Vision, Modeling, and Visualization (2013)
Michael Bronstein, Jean Favre, and Kai Hormann (Eds.)

# Mobile Image Retargeting

Daniel Graf Daniele Panozzo Olga Sorkine-Hornung

ETH Zurich

**Abstract**

*We propose an algorithm for axis-aligned content-aware image retargeting that is specifically optimized for mobile devices, and we show that interactive image retargeting is possible even with a low-power, mobile CPU. Our retargeting operator is based on non-uniform scaling and cropping and produces results that are on par with state-of-the-art on a large collection of images. Taking the limited screen space of mobile devices into account, we design a novel user interface that allows painting the saliency map directly onto the retargeted image while the system is continuously recomputing the retargeted result at interactive rates. Finally, we apply our algorithm in a picture gallery application to greatly improve the screen space utilization in mobile device settings.*

## 1 Introduction

Digital images are captured using sensors with different aspect ratios and visualized on consumer devices equipped with sceens that also have a wide variety of aspect ratios. To compensate the mismatch, the image is usually letterboxed, i.e., proportionally scaled to fit the screen by introducing horizontal or vertical black bars at the borders.

To make full use of the available screen space, many methods for non-uniform scaling of the image to the desired target resolution have been developed [SSHSH12]. They strive to preserve important areas of the image while concentrating the distortion in the homogeneous parts where it is less noticeable.

In this paper, we propose a retargeting algorithm that is tailored to mobile devices, enabling content-aware resizing of an image immediately after it has been captured. The mobile setting poses a challenge for real-time and user-friendly retargeting due to the limitations on computational power, battery life and screen space, requiring specially-designed algorithms and user interfaces. Our approach to solving the problem makes the following contributions:

1. We extend the axis-aligned warping algorithm [PWS12] to incorporate cropping into the optimization. Our method introduces a minimal overhead on the running time of the original algorithm, and we also adapt it to devices with limited computational resources.
2. We present a novel touch-based user interface for painting the region of importance directly onto the retargeted image, e.g. to enhance an automatically computed saliency map. This allows for interactive refinement of the result using a full-screen preview. Our method has a real-time performance of 60 fps, providing immediate visual feedback to the user and enabling to define high-quality importance maps in seconds.
3. We apply our algorithm to the generation of thumbnail photo galleries, which is of interest for every device with a mobile camera. We test our method on a large image collection from the RETARGETME benchmark [RGSS10] and achieve results with a quality comparable or higher than other approaches.


**Figure 1:** *With our touch-based user interface for image retargeting, users can paint the saliency map and observe the retargeted image in real time.*

The prototype developed in this work will be released as an open source project to foster future research and evaluation of mobile image retargeting.

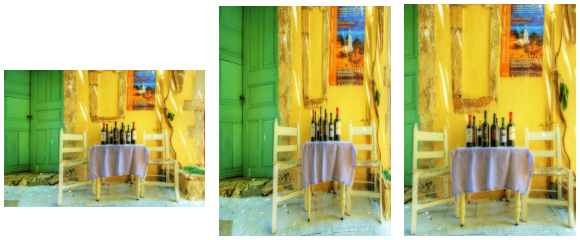
## 2 Related Work

We focus our attention on warp-based methods, and we refer to [RSA08,SCSI08,BSFG09,PKVP09] for a detailed discussion of discrete retargeting approaches.

**Warp-based retargeting** algorithms define an energy functional and minimize it while constraining the boundary to match the target image size. The energy measures local deviation of the warp from a shape-preserving deformation such as translation [GSCO06], rigid transformation [KFG09] or similarity [WTSL08, KFG09, ZCHM09, KLHG09]. Recently, many of these methods have been unified in a single finite-element framework [KWSH*13]. Quadratic energies, which can be minimized quickly by solving a sparse linear system, often introduce foldovers that lead to artifacts in the retargeted image (see Fig. 3 in [PWS12]). Self-intersections can be prevented using nonlinear optimization, for example by iteratively penalizing grid edge flips [WTSL08], by constraining the size of grid cells [KLHG09] or explicitly posing positive scaling constraints on grid cells' transformations [CFK*10]. However, currently, nonlinear constrained optimization can be performed in real time only using customized GPU solvers [KLHG09], and at present it cannot be used in mobile devices due to the prohibitive computational cost.

Observing that many successful warping methods strongly penalize rotations [WTSL08, KLHG09, CFK*10], Panozzo et al. [PWS12] proposed to restrict the optimization to axis-aligned deformations. This reduces the space of admissible warps and makes the optimization extremely efficient and suitable for mobile devices, while generally retaining high-quality results. However, cropping is not allowed by their formulation, leading to unwanted results for large deformations (Figure 2). Automatic cropping has been demonstrated to be a useful retargeting operator [LG06, DDN08,RGSS10], especially when combined with other operators [RSA09,WLSL10,YSWL11]. Wang et al. [WLSL10, YSWL11] successfully optimized content-aware cropping and warping for video volumes, yet at a significant computational cost. In this work we modify axis-aligned deformations [PWS12] to incorporate optimized cropping in an efficient manner, enabling implementation on mobile devices.

**Saliency map computation.** All image retargeting methods rely on an image importance map or saliency. Low-level features such as the $L^1$-norm of the intensity gradient have been successfully used in [AS07], but they fail to capture high-level features like faces or semantically important objects in a scene. Automatic saliency detection algorithms, e.g. [IKN98], have been used in more recent methods. Salient regions can be found by detecting globally unique regions of high contrast using a histogram-based contrast method [CZM*11]. Perazzi et al. [PKPH12] propose to cluster image pixels based on their color and then use high-dimensional Gaussian filtering to measure the uniqueness and spatial distribution of each element. Eye-tracking [CJG11] and face detection [VJ04] can also be used to help identifying the most important parts of an image. In this work, we combine an automatically generated saliency map with a simple touch-based interface that allows the user to manually refine it.



(a) (b) (c)

**Figure 2:** *(a) Original image. (b) Axis-Aligned Retargeting [PWS12] introduces an unpleasant distortion for large deformations. (c) By allowing to crop, our algorithm generates a map with a considerably lower distortion.*

## 3 Algorithm

We base our retargeting operator on [PWS12] and we extend it to use both scaling and cropping in order to reduce the distortion in presence of large deformations (Figure 2). We enrich the original formulation by adding constraints that allow the borders of the image to collapse. Our algorithm requires solving a small quadratic program twice: the first run decides which parts should be cropped, and the second generates the final, retargeted image. This simple iterative scheme is about twice slower than [PWS12], but it greatly reduces the distortion in most cases, as shown in Section 6 and in the additional material.

### 3.1 Axis-aligned retargeting

We wish to warp the source image $I$ of width $W$ and height $H$ into a target image $I'$ of width $W'$ and height $H'$. The deformation between $I$ and $I'$ is discretized on a uniform grid $\mathcal{G}$ with $M$ rows and $N$ columns. Each column of $\mathcal{G}$ has width $W/N$ and each row has height $H/M$. The deformed grid $\mathcal{G}'$ has the same connectivity as $\mathcal{G}$, but it is non-uniformly deformed so that its boundary matches the size of the target image ($W' \times H'$). We denote by $r_s = W/H$ and $r_t = W'/H'$ the aspect ratios of the source and target images, respectively.

Following [PWS12], we inhibit local rotations by restricting the set of admissible deformations to non-uniform scalings of the grid's rows and columns. The deformations contained in this subspace can be encoded by storing the size of each row and column of $\mathcal{G}'$:

$$s^{\text{rows}} = (s_1^{\text{rows}}, s_2^{\text{rows}}, \ldots, s_M^{\text{rows}}) \quad (1)$$

$$s^{\text{cols}} = (s_1^{\text{cols}}, s_2^{\text{cols}}, \ldots, s_N^{\text{cols}}) \quad (2)$$

Hence axis-aligned deformations are parameterized using $s = (s^{\text{rows}}, s^{\text{cols}})^T \in \mathbb{R}^{M+N}$. In this subspace, the *as similar as possible* energy [PWS12] can be written as:

$$E_{\text{ASAP}} = \sum_{i=1}^{M} \sum_{j=1}^{N} \left( \Omega_{i,j} \left( \frac{M}{H} s_i^{\text{rows}} - \frac{N}{W} s_j^{\text{cols}} \right) \right)^2, \quad (3)$$

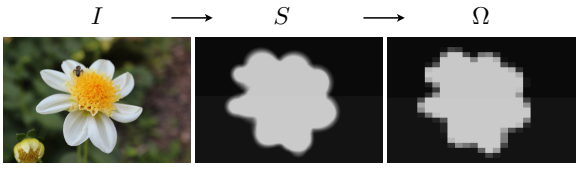



**Figure 3:** *(a) The original photo I. (b) Manually painted saliency map S on I. (c) The energy formulation only considers the rasterized saliency matrix* $\Omega$.

where $\Omega_{i,j}$ is the average saliency in the cell $(i,j)$ of $\mathcal{G}$ (see Figure 3). This energy penalizes all deformations except uniform scaling. This is particularly desirable if cropping is allowed, since it allows filling the target image with the more salient content by scaling the source uniformly and cropping the non-salient regions.

The energy can be written in matrix form as

$$E_{\text{ASAP}} = (Ks)^T (Ks) = s^T K^T K s, \tag{4}$$

where

$$K_{k,l} = \begin{cases} \Omega_{r(k),c(k)} \frac{M}{H} & \text{if } l = r(k), \\ -\Omega_{r(k),c(k)} \frac{N}{W} & \text{if } l = M + c(k), \\ 0 & \text{otherwise,} \end{cases} \tag{5}$$

for the row $r(k) = \lceil k/N \rceil$ and column $c(k) = ((k-1) \bmod N) + 1$.

In this setting, it is possible to guarantee that the deformation map is bijective simply by constraining all variables to be positive. We thus cast our optimization as a quadratic convex optimization problem (QP):

$$\text{minimize } s^T K^T K s \tag{6}$$

$$\text{subject to } \bar{H}_i^{\min} \le s_i^{\text{rows}} \le \bar{H}_i^{\max},\ i = 1,\dots,M, \tag{7}$$

$$\bar{W}_j^{\min} \le s_j^{\text{cols}} \le \bar{W}_j^{\max},\ j = 1,\dots,N, \tag{8}$$

$$s_1^{\text{rows}} + s_2^{\text{rows}} + \dots + s_M^{\text{rows}} = H', \tag{9}$$

$$s_1^{\text{cols}} + s_2^{\text{cols}} + \dots + s_N^{\text{cols}} = W'. \tag{10}$$

The equality constraints (9) and (10) fix the target size to $W' \times H'$. We use the bounds $\bar{H}_i^{\min}$, $\bar{H}_i^{\max}$, $\bar{W}_j^{\min}$ and $\bar{W}_j^{\max}$ to constrain the size of each row and column. Differently from [PWS12], we bound the maximal and minimal size of each row and column individually, instead of using a global bound. It is a key difference that enables us to introduce cropping into the optimization without changing the structure of the quadratic program, thus allowing us to use a fast solver customized for this specific task.

**Bound constraints.** We denote a bound without a subscript as a shorthand for global bounds across all rows and columns, e.g. $\bar{H}^{\min} = \bar{H}_i^{\min}$ for all $i = 1,\dots,M$.

[PWS12] used a universal minimal cell size, defined as $\bar{H}^{\min} = L \cdot H'/M$ and $\bar{W}^{\min} = L \cdot W'/N$, where $L \in (0,1)$ controls the maximal allowed stretch. A fixed $L$ can introduce unnecessarily distortion in salient parts of the image,

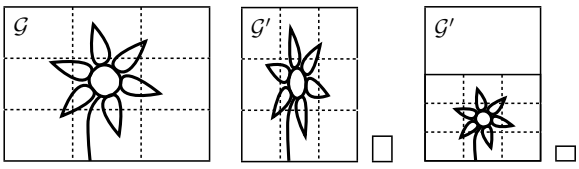



**Figure 4:** *Minimum cell size constraints. (a) The image in its original aspect ratio. (b) The uniform minimal cell size as used in [PWS12]. (c) The aspect-ratio-aware minimal cell size we propose. The two smaller boxes represent the L-downscaled minimal cell sizes for* $L = 0.5$.

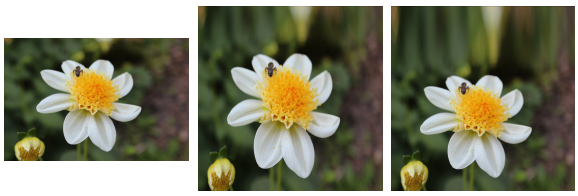



**Figure 5:** *Comparison between the two types of cell size constraints using* $L = 0.9$. *(a) The image in its original aspect ratio. (b) The uniform minimal cell size as used in [PWS12]. (c) The aspect-ratio-aware minimal cell size we propose. Both images use the same saliency (Figure 3).*

as illustrated in Figure 4(b). We thus propose a different, *aspect-ratio-aware* minimal cell size:

$$\bar{W}^{\min} = L \cdot s_r \frac{H'}{N}, \quad \bar{H}^{\min} = L \cdot \frac{H'}{M} \quad \text{if } r_t > r_s, \tag{11}$$

$$\bar{W}^{\min} = L \cdot \frac{W'}{N}, \quad \bar{H}^{\min} = L \cdot \frac{1}{s_r} \frac{W'}{M} \quad \text{if } r_t \le r_s. \tag{12}$$

We first uniformly scale the grid to fit into the target dimensions, and only then fix the minimum cell size as a fraction $L$ of the cell sizes of the uniformly scaled grid $\mathcal{G}'$. This way the minimal cell size keeps its original aspect ratio, as illustrated in Figure 4(c). Figure 5 compares the two approaches for a fixed $L = 0.9$.

Generally, we set all the upper bounds $\bar{H}_i^{\max}$ and $\bar{W}_j^{\max}$ to infinity. Only for rows and columns to be cropped, we set the corresponding upper bounds to zero to make them collapse.

**QP solver.** We use CVXGEN [MB12] to solve our QP problems. CVXGEN requires a static description of the problem and it generates a customized C solver that efficiently minimizes it. Its major limitations are that it cannot scale to problems with a large number of variables, and it is not possible to alter the structure of the problem, e.g. the number of variables, at runtime. While the first limitation does not affect our algorithm since a grid of size $25 \times 25$ is sufficient for images up to HD resolution (see Figure 7 of [PWS12]), the second limitation makes incorporation of cropping into the optimization difficult.

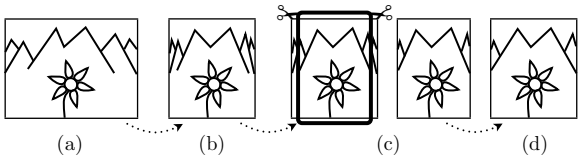



**Figure 6:** *Our threshold-based cropping approach. Starting with the original image (a) we find the optimal warp (b) using the minimal cell size* $\alpha \cdot \bar{W}^{min} \times \alpha \cdot \bar{H}^{min}$. *We decide what to crop (c) and then optimize the remaining image parts again to fill the target rectangle (d).*

### 3.2 Cropping

In situations where the image has large, uninteresting parts on its borders, cropping is preferable to plain warping, since it greatly reduces the distortion in the warp. While it would be trivial to include cropping directly into the variational energy formulation by adapting the constraints to the specific saliency map, this would not allow us to "prebake" a very efficient solver using CVXGEN and it would consequently prevent interactive feedback. In this section we show that it is possible to introduce cropping in the retargeting operator by solving the QP problem presented in Section 3.1 twice, using the output of the first minimization to define the constraints for the second run.

**Two-step optimization.** Our retargeting operator is divided into three steps (Figure 6):

1. Initial QP solve;
2. Detection of high-distortion areas with low saliency;
3. Final QP solve.

**Initial QP solve.** Given a user-defined, cropping threshold $\alpha \in [0,1)$, we solve Equations (6)–(10) using the lower bounds $\alpha \bar{W}^{\min}$ and $\alpha \bar{H}^{\min}$. Note that for smaller values of $\alpha$, non-salient regions are allowed to squeeze more and they are even allowed to collapse if $\alpha = 0$.

**Detection of high-distortion strips with low saliency.** We now mark each row that is smaller than $\bar{H}^{\min}$ and each column smaller than $\bar{W}^{\min}$, starting from the boundaries of the image and going inwards. All the marked rows and columns will be collapsed in the next optimization step.

**Final QP solve.** To collapse the marked rows and columns, we set the corresponding bounds on the maximal and minimal size to zero: we set $\bar{H}_i^{\min}$ and $\bar{H}_i^{\max}$ both to zero in order to crop row $i$. We use the original minimal bounds of $\bar{H}^{\min}$ and $\bar{W}^{\min}$ for the non-marked rows and columns and solve the QP again. This procedure effectively removes the corresponding variables from the system, forcing them to be zero. Note that this is performed without changing the structure of the optimization problem.

**Incremental cropping.** The procedure described above works well for fixed target sizes, but it has an unintuitive behavior when the size of the image is changed interactively. Changing the target size of the image by a single pixel could drastically change the cropping window: Since the rows and columns are cropped using a hard threshold, an entire group of rows or columns might get cropped at the same time, resulting in popping and a non-smooth user experience. We thus relax the hard cropping threshold by introducing a range that is higher on the border than in the middle of the image. To formalize this, we introduce the following limits:

$$\bar{W}^{\text{low}} = \tfrac{1+\alpha}{2} \cdot \bar{W}^{\min} \text{ and } \bar{W}^{\text{high}} = \tfrac{1+1/L}{2} \cdot \bar{W}^{\min}, \quad (13)$$

which ensures that for all admissible values of $L$ and $\alpha$ the following holds:

$$\bar{W}^{\text{low}} \leq \bar{W}^{\min} \leq \bar{W}^{\text{high}} \leq W'/N. \quad (14)$$

When deciding whether to crop the $j$-th column (starting from the left), we no longer just compare $s_j^{\text{cols}}$ with $\bar{W}^{\min}$ but use an interpolated threshold value from our threshold range $[\bar{W}^{\text{low}}, \bar{W}^{\text{high}}]$, so we crop column $j$ of width $s_j^{\text{cols}}$ iff

$$s_j^{\text{cols}} < \left( \frac{j-1}{N} \bar{W}^{\text{low}} + \left(1 - \frac{j-1}{N}\right) \bar{W}^{\text{high}} \right). \quad (15)$$

The same procedure is also adapted for the right border and for cropping rows. We use a fixed cell size factor $L = 0.7$ and a cropping threshold $\alpha = 0.5$ in all our experiments. By increasing $L$, one can make the grid cells more rigid, which forces more border cells to be cropped. Since $\alpha$ appears only in the first QP solve, it only plays a decisive role if large regions of the image are cropped. Compared to the fixed threshold $\bar{W}^{\min}$, this approach makes the cropping experience more fluid and dynamic, while also significantly improving the retargeting quality. A direct comparison is given in Figure 7.

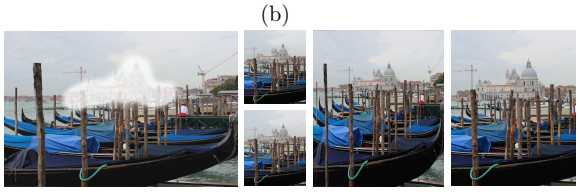
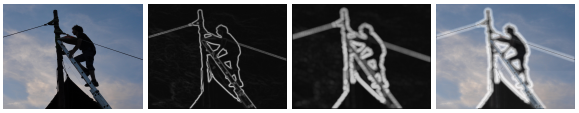



**Figure 7:** *We paint the saliency map that can be seen in the original image (a) and want to retarget the image from landscape to portrait. We compare our crop-threshold range (pictures (b) and (c)) with the fixed crop-threshold (pictures (d) and (e)). Note that pictures (b) and (d) have the exact 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 using the threshold-range generates two almost identical, good-looking warps (b) and (c).*

**Feasibility of the QP.** Our energy is strictly convex, since $K^T K$ is positive semidefinite. The feasible region is always non-empty: first, note that our cropping algorithm never crops all columns or all rows. As all the thresholds and



(a) (b) (c) (d)

**Figure 8:** *Gradient-based automatic saliency detection. (a) Original image. (b) Gradient magnitude after the convolution with the Sobel kernel along both axes. (c) Gradient magnitude after dilation operation. (d) Auto-saliency on top of the original image.*

lower bounds we used are always smaller than the average row height $H'/M$ and the average column width $W'/N$, at least one row and one column will not get cropped. This also holds for the incremental cropping thresholds. Therefore the feasible region always contains the warp that takes all non-cropped cells and spans them uniformly over the entire image.

### 3.3 Automatic saliency estimation

We automatically estimate an initial saliency map that can be subsequently edited with our UI (Section 4) by combining low-level image features with a face detector.

**Gradient-based saliency detection.** We scale the image to 250 pixels along the longer direction to reduce the computational effort and to eliminate high-frequency noise. We then convert the image to gray-scale and convolve it with the Sobel gradient kernel along the $x$- and the $y$-axes. The saliency is extracted as the $L^2$-norm of the gradient, and it is dilated to simulate a brush stroke. This also fills up small holes in the gradient map and looks similar to manually painting along each contrast-rich edge. The dilation operation is performed using a fixed radius $d = 4$ as follows:

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

Figure 8 shows an example of the extracted saliency.

**Face detection.** Humans are very good at detecting distortion on faces, even if it is small. We thus try to reduce the distortion by identifying image regions occupied by faces and assigning a high saliency value to them. Standard face detectors, like [VJ04] or the one implemented in iOS, return a set of axis-aligned rectangular subwindows, each assumed to contain a face. This rectangle, which roughly contains 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 $2r_x$ and height $2r_y$, we mark as salient all the pixels in the following set:

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

To simulate a brush stroke, we let the intensity slightly decay towards the border: we set the saliency of each pixel in the face region to $(1-r^3)$ times the maximal saliency value.

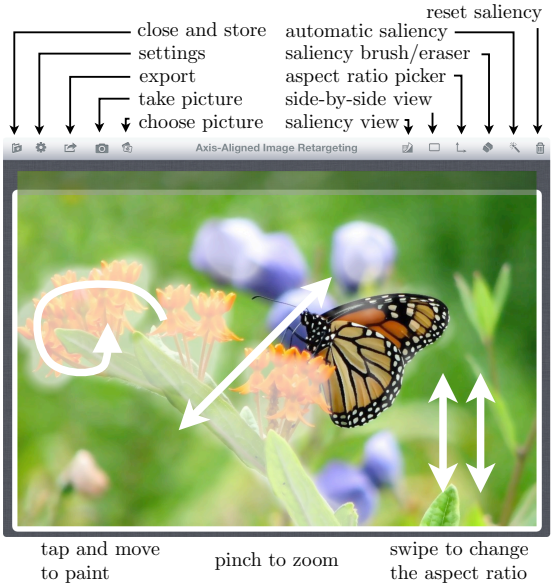



**Figure 9:** *The editor interface on the iPad.*

## 4 Touch-based user interface for saliency editing

We designed a customized user interface that allows to refine an automatically generated saliency map with a few user strokes. A normal UI cannot be used in this scenario since the available screen space is not sufficient. The user must be allowed to edit the saliency map, which is defined on every pixel of the original image, and at the same time visualize the result, which has a different aspect ratio. On mobile devices, the screen is too small to visualize both, and we thus combine them into a single view. We show a screenshot of our UI in Figure 9. The user can paint saliency by pointing and dragging the finger across the image. 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 greater detail.

**Fixed-point grid stabilization.** Combining the saliency map and the retargeted preview into a single image becomes problematic when the user starts painting by moving a finger across the canvas. As the saliency changes, the image is rescaled and the region where the user is painting consequently moves.

Such unexpected movement underneath the finger can make it difficult for the user to draw simple shapes like lines or circles onto the saliency map. To make the process more intuitive, we fix the point where the user is currently applying the brush and we translate the image after the deformation so that the region where the user is painting stays below the finger. Figure 10 outlines this fixed-point stabilization. To implement it, we first look up the position of the finger in screen coordinates and express it using barycentric coordinates inside the corresponding grid cell. We then compute the deformation, find the new deformed position by apply-

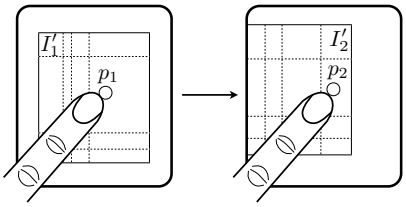



**Figure 10:** *We compensate the motion of the painted point from $p_1$ to $p_2$ by translating the whole image in the opposite direction. The point underneath the finger stays exactly the same, no matter how much the underlying grid is deformed.*

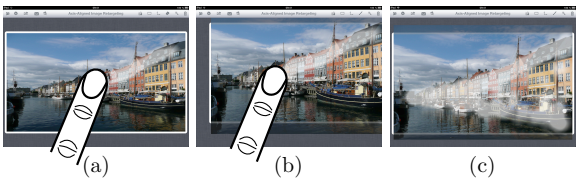



**Figure 11:** *Crop preview. (a) The user is painting saliency, but nothing has been cropped yet. (b) A few rows are cropped. The whole image is moved to keep the point beneath the finger fixed. (c) The image is recentered after the user stops painting. The preview of the cropped rows is shown above and below the salient part. The white frame outlines the retargeted image.*

ing the barycentric coordinates to the deformed grid and we translate it to be below the finger.

**Crop preview.** Another problem that arises while visualizing the saliency together with the retargeted result is that the cropped parts disappear, not allowing to paint saliency on them. We thus visualize the cropped parts around the image with an opacity of 50% (Figure 11), and we allow the user to paint on them. The size of the cropped rows and columns is set to the value computed after the first energy minimization. Our fixed-point grid stabilization algorithm provides a fluid transition of a cropped part back into the retargeted image when the user paints on it.

## 5 Implementation details

To compile the QP solver in an iOS binary, we translated the solver generated by CVXGEN from C to Objective-C. The solver runs in 10 milliseconds on an iPad 2 (Apple A5 chipset) and 6 milliseconds on an iPhone 5 or iPad 4 (Apple A6, A6X chipset). The pixels of the final image are warped using bilinear interpolation inside each grid cell. Similarly to [PWS12], we upsample the grid to $50 \times 50$ using a cubic B-spline to remove the artifacts of the bilinear interpolation.

**iOS technologies used.** 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 `CoreAnimation` framework. 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 size of $2048 \times 2048$ can be used and we thus rescale the images to this resolution. For the export at the original image resolution, we render multiple images and stitch them together. The class `GLKView` takes care of the render buffer management and the `EAGLContext` manages the OpenGL state when we draw into several seperate views. Finally, we use mipmaps to minimize aliasing artifacts for high resolution images.

## 6 Results

We implemented our prototype in Objective-C and tested it on an iPhone 5 and an iPad 4. When painting the saliency, resizing an image or tweaking the parameters, we achieved a steady frame rate of 60 fps. This allows for a fluid and interactive user experience. About 60% of the CPU time is spent in the CVXGEN solver, while the rest is spent assembling the energy matrix, performing the bilinear interpolation using OpenGL and in UI-related computations. The generation of the automatic saliency map (Section 3.3) takes about a second, where 0.6 seconds are spent on the iOS face detection and 0.3 on the dilation operation on the image gradient.

We recorded the time required to process the entire RETARGETME [RGSS10] dataset, measuring an average of 90 seconds per image. This includes the computation of the automatic saliency map, its interactive refinement and the generation of the full-resolution, retargeted image.

We also found our prototype to be engaging and fun for people who were previously unaware of image retargeting. A short one-minute introduction was sufficient to get them started and let them explore the functionalities of the prototype on their own.

**RETARGETME benchmark.** We retargeted all the 86 images in the RETARGETME benchmark [RGSS10] using our method. Figure 13 compares a few of the results with other operators. The full comparison table is provided in the additional material. In all images, we used the default parameter values and manually adjusted the automatically generated saliency map. We used the default brush size and did not zoom into the image, so we just painted what we

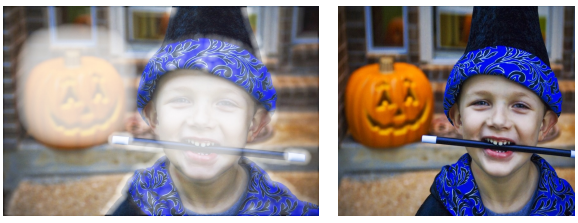



**Figure 12:** *Automatic retargeting. (a) Automatically detected saliency map. Note that also the face of the pumpkin is recognized and marked as salient. (b) Retargeted to 75% width, as in the* RETARGETME *benchmark.*

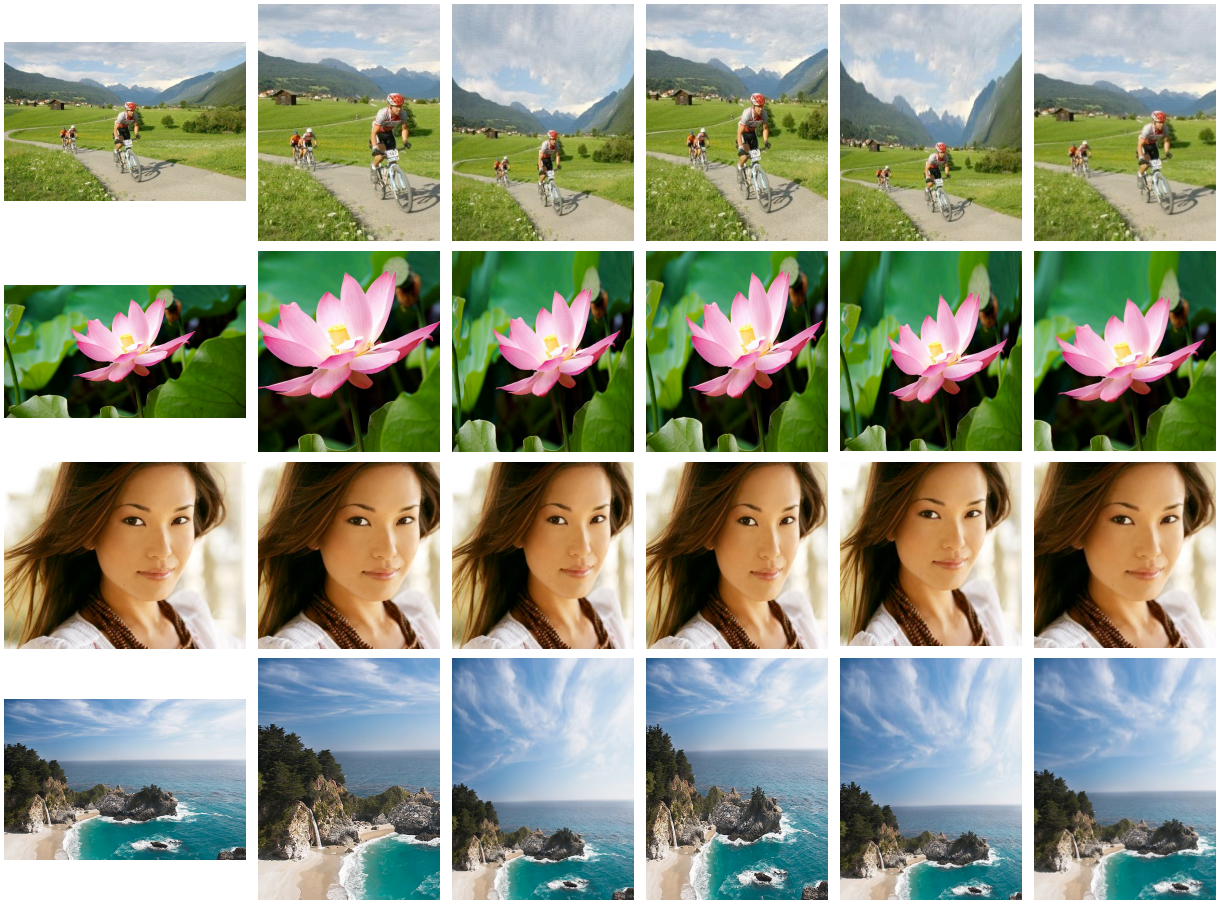



**Figure 13:** RETARGETME *benchmark comparison. We compare manual cropping (CR), streaming video (SV) [KLHG09], multi-operator media retargeting (MULTIOP) [RSA09] and axis-aligned image retargeting (AAIR) [PWS12] with our approach.*

could see on the display of the iPad. In the RetargetMe study [RGSS10], the saliency map has been manually optimized for each operator and for each picture to produce optimal results and a fair comparison. In general, the automatic saliency detection works well for images with a small depth of field, i.e. with a blurred background, for which manual refinement was not necessary (Figure 12). High-frequency backgrounds (like leafs, grass or geometric patterns on buildings) are challenging for our automatic saliency and we found it easier to paint the saliency from scratch in these cases.

**Thumbnail gallery.** A common problem in the generation of a grid-view for a collection of images is that not all images have the same aspect ratio. 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 methods are used in the iOS picture gallery (Figure 14 (a)), in the iOS image picker and in many web galleries (e.g. Picasa, 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 where we use squared thumbnails to fill all available screen space. Inside this square area, we render a retargeted image using our approach. With our algorithm, the amount of cropping and scaling is automatically selected, as shown in Figure 14 (b).

## 7 Concluding remarks

We presented a novel content-aware image retargeting operator that combines axis-aligned scaling with cropping to generate results on par with the state of the art at interactive rates. We implemented our algorithm for iOS and proposed a novel user interface optimized for touch input and small-screen devices. Our prototype provides a smooth experience and runs at 60 fps on the current generation of iOS devices.

To keep the computation cost low, we used a greedy approach to decide which part of the image should be cropped.


**Figure 14:** *iOS picture gallery (a) and our gallery (b).*

Especially in images where the salient content is concentrated in an area smaller than the target image, our operator might crop more than necessary. While these cases can be corrected manually with a few strokes, it would be interesting to find a non-greedy way of introducing crop into the retargeting operator. Similarly to [PWS12], our operator reduces to uniform scaling if the saliency map wants to preserve straight line features that are not axis-aligned. As our approach only crops whole rows and columns of the grid, the coarse grid resolution of $25 \times 25$ cells, which is needed to obtain real-time performance, restricts the precision of the cropping. In future work, we plan to investigate the possibility to continuously crop parts of a single row or column.

It would be interesting to integrate our retargeting operator into a mobile web browser to dynamically optimize the images and their layout on any webpage for a given screen size. Another possible direction for future 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.

## References

[AS07] AVIDAN S., SHAMIR A.: Seam carving for content-aware image resizing. *ACM Trans. Graph. 26*, 3 (2007). 2

[BSFG09] BARNES C., SHECHTMAN E., FINKELSTEIN A., GOLDMAN D.: PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph. 28*, 3 (2009). 2

[CFK*10] CHEN R., FREEDMAN D., KARNI Z., GOTSMAN C., LIU L.: Content-aware image resizing by quadratic programming. In *Proc. NORDIA* (2010). 2

[CJG11] CASTILLO S., JUDD T., GUTIERREZ D.: Using eye-tracking to assess different image retargeting methods. In *Proc. APGV* (2011), pp. 7–14. 2

[CZM*11] CHENG M.-M., ZHANG G.-X., MITRA N. J., HUANG X., HU S.-M.: Global contrast based salient region detection. In *Proc. IEEE CVPR* (2011), pp. 409–416. 2

[DDN08] DESELAERS T., DREUW P., NEY H.: Pan, zoom, scan – time-coherent, trained automatic video cropping. In *Proc. CVPR* (2008). 2

[GSCO06] GAL R., SORKINE O., COHEN-OR D.: Feature-aware texturing. In *Proc. EGSR* (2006), pp. 297–303. 2

[IKN98] ITTI L., KOCH C., NIEBUR E.: A model of saliency-based visual attention for rapid scene analysis. *IEEE Trans. Pattern Anal. Mach. Intell. 20* (1998). 2

[KFG09] KARNI Z., FREEDMAN D., GOTSMAN C.: Energy-based image deformation. In *Proc. SGP* (2009). 2

[KLHG09] KRÄHENBÜHL P., LANG M., HORNUNG A., GROSS M.: A system for retargeting of streaming video. *ACM Trans. Graph. 28*, 5 (2009). 2, 7

[KWSH*13] KAUFMANN P., WANG O., SORKINE-HORNUNG A., SORKINE-HORNUNG O., SMOLIC A., GROSS M.: Finite element image warping. *Comput. Graph. Forum 32*, 2 (2013). 2

[LG06] LIU F., GLEICHER M.: Video retargeting: automating pan and scan. In *Proc. Multimedia* (2006), pp. 241–250. 2

[MB12] MATTINGLEY J., BOYD S.: CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering* (2012), 1–27. 3

[PKPH12] PERAZZI F., KRÄHENBÜHL P., PRITCH Y., HORNUNG A.: Saliency filters: Contrast based filtering for salient region detection. In *Proc. CVPR* (2012). 2

[PKVP09] PRITCH Y., KAV-VENAKI E., PELEG S.: Shift-map image editing. In *Proc. ICCV* (2009). 2

[PWS12] PANOZZO D., WEBER O., SORKINE O.: Robust image retargeting via axis-aligned deformation. *Computer Graphics Forum 31*, 2 (2012), 229–236. 1, 2, 3, 6, 7, 8

[RGSS10] RUBINSTEIN M., GUTIERREZ D., SORKINE O., SHAMIR A.: A comparative study of image retargeting. *ACM Trans. Graph. 29*, 5 (2010). 1, 2, 6

[RSA08] RUBINSTEIN M., SHAMIR A., AVIDAN S.: Improved seam carving for video retargeting. *ACM Trans. Graph. 27*, 3 (2008). 2

[RSA09] RUBINSTEIN M., SHAMIR A., AVIDAN S.: Multi-operator media retargeting. *ACM Trans. Graph. 28*, 3 (2009). 2, 7

[SCSI08] SIMAKOV D., CASPI Y., SHECHTMAN E., IRANI M.: Summarizing visual data using bidirectional similarity. In *Proc. CVPR* (2008). 2

[SSHSH12] SHAMIR A., SORKINE-HORNUNG A., SORKINE-HORNUNG O.: Modern approaches to media retargeting. In *ACM SIGGRAPH ASIA Courses* (2012). 1

[VJ04] VIOLA P., JONES M. J.: Robust real-time face detection. *Int. J. of Computer Vision 57*, 2 (2004), 137–154. 2, 5

[WLSL10] WANG Y.-S., LIN H.-C., SORKINE O., LEE T.-Y.: Motion-based video retargeting with optimized crop-and-warp. *ACM Trans. Graph. 29*, 4 (2010). 2

[WTSL08] WANG Y.-S., TAI C.-L., SORKINE O., LEE T.-Y.: Optimized scale-and-stretch for image resizing. *ACM Trans. Graph. 27*, 5 (2008), 118. 2

[YSWL11] YU-SHUEN WANG JEN-HUNG HSIAO O. S., LEE T.-Y.: Scalable and coherent video resizing with per-frame optimization. *ACM Trans. Graph. 30*, 4 (2011). 2

[ZCHM09] ZHANG G.-X., CHENG M.-M., HU S.-M., MARTIN R. R.: A shape-preserving approach to image resizing. *Comput. Graph. Forum 28*, 7 (2009), 1897–1906. 2