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

Transfusive Weights for Content-Aware Image Manipulation

Kaan Yücer[1,2] Alexander Sorkine-Hornung[2] Olga Sorkine-Hornung[1]

[1]ETH Zurich [2]Disney Research, Zurich

Abstract

Many image editing operations, such as colorization, matting or deformation, can be performed by propagating user-defined sparse constraints (e.g. scribbles) to the rest of the image using content-aware weight functions. Image manipulation has been recently extended to simultaneous editing of multiple images of the same subject or scene by precomputing dense correspondences, where the content-aware weights play a core role in defining the sub-pixel accurate image warps from source to target images. In this paper, we expand the range of applications for content-aware weights to the multi-image setting and improve the quality of the recently proposed weights and the matching framework. We show that multiple images of a subject can be used to refine the content-aware weights, and we propose a customization of the weights to enable easily-controllable interactive depth segmentation and assignment, image matting and deformation transfer, both in single- and multi-image settings.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation

1 Introduction

Content-aware weight functions are a fundamental tool for image processing and manipulation applications. In such applications, the user defines constrained regions on the image, usually in form of scribbles or points, and scalar weight functions are computed around these constraints. The weight functions have their maximum value of 1 at the constraints and fade out to 0 away from the constraints, varying smoothly inside coherent image regions and falling off sharply at strong edges, which makes them "content-aware". Such weights are used to propagate operations on a small set of scribbles to the entire image, usually by linearly interpolating their effects. Colorization of black&white images [LLW04], tonal adjustments [LFUS06], matting, segmentation and compositing of images/objects [LLW06, FFL10, MB95] all fall into this category. Spatially varying weights have also been used for extracting novel information out of single images with the help of user interaction, e.g. depth maps [WLF*11, GSCO12].

With the ever-expanding digital image collections, it is of interest to simultaneously process multiple images of the same subject, taken from different viewpoints, under different illumination conditions and possibly with non-rigid deformations, like facial expressions. Recent research leverages the intrinsic coherence between images for automatic propagation of local [HJDF10, YJHS12] and global [HSGL11, HSGL13] edits to multiple views. Global editing on multiple images can be achieved by computing and generalizing sparse correspondences [HSGL11]. For simultaneous local editing of multiple images or video frames, an important part of the problem is finding correct *dense* mappings between different views [HJDF10, RAKRF08, LJH10]. In this context, the recently proposed transfusive image manipulation (TIM) [YJHS12] enables local editing of multiple images by computing a sub-pixel accurate mapping between multiple views of a given object or region of interest. TIM uses special content-aware weight functions to define a warp from a source image to a given target, where local affine transformations are linearly blended into a global warp by the weight functions and optimized using the Lukas-Kanade (LK) framework [LK81].

We observe that the weight functions proposed in TIM, which have been demonstrated to work with automatically sampled point constraints, can be combined with a scribble-based user interface. Hence, they can serve as a tool for a variety of other image manipulation applications apart from matching, such as matting, image deformation, depth segmentation and depth assignment. These applications require the content-aware weights to be adjusted according to the particular interactive setting. Further, the TIM matching framework enables to apply such manipulations to multiple images simultaneously, and even to refine the content-aware weights themselves, allowing us to *infer better weights* based on multiple images of the same scene. We notice that the assumptions on the required properties of the weight functions previously made by TIM are too strong, and by relaxing them and introducing a more sophisticated execution of the LK optimization we can improve the generated correspondences.

The contributions of this paper are: (i) We improve the image correspondence framework of TIM, relaxing the assumptions on the underlying weight functions and transformations, which leads to better matching warps; (ii) we show how the content-aware weight functions can be refined based on multiple images of the same scene, making the weights more suitable for image manipulation applications; (iii) we propose single- and multi-image matting, multi-image deformation and depth assignment applications, all based on the improved, transferrable weights. For the depth assignment application, we show how minimal user input can be incorporated into the weights using inequality constraints, which allows for easy, controllable depth segmentation.

2 Related work

We briefly review recently proposed content-aware weights and dense image alignment methods relevant to our approach.

Content-aware weights. Central to image manipulation tools is the computation of scalar weight functions, or interpolation bases, anchored at constrained image regions, for example scribbles. As surveyed e.g. in [LAA08, FFL10], the weights are usually the solutions of a harmonic PDE, $L\mathbf{w} = 0$, constrained so that the weight value is 1 on a particular scribble and 0 on all others. L is an image-based, weighted Laplace matrix. Depending on the application, this matrix is constructed locally (sparse L) or globally (dense L). In case of a global construction, L can capture the similarity between distant pixels, resulting in weight functions with global support that can be used for editing parts of images that are spatially disjoint [FFL10]. The L matrix can also be filled in by using local similarity between neighboring pixels, which leads to local information flow to nearby pixels. These can be used for methods like colorization [LLW04], tonal adjustment [LFUS06] or edge-aware multiscale image decomposition [FFLS08]. TIM [YJHS12] uses the higher-order bi-Laplacian L^2 to get C^1 smoothness of the weights at the constraints (single points in their case); the content-aware bi-Laplacian is obtained from the discrete metric of the image manifold. The resulting weight functions have been shown to work with automatically sampled control handles, and we demonstrate how they can be used with user interaction.

Content-aware weights can be combined with sparse user input to assign depth values to images (e.g. for 2D-to-3D conversion) [GSCO12, WLF*11, SSJ*10]. User interaction has been used in form of absolute or relative input: In absolute depth labeling [GWCO09, WLF*11], the user marks absolute depth values for parts of images, and these are propagated to unknown parts using content-aware weights. Relative depth ordering can be inferred from a single user [SSJ*10], where simple relations are used to create smooth depth transitions, or from crowd sourcing [GSCO12], where the results of basic ordering tasks are gathered and used to augment a Laplace equation to compute depth maps. We show how TIM weights can be efficiently combined with a low number relative constraints to solve ordering problems with minimal user input.

Image alignment. To transfer edits or other image information (e.g. mattes) to new views of a scene, correspondence between the different images need to be estimated. If the input is in form of dense frames of a video, it is possible to use optical flow techniques [LLW04, RAKRF08, BWSS09], which track dense features, but these can only be used with small image displacements. Other methods are appropriate for sparse image sets where displacements can be larger. Some use machine learning to detect and label features without computing pixel-accurate correspondence [BLDA11]. Partial, non-rigid correspondences can be used to transfer global color or blur edits between images [HSGL11], or to globally optimize color consistency across images [HSGL13], but they cannot be used for precise, localized editing due to potential holes in the registration. Sub-pixel accurate continuous

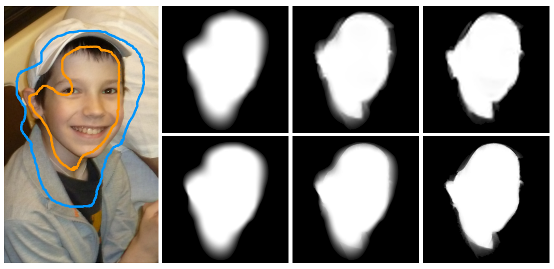

Figure 1: *Weight functions for different image resolutions and factors γ (Eq. (1)) in a matting application. The foreground and background weights have been computed from the two user-provided scribbles. The top row images have been processed at 20% of the resolution of the bottom row. From left to right: $\gamma = 0$ (no content-awareness), $\gamma = 0.15$ (weak content-awareness), $\gamma = 15$ (strong content-awareness).*

matching can be done with feature matching on local planar patches [HJDF10], or more generally using controllable image warps, as in TIM [YJHS12]. TIM parameterizes a non-rigid warp by a few affine transformations, linearly blended using content-aware weights; the parameters are optimized using an efficient variant of the Lucas-Kanade [LK81] method to obtain accurate matching. TIM makes assumptions on the properties of the weights and the resulting deformations, limiting its usability in challenging settings. We show how these assumptions can be lifted by improving the image alignment optimization, leading to better quality of correspondences. The computed matchings lead to the ability of symmetrical computations between different views, as was done before in optical flow [ADPS07] for small image displacements. We show how symmetry can be exploited for large displacements to refine the extracted image information like mattes.

3 Fundamentals of transfusive image manipulation

The original transfusive image manipulation [YJHS12] (TIM) approach for finding a mapping between a source and a target image consists of two central components, which we briefly recap here before describing our contributions. Please see [YJHS12] for details.

First, the source image I_s is adaptively sampled by automatically placing a small number of control points C_k in the image. For each C_k, a weight function w_k is computed, which grabs a locally similar image region surrounding the control point while stopping at strong image edges. This results in a soft segmentation of the source image, where every pixel $\mathbf{p}$ has an associated weight value $w_k(\mathbf{p})$ for each C_k. In the second step, an affine transformation T_k is computed for each control point, such that mapping each pixel by a weighted linear combination $\mathcal{M}(\mathbf{p}) = \sum_{k=1}^{m} w_k(\mathbf{p})\, T_k\, \mathbf{p}$ aligns corresponding regions in the source image I_s and a target image I_t. Once aligned using this procedure, edits applied to I_s can be automatically applied to I_t as well. Both algorithm components are explained in more detail below.

3.1 Content-aware weights

The weight functions w_k in TIM are computed by associating with each image pixel a 5D feature vector

$$\mathbf{v}(x,y) \rightarrow (x,\ y,\ \gamma \cdot l(x,y),\ \gamma \cdot a(x,y),\ \gamma \cdot b(x,y)), \quad (1)$$

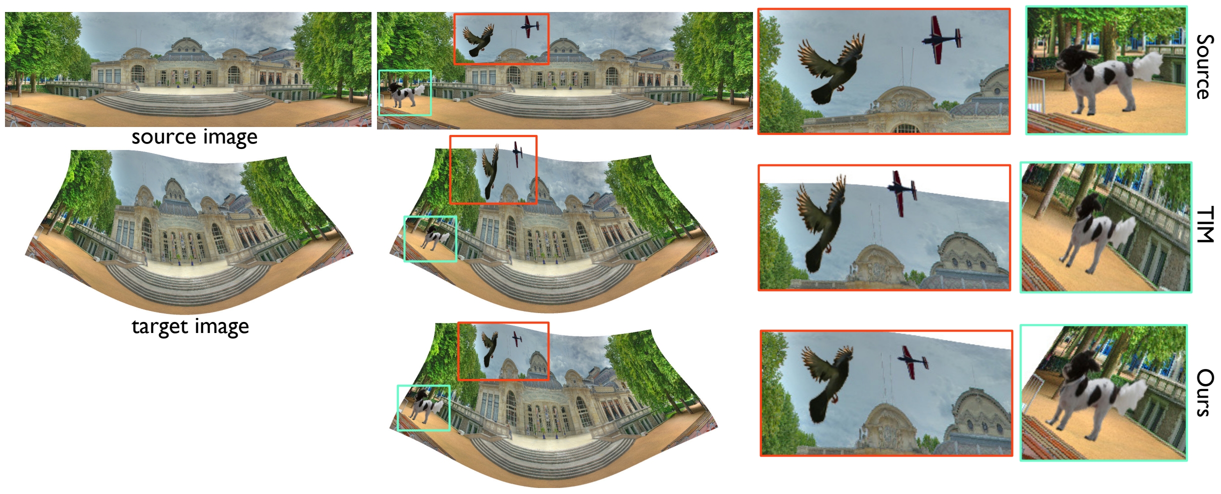

Figure 2: *The limitations of the TIM alignment framework prevent it from correctly recovering the deformations between the source and the target image. The TIM optimization gets stuck in a local minimum, leading to excessive deformations of the applied image edits (middle row). Our method alleviates this problem, resulting in correct and smooth edit propagations (lower row), as evident in the close-ups. All images via flickr. Opera de Vichy by Nicolas Ledez, bird by shutter41, plane by Denis Dervisevic and dog by Mike Baird.*

consisting of the 2D pixel position, its color in the CIELAB color space, and a weighting factor γ. The weights w_k are then computed by minimizing a bi-Laplacian energy subject to interpolation constraints at the control points C_k, partition-of-unity, and positivity:

$$\underset{w_k,\ k=1,\dots,m}{\arg\min} \quad \sum_{k=1}^{m} \tfrac{1}{2} \int_I (\Delta w_k)^2 \,\mathrm{d}x\mathrm{d}y \qquad (2)$$

$$\text{s.t. } w_k|_{C_\ell} = \delta_{k\ell}, \quad k,\ell = 1,\dots,m \qquad (3)$$

$$\sum_{k=1}^{m} w_k(\mathbf{p}) = 1 \qquad \forall \mathbf{p} \in I \qquad (4)$$

$$0 \le w_k(\mathbf{p}) \le 1, \ \ k = 1,\dots,m, \qquad \forall \mathbf{p} \in I. \qquad (5)$$

Content-awareness is achieved by taking Δ as the Laplace-Beltrami operator of the image viewed as a 2-manifold embedded in the above 5D feature space.

As also noted in [JS12], it is straightforward to employ the same optimization procedure for computing content-aware weight functions for more general image editing tasks. For instance, for scribble-based colorization or matting applications, each set of user-provided scribbles simply becomes a constrained region C_k. Figure 1 shows a matting example where the user placed one scribble on the foreground and one on the background region of the input image. The images on the right show a comparison of how different choices of γ and different image resolutions influence the sensitivity of the weight functions w_k to image content. Unless stated otherwise, we always use $\gamma = 15$ in our experiments.

3.2 Image alignment

As mentioned above, the original TIM approach defines the per-pixel mapping $\mathcal{M}$ between I_s and I_t as a weighted sum of transformations $\mathcal{M}(\mathbf{p}) = \sum_{k=1}^{m} w_k(\mathbf{p})\, T_k\, \mathbf{p}$. The transformations T_k are optimized to minimize the color mismatch function ϕ between the source and the re-mapped target image

$$\underset{\mathcal{M}}{\arg\min} \sum_{\mathbf{p}\in I_s} \phi\big(I_t(\mathcal{M}(\mathbf{p})), I_s(\mathbf{p})\big). \qquad (6)$$

To solve this problem, the iterative Lucas-Kanade method [BM04] is used, which at each iteration computes an update step for the affine transformations, ΔT_k, and then requires the following warp update:

$$\mathcal{M}(\mathbf{p}) \leftarrow \sum_{k=1}^{m} \Big[w_k(\mathbf{p})\, T_k \big(\sum_{\ell=1}^{m} w_\ell(\mathbf{p}) \Delta T_\ell \big)^{-1} \Big] \mathbf{p}. \qquad (7)$$

One problem in computing this update is that the inversion of the inner term does not necessarily lie in the linear subspace of the underlying deformation model anymore, i.e., it can no longer be expressed as a linear combination of some affine transformations with weights w_k. In the TIM approach this problem is resolved with a number of strong assumptions about the properties of the weight functions and the update of transformations, so that the inner sum can be replaced by a single transformation:

$$\mathcal{M}(\mathbf{p}) \leftarrow \sum_{k=1}^{m} \Big[w_k(\mathbf{p}) T_k \big(\Delta T_k\big)^{-1} \Big] \mathbf{p}. \qquad (8)$$

While this approximate solution was sufficiently robust for the editing tasks shown in [YJHS12], we demonstrate in the following section that the underlying assumptions may be violated in more challenging scenarios.

4 Generalized transfusive image manipulation

In the following, we present a new solution to the image alignment optimization, which leads to more robust matching of images. We then introduce two extensions to the weight computation, which enable multi-image refinement of weights and the definition of relative constraints, with applications to matting and depth map creation.

4.1 Robust warp projection

The projection of Eq. (7) back into the linear subspace using Eq. (8) assumes that a pixel is either (i) dominated by a single weight function w_k, so that other weight functions

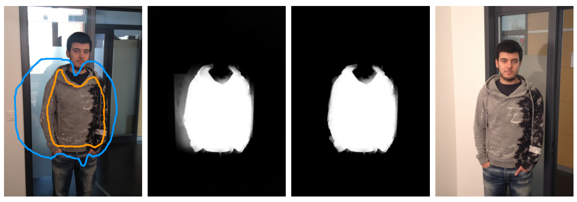

Figure 3: *Multi-image weight refinement for matting. The inaccurate scribbles (left) cause the weights to spill onto the background (middle left). Our multi-view weight refinement can automatically take additional views (right) into account and improve the weights (middle right) without the need for additional user interaction.*

are negligible and the projection can be derived as the inverse of the associated transformation T_k, or that (ii) a pixel influenced by multiple weight functions receives a similar transformation from each of them, so that the inverse can again be approximated by a single T_k [YJHS12]. However, in more challenging image matching and editing scenarios, constraints sharing common image regions may require independent transformations, preventing the alignment algorithm from finding a good solution, as shown in Figure 2.

We relax the above assumptions and instead project the warp back into the linear subspace by representing the inverse in Eq. (7) again as a weighted sum of transformations:

$$\left[\sum_{\ell=1}^{m} w_\ell(\mathbf{p})\,\Delta T_\ell\right]^{-1} \approx \sum_{\ell=1}^{m} w_\ell(\mathbf{p})\,A_\ell \quad \forall \mathbf{p}. \tag{9}$$

The task is now to find suitable A_ℓ. Using the definition of the inverse, multiplying the transformations above should lead to the identity transform:

$$\left[\sum_{\ell=1}^{m} w_\ell(\mathbf{p})\,\Delta T_\ell\right] \cdot \left[\sum_{\ell=1}^{m} w_\ell(\mathbf{p})\,A_\ell\right] \mathbf{p} \;=\; \mathbf{p} \;\; \forall \mathbf{p}. \tag{10}$$

This corresponds to an overdetermined system of linear equations with unknowns A_ℓ which we solve in the least squares sense. Note that this provides us with a different approximation to the inverse required to solve the image alignment problem, without the limiting assumptions required in the original TIM approach.

The next step is the actual application of this warp update, which may again lie outside our linear deformation subspace. We apply a similar idea as above and represent the warp in Eq. (7) as

$$\left[\sum_{k=1}^{m} w_k(\mathbf{p})\,T_k\right] \cdot \left[\sum_{\ell=1}^{m} w_\ell(\mathbf{p})\,A_\ell\right] \mathbf{p} \;\approx\; \sum_{i=1}^{m} w_i(\mathbf{p})\,Q_i\,\mathbf{p},\ \forall \mathbf{p},$$

and determine the unknowns Q_i again via linear least squares minimization.

As demonstrated in Figure 2, this new projection step is more general and can handle more challenging deformations than the original TIM approach, thanks to dropping assumptions regarding similarities of transformations of spatially adjacent weight functions.

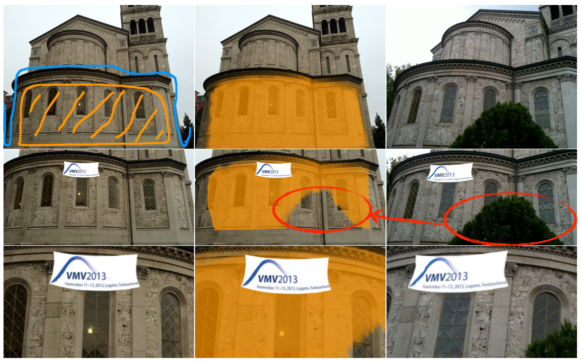

Figure 4: *Multi-image weight refinement in case of occlusions. In this example the user wants to ensure that a logo placed in the orange region of the source image (left and middle column) is not occluded when transferred to an image from another perspective (right column). The weight refinement supports the user by automatically identifying the part of the wall that is visible in both images (center image).*

4.2 Multi-image weight refinement

The quality of any method for computing content-aware weight functions for image editing is limited by the accuracy of the user input and the ambiguity of colors between different semantic regions. This is illustrated in the matting example in Figure 3 using TIM weights where the segmentation of the shirt is compromised by inaccurate scribbles and the color similarity of the reflection in the window. A similar problem occurs with partial occlusions is shown in Figure 4. However, instead of computing weights only on the source image, the computed mappings to additional target images can be exploited to resolve such ambiguities and improve the weights.

The idea of our multi-image weight refinement is to resolve such inconsistencies by computing weight functions in both images simultaneously and then strengthening those pixels on which both weight functions agree, while eliminating pixels without consensus. The refinement procedure works as follows:

1. Given a set of user-defined constraints C_k in image I_s, we first compute the content-aware weight functions w_k.
2. With the warp computation of the TIM framework, we compute warp(s) $\mathcal{M}$ from I_s to one (or more) additional images I_t around automatically generated control handles.
3. Using $\mathcal{M}$, the constrained regions C_k from I_s are warped to I_t. We mark (dis)occluded pixels in the target images and remove them from the user-defined constraints C_k. For those warped constraints C'_k we then compute weight functions w'_k in I_t.
4. The refined weights $\hat{w}_k(\mathbf{p})$ for a pixel $\mathbf{p} \in I_s$ can then be computed by warping the weight functions w'_k back to I_s and multiplying them with the weight functions w_k:

$$\hat{w}_k(\mathbf{p}) = w_k(\mathbf{p})\,w'_k(\mathcal{M}(\mathbf{p})). \tag{11}$$

Re-normalization restores the partition of unity property. The refined weights $\hat{w}_k$ are readily available in all other images I_t thanks to the computed warps $\mathcal{M}$.

4.3 Relative constraints

Scribble-based interfaces generally do not allow for the definition of relative dependencies between scribbles. However, in applications such as scribble-based depth assignment [GWCO09, WLF*11], relative constraints in form of depth inequalities have been shown to be a very powerful tool [SSJ*10]. For instance, it is not always clear at first sight what absolute depth value an image region should have. Depth differences between regions, on the other hand, are usually easier to spot and can be helpful in computing the weight functions. Humans are also more sensitive to relative changes than absolute values [GSCO12], which makes relative orderings an easier task than absolute value decisions. The flexible formulation of the weight computation allows us to incorporate inequality constraints.

In general the inequality constraints can then be defined as

$$d_{\mathbf{p},\mathbf{p}'} \leq w_k(\mathbf{p}) - w_k(\mathbf{p}'), \qquad \forall \mathbf{p} \in C_i, \quad \forall \mathbf{p}' \in C_j, \tag{12}$$

where C_i and C_j are the image regions described by the user scribbles and $d_{\mathbf{p},\mathbf{p}'}$ is a variable defining by how much the weights for pixels $\mathbf{p}$ and $\mathbf{p}'$ should differ inside these domains. Equality constraints for pairs of pixels belonging to the same scribble can be included similarly as

$$|w_k(\mathbf{p}) - w_k(\mathbf{p}')| \leq \varepsilon, \qquad \forall \mathbf{p}, \mathbf{p}' \in C_i, \tag{13}$$

which means that all pairs from the same constraint C_i should have similar weights. By not enforcing absolute equality using the threshold ε, we add an additional level of flexibility for the solver during the computation of weight functions, which prevents the often undesirable "piecewise-constant" look of depth maps created using scribble-based interfaces.

The above, general formulation renders the problem computationally too expensive due to the high connectivity in the associated constraint graph (Figure 5 left), which depends quadratically on the number of scribbles and pixels. However, we can dramatically simplify this problem and the number of constraints by adding 2 auxiliary nodes to the constraint graph (Figure 5 middle), which reduces the complexity of the problem while keeping the constraints identical:

$$\begin{aligned} d_{\mathbf{p},\mathbf{p}'} &\leq w_k(\mathbf{p}_k) - w_k(\mathbf{p}'_k), & \mathbf{p}_k, \mathbf{p}'_k \in C_k, \quad (14)\\ \varepsilon &\geq |w_k(\mathbf{p}) - w_k(\mathbf{p}_k)|, & \forall \mathbf{p} \in C_i, \quad (15)\\ \varepsilon &\geq |w_k(\mathbf{p}') - w_k(\mathbf{p}'_k)|, & \forall \mathbf{p}' \in C_j, \quad (16) \end{aligned}$$

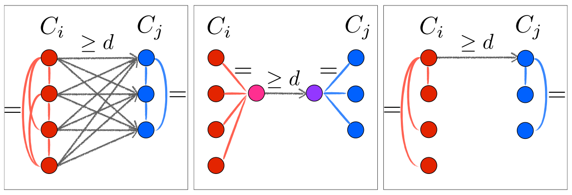

Figure 5: *Using all possible pairs of nodes as constraints makes the minimization problem computationally too expensive (left). The number of constraints can be dramatically reduced by using auxiliary variables without changing the original constraints (middle). The auxiliary nodes can then be collapsed and removed (right).*

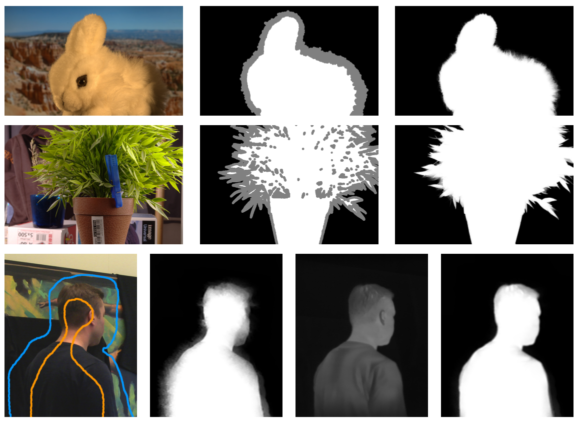

Figure 6: *Matting examples. The two upper rows show input images and trimaps from an evaluation database provided by [RRW*09] (left and middle) and our matting result using those inputs (right). In the bottom row, we captured an image with a regular camera and an aligned infrared camera. The similar fore- and backgrounds in the color input (left, with scribbles) lead to visible artifacts in the matte (middle left). We demonstrate how adding the infrared image (middle right) simply as a sixth dimension to our feature space (see Eq. (1)) considerably improves the result (right).*

where $\mathbf{p}_k$ and $\mathbf{p}'_k$ are the auxiliary nodes from the auxiliary set C_k. The auxiliary nodes can be removed by collapsing them on one of the existing nodes $\mathbf{p}_0$, thereby keeping the problem and the number of initial variables identical:

$$\begin{aligned} d_{\mathbf{p},\mathbf{p}'} &\leq w_k(\mathbf{p}_0) - w_k(\mathbf{p}'_0), & \mathbf{p}_0 \in C_i, \quad \mathbf{p}'_0 \in C_j, \quad (17)\\ \varepsilon &\geq |w_k(\mathbf{p}) - w_k(\mathbf{p}_0)|, & \forall \mathbf{p} \in C_i, \quad \mathbf{p} \neq \mathbf{p}_0, \quad (18)\\ \varepsilon &\geq |w_k(\mathbf{p}') - w_k(\mathbf{p}'_0)|, & \forall \mathbf{p}' \in C_j, \quad \mathbf{p}' \neq \mathbf{p}'_0. \quad (19) \end{aligned}$$

An important strength of this approach is that the user can now make soft decisions about relations between scribbles rather than having to assign absolute values. This is particularly useful for depth map creation using scribble-based interfaces. Moreover, we demonstrate in Section 5 how the multi-image refinement can further expedite the processes by automatically transferring inequality constraints from one image to another. The weight computation remains a sparse quadratic programming problem which is solved with [AA00].

5 Applications and results

We demonstrate our framework on a number of image editing tasks. We tested all applications on an iMac Intel Core i7 3.4GHz computer with 16GB memory.

Matting. The content-aware weights have been used with point handles in [YJHS12] and with scribbles in [JS12]. They can be further extended to work with trimaps for solving the image matting problem, see Figure 6 top. The given background (black) and foreground (white) regions are interpreted as two constrained regions C_1, C_2, and a weight function $\mathbf{w}_2$ is computed to segment the parts with unknown matte values. Although our method is not optimized specifically for matting, at the time of the submission it had an average rank of 22.7

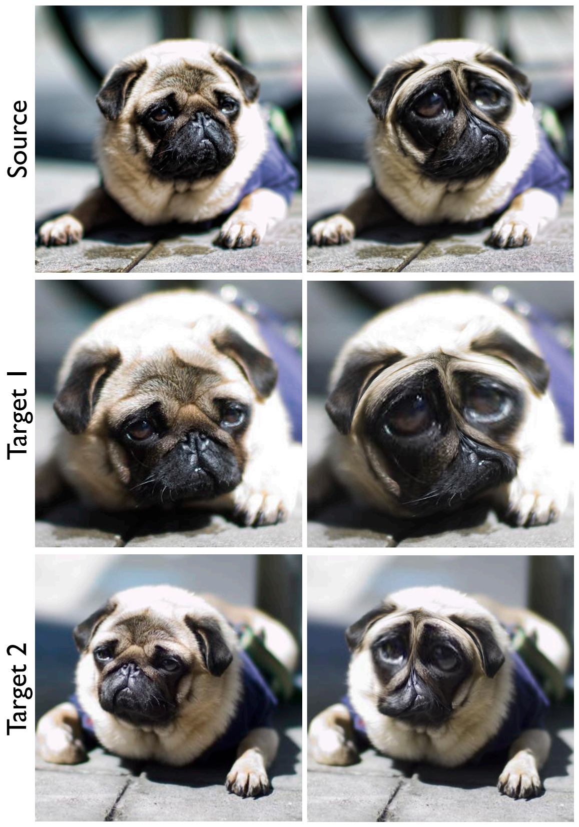

Figure 7: *Deformations of the source object (first row) can be transferred to other views of the same object. Here, the forehead of the pug is shrunk and moved upwards, its eyes are enlarged and moved closer, and the mouth region is shrunk and rotated. The mapping between the source and the targets is used to smoothly transfer the deformations to the targets. Image by *christopher* via flickr.*

in terms of MSE and 5.8 out of 33 in terms of connectivity error on the evaluation database provided by [RRW*09].

The weight computation is flexible enough to incorporate additional channels as feature dimensions (Eq. (1)), such as infrared data, which benefits matting by better identifying semantic image regions. Figure 6 (bottom) demonstrates substantial improvement of a matte by adding infrared data.

If multiple views of the same object are available, we can compute "multi-view" image mattes using our approach. The trimaps or scribbles drawn on one image can be transferred to the others and used to automatically refine the weights computed on the source image, as described in Section 4.2. Figure 3 demonstrates how this helps to differentiate between fore- and background and improves the matte. Figure 4 shows how occlusions in other views can be incorporated into the multi-view mattes. When editing multiple images simultaneously, this can be effectively used to direct the user to place important edits on parts that are visible on all available views.

Deformation transfer. Spatial warps and deformations applied to an image can also be transferred to other images of the scene using the correspondence map $\mathcal{M}$. Moreover, the linear blending formula with content-aware weights can itself be used to define expressive deformations of the image, using user-defined "deformation handles" C_k and user-provided transformations T_k for each handle, such as translation or rotation. Figure 7 (top) shows a smooth, spatially-varying warp, interactively defined in this way and applied to a source image of a pug. We used the TIM weights with $\gamma = 0.15$ for this purpose, to increase their smoothness. The deformation handles, their weights and associated transformations can then be transferred to multiple target images using our alignment framework, see Figure 7 (middle and bottom).

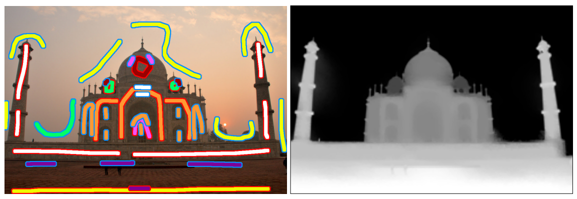

Figure 8: *Left: The input image overlaid with scribbles used for inequality constraints. Right: the resulting depth map. Scribbles belonging to the same inequality constraint are shown in matching colors. The strokes around each scribble show whether they are closer to the camera (red) or further away (blue). Observe how the inequality constraints are fulfilled in the final result. The scribbles are thickened for visualization purposes. Image by Christian Haugen via flickr.*

Depth assignment. The relative decisions and inequality constraints described in Section 4.3 enable interactive generation of depth maps with minimal user input. At each step, a new pair of scribbles are added to point out a difference in depth values and refine the depth map. The desired depth maps can be achieved in a low number of incremental steps, as shown in Figures 8 and 9. If multiple views of the same scene are available, the inequality scribbles can be automatically transferred to the additional views to compute matching depth maps there, see Figure 10. The user may only need to adjust depth differences and to add scribbles for novel regions that did not exist in the source image; usually few adjustments are needed thanks to the coherence between multiple views.

As shown in [YJHS12], Figure 2, the weight functions are rather stable over a range of different image resolutions, hence for interactive response times we compute depth on downsampled images. A low-resolution depth map (150×150) takes less then 3 seconds, whereas the time needed for a higher resolution depth map (250×250) varies between 5-15 seconds depending on the number of inequality constraints.

We compare our method with StereoBrush [WLF*11] using scribbles supplied by its authors, which uses absolute depth values as constraints to compute a depth hypothesis. We convert their absolute scribbles to relative constraints in Figure 11. Our depth maps exhibit higher edge-awareness, resulting in sharper depth discontinuities on object boundaries and decreased bleeding of depth values between different objects thanks to the diffusion characteristics of TIM weights.

Limitations. Transparent and highly reflective objects cannot be extracted in our mattes or matched properly due to the view-dependent appearance. The computation time of the content-aware weights is a limitation due to the quadratic

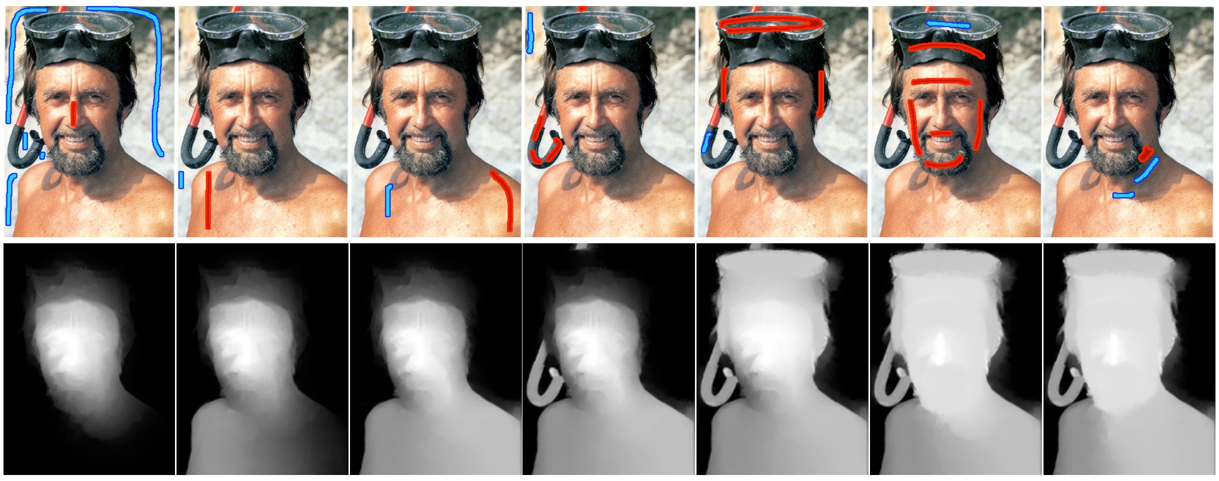

Figure 9: *Top row shows the scribbles incrementally added by the user to differentiate the depth levels. A red scribble describes the image regions that should be closer to the camera. The bottom row shows how the depth map is updated after each pair of scribbles. Note how the relative constraints are incorporated at each step to change the depth map according to the user's needs. The scribbles are thickened for visualization clarity. Image by Michael Jung via Wikimedia Commons.*

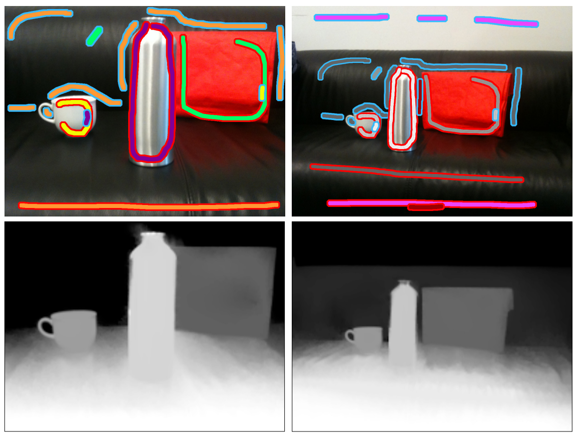

Figure 10: *The column on the left shows the source image overlaid with scribbles and its corresponding depth map. The inequality scribbles can be transferred to target views to generate their depth maps, as shown on the right. Two extra pairs of scribbles are added to label previously unseen parts.*

programming step. We use the interior point solver MOSEK [AA00], which does not benefit from initial guesses. To reduce computation time, we downsample the image for the purpose of weight computation, possibly leading to loss of precision and inability to detect fine details like fur and hair. Our LK energy minimization is more costly than in the original TIM due to the more accurate warp projection and takes 0.7 sec. compared to previous 0.2 per LK iteration on a 800×800 image with 11 control handles. It may still fail in dramatic viewpoint changes as in Figure 5 of [YJHS12]. In future work we are interested in exploring alternative numerical optimization procedures that would efficiently support incremental computation, e.g. when the user inputs additional scribbles. We are also interested in finding how to effectively update the warps with the refined weights in an iterative process.

6 Conclusions

We proposed a framework for alignment, simultaneous editing and deformation of multiple images of a scene using smooth, content-aware weight functions, resulting in a diverse multi-image manipulation toolset. We enhanced the dense alignment framework of TIM [YJHS12] by relieving their particular assumptions on the behavior of the weight functions and local deformations, thereby increasing the method's applicability. We proposed multi-image weight refinement, making the content-aware weight functions more suitable, e.g., for matting when multiple images are available. Our new (in)equality constraints enable incorporating user-defined relations into the weights, useful for depth map computation.

We see the advantage of our method in its unified mathematical foundation: variational optimization of content-aware weights and non-rigid alignment. This single scaffold can be used in a variety of single and multi-image applications; once the weights are available on the source image, they can be used for different purposes ranging from matting to multi-image deformation transfer. We hope that the ideas and applications in this paper will inspire new tools to solve different vision and graphics problems at once.

Acknowledgements

We are grateful to Alec Jacobson for insightful discussions and the authors of StereoBrush [WLF*11] for sharing their input data for comparisons.

References

[AA00] ANDERSEN E. D., ANDERSEN K. D.: The MOSEK interior point optimizer for linear programming: an implementation of the homogeneous algorithm. In *High Performance Optimization*. Kluwer Academic Publishers, 2000, pp. 197–232. 5, 7

[ADPS07] ALVAREZ L., DERICHE R., PAPADOPOULO T., SÁNCHEZ J.: Symmetrical dense optical flow estimation with occlusions detection. *Int. J. Comput. Vision 75*, 3 (2007). 2

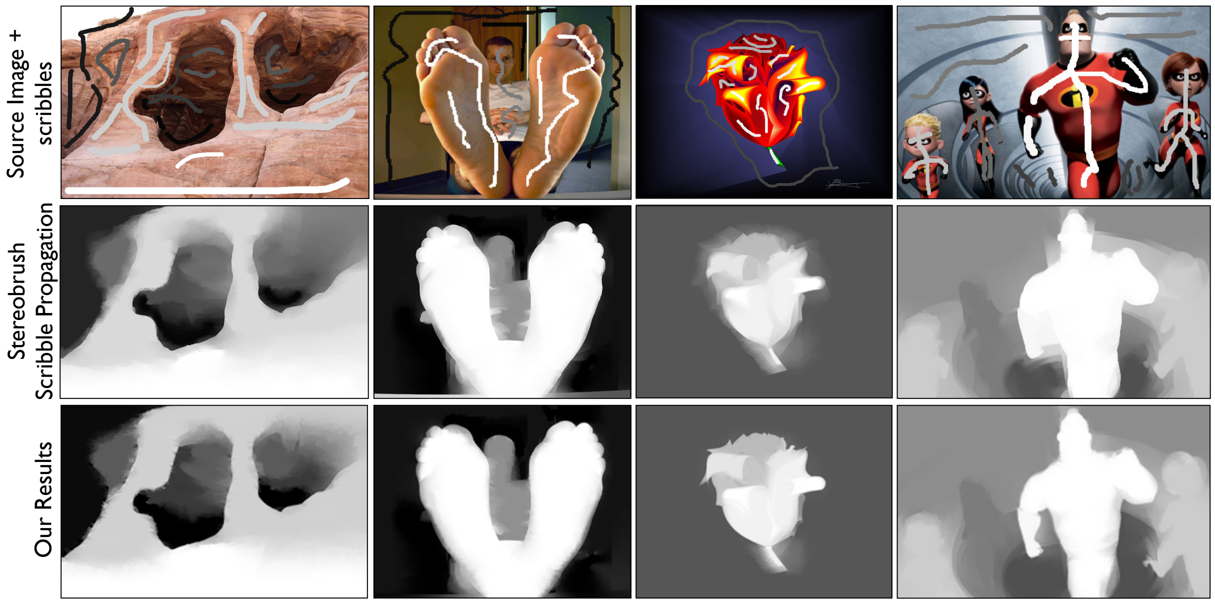

Figure 11: *Top row: input images and scribbles; second row: depth maps computed by StereoBrush [WLF*11]; third row: our depth maps, computed using relative constraints. Our algorithm respects image edges and decreases bleeding of depth values to the background or nearby pixels. Note that we show the scribble propagation results of StereoBrush computed using Levin's weights [LLW04], and not their final disparity maps, which use additional ingredients, to allow for a fair comparison.*

[BLDA11] Berthouzoz F., Li W., Dontcheva M., Agrawala M.: A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. *ACM Trans. Graph. 30*, 5 (2011), 120. 2

[BM04] Baker S., Matthews I.: Lucas-Kanade 20 years on: A unifying framework. *Int. J. Comput. Vision 56*, 3 (2004), 221–255. 3

[BWSS09] Bai X., Wang J., Simons D., Sapiro G.: Video snapcut: robust video object cutout using localized classifiers. *ACM Trans. Graph. 28*, 3 (2009). 2

[FFL10] Farbman Z., Fattal R., Lischinski D.: Diffusion maps for edge-aware image editing. *ACM Trans. Graph. 29*, 6 (2010). 1, 2

[FFLS08] Farbman Z., Fattal R., Lischinski D., Szeliski R.: Edge-preserving decompositions for multi-scale tone and detail manipulation. *ACM Trans. Graph. 27*, 3 (2008). 2

[GSCO12] Gingold Y., Shamir A., Cohen-Or D.: Micro perceptual human computation. *ACM Trans. Graph. 31*, 5 (2012). 1, 2, 5

[GWCO09] Guttmann M., Wolf L., Cohen-Or D.: Semi-automatic stereo extraction from video footage. In *Proc. ICCV* (2009). 2, 5

[HJDF10] Hasinoff S. W., Jóźwiak M., Durand F., Freeman W. T.: Search-and-replace editing for personal photo collections. In *Proc. ICCP* (2010). 1, 2

[HSGL11] HaCohen Y., Shechtman E., Goldman D. B., Lischinski D.: Non-rigid dense correspondence with applications for image enhancement. *ACM Trans. Graph. 30*, 4 (2011). 1, 2

[HSGL13] HaCohen Y., Shechtman E., Goldman D. B., Lischinski D.: Optimizing color consistency in photo collections. *ACM Trans. Graph. 32*, 4 (2013). 1, 2

[JS12] Jacobson A., Sorkine O.: *A Cotangent Laplacian for Images as Surfaces*. Tech. Rep. 757, ETH Zurich, April 2012. 3, 5

[LAA08] Li Y., Adelson E. H., Agarwala A.: ScribbleBoost: Adding classification to edge-aware interpolation of local image and video adjustments. *Comput. Graph. Forum 27*, 4 (2008). 2

[LFUS06] Lischinski D., Farbman Z., Uyttendaele M., Szeliski R.: Interactive local adjustment of tonal values. *ACM Trans. Graph. 25*, 3 (2006). 1, 2

[LJH10] Li Y., Ju T., Hu S.-M.: Instant propagation of sparse edits on images and videos. *Comput. Graph. Forum 29*, 7 (2010). 1

[LK81] Lucas B. D., Kanade T.: An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence* (1981). 1, 2

[LLW04] Levin A., Lischinski D., Weiss Y.: Colorization using optimization. *ACM Trans. Graph. 23*, 3 (2004). 1, 2, 8

[LLW06] Levin A., Lischinski D., Weiss Y.: A closed form solution to natural image matting. In *Proc. CVPR* (2006). 1

[MB95] Mortensen E. N., Barrett W. A.: Intelligent scissors for image composition. In *Proc. ACM SIGGRAPH* (1995). 1

[RAKRF08] Rav-Acha A., Kohli P., Rother C., Fitzgibbon A. W.: Unwrap mosaics: a new representation for video editing. *ACM Trans. Graph. 27*, 3 (2008). 1, 2

[RRW*09] Rhemann C., Rother C., Wang J., Gelautz M., Kohli P., Rott P.: A perceptually motivated online benchmark for image matting. In *Proc. CVPR* (2009). 5, 6

[SSJ*10] Sýkora D., Sedlacek D., Jinchao S., Dingliana J., Collins S.: Adding depth to cartoons using sparse depth (in)equalities. *Computer Graphics Forum 29*, 2 (2010). 2, 5

[WLF*11] Wang O., Lang M., Frei M., Hornung A., Smolic A., Gross M.: StereoBrush: interactive 2D to 3D conversion using discontinuous warps. In *Proc. Symp. Sketch-Based Interfaces and Modeling* (2011). 1, 2, 5, 6, 7, 8

[YJHS12] Yücer K., Jacobson A., Hornung A., Sorkine O.: Transfusive image manipulation. *ACM Trans. Graph. 31*, 6 (2012). 1, 2, 3, 4, 5, 6, 7