac{\partial F_x}{\partial y} \\ \frac{\partial F_y}{\partial x} & \frac{\partial F_y}{\partial y} \end{pmatrix} \stackrel{!}{=} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = I_2.$$

But of course, for  $R \neq R'$ , this is impossible. The different warping operators use variational methods to find a map  $F$  whose Jacobian is close to  $I_2$  on salient parts.

### Energy formulation

All continuous methods use an energy formulation that defines which distortion measure the mapping should minimize. In [Shamir and Sorkine 2009], the following objective functional is minimized

$$E(F) = \int_{x=0}^H \int_{y=0}^W S(x, y) \|J_F(x, y) - I_2\|^2 dx dy,$$

where  $\|\cdot\|$  is the Frobenius matrix norm and  $S$  is the saliency measure. This way, the distortion of each image part contributes to the energy proportional to its saliency. Then the optimal  $F$  is defined as

$$F = \underset{F}{\operatorname{argmin}} E(F) \text{ such that } F \text{ satisfies the boundary constraints.}$$



The method proposed by [Gal et al. 2006] uses the following energy term to express that salient parts should be scaled by some fixed factor  $s$  and background parts should be uniformly scaled:

$$E(F) = \int_{x=0}^H \int_{y=0}^W S(x, y) \|J_F(x, y) - sI_2\|^2 + (1 - S(x, y)) \|J_F(x, y) - A\|^2 dx dy,$$

where  $s = \min(H'/H, W'/W)$  and  $A$  is the homogeneous scaling matrix

$$A = \begin{pmatrix} H'/H & 0 \\ 0 & W'/W \end{pmatrix}.$$

[Wolf et al. 2007] present a method that allows flexible scaling along the retargeted dimension and encourages the mapping to stay smooth along the unchanged dimension. For horizontal retargeting, this results in the formulation

$$E(F) = \int_{x=0}^H \int_{y=0}^W S(x, y) \left( \frac{\partial F_x}{\partial x}(x, y) - 1 \right)^2 + w \left( \frac{\partial F_y}{\partial x}(x, y) \right)^2 dx dy,$$

for some weighting factor  $w$ .

These approaches allow fast numerical solutions due to the linear least-squares energy formulation that can be efficiently minimized with linear sparse solvers. On the other hand, the resulting mapping is not guaranteed to be injective, meaning that some of the piece-wise linear elements can intersect. They are called *fold overs*, and they are problematic, since multiple points of  $I$  are mapped into the same region of  $I'$ .

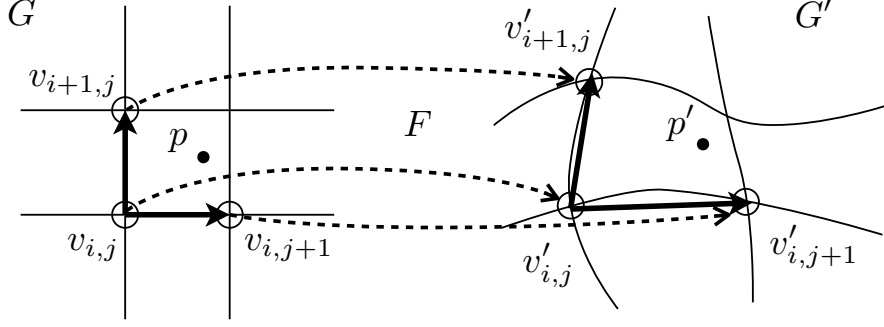
This scaling approach has been extended to the *scale-and-stretch* operator by [Wang et al. 2008] where the scaling factor  $s$  is allowed to locally vary. To find this field of scaling factors  $s(x, y)$ , they alternatively optimize the mapping  $F$  and the factors  $s$  until convergence. The flexibility allowed by this local scaling factors enables local rotations and contortions. They added a *grid line bending energy* to control such distortions and to limit fold overs.

The algorithm of [Krähenbühl et al. 2009] went back to using a single global scale factor to prevent proportionality artifacts. They introduced two new energy components to prevent bending and blurring of edges. As this work focused on streaming video retargeting, they also ensured temporal coherence of the warp. To allow precise manual control of the deformation, they further introduced more sophisticated constraints. The user can specify lines that should be preserved and mark objects that should keep their relative position in the warped image.

These advanced methods require more expensive optimization techniques. In [Wang et al. 2008], the mapping is iteratively refined by alternately optimizing the mapping and updating the scaling and grid line bending factors. [Krähenbühl et al. 2009] use a GPU-optimized multigrid solver to incorporate all constraints.

A general framework for energy-based image deformations is presented in [Karni et al. 2009]. They introduce an alternating *local-global-optimization*. Incorporating the saliency information only in the local optimization step allows this method to avoid fold overs.

## 2. Related Work



**Figure 2.2.:** Illustration of the discretized mapping from grid  $G$  to grid  $G'$ ,

### Discretization and Finite Differences

Once the energy term is defined, all methods discretize the mapping using a regular grid. We call  $G$  the original uniform grid on  $R$  that gets mapped to a grid  $G'$  on  $R'$ , that is the retargeted grid that represents  $F(G)$ . Let us denote the number of rows and columns of these grids by  $N$  and  $M$  and by  $v_{i,j}$  and  $v'_{i,j}$  the vertices of  $G$  respectively  $G'$ , where  $0 \leq i \leq M, 0 \leq j \leq N$ .

Inside each grid cell we use *bilinear interpolation* to locally linearize the mapping. A point in the cell at row  $i - 1$  and column  $j - 1$  of grid  $G$  is described in terms of the positions of the cell corners as

$$p = (1 - s) \cdot (1 - t) \cdot v_{i,j} + s \cdot (1 - t) \cdot v_{i+1,j} + (1 - s) \cdot t \cdot v_{i,j+1} + s \cdot t \cdot v_{i+1,j+1}.$$

for some  $s, t \in [0, 1]$  and will be mapped to  $p'$  in  $G'$  using the transformed cell corner positions

$$p' = (1 - s) \cdot (1 - t) \cdot v'_{i,j} + s \cdot (1 - t) \cdot v'_{i+1,j} + (1 - s) \cdot t \cdot v'_{i,j+1} + s \cdot t \cdot v'_{i+1,j+1}.$$

This discretization allows us to express the derivatives as finite differences:

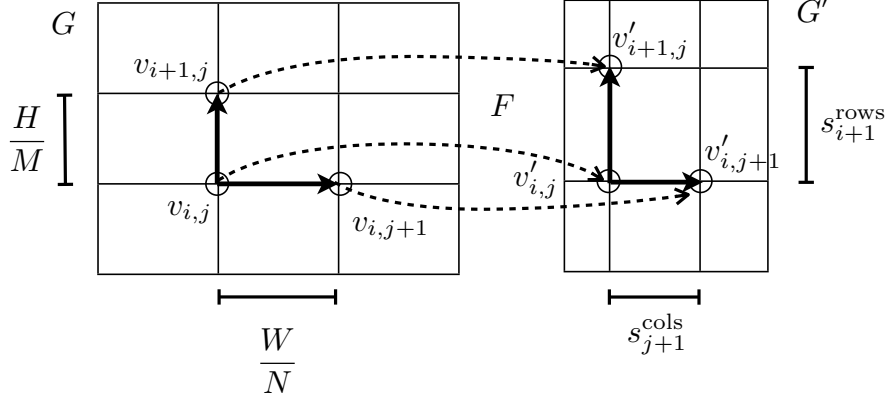
$$\frac{\partial F}{\partial x} = \frac{v'_{i+1,j} - v'_{i,j}}{\|v_{i+1,j} - v_{i,j}\|} \text{ and } \frac{\partial F}{\partial y} = \frac{v'_{i,j+1} - v'_{i,j}}{\|v_{i,j+1} - v_{i,j}\|}.$$

Figure 2.2 illustrates the discretized mapping. By choosing the resolution of the discretization, a tradeoff between the approximation quality to the continuous warp and the computational complexity can be made. [Wang et al. 2008] stated that a quad grid of  $20 \times 20$  cells produced sufficiently good results in all experiments. Such resolutions can be far below the dimensions of both  $I$  and  $I'$ , but they can still capture the optimal deformations reasonably well since neighboring regions should be deformed similarly to avoid visual artifacts.

However, [Krähenbühl et al. 2009] note that the coarse grid resolution considerably restricts the warp, making it difficult to preserve small scale features. They propose using a regularized pixel-accurate warping, which can run at interactive rates only using a customized GPU-based multigrid solver.

### Bijectivity

None of these methods can guarantee the absence of fold overs. Only recently, [Chen et al. 2010] introduced inequality constraints to formally guarantee non-negativity of each grid cell. They



**Figure 2.3:** Illustration of the axis aligned deformation space.

derive a convex quadratic programming problem, which is more expensive to solve than all the previously presented energies. Using the CVX MATLAB toolbox by [Grant et al. 2008], they report optimization times between 2 to 12 seconds on a grid of 10 to 32 cells in both dimensions.

Looking at previous retargeting methods and noticing that all of them hardly introduce local rotations, [Panozzo et al. 2012] proposed choosing the space of *axis-aligned deformations* as the meaningful space for content-aware image retargeting. This has the advantage of allowing a one-dimensional parametrization of the deformation, thus reducing the number of variables from  $M \cdot \dots \cdot N$  to  $M + N$ . All the vertices in the same row of the grid stay on the same line and have equally distant horizontal neighbors across all rows. This means that the grid  $G'$  satisfies the equalities

$$\begin{aligned} \forall i \in \{1, \dots, M\} \forall j \in \{0, \dots, N\}: s_i^{\text{rows}} &= v'_{i,j} - v'_{i-1,j} \text{ and} \\ \forall j \in \{1, \dots, N\} \forall i \in \{0, \dots, W\}: s_j^{\text{cols}} &= v'_{i,j} - v'_{i,j-1}, \end{aligned}$$

where  $s_i^{\text{rows}}$  and  $s_j^{\text{cols}}$  for  $1 \leq i \leq M$  and  $1 \leq j \leq N$  completely describe  $G'$ . The axis-aligned deformation space is illustrated in Figure 2.3.

They use an As-Similar-As-Possible (ASAP) energy formulation extended with a Laplacian regularization. They impose inequality constraints to enforce a minimum cell size that prevents discontinuities and consequently also fold overs. As other grid-based approaches before, they argue that coarse grids of  $25 \times 25$  cells suffice in most cases, even for multi-megapixel images. So with as few as 50 optimization variables, the convex optimization can be performed in 4ms (250 fps) on a recent single desktop CPU. This allows real-time frame rates for interactive saliency drawing and makes this method suitable for our mobile applications.

To increase the smoothness of the final warp, they propose using 1D cubic B-spline upsampling of the grid along both dimensions. From this refined grid, the final retargeted image  $I'$  gets sampled using bilinear interpolation.

The limitations of this method can be seen in situations where extreme shearing would better preserve salient parts. While the method of [Krähenbühl et al. 2009] allows such deformations, it requires a fast GPU to reach interactive rates. Also, non-axis-aligned straight lines are not preserved by this method.

### 2.1.5. Comparison

As the perceived quality of a retargeted image depends on many different factors, giving objective quality measures is very hard. The perceived quality is very subjective. For instance, this could get as personal as whether the viewer knows the people in the image and therefore could tell if their faces are even slightly deformed.

[Rubinstein et al. 2010] performed the first large-scale comparative user study. They compiled a dataset of 37 images in a benchmark called *RetargetMe*. After applying eight different operators to each of these images, they asked several hundred users to compare these images pairwise in a web-based survey. Streaming video (SV) [Krähenbühl et al. 2009], multi operator retargeting (MULTIOP) [Rubinstein et al. 2009] and manual cropping (CR) were preferred over the other approaches (seam carving (SC) [Avidan and Shamir 2007] and scale-and-stretch (SNS) [Wang et al. 2008]). It is interesting to note that all types of operators appear in the top three. Therefore, there seems to be no generally preferable approach among the discrete and continuous warping methods, and manual selection of an optimal subwindow can keep up with the more sophisticated approaches. According to the authors of the study, “this suggests that loss of content is generally preferred over deformation artifacts. The search for optimal cropping windows [...] is still very much a valid and relevant research venue.” They also underline that several automatic similarity metrics did not correspond to their comparative results. This confirms the complexity of defining retargeting quality in an objective way.

[Castillo et al. 2011] used eye-tracking to assess the quality of image retargeting operators on the *RetargetMe* benchmark. They especially compared the sets of fixation points before and after the retargeting to see whether artifacts caught the eyes of the viewers.

The study has been repeated in [Panozzo et al. 2012] together with the introduction of the axis-aligned retargeting operator. The quality of the axis-aligned operator (AA) was ranked indistinguishable from the three previous best methods SV, MULTIOP and CR. Therefore, this new approach provides state-of-the-art retargeting quality while also being considerably faster than other methods.

We refer to [Shamir and Sorkine 2009] and [Shamir et al. 2012] for an in-depth treatment of existing retargeting approaches.

## 2.2. Saliency Evaluation

The extraction of a saliency map is the first step in every retargeting pipeline. As with retargeting operators, the quality of a saliency detector is not trivial to measure. It depends on the observer and the context as well as on the specific application. To understand the human visual perception, many fields, ranging from neurosciences, biology, machine learning to computer vision, need to be considered. In general, a classification into *low-level stimuli-driven* and *high-level object-detection* types of saliency inference methods can be made. This classification goes back to the feature-integration theory of [Treisman and Gelade 1980]. They model human perception so that bottom-up features “are registered early, automatically, and in parallel across the visual field, while objects are identified separately and only at a later stage, which requires

focused attention.” (page 98 of [Treisman and Gelade 1980])

### 2.2.1. Low-Level Salient Region Detection

According to [Cheng et al. 2011], “saliency originates from visual uniqueness, unpredictability, rarity, or surprise, and is often attributed to variations in image attributes like color, gradient, edges and boundaries.” This indicates that filtering methods, like edge detection, can already be a reasonable approximation to the human visual perception. The seam carving operator by [Avidan and Shamir 2007] takes the L1-norm of the image gradient as saliency measure

$$S(I) = \left| \frac{\partial d}{\partial x} I \right| + \left| \frac{\partial d}{\partial y} I \right|,$$

while others like [Wang et al. 2008] use the Euclidean L2-norm

$$S(I) = \sqrt{\left( \frac{\partial d}{\partial x} I \right)^2 + \left( \frac{\partial d}{\partial y} I \right)^2}.$$

Other ways to detect locally significant parts are the edge-detector by [Canny 1986] or the corner-detector by [Harris and Stephens 1988].

In the user study conducted by [Einhäuser and König 2003], they asked the participants to carefully study some images while an eye tracker was recording fixation points. They found a correlation between the fixation points and regions of high *luminance-contrast*. Therefore many salient region detection algorithms try to find globally unique regions of high contrast. As exhaustively comparing each pair of image pixels is very expensive, [Cheng et al. 2011] propose a *histogram-based contrast method* with some additional smoothing to minimize quantization artifacts. They also propose segmenting the image first into regions of small contrast to further minimize the computational effort. Recently, [Perazzi et al. 2012] introduced a method called *Saliency Filters*. They first cluster all pixels of the image into elements of similar color and then apply high-dimensional Gaussian filtering to measure the uniqueness and the spatial distribution of each element. To get the saliency of every pixel, they combine the uniqueness and spatial distribution of nearby elements using Gaussian weights. This way, they get a pixel-accurate saliency map, which can also compensate segmentation inaccuracies. As all these steps can be formulated as decomposable Gaussian blurrings, they derive an efficient implementation and at the same time produce saliency maps that are closer to the ground truth than any previous contrast-based method.

### 2.2.2. High-Level Object Detection

By using eye-tracking data, [Judd et al. 2009] learned that humans focus on common objects when asked to freely look at pictures. Text, people and specifically faces catch the viewer’s attention quickly. In the absence of such objects, humans focus on the center of the image and look for salient low-level features. So recognizing each of these objects, like text or faces, can help to compute saliency maps. [Viola and Jones 2004] proposed a *robust real-time face detection* method and methods for text detection are also well-known.

## 2. Related Work

To allow maximal flexibility, *manual saliency drawing* should be considered as well. Especially for the application in image retargeting, this is still important. In images where there are no objectively less important parts, manual saliency drawing can provide an intuitive way for an artist to control the deformation on a fine-grained level and to express personal preference.

## 2.3. Convex Optimization

[Chen et al. 2010] and [Panozzo et al. 2012] formulate the generation of a retargeted image as a quadratic optimization program. In this Section, we give a short introduction to convex optimization problems and numerical solvers for such problems.

### 2.3.1. Problem Formalization

Following the notation of [Boyd and Vandenberghe 2004], we specify a *mathematical optimization problem* as follows

$$\begin{aligned} & \text{minimize } f_0(x) \\ & \text{subject to } f_i(x) \leq b_i, \quad i = 1, \dots, m. \end{aligned}$$

We call  $x = (x_1, \dots, x_n)$  the *optimization variable*,  $f_0 : \mathbb{R}^n \rightarrow \mathbb{R}$  the *objective function*, the functions  $f_i : \mathbb{R}^n \rightarrow \mathbb{R}, i = 1, \dots, m$  are the *constraints functions* and the constants  $b_1, \dots, b_m$  are the *bounds* for the constraints. A vector  $x^*$  is called *optimal* if it has the smallest objective value among all vectors satisfying the constraints. We call the set of vectors that satisfy the constraints the *feasible region* of the problem.

An optimization problem is *quadratic* if its objective function can be written in the form

$$f_0(x) = x^T Q x + x^T b,$$

where  $Q$  is a symmetric matrix in  $\mathbb{R}^{n \times n}$  and  $b \in \mathbb{R}^n$ .

The optimization problem is *convex* if

$$\begin{aligned} & f_i(\alpha x + \beta y) \leq \alpha f_i(x) + \beta f_i(y) \quad \forall x, y \in \mathbb{R}^n \text{ and } \forall \alpha, \beta \in \mathbb{R} \\ & \text{with } \alpha + \beta = 1 \text{ and } \alpha, \beta \geq 0, \forall i \in \{0, 1, \dots, m\}. \end{aligned}$$

In the case of a quadratic problem, this means that  $Q$  needs to be positive semidefinite. If  $f_0$  is bounded from below on the feasible region and the feasible region is non-empty, a global minimizer exists. If additionally  $Q$  is positive definite, i.e.  $x^T Q x > 0 \quad \forall x \neq 0$ , the minimizer is unique.

### 2.3.2. Interior Point Method

One algorithm to solve such convex quadratic optimization problems is the *interior point method*. According to [Dantzig and Thapa 2003], this method was invented by John van Neumann. The

main idea is to incorporate the convex constraints into the objective function. So whenever any condition is violated, a near-infinity penalty should be added and whenever all conditions are satisfied, nearly no change to  $f_0$  should be made. This is achieved using so called *logarithmic barrier functions* that smoothly approximate the discontinuous constraint indicator function. The work of [Karmarkar 1984] shows that it is possible to approximate the solution efficiently, meaning that the number of iterations needed is polynomial in the dimension  $n$  and the desired accuracy of the approximation.

### CVXGEN

[Mattingley and Boyd 2012a] presented a code generator that can generate a solver for any convex quadratic optimization problem of modest size. The solver uses the interior point method and the generated C-code is library-free and highly optimized, making it suitable for embedded real-time applications. The problem statement is formulated in a high level description language, which allows the generator to analyze the sparsity patterns of the matrices involved and optimize the generated code accordingly. We used CVXGEN to generate the solver for our iOS application.

### MATLAB Optimization Toolbox

The optimization toolbox in MATLAB also includes a solver for quadratic programs, called `quadprog`, where the interior point method can be chosen by setting the corresponding algorithm option. We used this solver only for testing purposes.

2. *Related Work*



# 3

## Algorithm

In this Chapter, we present our retargeting algorithm. The method minimizes the As-Similar-As-Possible energy on an axis-aligned grid, similarly to [Panozzo et al. 2012], by combining a bijective deformation and cropping. We also show how to regularize the result with a Laplacian term and how to apply it to high-resolution images using cubic B-spline upsampling.

We first introduce the retargeting operator without cropping in Section 3.1 and then we add cropping in Section 3.2.

In Section 3.2.1, we discuss possible other approaches and why they are not suitable for our specific scenario. Our cropping operator is simple, yet effective and allows an efficient implementation.

Finally, we summarize in Section 3.3 the automatic saliency detection methods we implemented. We use the magnitude of the gradient and a face detector to get a rough saliency map. Our main focus however lies on an intuitive interface for manual saliency painting, which will be discussed later in Section 4.2.

### 3.1. Retargeting Operator

We want to warp an input image  $I$  of width  $W$  and height  $H$  to a new image  $I'$  of width  $W'$  and height  $H'$ . To describe the warp, we lay a grid  $G$  of  $N$  columns and  $M$  rows over the image  $I$  and search for an optimal grid  $G'$  of the same number of rows and columns but over  $I'$ . We will use  $M = N$  in our implementation, as  $M \neq N$  is a reasonable choice only if we a priori know something about the aspect ratio of  $I$  or  $I'$ . The grid  $G$  is uniform, meaning that each column has width  $W/N$  and each row has height  $H/M$ . The retargeted grid  $G'$  will have the desired

### 3. Algorithm

size of  $W' \times H'$  and will be non-uniformly deformed to preserve salient content. We denote by  $r_s$  and  $r_t$  the aspect ratios in the source image  $I$  and the target image  $I'$ , so

$$r_s = \frac{W}{H} \text{ and } r_t = \frac{W'}{H'}.$$

We restrict the set of admissible grids  $G'$  to *axis-aligned deformations*, such that we can fully describe the grid by stating the height of each row and the width of each column. Therefore, the following  $M + N$  variables are sufficient to describe the position and extent of all the  $M \cdot N$  grid cells:

$$\begin{aligned} s^{\text{rows}} &= (s_1^{\text{rows}}, s_2^{\text{rows}}, \dots, s_M^{\text{rows}}) \\ s^{\text{cols}} &= (s_1^{\text{cols}}, s_2^{\text{cols}}, \dots, s_N^{\text{cols}}) \end{aligned}$$

We denote by  $s = (s^{\text{rows}}, s^{\text{cols}})^T \in \mathbb{R}^{M+N}$  the vector of the unknown heights and widths.

#### 3.1.1. Quadratic Convex Optimization Problem

Our energy formulation will lead to a quadratic convex optimization problem (QP) of this form:

$$\text{minimize } s^T Q s \tag{3.1}$$

$$\text{subject to } s_i^{\text{rows}} \geq \bar{H}_i^{\min}, \quad i = 1, \dots, M, \tag{3.2}$$

$$s_i^{\text{rows}} \leq \bar{H}_i^{\max}, \quad i = 1, \dots, M, \tag{3.3}$$

$$s_j^{\text{cols}} \geq \bar{W}_j^{\min}, \quad j = 1, \dots, N, \tag{3.4}$$

$$s_j^{\text{cols}} \leq \bar{W}_j^{\max}, \quad j = 1, \dots, N, \tag{3.5}$$

$$s_1^{\text{row}} + s_2^{\text{row}} + \dots + s_M^{\text{row}} = H', \tag{3.6}$$

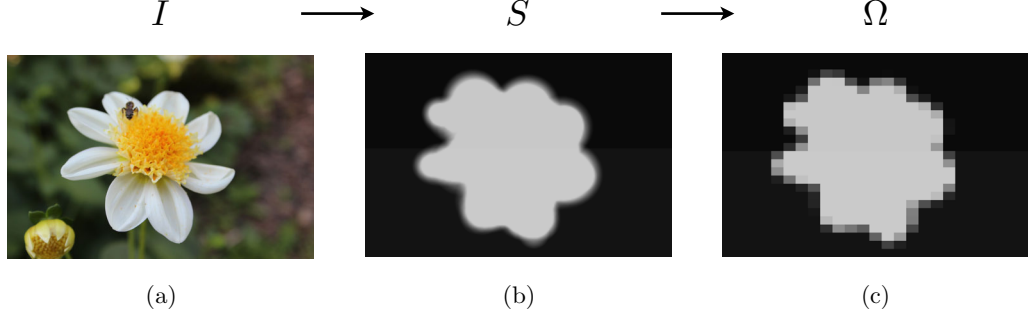
$$s_1^{\text{cols}} + s_2^{\text{cols}} + \dots + s_N^{\text{cols}} = W'. \tag{3.7}$$

The matrix  $Q \in \mathbb{R}^{(M+N) \times (M+N)}$  will describe the saliency-based energy. We will use the bounds  $\bar{H}_i^{\min}$ ,  $\bar{H}_i^{\max}$ ,  $\bar{W}_j^{\min}$  and  $\bar{W}_j^{\max}$  to restrict the size of each row and column. By choosing  $\bar{H}_i^{\min} > 0$  and  $\bar{W}_j^{\min} > 0$  for all non-cropped rows and columns, the inequalities (3.2) and (3.4) guarantee that  $G'$  will not contain fold overs. The cell  $(i, j)$  will be of size at least  $\bar{W}_j^{\min} \times \bar{H}_i^{\min}$ . As we use bilinear interpolation inside each cell, no part of  $I'$  will degenerate and discontinuities are impossible. The equality constraints (3.6) and (3.7) fix the target size  $W' \times H'$ .

Compared to the formulation in [Panozzo et al. 2012], we simplified the objective function by omitting the additive term  $s^T b$ . In our energy formulation,  $b$  would be the zero vector regardless of all parameters. We added more detailed boundary constraints as we want to control the size of each row or column individually. We can also bound them from above, which we will need when we decide to crop some of them.

#### 3.1.2. Energy Formulation

We base the energy formulation on the pixel-accurate *saliency map*  $S : I \rightarrow [0, 1]$ . This map measures how important each image location is. It tells us if we should try not to deform the



**Figure 3.1.:** (a) The original photo  $I$ . (b) Manually painted saliency map  $S$  on  $I$ . (c) The energy formulation only considers the rasterized saliency matrix  $\Omega$ .

region around around the pixel or if we are free to stretch it as we please. Figure 3.1 (b) shows such a saliency map. The discretized energy only considers the integrated values over each grid cell and therefore, we can reduce the relevant saliency values to the *saliency matrix*  $\Omega \in \mathbb{R}^{M \times N}$  (Figure 3.1 (a)). In our piecewise-linear deformation, these matrix entries are simply the cell averages:

$$\Omega_{i,j} = \frac{M \cdot N}{W \cdot H} \sum_{x=\lfloor (i-1) \cdot H/M \rfloor}^{\lfloor i \cdot H/M \rfloor} \sum_{y=\lfloor (j-1) \cdot W/N \rfloor}^{\lfloor j \cdot W/N \rfloor} S(x, y)$$

### ASAP Energy

We want to preserve salient regions of the image by only scaling them uniformly and avoiding every other transformations. In other words, we want the warp to locally correspond to a similarity transform. A salient part can then be translated, proportionally scaled, mirrored or rotated. Note that our deformation space does not allow the latter two. In our setting, the *As-Similar-As-Possible* energy measures only non-uniform scaling:

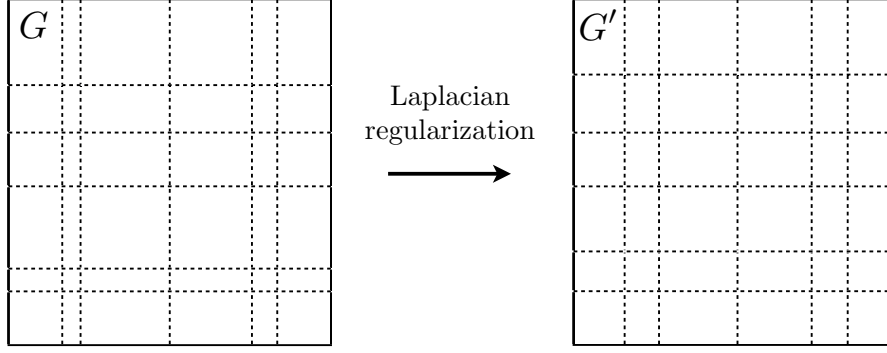
$$E_{\text{ASAP}} = \sum_{i=1}^M \sum_{j=1}^N \left( \Omega_{i,j} \left( \frac{M}{H} s_i^{\text{rows}} - \frac{N}{W} s_j^{\text{cols}} \right) \right)^2.$$

The two factors  $M/H$  and  $N/W$  take the aspect ratio of the cells in the original grid  $G$  into account. They are needed since in general the cells will not be square. To convert the energy in the QP form of Equation 3.1, we rewrite the summation as an inner product of a matrix  $K \in \mathbb{R}^{(MN) \times (M+N)}$ :

$$K_{k,l} = \begin{cases} \Omega_{r(k),c(k)} \frac{M}{H} & \text{if } l = r(k), \\ -\Omega_{r(k),c(k)} \frac{N}{W} & \text{if } l = M + c(k), \\ 0 & \text{otherwise,} \end{cases}$$

for the row  $r(k) = \lceil k/N \rceil$  and column  $c(k) = ((k-1) \bmod N) + 1$ . With increasing  $k$ , we go through  $\Omega$  row after row and set (at most) two non-zero entries in each row of  $K$ . Each entry in  $Ks$  is then the square root of some summand of  $E_{\text{ASAP}}$ . Therefore  $E_{\text{ASAP}} = (Ks)^T (Ks) = s^T K^T K s$ .

### 3. Algorithm



**Figure 3.2.:** The Laplacian regularization minimizes differences in width and height of neighboring columns and rows.

### Regularization

To control the smoothness of the resulting deformation, we add a *Laplacian regularization* term. This will ensure that neighboring rows or columns have a similar height resp. width. We again use a sum of squared differences:

$$E_{\text{reg}} = \sum_{i=1}^{M-1} \frac{M}{H} (s_{i+1}^{\text{rows}} - s_i^{\text{rows}})^2 + \sum_{j=1}^{N-1} \frac{N}{W} (s_{j+1}^{\text{cols}} - s_j^{\text{cols}})^2$$

which we can write in matrix form  $(Ls)^T(Ls) = s^T L^T L s$  for a matrix  $L \in \mathbb{R}^{(M+N) \times (M+N)}$  with

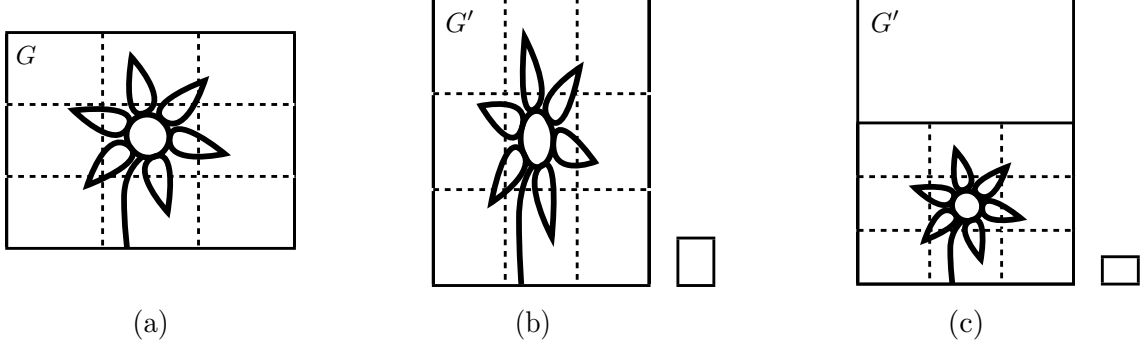
$$L_{kl} = \begin{cases} \frac{M}{H} & k = l \text{ and } 1 \leq k \leq M \\ \frac{M}{H} & k = l - 1 \text{ and } 1 \leq k \leq M \\ \frac{N}{W} & k = l + M \text{ and } 1 \leq k \leq N \\ \frac{N}{W} & k = l - 1 + M \text{ and } 1 \leq k \leq N \end{cases}$$

Since manually painted saliency maps often include sharp edges, the ASAP minimum could contain abrupt changes. The Laplacian regularization helps to distribute the deformation more evenly across the image. This is illustrated in Figure 3.2. The Laplacian is minimized by homogeneous scaling. Therefore we can think of the regularization term as a way to blend between the ASAP deformation and homogeneous scaling. To do this we introduce a weighting factor  $w_{\text{reg}}$ , which completes our energy formulation:

$$E = (1 - w_{\text{reg}})E_{\text{ASAP}} + w_{\text{reg}}E_{\text{reg}} \quad (3.8)$$

$$= (1 - w_{\text{reg}})(s^T K^T K s) + w_{\text{reg}} s^T L^T L s \quad (3.9)$$

$$= s^T \underbrace{(1 - w_{\text{reg}})(K^T K) + w_{\text{reg}} L^T L}_{=: Q} s \quad (3.10)$$



**Figure 3.3.:** Minimum cell size constraints. (a) The image in its original aspect ratio. (b) The uniform minimum cell size as used in [Panozzo et al. 2012]. (c) The aspect-ratio-aware minimum cell size we propose. The two smaller boxes represent the  $L$ -downscaled minimum cell sizes for  $L = 0.5$ .

### 3.1.3. Boundary Constraints

As long as we do not crop parts of the image, we use homogeneous boundary constraints as in [Panozzo et al. 2012], which means that we choose some  $H^{\min}$  and  $W^{\min}$  such that

$$\begin{aligned} i &= 1 \ggg M : H_i^{\min} = H^{\min} \\ &H_i^{\max} = H \\ j &= 1 \ggg N : W_j^{\min} = W^{\min} \\ &W_j^{\max} = W \end{aligned}$$

Now we can set  $H^{\min}$  and  $W^{\min}$  to define the minimal size of each cell. This gives strict control over the flexibility of the warp and ensures the absence of fold overs and discontinuities. While  $H^{\min} = H/M$  and  $W^{\min} = W/N$  would force homogenous scaling, setting  $H^{\min} = 0$  and  $W^{\min} = 0$  would allow cells to be cropped and could introduce degenerated cells and thus discontinuities. So we want to go for something in between.

[Panozzo et al. 2012] choose a minimum cell size, which we will call *uniform minimum cell size*. It is defined as

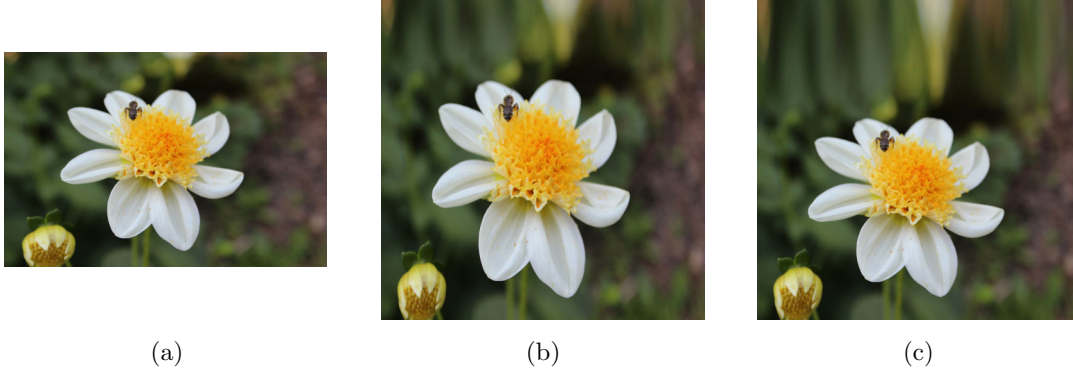
$$H^{\min} = L \cdot H/M \text{ and } W^{\min} = L \cdot W/N \text{ for some } L \in (0, 1)$$

where we call  $L$  the *minimum cell size scaling factor*.

This forces each cell to respect a minimum cell size, which is in the aspect ratio  $W/H \cdot M/N$  of the retargeted image  $I$ . This bound has no explicit meaning as each cell originates from a part of the original image. In the original image the cell was in a different aspect ratio  $W/H \cdot M/N$ . The uniform minimum cell size forces the warp to respect a minimum image size that corresponds to a uniform homogeneous scaling to the target ratio. This is illustrated in Figure 3.3 (b).

For different source and target ratios and a large  $L$  this can force the warp to unnecessarily stretch salient parts of the image. Furthermore this uniform minimum cell size would be problematic in our cropping approach, as we will see later in Section 3.2.2. We propose a different minimum cell size, which we call *aspect-ratio-aware*. We first proportionally scale  $G$  to fit into

### 3. Algorithm



**Figure 3.4.:** Comparison between the two types of cell size constraints using  $L = 0.9$ . (a) The image in its original aspect ratio. (b) The uniform minimum cell size as used in [Panozzo et al. 2012]. (c) The aspect-ratio-aware minimum cell size we propose. Both images use the same saliency map as in Figure 3.1.

$G'$  and only then apply some scaling factor  $L$ . This way the minimum cell size keeps its original aspect ratio, as illustrated in Figure 3.3 (c).

To explicitly calculate the size of this aspect-ratio-aware minimum cell, we condition on the side of  $I'$  that is smaller compared to  $I$  and then apply proportional scaling to fit into  $G'$ . This formula is given in Algorithm 1.

---

**Algorithm 1** Determine the minimum cell size

---

```

if  $r_t > r_s$  then                                 $\triangleright$  The height of  $I'$  is the limiting factor.
     $\bar{H}^{\min} \leftarrow \frac{H'}{M} \cdot L$ 
     $\bar{W}^{\min} \leftarrow s_r \cdot \frac{H'}{N} \cdot L$             $\triangleright \bar{W}^{\min} = (s_r \frac{M}{N}) \cdot \frac{H'}{M} \cdot L = s_r \cdot \frac{H'}{N} \cdot L$  and still  $\bar{W}^{\min} \leq W'/N$ .
else
     $\bar{W}^{\min} \leftarrow \frac{W'}{N} \cdot L$ 
     $\bar{H}^{\min} \leftarrow \frac{1}{s_r} \cdot \frac{W'}{M} \cdot L$             $\triangleright \bar{H}^{\min} = \frac{1}{s_r(M/N)} \cdot \frac{W'}{N} \cdot L = \frac{1}{s_r} \cdot \frac{N}{M} \cdot \frac{W'}{N} \cdot L = \frac{1}{s_r} \cdot \frac{W'}{M}$ 
end if

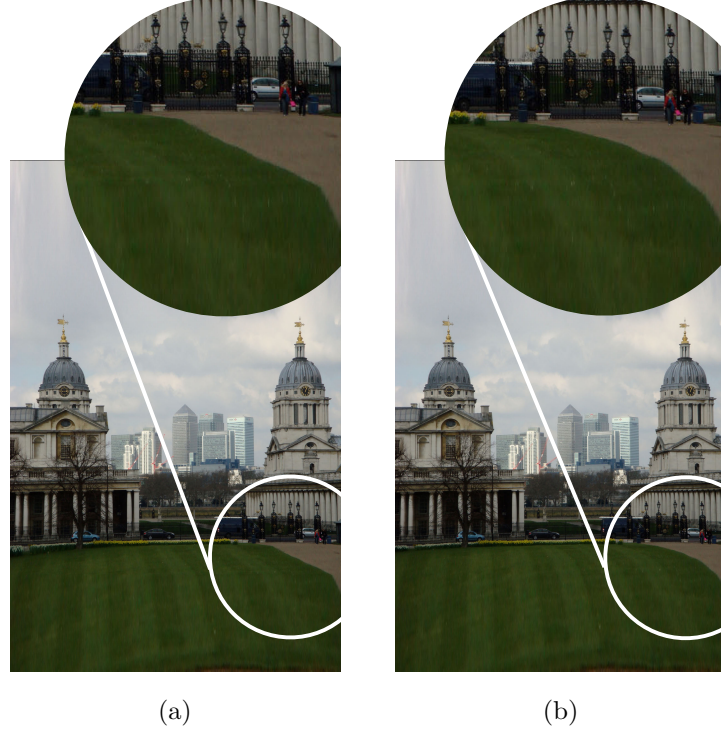
```

---

The difference between these two constraints becomes apparent for  $L$ -values close to one. As [Panozzo et al. 2012] used  $L = 0.2$  for their benchmark and as default value, the choice between the two constraints does not change much. We will use bigger  $L$ -values later on, where the two bounds can clearly be distinguished. Figure 3.4 compares the two bounds for  $L = 0.9$  and visualizes the distortion caused by the uniform minimum cell size that we prevent with our new approach.

### Existence of a Feasible Solution

In order for the quadratic program (QP) to be convex,  $Q$  needs to be positive semi-definite. As this is the case for both the ASAP-matrix  $K^T K$  and the Laplacian  $L$ , their weighted sum  $Q$  is also positive semi-definite. The feasible region is non-empty as long as all  $\bar{W}_j^{\min} \leq W'/M$  and  $\bar{H}_i^{\min} \leq H'/M$  and for at least one  $\bar{W}_j^{\max} = W'$  and one  $\bar{H}_i^{\max} = H'$ . With the homogeneous boundary conditions above, homogeneous scaling is a feasible solution. In the cropping case



**Figure 3.5.:** (a) Retargeted image using bilinear interpolation on a  $25 \times 25$  grid. One can clearly see the irregularities at the border of the lawn. (b) Resulting image after spline upsampling from a  $25 \times 25$  grid to a grid of  $75 \times 75$  cells.

below, homogeneous scaling in the non-cropped rows and columns will also be feasible. Therefore, a minimal solution exists as the objective function is bounded on the feasible domain.

### 3.1.4. Spline Interpolation

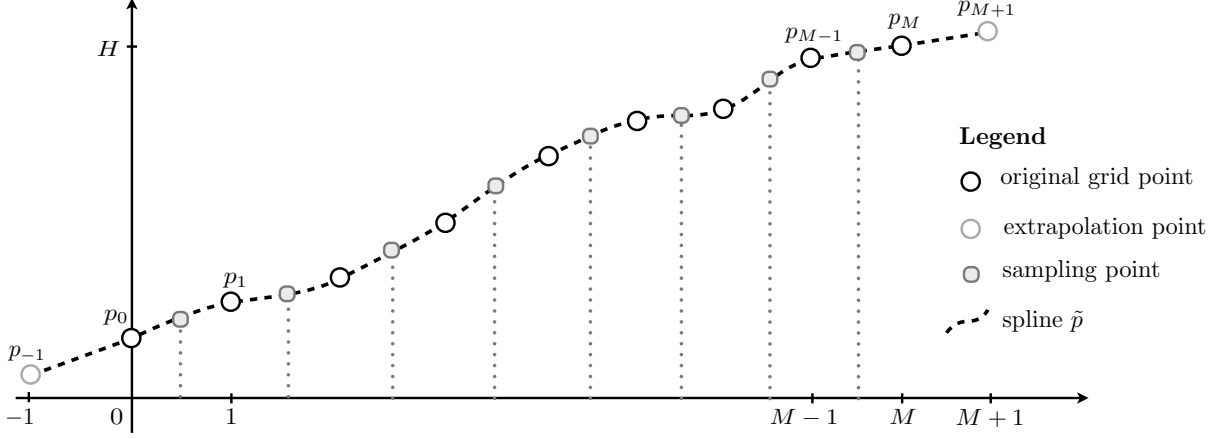
[Panozzo et al. 2012] noted that a coarse resolution of the grid, like  $N = M = 25$ , is enough to approximate the continuous warp very well in most cases. But, as the bilinear interpolation is not smooth across the grid cell borders, we can sometimes see smoothness artifacts in high resolution images, like the ones in Figure 3.5 (a).

To remove these artifacts, we use B-spline interpolation to upsample the retargeted grid. We can separately interpolate the rows and columns in one dimension each, using cubic B-spline interpolation. We start by calculating the cumulative sums of the row heights  $s^{\text{rows}}$  and column widths  $s^{\text{cols}}$ :

$$p_i^{\text{rows}} = \sum_{k=1}^i s_k^{\text{rows}}, \quad i = 0, \dots, M \quad \text{and} \quad p_j^{\text{cols}} = \sum_{k=1}^j s_k^{\text{cols}}, \quad j = 0, \dots, N.$$

Without loss of generality, we now show how to interpolate the row border positions  $p := p^{\text{rows}} = (p_0^{\text{rows}}, p_1^{\text{rows}}, \dots, p_M^{\text{rows}})$ . Between two such grid points  $p_i$  and  $p_{i+1}$ , we use the interpolation parameter  $t \in [0, 1]$  for the interpolation function  $\tilde{p}$  and equally weigh each of the four nearest points. This is called *piece-wise uniform cubic B-spline interpolation* and corresponds

### 3. Algorithm



**Figure 3.6.:** Uniform cubic B-spline upsampling in one dimension to get a refined grid of resolution  $2M$ .

to

$$\tilde{p}(t) = \begin{pmatrix} t^3 & t^2 & t^2 & t \end{pmatrix} \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix} \begin{pmatrix} p'_{i-1} \\ p_i \\ p_{i+1} \\ p'_{i+2} \end{pmatrix}. \quad (3.11)$$

We have to carefully handle the two corner cases, where we will just linearly extrapolate  $p$  in order not to introduce any new discontinuities:

$$p'_{i-1} = \begin{cases} p_{i-1} & \text{if } i-1 \geq 0 \\ p_0 - (p_1 - p_0) = 2p_0 - p_1 & \text{otherwise, so if } i-1 = -1 \end{cases}$$

$$p'_{i+2} = \begin{cases} p_{i+2} & \text{if } i+2 \leq M \\ p_M + (p_M - p_{M-1}) = 2p_M - p_{M-1} & \text{otherwise, so if } i+2 = M+1 \end{cases}$$

Figure 3.6 illustrates the uniform cubic B-spline interpolation. To refine our retargeted grid, we sample the spline  $\tilde{p}$  for denser positions and obtain a new grid of arbitrary resolution. As cubic B-splines fulfill the variation diminishing property, the upsampled grid will also be foldover-free. The quality improvement can be clearly seen in Figure 3.5 (b).

## 3.2. Cropping

In situations where the image is either full of salient content or some regions along the border do not contain any salient parts, additional cropping might be preferable to plain non-uniform warping. In this Section, we present our cropping extension to the axis-aligned image retargeting operator. We also discuss other approaches that we considered and their shortcomings.





**Figure 3.7.:** *If we set the lower bound column width to zero, then also columns can be removed that are not at the border. This introduces highly visible artifacts. (a) The image in its original aspect ratio with the painted saliency map. (b) The retargeted image where the optimization removed several columns in the middle completely.*

### 3.2.1. General Approaches

#### Integrate Cropping into the Energy Term

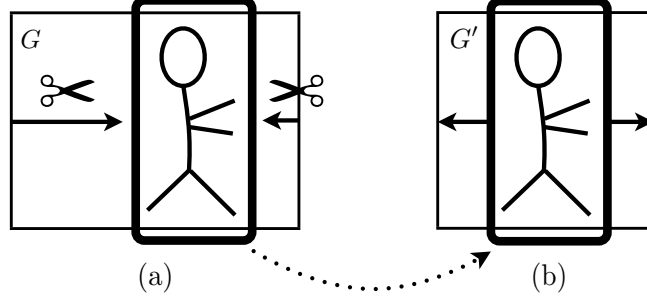
We use a variational method to retarget the image in Section 3.1, so the most natural approach would be to incorporate cropping directly into the energy term. We could add  $N + M$  binary variables  $c_1, \dots, c_{N+M}$ , one for each row and column of the grid to indicate which parts of the image will be cropped. But this would chain the variables together, as we would have conditions of the form “only allow  $s_i = 0$  if already  $s_{i-1} = 0$ ”, which we can not express in our optimization framework. Note that simply removing the lower bounds of the cell sizes does not help here since this would allow the removal of every row or column, independent of whether it is adjacent to the border or not. Removing a row that is not on the border will introduce a discontinuity as shown in Figure 3.7 (b).

It is thus non-trivial to directly include the cropping into our energy term while keeping the structure of the constraints fixed. This is very important in order to achieve high efficiency in the optimization, as discussed in Section 4.1.

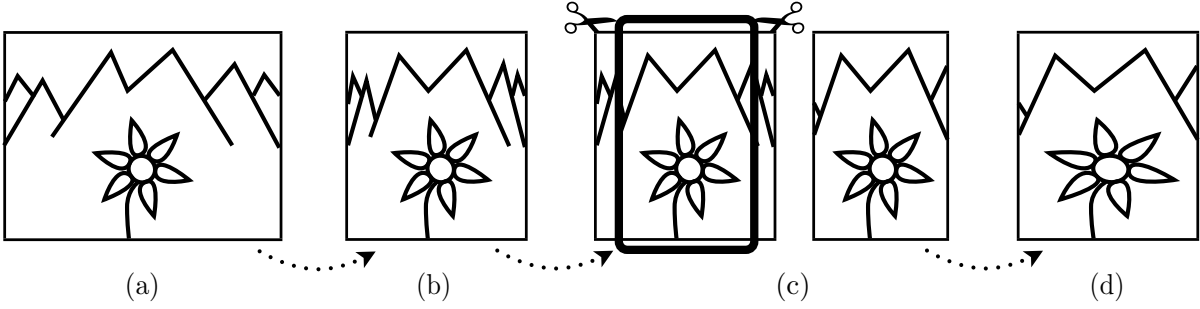
#### A Priori Cropping

Heuristics could be used to decide what to crop before minimizing the energy (Equation 3.8). For instance, we could crop along the side we have to decrease to get an image whose aspect ratio is closer to the target ratio. As indicated in Figure 3.8 we could crop simultaneously from both sides as long as we do not hit any (large) salient parts. However, we might end up having cropped more than necessary and we might only realize this after the subsequent optimization. In general, such *a priori* methods lack the knowledge gained by the energy minimization when deciding what to crop.

### 3. Algorithm



**Figure 3.8.:** A priori cropping. (a) We crop until we hit a salient object. (b) We might crop more than necessary and then need to stretch the remaining part during the energy optimization.



**Figure 3.9.:** A posteriori cropping. Starting with the original image (a) we find the optimal warp (b). We decide what to crop (c) and then scale the remaining image parts uniformly to fill the target rectangle (d).

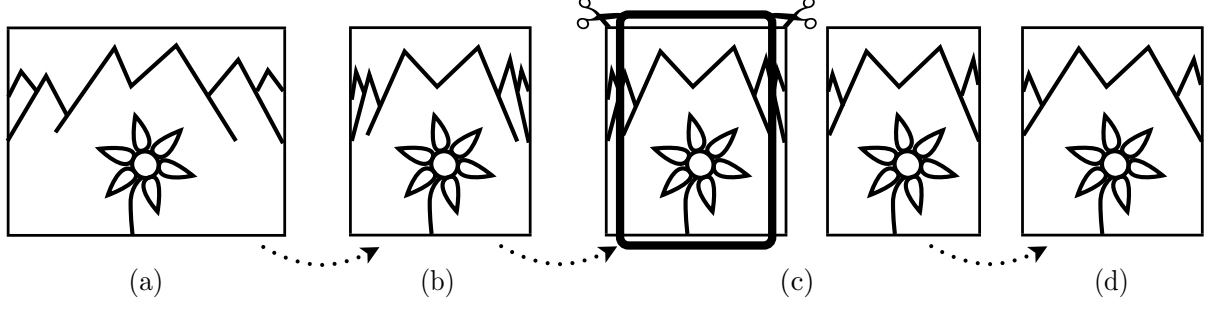
#### A Posteriori Cropping

Another possible approach would be to first run the energy minimization and then decide what to crop based on the optimized grid. For this, we could start at each of the four borders and decide to crop rows and columns whose heights and widths are close to the minimum sizes  $\bar{H}^{\min}$  [ $\bar{W}^{\min}$ ]. After that we could rescale the remaining part of the image to fill the target rectangle  $R'$  again. These three steps are visualized in figure 3.9.

Since only small border parts get cropped, hopefully only minimal rescaling is needed at this point. However, we cannot give any quality guarantees and the final warp is not an energy minimizer anymore. Therefore, what we propose is performing another energy optimization on the part of the image that is left after cropping.

#### 3.2.2. Threshold-Based Cropping

Our cropping operator will solve two similar quadratic programs. Let us assume that the saliency matrix  $\Omega$ , the regularization factor  $w_{\text{reg}}$  and the minimum cell size  $\bar{W}^{\min} \times \bar{H}^{\min}$  are fixed. In the first minimization, we detect which rows and columns would be squeezed below the minimum cell size if we imposed softer lower bound constraints. In other words, we want to detect what the optimization would shrink more if it were allowed to do so. For this, we choose a threshold factor  $\alpha \in (0, 1)$  and then optimize with a downscaled minimum cell size of



**Figure 3.10.:** Our threshold-based cropping approach. Starting with the original image (a) we find the optimal warp (b) using the minimum cell size  $\bar{W}^{\text{crop}} \times \bar{H}^{\text{crop}}$ . We decide what to crop (c) and then optimize the remaining image parts again to fill the target rectangle (d).

$\bar{W}^{\text{crop}} \times \bar{H}^{\text{crop}} := (\alpha \bar{W}^{\min}) \times (\alpha \bar{H}^{\min})$ . We call the resulting first grid  $G'_1$ .

In  $G'_1$ , we now crop each row that is smaller than  $\bar{H}^{\min}$  starting from the top and the bottom, and by starting from the left and right border, we crop each column that is narrower than  $\bar{W}^{\min}$ . We denote the number of cropped rows and columns on each side by  $c_t$ ,  $c_b$ ,  $c_l$  and  $c_r$ . So we are left with a grid of  $(M - c_t - c_b)$  rows and  $(N - c_l - c_r)$  columns.

For the second application of the convex solver, we now only consider the non-cropped parts of the image. Thanks to the element-wise boundary condition in our optimization problem, we can easily reflect our cropping decision in the constraints. We just set  $\bar{H}_i^{\min} = \bar{H}_i^{\max} = 0$  for all cropped rows and  $\bar{W}_j^{\min} = \bar{W}_j^{\max} = 0$  for all cropped columns. On the non-cropped part, we use the original boundary constraints of  $\bar{W}^{\min} \times \bar{H}^{\min}$  and only upper bound the rows and columns by the total width  $W'$  and height  $H'$  as in the standard operator. By solving this adopted QP, we get a second grid  $G'_2$ . Using the mapping from  $G$  to  $G'_2$ , we sample the final retargeted image  $I'$ .

Figure 3.10 visualizes this threshold-based cropping technique and Algorithm 2 summarizes our method. In Figure 3.11, we show how the saliency map defines the parts of the image that can get cropped and that we can largely reduce the deformation in the unsalient part.

## Boundary Constraints

With our cropping approach in mind, it becomes clear why it is important that the minimum cell size and also its  $\alpha$ -downscaled variant, which we use in the first optimization, are of the same aspect ratio as the cells in  $G$ . Figure 3.12 illustrates how the salient part of an image would get cropped if we used a minimum cell size in the ratio of the target rectangle.

### 3.2.3. Implementation

When implementing this cropping algorithm, we can use exactly the same  $N \times M$  grid QP solver in both optimizations. This is essential when using CVXGEN to generate an optimized solver for a single fixed problem size. What we do need to change however, is how we construct the Laplacian regularization matrix  $L$  and how we use spline upsampling to refine the grid. Both

### 3. Algorithm

---

**Algorithm 2** Threshold-based cropping
 

---

**Input:**  $I, W', H', \Omega, w_{\text{reg}}, \bar{W}^{\min} \times \bar{H}^{\min}$

**Output:**  $I'$

**Procedure:**

$G'_1 \leftarrow$  solution of optimization with minimum cell size  $\bar{W}^{\text{crop}} \times \bar{H}^{\text{crop}}$

$(s^{\text{rows}}, s^{\text{cols}}) \leftarrow G'_1$   $\triangleright$  Extract the size of the rows and columns from this first grid.

$(c_t, c_b, c_l, c_r) \leftarrow (0, 0, 0, 0)$

**while**  $s_{c_t+1}^{\text{rows}} < \bar{H}^{\min}$  **do**  $\triangleright$  Crop vertically.

$c_t \leftarrow c_t + 1$

**end while**

**while**  $s_{M-c_b}^{\text{rows}} < \bar{H}^{\min}$  **do**

$c_b \leftarrow c_b + 1$

**end while**  $\triangleright$  Not all rows will get cropped as they sum up to at least  $M \cdot \bar{H}^{\min}$ .

**while**  $s_{c_l+1}^{\text{cols}} < \bar{W}^{\min}$  **do**  $\triangleright$  Crop horizontally.

$c_l \leftarrow c_l + 1$

**end while**

**while**  $s_{N-c_r}^{\text{cols}} < \bar{W}^{\min}$  **do**

$c_r \leftarrow c_r + 1$

**end while**  $\triangleright$  Not all columns will get cropped as they sum up to at least  $N \cdot \bar{W}^{\min}$ .

$G'_2 \leftarrow$  solution of optimization with minimum cell size  $\bar{W}^{\min} \times \bar{H}^{\min}$  for non-cropped cells

$I' \leftarrow$  sample from  $I$  using the mapping  $G \rightarrow G'_2$

---



(a)

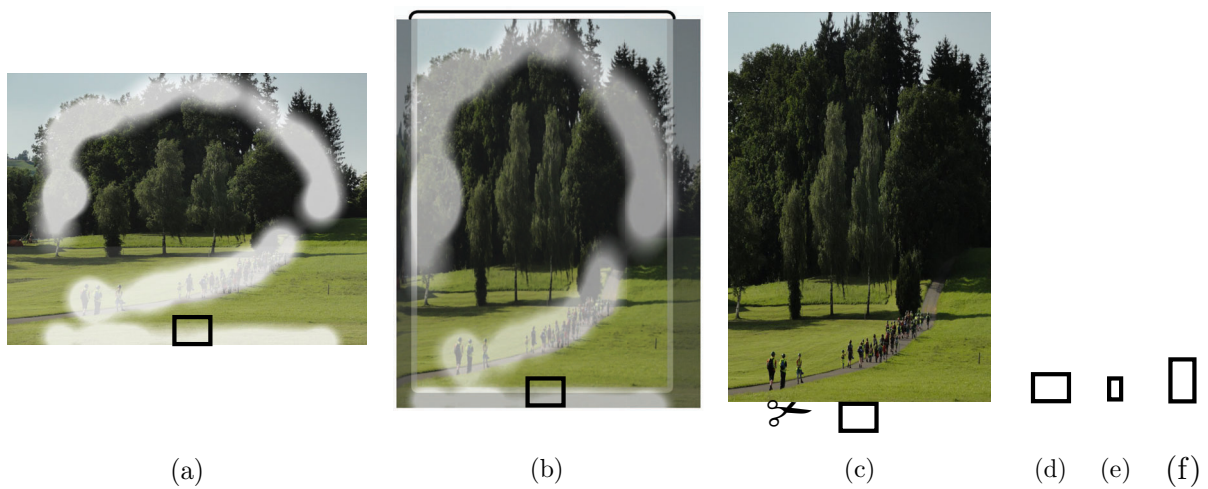


(b)



(c)

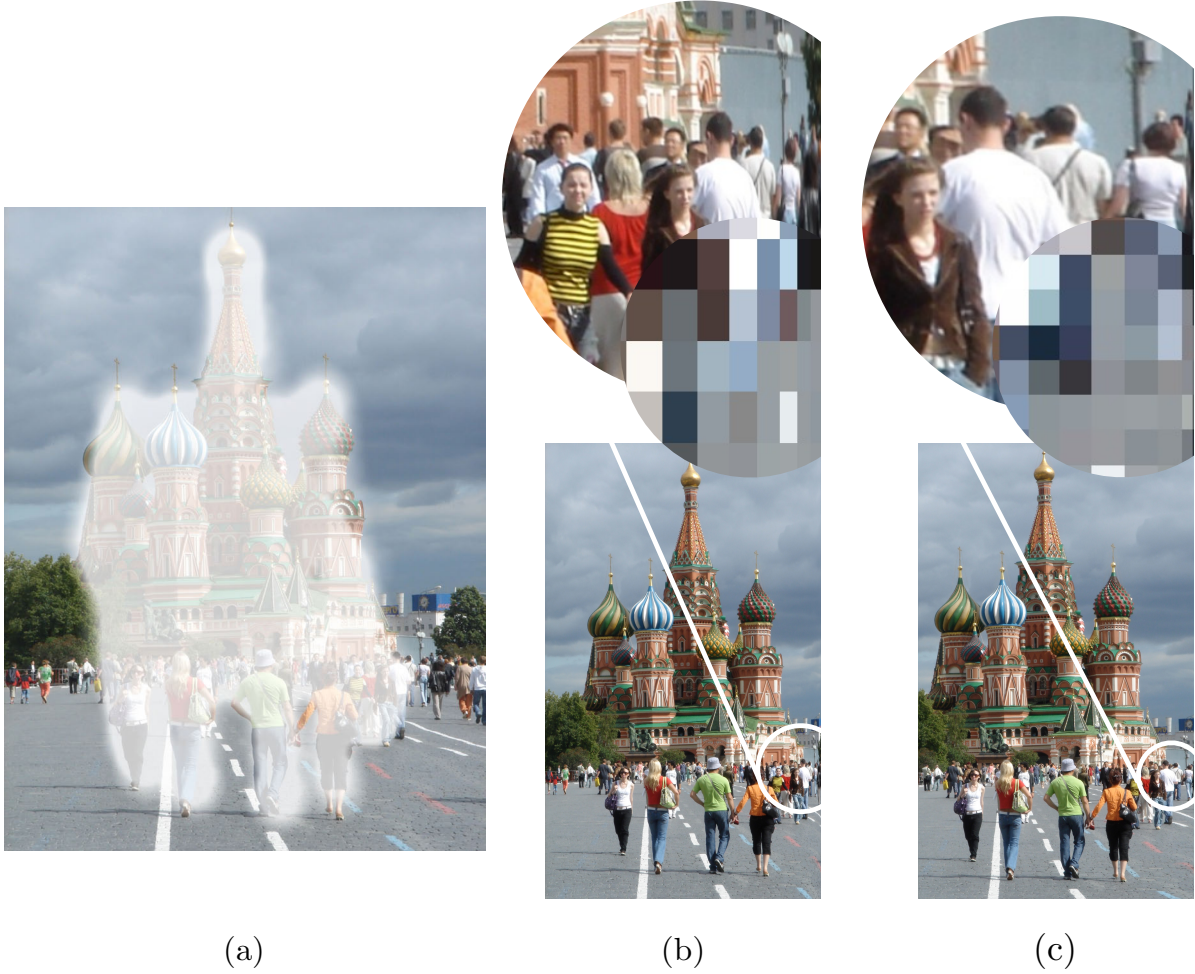
**Figure 3.11.:** We can influence the cropping decision by drawing into the saliency map. This way we can select whether only the cow (b) or also the hills in the background (c) should be kept. We observe that the more we can crop, the less deformation in the unsalient parts is needed.



**Figure 3.12.:** This image illustrates the problems that arise when we use the uniform minimum cell size. (a) The original image with saliency painted on top of it. Note the salient cell marked at the bottom of the image. (b) In the retargeted image the salient cell keeps its original aspect ratio. (c) But if we use the uniform minimum cell size the salient cell will get cropped. To see why this happens we take a look at the three sizes involved in the decision. The size of the retargeted cell (d) after the first optimization, the crop size (e) and the minimum cell size (f). Remember that the crop size is used as the minimum cell size in the first optimization. Therefore by keeping its aspect ratio the optimized cell (d) can have a height below the cropping threshold, which is the height of the minimum cell (f). So the row containing (d) gets cropped even though it is salient and even though the image is resized horizontally.



### 3. Algorithm



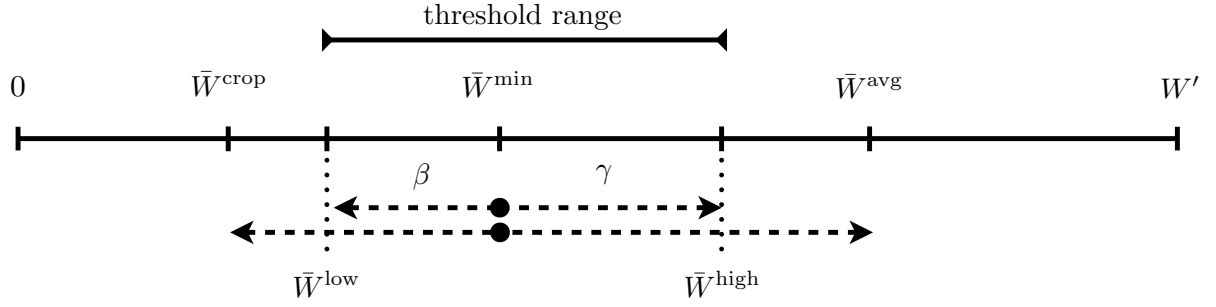
**Figure 3.13.:** Crop awareness. (a) The original image with the saliency map on top of it. (b) Wrong crop-unaware implementation of the Laplacian regularization. The columns near the border get unnecessarily thinned out. The detail view shows the underlying rectangular grid. (c) Wrong non crop-aware implementation of the cubic B-spline upsampling. Also here, the columns near the border get thinner and even two cropped columns become very thin lines again.

techniques must be modified to be aware of the cropping.

For the regularization matrix  $L$  for the second optimization, this means that we should consider neighboring pairs of rows and columns only if both of them are not cropped. Otherwise this would lead to thinning effects at the cropping border as can be seen in Figure 3.13 (b).

The same also applies to the uniform cubic B-spline upsampling, where we should only take the non-cropped segment into account. If we wrongly interpolated along all positions, including the cropped ones, the last cropped segments would appear as thin lines. Such artifacts can be seen in the wrongly upsampled image in Figure 3.13 (c).

If implemented correctly, none of these artifacts appear. In Section 4.2.2, we will go into detail on how we present the cropping decision to the user and how the user can interactively refine the cropping decision just by painting or removing saliency.



**Figure 3.14.:** All the limits involved in defining the crop-threshold range.  $\beta$  interpolates between  $\bar{W}^{\text{min}}$  and  $\bar{W}^{\text{crop}}$  and  $\gamma$  between  $\bar{W}^{\text{min}}$  and  $\bar{W}^{\text{avg}}$ .

### 3.2.4. Crop-Treshold Range

When dynamically changing the aspect ratio  $r_t$  of our retargeted image  $I'$ , we can observe that our cropping algorithm so far often does not crop one column after the other, step by step. Instead, the decision to crop several columns is often made over a very small change in aspect ratio. This makes perfect sense considering the underlying energy minimization. The optimization tries to keep equally salient parts equally deformed, as this minimizes the quadratic ASAP energy term. Additional regularization further encourages neighboring cells to have the same size. This means however, that neighboring cells also reach the threshold width at roughly the same aspect ratio and therefore will get cropped roughly at the same time.

Thus, the operator does not provide a smooth cropping experience. We relax the fixed cropping threshold by introducing a *crop-threshold range*. The operator should decide to crop the first column even if its width is still bigger than  $\bar{W}^{\text{min}}$ , somewhere between the average column width and the minimum column width  $\bar{W}^{\text{min}}$ . But later on, we want to crop additional columns only if they are clearly narrower than  $\bar{W}^{\text{min}}$ . In the end, the column would need to be as narrow as the first optimization allows, so  $\bar{W}^{\text{crop}}$ .

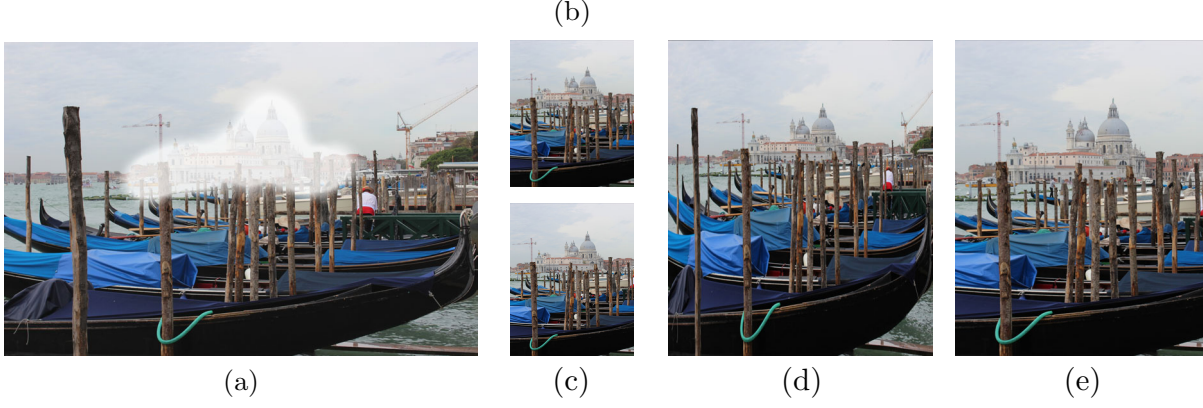
To formalize this, we introduce two new parameters  $\beta, \gamma \in [0, 1]$  to control the extent of this threshold range and define the following limits:

|   |   |
|---|---|
| the aspect-ratio-aware average cell width                               | $\bar{W}^{\text{avg}} = \begin{cases} \frac{W'}{N} & \text{if } r_t > r_s \\ s_r \frac{H'}{M} & \text{if } r_t \leq r_s \end{cases},$ |
| the minimum column width  | $\bar{W}^{\text{min}} = L \cdot \bar{W}^{\text{avg}},$  |
| the downsampled $\bar{W}^{\text{min}}$ we use in the first optimization | $\bar{W}^{\text{crop}} = \alpha \cdot \bar{W}^{\text{min}},$  |
| the lower end of the cropping-threshold range                           | $\bar{W}^{\text{low}} = \bar{W}^{\text{min}} - \beta (\bar{W}^{\text{min}} - \bar{W}^{\text{crop}}),$                                 |
| the upper end of the cropping-threshold range                           | $\bar{W}^{\text{high}} = \bar{W}^{\text{min}} + \gamma (\bar{W}^{\text{avg}} - \bar{W}^{\text{min}}).$                                |

In Figure 3.14, all these limits and the threshold range  $[\bar{W}^{\text{low}}, \bar{W}^{\text{high}}]$  are visualized.

When deciding whether to crop an additional  $(i+1)$ -th column, e.g. from the left after cropping columns 1 to  $i$ , we no longer just compare  $s_i$  with  $\bar{W}^{\text{min}}$  but use an interpolated threshold value

### 3. Algorithm



**Figure 3.15.:** We paint the saliency map that can be seen in the original image (a) and want to retarget the image from landscape to portrait. We want to compare our default crop-threshold range ( $\beta = \gamma = \frac{1}{2}$ , pictures (b) and (c)) with the fixed crop-threshold ( $\beta = \gamma = 0$ , pictures (d) and (e)). Note that pictures (b) and (d) have the exact same aspect ratio. Pictures (c) and (e) are only slightly taller. But in the case of the fixed crop-threshold we see a big crop-difference between (d) and (e). Near the border the columns in (d) are visibly narrower than in (a), but just not narrow enough to get cropped. In (e) however, these columns were cropped and the image looks fine. Our improved approach, the threshold-range, generates two almost identical, good-looking warps (b) and (c), just as desired.

from our threshold range  $[\bar{W}^{\text{low}}, \bar{W}^{\text{high}}]$ , so we crop column  $i + 1$  of width  $s_{i+1}$  if and only if

$$s_{i+1} < \left( \frac{i}{N} \cdot \bar{W}^{\text{low}} + \left( 1 - \frac{i}{N} \right) \cdot \bar{W}^{\text{high}} \right).$$

The cropping from the right is done in the same way. To decide whether we should crop  $i + 1$  columns, when starting from the last column  $N$ , we compare  $s_{N-i-1}$  with the same interpolated threshold. Note that none of these bounds depend on the target ratio  $r_t$ , such that we can prevent the problems we outlined in Section 3.2.2 and shown in Figure 3.12. Without loss of generality all of the above also applies to cropping rows.

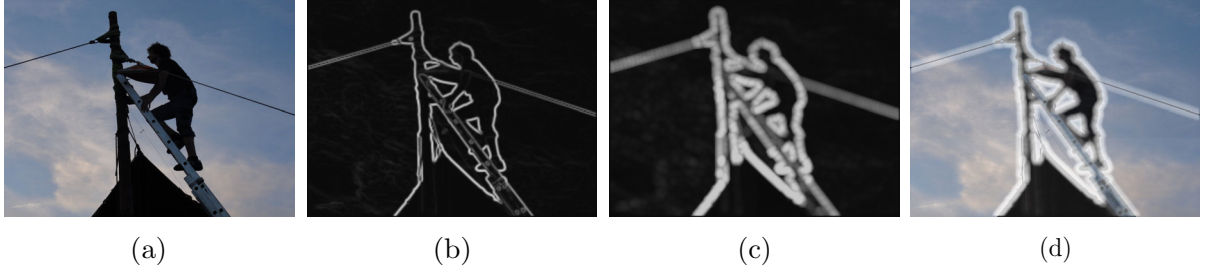
We found that choosing a cell size factor  $L = 0.7$ , a crop-threshold ratio  $\alpha = 0.5$  and additional parameters  $\beta = 0.5$  and  $\gamma = 0.5$  are appropriate default values. Note that our choice for the minimum cell size factor  $L$  is with 0.7 considerably higher than in the crop-free method of [Panozzo et al. 2012], where they used a default value of 0.2. Choosing a bigger factor, and therefore less flexibility of the warp, in our method can be justified by the different definition of the minimum cell size and the additional cropping.

Compared to the fixed threshold  $\bar{W}^{\text{crop}}$ , this threshold range approach makes the cropping experience much more fluid and dynamic and significantly improves the retargeting quality. Therefore, it is an essential part of our operator. A direct comparison is given in Figure 3.15.

## 3.3. Saliency

We implemented two basic salient feature detection methods in our application. The former is a gradient magnitude filter while the latter is a face detector.





**Figure 3.16:** Gradient-based automatic saliency detection. (a) Original image. (b) Gradient magnitude after the convolution with the Sobel kernel along both axis. (c) Gradient magnitude after dilation operation. (d) Auto-saliency on top of the original image.

### 3.3.1. Gradient-Based Saliency Detection

We start by creating a copy  $\tilde{I}$  of the input image  $I$  and resizing it to a fixed size of 250 pixels along the longer direction. This helps reducing the computational effort and removes high-frequency noise from high-resolution images. Then we convert this image from RGB to grayscale using the conversion weights from the luminance  $Y$ -channel conversion in the  $YUV$  and  $YIQ$  color spaces

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B.$$

Next we convolve this image with the Sobel gradient kernel along both the  $x$ - and  $y$ -axis which creates two matrices  $G_x$  and  $G_y$  that contain the two components of the approximated gradient in each image point:

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} * \tilde{I}_{\text{gray}} \quad \text{and} \quad G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * \tilde{I}_{\text{gray}}.$$

Then we build the gradient magnitude array  $G$ , as  $G(x, y) = \sqrt{G_x(x, y)^2 + G_y(x, y)^2}$  and normalize and round it to the byte value range  $[0, 255]$ . Finally, we perform a dilation operation to emulate a brush-stroke-like effect. This also fills up small holes in the gradient map and looks similar to manually painting along each contrast-rich edge of  $I$ . Formally, we can define our dilation operation for some radius  $d$  as follows:

$$D(G)_{(x,y)} = \max_{dx, dy \text{ s.t. } \sqrt{dx^2 + dy^2} \leq d} G(x + dx, y + dy),$$

where we used  $d = 4$  in our implementation. Figure 3.16. illustrates the kind of saliency maps we get this way.

### 3.3.2. Face Detection

Since even slightly stretched faces are easily observable, detecting and marking faces as salient image parts is very helpful. Standard face detectors like the one by [Viola and Jones 2004] or

### 3. Algorithm



**Figure 3.17.:** Face detection. Given the rectangle, where the face was detected, we paint saliency inside a cubic ellipsoidal shape with a cubic intensity decay towards the border.

the one implemented in the iOS system frameworks return a set of axis-aligned rectangular subwindows, each assumed to contain a face. Such a rectangle roughly containing the mouth and both eyes extends from below the chin up to the hair line. To better approximate the form of a head we use a cubic ellipsoidal shape. For a rectangle with center  $(c_x, c_y)$ , width  $r_x$  and height  $r_y$ , this is the following set

$$\left\{ (x, y) \in \mathbb{R}^2 \mid \underbrace{\left| \frac{x - c_x}{r_x} \right|^3 + \left| \frac{y - c_y}{r_y} \right|^3}_{=: r} \leq 1 \right\}.$$

To further provide a brush-like look we let the intensity slightly decay towards the border. We use again a cubic form. For a pixel  $(x, y)$ , we set its intensity to  $(1 - r^3)$  times the maximal saliency value. In Figure 3.17, the saliency maps generated with this face painting technique can be seen.

These two automatic filtering methods provide some basic saliency detection but are far inferior to state of the art filters like the ones by [Cheng et al. 2011] and [Perazzi et al. 2012]. However, they are useful to provide a good starting point for the manual refinement of the saliency map.

# 4

## Implementation

In this Chapter, we outline parts of our application that are not directly related to the retargeting operator we propose. We focus on the limits and challenges of retargeting images on mobile devices.

Section 4.1 explains how we solve the quadratic program. We take a detailed look on how to construct the input matrix  $Q$  efficiently and other potential bottlenecks of the retargeting pipeline.

In Section 4.2, we present a novel user interface for painting saliency directly onto the retargeted image. Also, we display the cropping decision and allow interactive refinement.

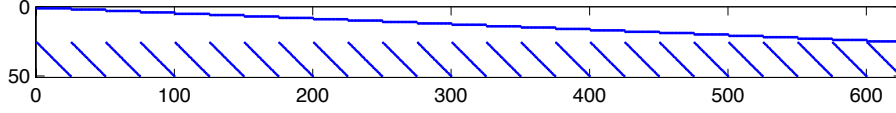
### 4.1. Solving the Quadratic Program

To solve the quadratic convex optimization problem given in Equations 3.1 to 3.7, we generated a self-contained solver using CVXGEN by [Mattingley and Boyd 2012a]. The solver generator is available online at <http://cvxgen.com>. The problem statement in the input language of CVXGEN can be found in the appendix A.1.

We use a fixed grid size of  $25 \times 25$  cells. To run it on iOS, we translated the generated solver from C to Objective-C to obtain an encapsulated solver class. The solver takes 10 milliseconds on an iPad 2 (Apple A4 chipset) and 6 milliseconds on an iPhone 5 or iPad 4 (Apple A6, A6X chipset) to optimize the grid.

This does not include the time we need to prepare the input matrix  $Q$ . As we saw in Section 3.1.2,  $Q$  is built as the sum of two inner products of two matrices  $K \in \mathbb{R}^{625 \times 50}$  and  $L \in \mathbb{R}^{50 \times 50}$ . If we just implemented  $K^T K$  as straightforward matrix multiplication, this would noticable

#### 4. Implementation



**Figure 4.1.:** Distribution of the non-zero entries in  $K^T$ .

degrade our performance. If we look at the sparsity pattern of  $K$ , visualized in Figure 4.1, we see that every row of  $K$  has only two non-zero entries. To exploit this fact, we wrote a lightweight matrix library, which stores sparse matrices in the *compressed column storage* format. We compress each column by only storing the non-zero values and the corresponding row indices. This is best explained in a small example:

$$\begin{pmatrix} 5 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 7 \\ 0 & 3 & 0 \end{pmatrix} \Rightarrow \begin{array}{ll} \text{values} & = \begin{cases} (5, 1) \\ (2, 3) \\ (7) \end{cases} \\ \text{indices} & = \begin{cases} (1, 4) \\ (2, 5) \\ (4) \end{cases} \\ \text{sizes} & = (2, 2, 1). \end{array}$$

For the inner product multiplication, we then only need to consider summands where at least one of the two factors is non-zero. Also, the Laplacian matrix  $L$  is sparse and only contains at most two non-zero entries per row and column.

We also expanded the matrix multiplication in Formula (3.11) to minimize the computational impact of the spline upsampling. In our implementation, about 60 percent of the CPU time is spent in the CVXGEN-solver while the rest is spent assembling the energy matrix, rendering in OpenGL and in UI-related computations.

## 4.2. Interface

Our application features two main screens. In the *library* view (Figure 4.2), the user can store, browse, select or delete the images she wants to retarget.

In the *editor* screen (Figure 4.3), the retargeted image is either shown filling the entire screen or side by side with the original image. The user can paint saliency by pointing and dragging the finger across the image. She can switch back and forth to erasing saliency by tapping the brush icon in the toolbar. To change the target ratio, the user can either select from a list of predefined common aspect ratios or continuously resize the image using a two finger up/down swipe gesture. With a two finger pinch gesture, the user can enlarge the image and then paint smaller salient regions in more detail.

We also optimized the iPhone-sized counterparts of these screens to accommodate all necessary tools in the smaller space, see Figure 4.4.

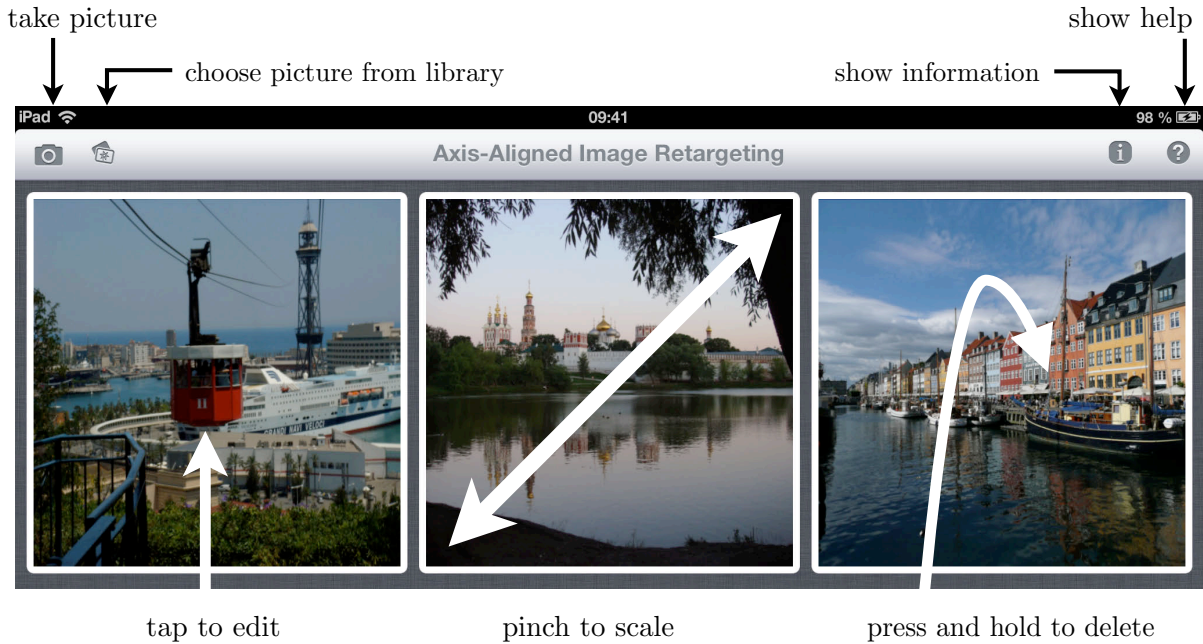


Figure 4.2.: The library interface on the iPad.

### 4.2.1. Fixed-Point Grid Stabilization

Given the small screen sizes, the side by side view in the editor does not allow very detailed saliency painting. Therefore, we combined the saliency map and the retargeted preview into a single image. Whenever the user starts painting by moving a finger across the canvas, we overlay the saliency map and continuously update the underlying warp.

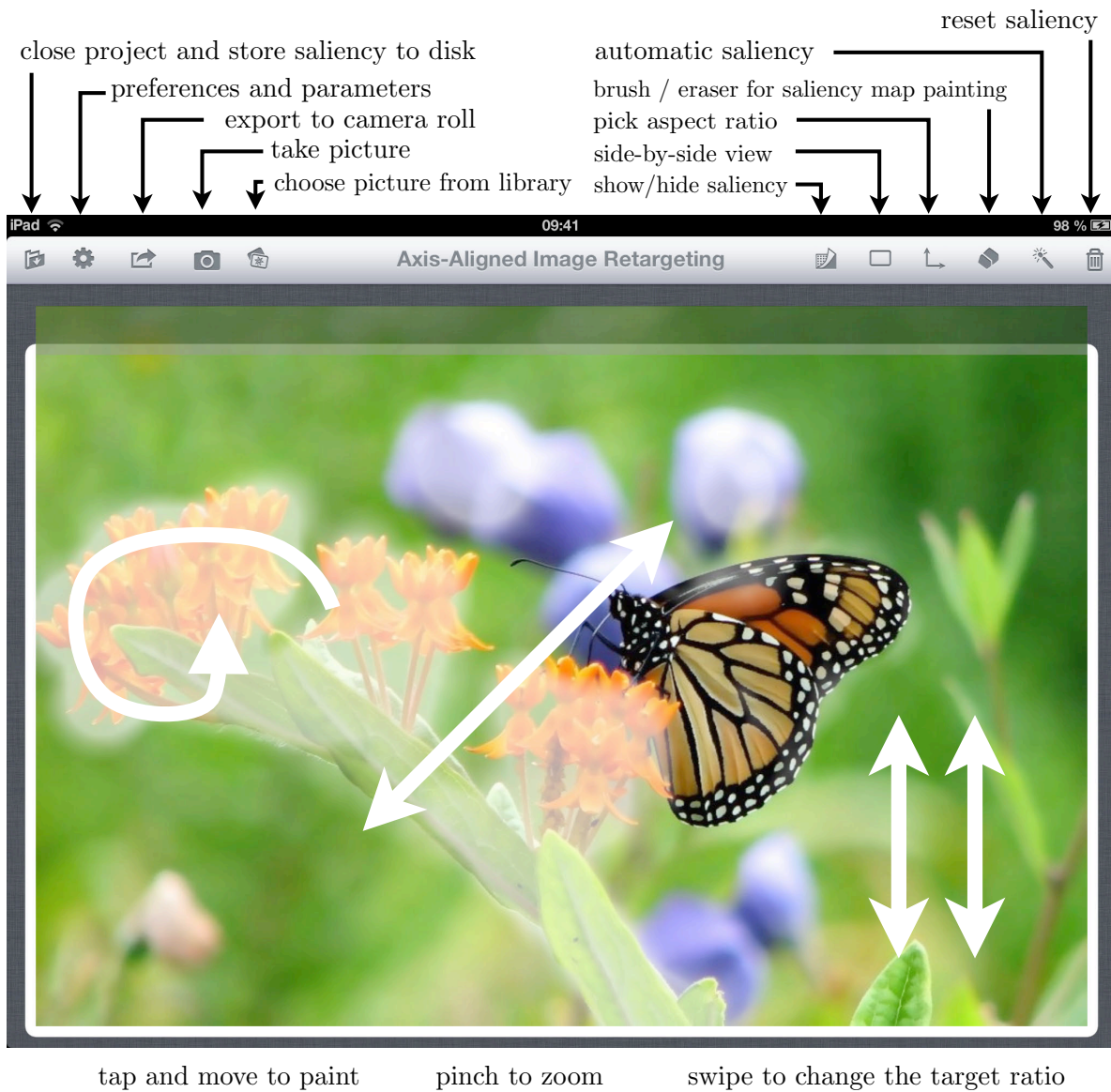
However, if the positions of the borders of the image are fixed and we change the image underneath the user's finger, a major problem appears. When the user paints saliency at a point  $p_1$  into the already retargeted image  $I'_1$ , this new saliency information leads to an updated retargeted image  $I'_2$ . So potentially, another part of the image than the one the user started painting on is located at position  $p_1$  now. The painting motion would continue at a point not intended or even foreseeable by the user. By  $p_2$ , we denote the position where in  $I'_2$  the same part of the image as at  $p_1$  in  $I'_1$  is located. Figure 4.5 illustrates this behaviour.

Such unexpected movement underneath the finger can make it hard for the user to draw simple shapes like lines or circles onto the saliency map. We propose the following solution for this problem, which we call *fixed-point grid stabilization*. Instead of fixating the border position of the image during painting, we fix the point where the user currently applies the brush and study how it changes from  $I'_1$  to  $I'_2$ . Now we keep the part under the finger the same by comparing the two position  $p_1$  and  $p_2$  and then translating the whole image  $I'_2$  such that  $p_2$  is now at the screen coordinates of  $p_1$ . In Figure 4.6, this fixed-point stabilization is outlined. To implement this stabilization, we first look up the position  $p_1$  given in screen coordinates in the grid  $G'_1$  of  $I'_1$  and we store this position as a barycentric coordinate in a cell of the original picture grid  $G$ . After updating the deformation, we use our new mapping from  $G$  to  $G'_2$  to retrieve  $p_2$ , and we center the view so that  $p_2$  is below the finger.

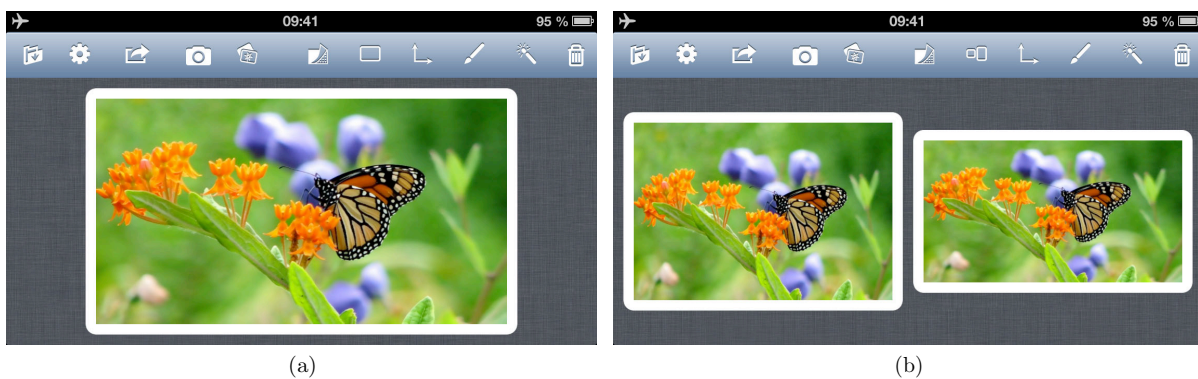
We experimentally discovered that this technique helps to make the painting process more intu-



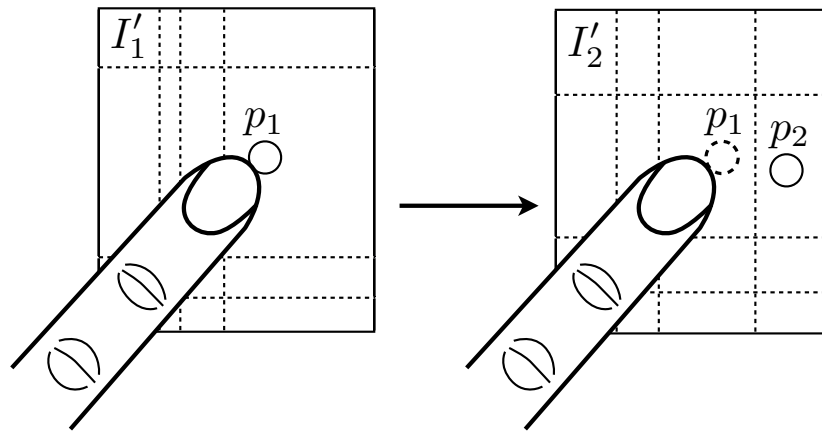
#### 4. Implementation



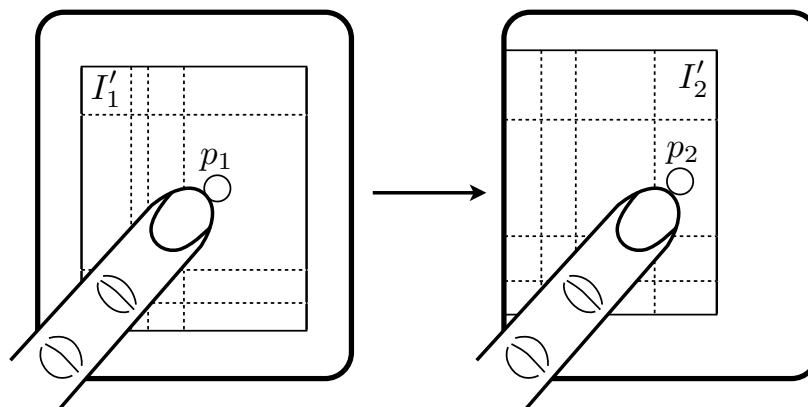
**Figure 4.3.:** The editor interface on the iPad.



**Figure 4.4.:** The editor interface on the iPhone. (a) The combined view where we directly paint onto the retargeted image. (b) Split view where we see the original image on the left and the retargeted image on the right.

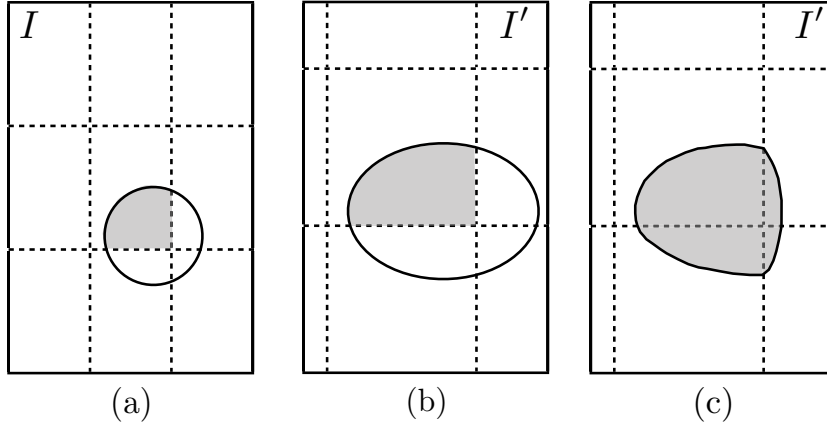


**Figure 4.5.:** Changes in the saliency map can cause the point underneath the finger to move around. Here the user paints on  $p_1$  in image  $I_1$ . In the updated image  $I_2$  this point moved to position  $p_2$  as the underlying grid changed due to the new saliency information. At position  $p_1$  another part of the image is located now, which the user probably does not want to paint on.



**Figure 4.6.:** We compensate the motion of the painted point from  $p_1$  to  $p_2$  by translating the whole image in the opposite direction. Then the point underneath the finger will stay exactly the same - independent of how much the underlying grid changes.

#### 4. Implementation



**Figure 4.7.:** Projection of the brush shape. (a) Circular shape in the original picture. (b) Projection inside a grid cell is an ellipse. (c) Projection across all grid cells involved gives a piece-wise ellipsoidal shape.

itive. It is particularly effective when the user tries to accurately trace the border of an object.

We also similarly apply the warp to the saliency map when we display it on top of the retargeted image. For the shape of the paint brush, we use a circular form with a linear intensity decay along the radius. As this shape is applied in the unretargeted saliency map, its appearance in the retargeted image gets piece-wise ellipsoidal. This means that inside each grid cell the brush is projected to an ellipsoidal shape. As the brush extends over several grid cells, the projection will be a composition of partial ellipses as indicated in Figure 4.7.

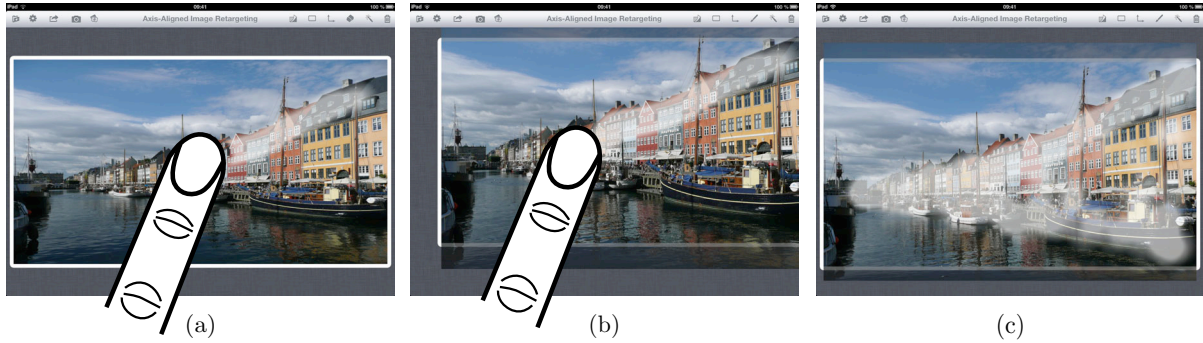
In practice this piece-wise ellipsoidal shape is hardly ever distinguishable from a single ellipse or even a single circle. So we do not need to worry about possibly strange-looking brush shapes as in Figure 4.7. Then in order to be different from a single ellipse, the grid cells beneath the brush need to be deformed differently from each other. But as we paint saliency information by applying the brush, the optimization will try to deform these cells equally and therefore creating a shape similar to a single ellipse. We dynamically adapt the size of the brush to always keep it the same size on the display. For instance, if the user zooms into the image less of the original content is visible on the screen. So we paint onto the original image with a smaller brush radius.

#### 4.2.2. Crop Preview

To show the cropped parts of the image in real-time, we show them during saliency painting as rows and columns that stick out of the retargeted picture frame (Figure 4.8). We show cropped rows and columns semi-transparent on top of the white picture frame; the background is also textured to clearly allow to see the cropped part, even if it is semi-transparent.

The widths of the cropped columns are not produced by our final optimization (they should be zero), but we need a non-zero value for visualization purposes. We use the values computed in the first optimization, i.e. the values in the grid  $G_1$  (see Section 3.2.2). The cropped columns are then narrower than the threshold, but also at least  $W^{\min}$ . So we do not waste too much display space for the cropped part, but we are also guaranteed that they will not disappear. The





**Figure 4.8.:** *Crop Preview. (a) The user is painting saliency, but nothing has been cropped yet. (b) A few rows get cropped. The whole image is moved to keep the point beneath the finger fixed. (c) After the user stopped painting the image is recentered. We can see the preview of the cropped rows below and above the salient part. The white border frame depicts the size of the retargeted image.*

cropped rows are treated in the same way.

Showing the cropped part allows the user to refine his cropping decision by simply painting into the cropped part. Our fixed-point grid stabilization algorithm provides a fluid transition of a cropped part back into the retargeted image. Figure 4.8 shows our interface for the crop preview. By just tapping onto the screen the user can show and hide both the saliency map and the preview of the cropped image for a quick estimate of the final result.

### 4.3. iOS technologies used

We conclude this Chapter by discussing a few of the iOS technologies we used in our implementation. To encode, decode and rescale images, we used the `UIImage` class and some `CoreGraphics` routines. The `CIDetector` face detector class is part of the `CoreImage` framework. The smooth transitions between different layouts and saliency modes are realized using the `animateWithDuration:animations:` method of the `UIView` class, which is part of the `CoreAnimation` framework. To display the help and info panels, we use a `UIWebView` and the project settings are stored in the app's document folder and in `NSUserDefaults`.

For rendering, we use OpenGL ES 1.1, which allows 2D texture mapping with bilinear interpolation, using an orthogonal projection. On recent iOS devices, a maximum texture of size 2048 by 2048 can be used and we thus rescale the images to this resolution for rendering purposes. For the final export at the original image resolution, we render multiple images and we stick them together without using OpenGL. The class `GLKView` takes care of the render buffer management and the `EAGLContext` manages the OpenGL state when we draw into several separate views. We create the geometry on the fly by triangulating the rectangular grids  $G$  and  $G'$  in both the image and texture space. To minimize aliasing artifacts for high resolution images, we use mipmaps.

#### 4. *Implementation*

# 5

## Results

### 5.1. App Performance

Our implementation for iOS allows refining the retargeted image in real time. When painting saliency, resizing the image or tweaking parameters, we achieve a steady frame rate of 60 fps on both the iPhone 5 and iPad 4. This allows a fluid and interactive user experience.

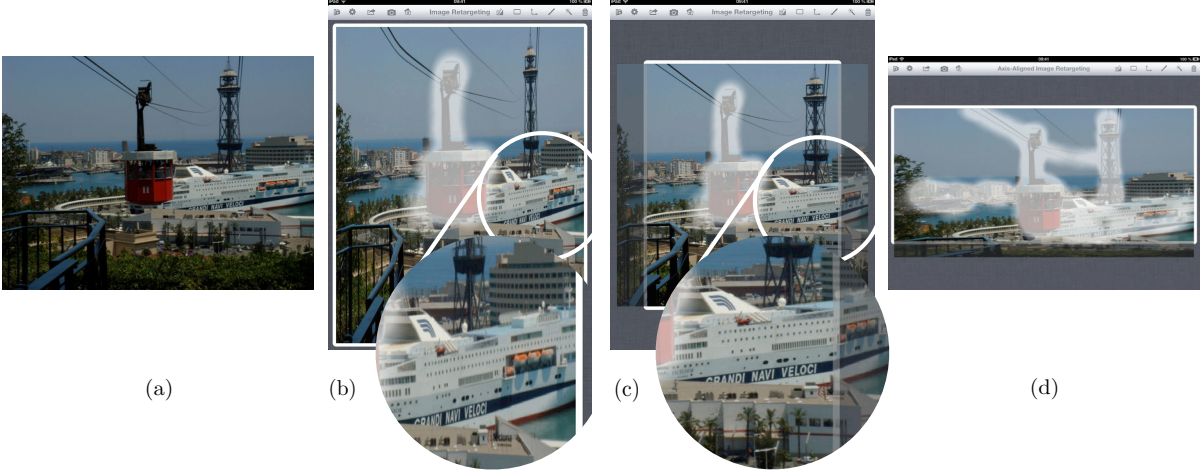
To export a retargeted image in full resolution, the app takes about 5 seconds to create a 6 megapixel image; or roughly one second per megapixel. The rendering is split into several small patches that fit into the OpenGL framebuffer. 80 percent of the time is spent rendering and stitching these patches back together. Encoding and saving the final image to the iOS camera roll takes 15 percent of the time. The warp optimization itself takes less than a percent of the time as we only need to calculate it once.

The automatic saliency map generation takes about a second. 60 percent is spent on the iOS face detection API and 35 percent on the dilation operation on the image gradient.

The whole workflow from import and automatic saliency detection to the interactive refinement of the saliency map, the choice of the target aspect ratio and the final export takes about 30 seconds of user time. To perform the *RetargetMe* benchmark, we spent 90 seconds per image on average.

We also found our app to be engaging and fun for people who were previously unaware of image retargeting. A short introduction of 20 seconds was enough to get them started and let them explore the functionalities of the app on their own.

## 5. Results



**Figure 5.1.:** Advantages of cropping (a) Original image. (b) Retargeting to portrait ratio without cropping. We can see the deformation near the border, e.g. in the inscription of the ferry as well as the at the cables of the ropeway. (c) Just by enabling our cropping operator, we get rid of the artifacts we saw before. (d) We painted a different saliency map for this wider aspect ratio to indicate what can be cropped vertically.

### 5.2. Retargeting quality using Cropping

The cropping capabilities we added to our operator (see Section 3.2) did prove to be very useful. Figure 5.1 (b), (c) compares the results with and without cropping.

We found that in most situations the regularization term is not needed at all when using cropping. This is because we use a big minimum cell size for cropping and therefore the flexibility of the grid is already constrained a lot. The default values of  $w_{\text{reg}} = 0$ ,  $L = 0.7$ ,  $\alpha = \beta = \gamma = 0.5$  and spline upsampling to a grid of  $50 \times 50$  cells are sufficient in most pictures.

The user can therefore focus on painting the saliency map. However, the semantics of this map changed when using cropping. Before, the purpose of the saliency map was to indicate which regions should not change their proportions. We found that now we end up painting the regions that we do not want the optimization to crop. On one hand this allows quicker editing as not all salient details need to be painted anymore. On the other hand the saliency map should now be optimized to the specific target ratio. Figure 5.1 (c), (d) shows how we use different saliency maps for different target ratios to indicate what we want to keep in the retargeted image.

### 5.3. RetargetMe Benchmark

Using our cropping operator, we retargeted all 86 images specified in the *RetargetMe* benchmark by [Rubinstein et al. 2010]. Figure 5.2 compares a few of these pictures with other operators. The full comparison table is provided in the additional material on the CD.

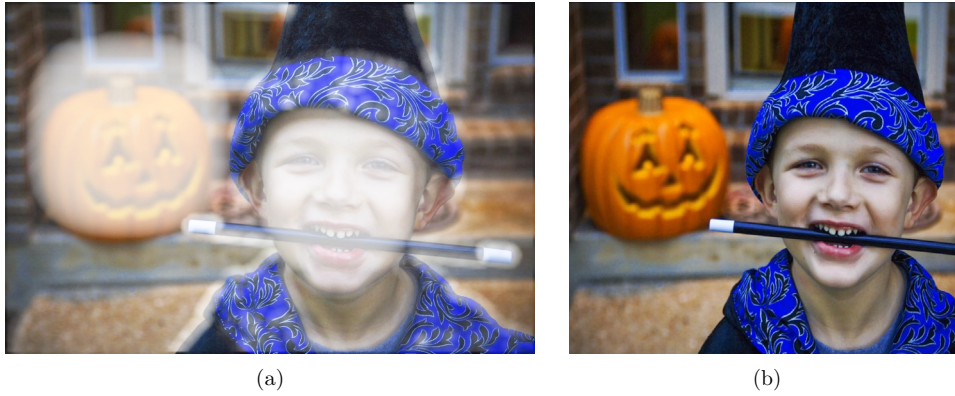
In all images, we used the default parameter values and just painted the saliency map. We used the default brush size and did not zoom into the image, so we just painted what we could see on





**Figure 5.2.:** RetargetMe benchmark comparison. We compare manual cropping (CR), streaming video (SV) [Krähenbühl et al. 2009], multi-operator media retargeting (MULTIOP) [Rubinstein et al. 2009] and axis-aligned image retargeting (AAIR) [Panozzo et al. 2012] with our approach.

## 5. Results



**Figure 5.3.:** Automatic retargeting. (a) Automatically detected saliency map. Note that also the face of the pumpkin was recognized and marked as salient. (b) Retargeted to 75% width, as in the RetargetMe benchmark.

the display of the iPad. For some images, we just kept the automatic saliency map, for instance for the child picture in Figure 5.3. In general, the automatic saliency detection works well for pictures with a small depth of field, i.e. pictures with a blurred background. High-frequency backgrounds (like leaves, grass or geometric patterns on buildings) are problematic. In those pictures, we found it easier to paint the saliency map from scratch by hand.

## 5.4. Applications

We show a simple practical application of our operator in the library view of our app. To display a gallery of pictures, two ways are often used: Either the pictures are homogeneously scaled down to fit into a regular grid or a squared central part of the image is cut out and scaled. Both of these ways can be seen in the iOS picture gallery (Figure 5.4 left) and in the iOS image picker (Figure 5.4 right) as well as on many web galleries like Picasa or flickr. While the first way does not use the screen space efficiently, the second way often crops important content.

We propose a new gallery format. We use squared thumbnails to fill all available screen space. Inside this square area, we render a retargeted image based on the saliency map the user painted the last time she edited this image. This way we crop if possible and warp if necessary to get the best out of both formats. Figure 5.5 shows this gallery for some of the benchmark pictures.

## 5.5. Limitations and Future Work

Our greedy approach to cropping has some limitations. Especially in images where the salient content is concentrated in an area smaller than the target image, our operator might crop more than necessary. The retargeted image then contains stretched unsalient parts which can be seen in Figure 5.6 (b). If we cropped and optimized the warp at the same time, as suggested in Section 3.2.1, this problem might be addressed. But by doing this, we could not use our specific high-performance convex solver and would lose our real-time performance. Also, such cases





**Figure 5.4.:** iOS picture gallery (a) and image picker (b).



**Figure 5.5.:** Our gallery using retargeted square thumbnails. Both on the iPad (a) and iPhone (b).

## 5. Results



**Figure 5.6.:** Limitation of our greedy cropping approach. (a) Original image. (b) When only painting the people too many columns will be cropped and the top and bottom of the resulting image is distorted. (c) By simply painting a bit more saliency, the user can quickly correct this cropping decision.

can be quickly corrected manually just by painting a bit more saliency, as shown in 5.6 (c).

The limited flexibility of the underlying axis-aligned deformation noticeably constrains the warp in situations where shearing and local rotations might be preferred. Also, our operator can not guarantee the preservation of straight lines that are not axis-aligned.

The fixed-point grid motion stabilization proves to be very helpful in order to paint directly onto the retargeted image. In a few special cases, our implementation is not able to keep the finger’s location fixed. These situations occur when the cropped region changes quickly and the preview of the cropped rows or columns expands over a large part of the screen. A more sophisticated balance between screen layouting and motion stabilization could address this problem.

In their user study, [Rubinstein et al. 2010] studied if it is important whether the original un-retargeted image is shown to the viewer or not. They found the following main difference: without the reference image, cropping was almost always the preferred choice, as it does not deform the image. When the original image was shown side by side the loss of content became apparent and so in that case, the SV and MULTIOP operators ranked just as good as manual cropping. So having the original image directly available for comparison while drawing the saliency map can be helpful in some applications. For this reason, we provide an easy way to switch to this side by side view in our app, even though the combined view makes much better use of the available screen real estate.

As our approach only crops whole columns and rows of our grid, the coarse grid resolution of  $25 \times 25$  cells restricts the precision of the cut. In future work, we could try to increase the grid resolution or to find a way to partially crop rows and columns. Also, it would be interesting to try out other, more sophisticated cropping approaches and to compare their quality and efficiency.

It would be interesting to integrate our retargeting operator into a mobile web browser to dynamically optimize the pictures and their layout on any webpage to the given screen size. Another intriguing continuation of our work would be to expand our operator to video retargeting. The speed of our method might allow real-time video retargeting even on mobile devices.



## Conclusion

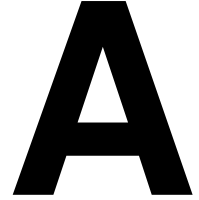
We presented a novel content-aware image retargeting operator. Our operator uses an axis-aligned deformation to retarget an image, and it combines them with cropping to enrich the solution space and to reduce the distortion. Based on an interpolated crop threshold and two energy minimizations, we get a fast and robust algorithm. We can guarantee the absence of fold overs and we can control the smoothness of the warp.

We implemented our algorithm on iOS, and we showed that it runs at interactive frame rates. The user interface has been optimized for touch input and to be used on mobile devices with small screens. To expose all parameters of the retargeting operator to the user in a intuitive and suitable way, we developed new tools to modify the saliency map directly on top of the retargeted result; showing a single picture allows us to use the limited screen space in the best possible way.

We achieve a fluid experience by customizing every component of the retargeting pipeline for mobile devices. The application is easy and fun to use and provides many options for more experienced users.

To the best of our knowledge, this thesis is the first attempt to bring interactive content-aware image retargeting to a mobile device. The source code of our implementation is available online at <http://github.com/grafdan/retargeting> and our application is submitted for review to the iOS App Store.

6. *Conclusion*



# Appendix

## A.1. CVXGEN Problem Statement

Below, we state the optimization problem from Section 3.1.1 in the CVXGEN language.

```
dimensions
  m = 25    # number of DOF along the height
  n = 25    # number of DOF along the width
end

variables
  # the first m elements are the heights of the rows
  # the next n elements are the widths of the columns
  st (m+n)
end

parameters
  Q (m+n, m+n) psd # Q=A'A
  B (m+n, 1)       # the linear term that is zero for our energy
  minH (m)
  maxH (m)
  minW (n)
  maxW (n)
  imageHeight
  imageWidth
end
```

## A. Appendix

```
# energy term
minimize
    #the energy is st'*Q*st + st'*B
    quad(st, Q) + st'*B

# boundary terms
subject to
    st[i] >= minH[i], i=1..m
    st[i] <= maxH[i], i=1..m
    st[m+i] >= minW[i], i=1..n
    st[m+i] <= maxW[i], i=1..n
    sum[i=1..m] (st[i]) == imageHeight
    sum[i=m+1..m+n] (st[i]) == imageWidth
end
```

## Bibliography

- AVIDAN, S., AND SHAMIR, A. 2007. Seam carving for content-aware image resizing. vol. 26.
- BOYD, S., AND VANDENBERGHE, L. 2004. *Convex optimization*. Cambridge university press.
- CANNY, J. 1986. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 6, 679–698.
- CASTILLO, S., JUDD, T., AND GUTIERREZ, D. 2011. Using eye-tracking to assess different image retargeting methods. In *Proceedings of the ACM SIGGRAPH Symposium on Applied Perception in Graphics and Visualization*, ACM, 7–14.
- CHEN, R., FREEDMAN, D., KARNI, Z., GOTSMAN, C., AND LIU, L. 2010. Content-aware image resizing by quadratic programming. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, IEEE, 1–8.
- CHENG, M.-M., ZHANG, G.-X., MITRA, N. J., HUANG, X., AND HU, S.-M. 2011. Global contrast based salient region detection. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, IEEE, 409–416.
- DANTZIG, G., AND THAPA, M. 2003. Linear programming 2: Theory and extensions. 67.
- EINHÄUSER, W., AND KÖNIG, P. 2003. Does luminance-contrast contribute to a saliency map for overt visual attention? *European Journal of Neuroscience* 17, 5, 1089–1097.
- FAN, X., XIE, X., QIN ZHOU, H., AND YING MA, W. 2003. Looking into video frames on small displays. In *In Proc. of ACM Multimedia 2003*, Press, 247–250.
- GAL, R., SORKINE, O., AND COHEN-OR, D. 2006. Feature-aware texturing. In *Proceedings of Eurographics Symposium on Rendering*, 297–303.
- GRANT, M., BOYD, S., AND YE, Y. 2008. Cvx: Matlab software for disciplined convex programming. Online accessible: <http://stanford.edu/~boyd/cvx>.
- HARRIS, C., AND STEPHENS, M. 1988. A combined corner and edge detector. In *Alvey vision conference*, vol. 15, Manchester, UK, 50.
- HUANG, H., FU, T., ROSIN, P., AND QI, C. 2009. Real-time content-aware image resizing. *Science in China Series F: Information Sciences* 52, 2, 172–182.
- JUDD, T., EHINGER, K., DURAND, F., AND TORRALBA, A. 2009. Learning to predict where humans look. In *Computer Vision, 2009 IEEE 12th international conference on*, IEEE, 2106–2113.
- KARMAKAR, N. 1984. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, ACM, 302–311.
- KARNI, Z., FREEDMAN, D., AND GOTSMAN, C. 2009. Energy-based image deformation. In *Computer Graphics Forum*, vol. 28, Wiley Online Library, 1257–1268.
- KRÄHENBÜHL, P., LANG, M., HORNING, A., AND GROSS, M. 2009. A system for retargeting of streaming video. In *ACM Transactions on Graphics (TOG)*, vol. 28, ACM, 126.

## BIBLIOGRAPHY

- MATTINGLEY, J., AND BOYD, S. 2012. Cvxgen: a code generator for embedded convex optimization. *Optimization and Engineering*, 1–27.
- MATTINGLEY, J., AND BOYD, S., 2012. Cvxgen: Code generation for convex optimization. <http://cvxgen.com>.
- PANOZZO, D., WEBER, O., AND SORKINE, O. 2012. Robust image retargeting via axis-aligned deformation. *Computer Graphics Forum (proceedings of EUROGRAPHICS) 31*, 2, 229—236.
- PERAZZI, F., KRÄHENBÜHL, P., PRITCH, Y., AND HORNUNG, A. 2012. Saliency filters: Contrast based filtering for salient region detection. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 733–740.
- RUBINSTEIN, M., SHAMIR, A., AND AVIDAN, S. 2008. Improved seam carving for video retargeting. *ACM Transactions on Graphics, (Proceedings SIGGRAPH 2008) 27*, 3.
- RUBINSTEIN, M., SHAMIR, A., AND AVIDAN, S. 2009. Multi-operator media retargeting. *ACM Transactions on Graphics, (Proceedings SIGGRAPH 2009) 28*, 3.
- RUBINSTEIN, M., GUTIERREZ, D., SORKINE, O., AND SHAMIR, A. 2010. A comparative study of image retargeting. *ACM Transactions on Graphics, Proceedings Siggraph Asia 29*, 5.
- SANTELLA, A., AGRAWALA, M., DECARLO, D., SALESIN, D., AND COHEN, M. 2006. Gaze-based interaction for semi-automatic photo cropping. In *In CHI 2006*, ACM, 771–780.
- SHAMIR, A., AND SORKINE, O. 2009. Visual media retargeting. In *ACM SIGGRAPH ASIA Courses*.
- SHAMIR, A., HORNUNG, A., AND SORKINE, O. 2012. Modern approaches to media retargeting. In *ACM SIGGRAPH ASIA Courses*.
- TREISMAN, A. M., AND GELADE, G. 1980. A feature-integration theory of attention. *Cognitive psychology 12*, 1, 97–136.
- VIOLA, P., AND JONES, M. J. 2004. Robust real-time face detection. *International journal of computer vision 57*, 2, 137–154.
- WANG, Y., TAI, C., SORKINE, O., AND LEE, T. 2008. Optimized scale-and-stretch for image resizing. *ACM Transactions on Graphics (TOG) 27*, 5, 118.
- WOLF, L., GUTTMANN, M., AND COHEN-OR, D. 2007. Non-homogeneous content-driven video-retargeting. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, IEEE, 1–6.