Retargeting Visual Data with Deformation Fields

Tim Elsner
Visual Computing Institute
RWTH University
elsner@cs.rwth-aachen.de

Julia Berger
Visual Computing Institute
RWTH University
julia.berger@rwth-aachen.de

Tong Wu
Institute of Computing Technology
Chinese Academy of Sciences
wutong19s@ict.ac.cn

Victor Czech
Visual Computing Institute
RWTH University
victor.czech@rwth-aachen.de

Lin Gao
Institute of Computing Technology
Chinese Academy of Sciences

Leif Kobbelt
Visual Computing Institute
RWTH University

Abstract

Seam carving is an image editing method that enable content-aware resizing, including operations like removing objects. However, the seam-finding strategy based on dynamic programming or graph-cut limits its applications to broader visual data formats and degrees of freedom for editing. Our observation is that describing the editing and retargeting of images more generally by a displacement field yields a generalisation of content-aware deformations. We propose to learn a deformation with a neural network that keeps the output plausible while trying to deform it only in places with low information content. This technique applies to different kinds of visual data, including images, 3D scenes given as neural radiance fields, or even polygon meshes. Experiments conducted on different visual data show that our method achieves better content-aware retargeting compared to previous methods.

1. Introduction

Media retargeting is a way to edit images, videos, 3D objects, or even entire 3D scenes by a global deformation such that relevant content, details, and features are properly preserved. The overall goal is to change the aspect ratio of the scene's bounding box so that it fits an allocated space, *e.g.* fitting an image to a display with prescribed format. Retargeting is based on the idea of identifying regions with little detail and to accumulate the necessary distortion induced by the deformation in these regions. Small editing operations are possible *e.g.* through concentrating the distortion there, specifying a region containing a particular object to have this object removed.

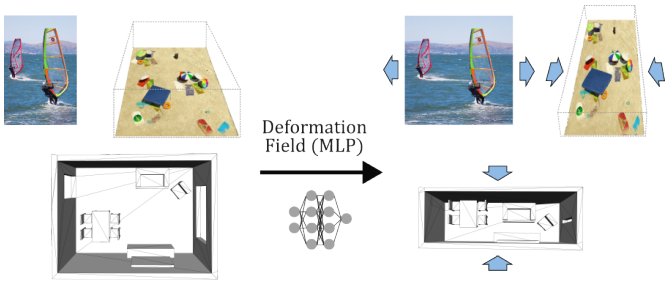

Figure 1. Different objectives applied to different visual domains with our approach: We demonstrate retarget images, NeRFs, and meshes. Image from [44].

While most existing retargeting methods typically use some form of seam carving, *i.e.* the deletion of discrete pixels [2, 8, 9, 43], we formulate the problem via a continuous deformation field that maps from the retargeted image or scene back to the undistorted input such that, *e.g.*, colours or normals can be queried. This allows us to handle discrete media (images) and continuous media (3D objects and scenes) equally.

In this paper, we introduce a more general approach for retargeting various forms of visual media, including the first approach that allows retargeting of neural radiance

fields [29] (NeRFs). We also demonstrate our approach for images and polygon meshes, as can be seen in Fig. 1. Our contributions are as follows:

(a) We introduce the use of a neural deformation field compressing or stretching low information content regions to achieve a smaller or larger output or to follow other editing objectives.

(b) We demonstrate the domain agnostic nature of our formulation by applying it to images, 3D meshes, and 3D scenes in the form of NeRFs.

(c) We show that the high flexibility of our approach from optimising a global deformation field produces better outputs than iteratively computing seams.

To achieve this, we regularise the neural deformation fields to follow general sanity guidelines, while attempt to minimize distortion in places with presumed high information content. We evaluate our results on both qualitative and quantitative levels. To showcase our approach on interesting scenes captured as NeRFs and inspired by the dataset of Richter *et al*. [41], we also provide a small synthetic NeRF retargeting dataset captured in the video game *GTA V* [34]. Our method can be applied in a matter of seconds, requires only minor modifications for vastly different applications and domains, and is straightforward to implement. We will provide the code of our method demonstrated on images on GitHub[1].

2. Related Work

As our approach is inspired by *seam carving*, it produces results from a *single example* without any learned prior, and was designed to allow seam carving-like *deformations for learned 3D representations*. We hence discuss these three branches of related work.

Seam Carving Image/video retargeting is an important tool in computer vision and computer graphics. Early works [25] first annotate or detect the region of interest (ROI), which is uniformly scaled to the target size afterwards. While contents out of the ROI are deformed by fisheye warping, Liu and Gleicher [26] later extend it to the video domain by introducing a motion salience map. Avidan and Shamir [2] propose seam carving, a content-aware image resizing method. It first defines an energy map by summing the gradient norms along the horizontal and vertical directions. The cost of a seam is defined by adding up the energy values of pixels on it. The seam with minimal energy can be found by dynamic programming and will be removed in the resizing process. Rubinstein *et al*. [42] extend the seam carving idea to the video domain and transform the dynamic programming problem to a graph cut problem. Follow-up works [8, 9, 43] incorporate homogeneous cropping and scaling with seam carving and excavate more high-level information like symmetry [51], depth [4], and object similarity [10] to assist seam carving. With the advances in deep learning, a few methods [30, 32, 33] made attempts to detect and localise the seam carving forgery. Song *et al*. [46] propose Carving-Net to replace the previous hand-crafted energy map with a neural network while the seam finding algorithm is still dynamic programming. Instead of using networks to improve seam selection, our approach replaces the discrete seams and dynamic programming itself with a neural network, globally optimising a continuous deformation field.

Single Example Generation The goal of single example generation is to discover similar local patterns from a single input and synthesise novel samples. Texture synthesis on 2D images and 3D shapes has been widely studied in computer vision and computer graphics by non-neural methods [11, 12, 23, 49]. There are also methods focusing on synthesising geometric textures for 3D shapes [5, 18, 24], which transfer local geometry details from an example shape to other surfaces. Recently, neural-based methods have been proposed to generate 2D textures [14, 45, 54] and transfer them to 3D shapes [16, 17], resulting in improved results. With recent advances in implicit representation [7, 28, 35] and neural radiance fields [29], Wu *et al*. propose to learn implicit 3D shapes from a shape with [53] or without texture [52]. To enhance the realism of synthesised shapes, Huang *et al*. [20] reconstructs the 3D shape with a NeRF by optimising texture features on a reconstructed mesh. These features can be applied to arbitrary surfaces to render the target surface with synthesised textures. These methods often only work for a certain kind of data format, while our method proposes a domain agnostic approach to synthesise visual data from a single example.

Deformations for Learned 3D Representations Neural radiance fields (NeRF), as a newly proposed 3D shape representation, has been widely used in scene reconstruction and novel view synthesis. However, the original NeRF only works for static scenes thus unable to deal with deformation in dynamic scenes or generate motions for a static scene. To overcome this, the deformation field is introduced to NeRF. NeRF-Editing [57] is the first to explore deformation in NeRF. It first reconstructs the explicit mesh of a static scene with NeuS [48] and deforms the explicit mesh with the ARAP algorithm [47]. Then sample points in volume rendering are deformed along with the mesh via barycentric coordinate interpolation, resulting in the change of rendered images. Follow-up NeRF deformation methods [13, 39, 55] utilise a similar pipeline but use different geometry prox-

[1]Prior to the final version, our code, including our NeRF retargeting dataset and example videos showing 3D scenes in motion, can be found in the supplemental material

ies. To reconstruct dynamic scenes, Albert *et al*. [40] first explore the possibility by adding an extra time-conditioned deformation field to the static NeRF. The time-conditioned deformation field transforms sample points in the observation space into the canonical space, where their colours and opacities are queried and then rendered into images. This idea stimulated a series of dynamic NeRF reconstruction methods [6, 27, 36, 37, 50, 56]. While our approach also uses a deformation field, we inject general plausibility constraints into it to keep the results plausible, enabling the idea behind seam carving on different types of visual data.

3. Deformation Fields for Retargeting Visual Data

Editing strategies for visual data require avoiding modifications on meaningful contents, which create visible artefacts. An approach that focuses on retargeting the input to match a target size can be guided by the following set of rules:

(a) *Content Aware Objective*: Prevent the accidental creation of new or the removal of existing regions with high information content (controlling **where** deformation happens).

(b) *Sanity Objective*: Avoid introducing deformations that yield implausible results (controlling **how** deformation happens).

Note that for the following sections, we only consider retargeting visual data to a smaller size (*shrinking*). This formulation can be extended naturally to provide editing (see Sec. 3.5) or to retarget to a larger size (*expansion*, see Sec. 8).

Discrete Solution Seam carving [2] performs editing by manipulating (removing or doubling) one seam of pixels at a time, *i.e*. a path with the width of one single pixel passing through the image. Through dynamic programming, seam carving chooses this path based on the lowest energy of the pixels, with high energy indicating information-rich areas. In a simple case, this energy is the colour gradient. This yields a *discrete* solution to the retargeting problem that is both content aware and sane. While this naturally extends to voxelised 3D scenes, it would become costly at high resolutions while suffering from voxelisation artefacts.

Continuous Solution Our approach follows the same guidelines, but instead of discrete, pixel-wise paths through the image, we learn a deformation function for continuous inputs. By working directly in the continuous domains (*e.g*. not requiring voxelisation of a 3D scene), our formulation is less prone to artefacts, while allowing for more degrees of freedom by accommodating non-continuous compression and expansion (see Fig. 4). We consistently utilise the same core formulation, regardless of the underlying domain and application, *e.g*. moving an object in an image or retargeting a 3D scene. For clarity, we will refer to our continuous seams as *folds*.

3.1. General Formulation

For input visual data I, we train a neural deformation field D, a simple MLP. The deformation field provides an offset to each input coordinate, *i.e*. it learns an I-specific deformation field to look up the content of I at a different place (see Fig. 2). For any point $p \in P$ of the domain of I, the deformation $D(p)$ yields a scalar offset in a deformation direction v for p. We define P to contain all possible input positions for I, *e.g*. $P = [0, 1]^2$ for a square image. Adding this scaling of v to p we obtain a deformed point p':

$$p' = p + vD(p) \tag{1}$$

This deformation field is always tuned to the visual data $I(p)$, *e.g*. a previously trained NeRF or an image. We only optimise the deformation field that gives an offset for the mapping from the edited output data to the input data. To fulfil the objectives stated in Sec. 3.1, we introduce two sets of losses:

Content Aware Objectives We apply regularisation to guide the deformation process in order to adjust the learnt deformation to the content of the data. To discourage noticeable changes in regions with high information content, we penalise the product between deformation magnitude and information content, expressed as an energy function E. The change in the deformation field between two points also change the distance between them in the output, suggesting a potentially noticeable deformation at this point. We penalise these distortions in areas with high information content. We define $E(p)$ to be a measure of content information at p, expressed as, *e.g*., the gradient in the visual data:

$$E(p) = ||\nabla I(p)||_2 \tag{2}$$

We therefore define our general energy term that measures *content deformation* as follows:

$$L_C = \int_{p \in P} [E(p + v \cdot D(p)) \cdot ||\nabla D(p)||_1] \; dp \tag{3}$$

Thus, in Eq. (3), we effectively minimise the product of information content at the deformed point as $E(p + v \cdot D(p))$ and the change in the deformation itself as $||\nabla D(p)||_1$, penalising *folds* in the deformation at regions with high information content. Using the L^1-norm for the distortion promotes piecewise constant, possibly sparse, deformations.

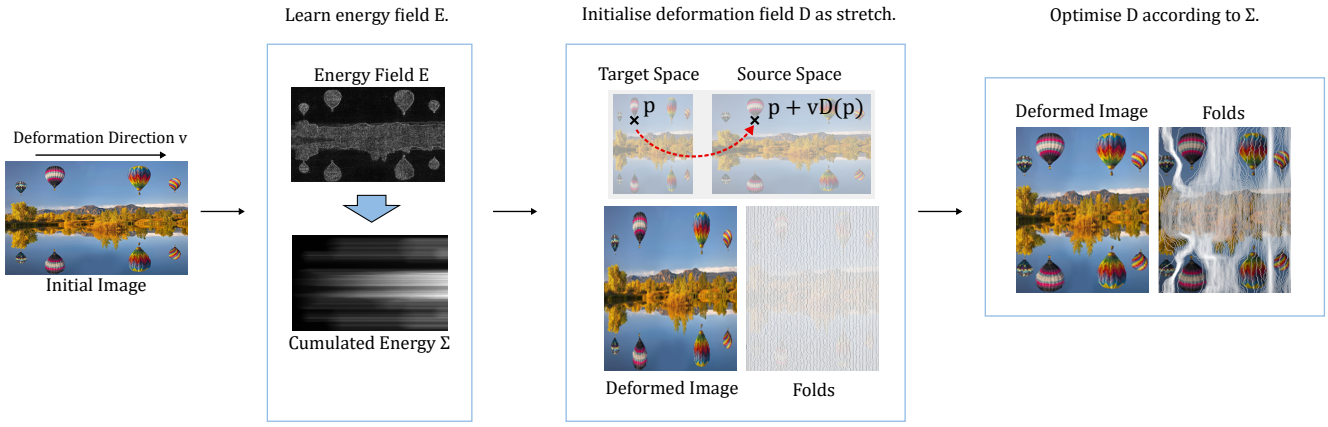

Figure 2. Our proposed pipeline for retargeting visual data: For a given input (left), we train two simple networks that learn the energy and cumulative energy along the deformation axis of the input (centre-left). We then initialise a network that stretches samples to the desired position (centre-right), then optimise this deforms to distribute the distortion to low information content regions (right). Image from [44].

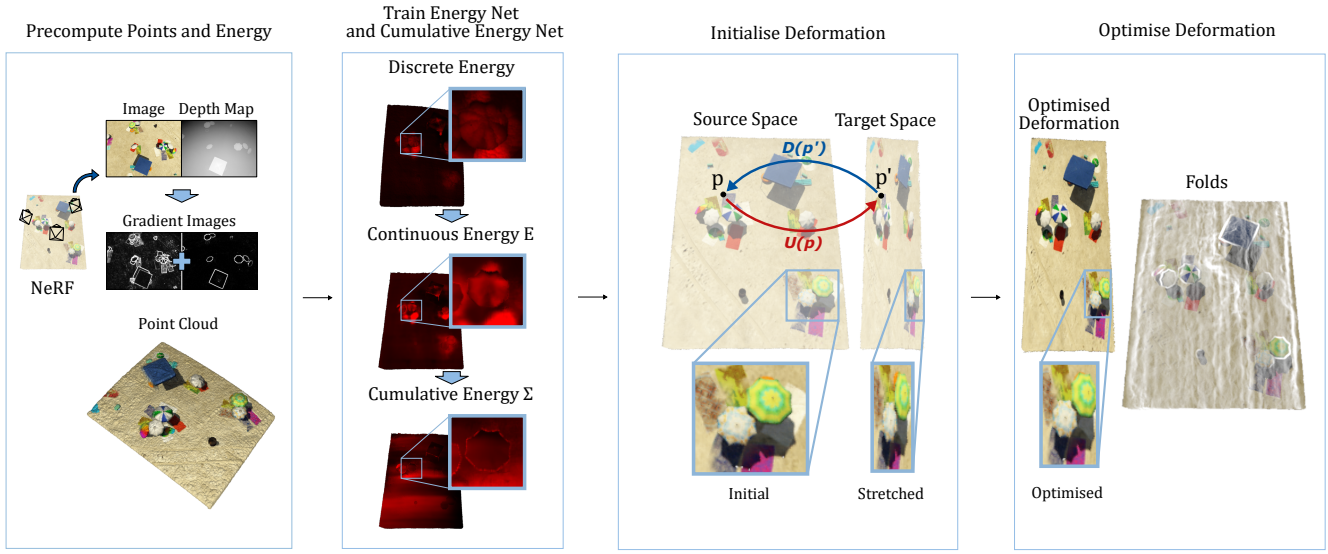

Figure 3. The part of our pipeline to apply it to neural radiance fields: We first obtain a point cloud, then estimate the energy values. We then learn a continuous energy function and a cumulative energy function that we use to optimise our deformation network. We visualise the resulting deformation field on the bottom right.

We can then split up the change in deformation to obtain:

$$L_C = \int_{p \in P} \Big[E(p + v \cdot D(p)) \cdot \left(\left| \frac{\partial}{\partial v} D(p) \right| + \left| \frac{\partial}{\partial v^\perp} D(p) \right| \right) \Big] \, dp \tag{4}$$

This equation is then split into two components, L_e penalising *stretching* or *compression* and L_s punishing *shearing* (deformation change in a direction that is not our deformation direction). For a now discrete sample set P from our domain, these terms are expressed as

$$\begin{aligned} L_e &= \sum_{p \in P} \left[E\left(p + v \cdot D(p)\right) \cdot \left| \frac{\partial}{\partial v} D(p) \right| \right] \, dp, \\ L_s &= \sum_{p \in P} \left[E\left(p + v \cdot D(p)\right) \cdot \left| \frac{\partial}{\partial v^\perp} D(p) \right| \right] \, dp. \end{aligned} \tag{5}$$

L_e directs deformation to low information content areas, while L_s discourages shearing for high information content regions. Note that while L_s would also penalise non-straight deformation seams, we tune our losses such that this only affects the output substantially if an area is sheared (*e.g.* for the balloons in Fig. 2, the deformation going

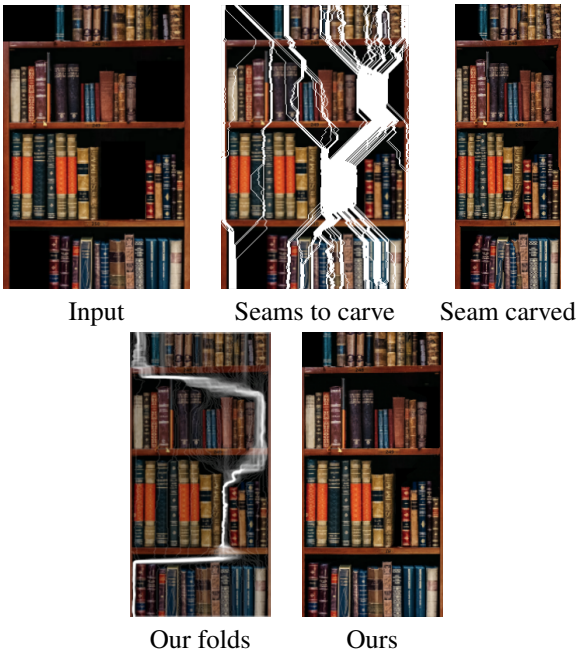

Figure 4. Example of applying retargeting to 75 percent width. Seam carving fails to use seams that are non continuous or with a too steep angle, not properly using the empty space in the shelf to close the gaps before shrinking the bookshelves content.

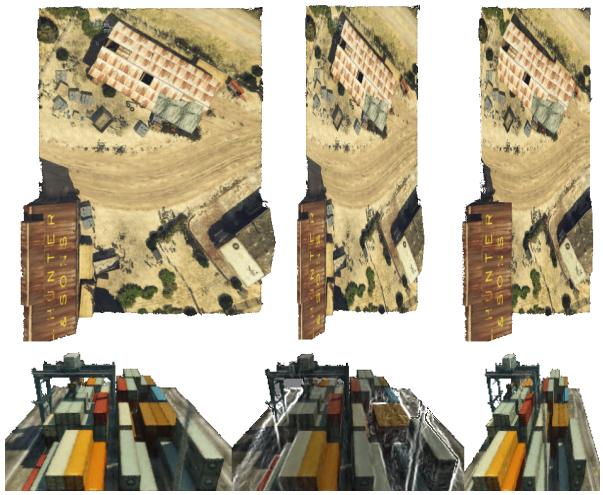

Figure 5. Applying our retargeting to NeRFs. Top row: Input, stretched, retargeted with our method. Bottom row: Input, folds on the input suggested by our approach, retargeted with our method.

around the balloons introduces shearing only right at the seam in an area with low information content).

In case of image deformation, neighbouring pixels with low energy in the input may translate to pixels widely apart in the output, potentially bypassing a substantial amount of (important) content. To overcome this, we want to penalise

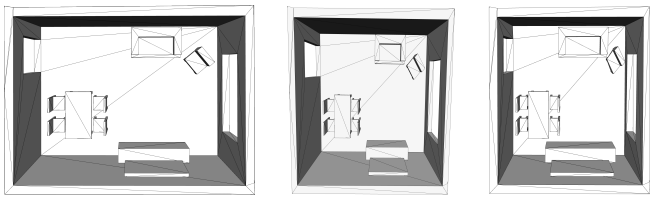

Figure 6. Applying our retargeting to 3D meshes.

the amount of energy $\hat{E}$ *between* two points p, q in the target domain, rather than the energy at individual positions:

$$\hat{E}(p,q) = \int_{p' \in [p,q]} E(p')\, dp' \tag{6}$$

Now, we re-define L_e to be the product of the energy between points and their deformation change,

$$L_e = \sum_{p \in P} \hat{E}\left(p + vD(p), p_\varepsilon + vD(p_\varepsilon)\right) \cdot \frac{|D(p) - D(p_\varepsilon)|}{\varepsilon} \tag{7}$$

where $p_\varepsilon := p + \varepsilon$ is a slightly offset p for small ε. This formulation guides the deformation to avoid introducing deformation to high information content regions or fold over them, while L_s discourages non-local deformations that would shear data with high information content.

Sanity objectives To further prevent undesired outputs, we introduce two additional losses. Assume we shrink the input by a factor $\alpha \in [0, 1]$:

1. The boundaries of the affected region should not change, *e.g.* an image should not be simply cropped. Based on α, the deformation field should map the start/end of the deformation axis in the output to the start/end of the input. For P_0, P_1 containing all points at the start/end of the deformation axis of the output, we express this requirement as follows:

$$L_b = \int_{p \in P_0} |D(p)|\, dp + \int_{p \in P_1} |D(p) - (1 - \alpha)|\, dp \tag{8}$$

2. We additionally demand monotonicity in the deformation field, expressed as

$$L_m = \int_{p \in P} \max\left(0, -\frac{\partial}{\partial v} D(p)\right) dp \tag{9}$$

indicating that any change in the deformation direction should be positive, *i.e.* the deformation $D(p)$ should be monotonic and only stagnate or increase exclusively

along v. This prevents the repetition of the same image portion ("jumping back and forth" through our deformation), which might result in bad optima.

We then obtain the total loss L with weights λ as

$$L = \lambda_e L_e + \lambda_s L_s + \lambda_b L_b + \lambda_m L_m. \tag{10}$$

To promote convergence and avoid undesirable local optima, we always initialise our deformation function with a uniform stretch to the target size. In the following, we demonstrate the application of this general formulation to different domains, adapting the energy formulation. All implementation and architecture details are given in Sec. 6 and the exact loss terms in Sec. 7.

3.2. Optimising Deformation Fields for Images

To ensure consistency with our continuous approaches, we first train a continuous image representation MLP I to serve as our input data, mapping pixel coordinates to colours.
We train a neural network to first learn a energy field E, then a cumulative energy field Σ learning to reproduce $\hat{E}$ with an MLP. The difference in cumulative energy Σ between two points p, q on the same axis v is then the absolute difference in information content between them:

$$\hat{E}(p, q) \approx |\Sigma(q) - \Sigma(p)| \tag{11}$$

For all objectives, we simply use discrete approximations of the loss terms introduced in Sec. 3.1.
The whole pipeline is visualised in Fig. 2, and extra results can be found in Tab. 4.

3.3. Optimising Deformation Fields for Neural Radiance Fields

A naive extension of the energy term from 2D to 3D as the product of density and change in colour, as given by the NeRF has two major issues: change of colour and density may not be well-defined for single points (*e.g.* consider mip-NeRF [3] only using volumes), while a large number of samples are in empty space, hence are unimportant. To address this, we propose in the following a strategy to optimise for points on the surface instead, as visualised in Fig. 3:

1. Extracting a Sparse Set of Representatives Using the RGB and depth images of applying the NeRF from the training dataset positions, we extract the surface point cloud that, by definition, contains the points with the most significant role in the deformation. We then compute the energy value for each point corresponding to a pixel as a sum of colour change in the RGB image and depth change in the depth image. As this leads to highly noisy energy values, we apply smoothing by taking the minimum energy for each point in a neighbourhood radius of 5 points.

2. Defining an Energy Function Since colour change alone is not a sufficient energy formulation, our defined energy combines colour change in the rendered image and change in the depth image. Hence, each point contributing to the rendered image is equipped with its corresponding energy value. To obtain a continuous energy function we can evaluate for every point in the domain, we train an auxiliary energy network E_{net}. The network maps all 3D points to their energy values, and clamps the energy to zero for all points further than twice the average neighbour distance. The resulting energy field has zero energy in empty space and inside objects, and non-zero energy values for points near the surface. Following the deformation axis, we additionally learn a cumulative energy field, expressed as a network E_{cnet}. The network is trained by shooting rays in the direction of deformation, and accumulating energy values from E_{net} from uniformly sampled points. Having the cumulative energy field, we can implicitly compute the energy for a segment between two points along v.

3. Inverting the Deformation We define an inverse deformation function to determine those points that lie on the surface after deformation. In contrast to images, we only perform optimisation on a sparse set of points, namely surface points. As we optimise a deformation from the *target space* to the *source space*, we need the position of surface points in both spaces. By training an MLP U to reverse the deformation, *i.e.* minimising $(U(D(x)) = x)^2$, we can effortlessly extract surface points, and hence their energy values, in the target space. This relation is also visualised in Fig. 3. We always update U once after training one iteration of D.

We therefore adapt our general approach from Sec. 3.1 for NeRFs as follows:

- We train a continuous energy function E from sparse samples on the NeRF surface, then train a cumulative energy function Σ from that.
- We use an inverse deformation network U to find points in the target space that map to surface points in the source space.
- We apply our regularisation terms on a mix of surface and random points to optimise our loss function.

As we use a discrete set of points, we apply the losses from Sec. 3.1 to Eq. (10) for a discrete interval, *i.e.* mean over a random subset of sample points. Example images from deformed scenes can be found in Fig. 5, while a larger number of results can be found in Sec. 10.

3.4. Optimising Deformation Fields for Meshes

To show the versatility of our approach, we also it to meshes, only redefining the energy term and using surface

Figure 7. We demonstrate different editing operations by simply adding and adapting the loss formulation, leading to different editing operations (always input on top, output on the bottom, green indicating a given input mask). Left: Removing an Object, Right: Moving an object. Images from [44].

samples and vertex positions to optimise. For objects like chairs, we used curvature, and for the example in Fig. 6, we simply set the energy for the floor to 0 and to 1 for everything else.

3.5. Other Applications

While retargeting is our main contribution, just as for seam carving [2], our framework opens up multiple different editing opportunities that even extend seam carving capabilities. While all details can be found in Sec. 9, the following exemplary operations were implemented through minor changes in the loss formulation, are applicable to any domain, and could form the basis for further editing methods or user-driven applications. Examples can be found in Fig. 7.

Removal To remove a part of a scene, we remove the monotonicity loss, use the same size for input and output, and add another loss term that penalises every deformation target that leads to the region, that should be removed. Hence, points originally mapping to the target region will be distorted to neighbouring regions.

Moving an Object Although classic seam carving cannot trivially move an object, since seams are assumed to cross the whole image, our formulation has no such restriction. We remove the monotonicity term, then enforce the deformation to point at the target feature from the desired new place of the object and punish it occurring anywhere else to avoid just repeating the object.

4. Evaluation

As retargeting quality is difficult to measure and lacks similar methods for *e.g.* meshes, as the backbone for our evaluation, we compare our results to seam carving [2] on images using the *RetargetMe* [44] benchmark for image retargeting. In 3D, lacking an alternative, we compare with the video-seam-carved animation frames of the same scene. To allow a fairer comparison with video seam carving, we only pan down the camera instead of rotating around the object (*e.g.* spinning the camera around an object would be disadvantageous for video seam carving). We always produce a short, 3 second clip at 30fps.

4.1. Comparisons

Quantitative Comparison In accordance with recent publications in the generative model and image editing domain [15, 21, 38], we compare our results and seam carving results with the original data using FID [19] in Tab. 1 for images. While we observe the results to align with our perception, we also validate this on a qualitative level.

Qualitative Comparison As image quality is a subjective criterion and may not align with our quantitative results, we validate our quantitative results by having humans classify which retargeted version they prefer. For this, we evaluate in a simple user study where participants were asked to classify which retargeted version of a randomly chosen image for a randomly chosen deformation axis they preferred, always choosing between our and the seam carved[2] version. We used the 80 images of the RetargetMe [44] for 2D, producing 360 variations per approach.

For retargeting quality of 3D scenes given as NeRFs, we compare ours results with what we consider to be the approach nearest of kin, video seam carving[42]. We use a 50% and a 150% version for each axis of our provided GTA V NeRF retargeting dataset.

The results can be found in Tab. 2 (2D images) and Tab. 3 (3D scenes).

Visual Ablation We provide a simple visual ablation that shows how our method requires all of its loss terms in Fig. 9. We observed similar behaviour for 3D NeRFs and meshes.

Methodological Comparison Compared to classic seam carving, our approach provides a superset of all possible solutions from discrete seam carving, as *e.g.* any removed seam can simply be expressed as a jump in the output of our deformation network. Hence, we extend the space of possible solutions. Our approach also tends to be less "jaggy" compared to individually removed seams, as neural networks fall into optima that are continuous rather than changing with high frequency. While removal of individual seams

x FID(↓)	50%	60%	70%	80%	90%	mean	110%	120%	130%	140%	150%	mean
Seam carving	85.75	66.41	51.67	35.20	23.80	52.57	20.86	30.01	37.38	43.45	**46.37**	35.61
Ours	**79.39**	**58.06**	**43.69**	**30.53**	**21.71**	**46.68**	**11.17**	**19.42**	**29.19**	**37.87**	49.74	**29.47**
y FID(↓)	50%	60%	70%	80%	90%	mean	110%	120%	130%	140%	150%	mean
Seam carving	116.70	84.33	60.59	40.86	**24.92**	65.48	23.04	33.58	40.82	48.49	52.59	39.70
Ours	**89.44**	**67.77**	**49.62**	**35.59**	25.42	**53.57**	**13.55**	**21.02**	**27.55**	**38.20**	**46.40**	**29.00**

Table 1. Comparison of FID scores on the RetargetMe [44] dataset, for retargeting x (top) and y (bottom) axis to a smaller (left) and bigger (right) size.

Figure 8. Selected results showing scenes where our approach benefits from higher degrees of flexibility (always input, seam carving, ours), all images from [44].

Retargeting width	Ours	SC	Draw
50 %	**48.48 %**	20.87 %	30.67 %
150 %	33.8 %	**36.27 %**	29.92 %
Retargeting height	Ours	SC	Draw
50 %	**49.5 %**	24.25 %	26.25 %
150 %	**45.84 %**	20.0 %	34.15 %
Total	**44.57** %	25.1 %	30.32 %

Table 2. User study comparing our approach to seam carving (*SC*) for 2D images, for different sizes and axis. 19 participants were asked to indicated which solution they prefer for random examples from the RetargetMe dataset[44], casting about 1200 votes in total.

	X Axis to 50 %			X Axis to 150 %		
Scene	Ours	VSC	Draw	Ours	VSC	Draw
Beach	**17**	3	1	**20**	1	0
Farm	**21**	0	0	**21**	0	0
Harb.	**21**	0	0	**21**	0	0

	Y Axis to 50 %			Y Axis to 150 %		
Scene	Ours	VSC	Draw	Ours	VSC	Draw
Beach	**19**	1	1	**19**	1	1
Farm	**16**	3	2	**21**	0	0
Harb.	**21**	0	0	**21**	0	0

Table 3. User study comparing retargeting a scene, then recording it (ours), to recording it and then retargetet it (video seam carving [42], *VSC*), using our provided dataset. 21 participants were asked to indicated which solution they prefer, casting 252 votes in total.

one after the other also prevents shearing by construction, we do not punish shearing that occurs in unimportant areas,

Figure 9. Visual ablation for retargeting to 80 percent width on an image[1], from top left, clockwise: Wobbly contours (no L_s), only uniform stretch (no L_e), swallowed features (no L_m), no boundary (no L_b).

gaining flexibility. Especially compared to video seam carving, our approach excelled in 3D, as we are able to produce geometry aware results.

4.2. Discussion and Limitations

Successors to seam carving focused on adding clever regularisations [4, 10, 51] or better energy terms [46]. While we only relied on colour gradient for our comparison to the baseline, showing our advantage without any improved energy formulation, and focused on redefining and generalising the approach itself rather than exploring *e.g.* energy terms we can incorporate in our approach by design. However, when *e.g.* applying our approach to images, where the assumption of important parts having high colour gradient breaks, will produce flawed outputs, just as seam carving does. This could be addressed by using a similar loss function like the discriminator from SinGAN [45] or using a

more complex measure for information content, as also partially explored with a face detector in [2]. Further, we do not learn a general deformation network but have to newly optimise a neural field for each individual input; However, this also means we do not require any training data other than our input image. Lastly, regarding computation time, our approach is slightly slower than seam carving for images, which we attribute to a more global solution, but excelled in 3D, as *e.g.* video seam carving on a 600 by 400 pixel image can takes multiple hours, while our approach produces a retargeted 3D scene within 15 minutes.

5. Conclusion

We present an approach that generalises the principle underlying seam carving to create image variants by manipulating its low content areas. By training neural deformation fields that avoid applying distortion to important parts of an input, our approach is applicable to different editing objectives, centred around visual data retargeting. Our formulation is largely domain invariant and only needs minor alterations to work on different types of visual data, as we demonstrated through its application to neural radiance fields and even meshes, which is novel for approaches from the seam carving family that do not operate in continuous domains. Compared to the existing idea of seam carving, our approach offers increased flexibility by using continuous deformation fields instead of discrete decisions for seams, while bringing compatibility *e.g.* for the use of more complex energy formulations in the future.

References

[1] Newton2 at English Wikipedia. Broadway tower, 2007. 8

[2] Shai Avidan and Ariel Shamir. Seam carving for content-aware image resizing. *ACM Trans. Graph.*, 26(3):10, 2007. 1, 2, 3, 7, 9

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5855–5864, 2021. 6, 1

[4] Tali Dekel Basha, Yael Moses, and Shai Avidan. Stereo seam carving a geometrically consistent approach. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(10):2513–2525, 2013. 2, 8

[5] Sema Berkiten, Maciej Halber, Justin Solomon, Chongyang Ma, Hao Li, and Szymon Rusinkiewicz. Learning detail transfer based on geometric features. *Comput. Graph. Forum*, 36(2):361–373, 2017. 2

[6] Hongrui Cai, Wanquan Feng, Xuetao Feng, Yan Wang, and Juyong Zhang. Neural surface reconstruction of dynamic scenes with monocular rgb-d camera. In *Neural Information Processing Systems (NeurIPS)*, 2022. 3

[7] Chen, Zhiqin and Zhang, Hao. Learning implicit fields for generative shape modeling. In *CVPR*, pages 5939–5948, 2019. 2

[8] Weiming Dong, Ning Zhou, Jean-Claude Paul, and Xiaopeng Zhang. Optimized image resizing using seam carving and scaling. *ACM Trans. Graph.*, 28(5):125, 2009. 1, 2

[9] Weiming Dong, Guan-Bo Bao, Xiaopeng Zhang, and Jean-Claude Paul. Fast multi-operator image resizing and evaluation. *J. Comput. Sci. Technol.*, 27(1):121–134, 2012. 1, 2

[10] Weiming Dong, Ning Zhou, Tong-Yee Lee, Fuzhang Wu, Yan Kong, and Xiaopeng Zhang. Summarization-based image resizing by intelligent object carving. *IEEE Trans. Vis. Comput. Graph.*, 20(1):1, 2014. 2, 8

[11] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, pages 341–346. ACM, 2001. 2

[12] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *ICCV*, pages 1033–1038, 1999. 2

[13] Stephan J Garbin, Marek Kowalski, Virginia Estellers, Stanislaw Szymanowicz, Shideh Rezaeifar, Jingjing Shen, Matthew Johnson, and Julien Valentin. Voltemorph: Real-time, controllable and generalisable animation of volumetric representations. *arXiv:2208.00949*, 2022. 2

[14] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks. In *Neural Information Processing Systems*, pages 262–270, 2015. 2

[15] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Josh Susskind, and Navdeep Jaitly. Matryoshka diffusion models. *arXiv preprint arXiv:2310.15111*, 2023. 7

[16] Philipp Henzler, Niloy J Mitra, , and Tobias Ritschel. Learning a neural 3d texture space from 2d exemplars. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[17] Philipp Henzler, Valentin Deschaintre, Niloy J. Mitra, and Tobias Ritschel. Generative modelling of BRDF textures from flash images. *ACM Trans. Graph.*, 40(6):284:1–284:13, 2021. 2

[18] Amir Hertz, Rana Hanocka, Raja Giryes, and Daniel Cohen-Or. Deep geometric texture synthesis. *ACM Trans. Graph.*, 39(4):108:1–108:11, 2020. 2

[19] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017. 7

[20] Yihua Huang, Yan-Pei Cao, Yu-Kun Lai, Ying Shan, and Lin Gao. Nerf-texture: Texture synthesis with neural radiance fields. In *ACM SIGGRAPH 2023 Conference Proceedings*, pages 43:1–43:10, 2023. 2

[21] Bahjat Kawar, Shiran Zada, Oran Lang, Omer Tov, Huiwen Chang, Tali Dekel, Inbar Mosseri, and Michal Irani. Imagic: Text-based real image editing with diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6007–6017, 2023. 7

[22] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 1

[23] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong. Solid texture synthesis from 2d exemplars. *ACM Transactions on Graphics*, 26(3):2:1–2:9, 2007. 2

[24] Yu-Kun Lai, Shi-Min Hu, Xianfeng Gu, and Ralph R. Martin. Geometric texture synthesis and transfer via geometry images. In *Proceedings of the Tenth ACM Symposium on Solid and Physical Modeling*, pages 15–26. ACM, 2005. 2

[25] Feng Liu and Michael Gleicher. Automatic image retargeting with fisheye-view warping. In *Proceedings of the 18th Annual ACM Symposium on User Interface Software and Technology*, pages 153–162. ACM, 2005. 2

[26] Feng Liu and Michael Gleicher. Video retargeting: automating pan and scan. In *Proceedings of the 14th ACM International Conference on Multimedia*, pages 241–250. ACM, 2006. 2

[27] Jonathon Luiten, Georgios Kopanas, Bastian Leibe, and Deva Ramanan. Dynamic 3d gaussians: Tracking by persistent dynamic view synthesis. In *3DV*, 2024. 3

[28] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3D reconstruction in function space. In *CVPR*, pages 4460–4470, 2019. 2

[29] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 2, 1

[30] Thierry Pinheiro Moreira, Marcos Cleison S. Santana, Leandro A. Passos, João Paulo Papa, and Kelton Augusto Pontara da Costa. An end-to-end approach for seam carving detection using deep neural networks. In *Pattern Recognition and Image Analysis*, pages 447–457, 2022. 2

[31] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *ACM Transactions on Graphics (ToG)*, 41(4):1–15, 2022. 1

[32] Seung-Hun Nam, Wonhyuk Ahn, In-Jae Yu, Myung-Joon Kwon, Minseok Son, and Heung-Kyu Lee. Deep convolutional neural network for identifying seam-carving forgery. *IEEE Trans. Circuits Syst. Video Technol.*, 31(8):3308–3326, 2021. 2

[33] Lakshmanan Nataraj, Chandrakanth Gudavalli, Tajuddin Manhar Mohammed, Shivkumar Chandrasekaran, and B. S. Manjunath. Seam carving detection and localization using two-stage deep neural networks. *CoRR*, abs/2109.01764, 2021. 2

[34] Rockstar North. Grand theft auto v. Steam, 2015. 2

[35] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. DeepSDF: Learning continuous signed distance functions for shape representation. In *CVPR*, pages 165–174, 2019. 2

[36] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. In *ICCV*, pages 5845–5854, 2021. 3

[37] Keunhong Park, Utkarsh Sinha, Peter Hedman, Jonathan T. Barron, Sofien Bouaziz, Dan B. Goldman, Ricardo Martin-Brualla, and Steven M. Seitz. Hypernerf: a higher-dimensional representation for topologically varying neural radiance fields. *ACM Trans. Graph.*, 40(6):238:1–238:12, 2021. 3

[38] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205, 2023. 7

[39] Yicong Peng, Yichao Yan, Shenqi Liu, Yuhao Cheng, Shanyan Guan, Bowen Pan, Guangtao Zhai, and Xiaokang Yang. Cagenerf: Cage-based neural radiance fields for generalized 3d deformation and animation. In *Advances in Neural Information Processing Systems*, 2022. 2

[40] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-NeRF: Neural Radiance Fields for Dynamic Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021. 3

[41] Stephan R Richter, Hassan Abu AlHaija, and Vladlen Koltun. Enhancing photorealism enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2): 1700–1715, 2022. 2

[42] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Improved seam carving for video retargeting. *ACM Trans. Graph.*, 27(3):16, 2008. 2, 7, 8

[43] Michael Rubinstein, Ariel Shamir, and Shai Avidan. Multi-operator media retargeting. *ACM Trans. Graph.*, 28(3):23, 2009. 1, 2

[44] Michael Rubinstein, Diego Gutierrez, Olga Sorkine, and Ariel Shamir. A comparative study of image retargeting. *ACM Transactions on Graphics (Proc. SIGGRAPH ASIA)*, 29(6):160:1–160:10, 2010. 1, 4, 7, 8, 3

[45] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *ICCV*, pages 4569–4579, 2019. 2, 8

[46] Eungyeol Song, Minkyu Lee, and Sangyoun Lee. Carvingnet: Content-guided seam carving using deep convolution neural network. *IEEE Access*, 7:284–292, 2019. 2, 8

[47] Olga Sorkine-Hornung and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry Processing*, 2007. 2

[48] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. In *Advances in Neural Information Processing Systems*, pages 27171–27183, 2021. 2

[49] Li-Yi Wei and Marc Levoy. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH*, pages 479–488. ACM, 2000. 2

[50] Guanjun Wu, Taoran Yi, Jiemin Fang, Lingxi Xie, Xiaopeng Zhang, Wei Wei, Wenyu Liu, Qi Tian, and Wang Xinggang. 4d gaussian splatting for real-time dynamic scene rendering. *arXiv preprint arXiv:2310.08528*, 2023. 3

[51] Huisi Wu, Yu-Shuen Wang, Kun-Chuan Feng, Tien-Tsin Wong, Tong-Yee Lee, and Pheng-Ann Heng. Resizing by symmetry-summarization. *ACM Trans. Graph.*, 29(6):159, 2010. 2, 8

[52] Rundi Wu and Changxi Zheng. Learning to generate 3d shapes from a single example. *ACM Transactions on Graphics (TOG)*, 41(6), 2022. 2

[53] Rundi Wu, Ruoshi Liu, Carl Vondrick, and Changxi Zheng. Sin3dm: Learning a diffusion model from a single 3d textured shape. *CoRR*, abs/2305.15399, 2023. 2

[54] Wenqi Xian, Patsorn Sangkloy, Varun Agrawal, Amit Raj, Jingwan Lu, Chen Fang, Fisher Yu, and James Hays. Texturegan: Controlling deep image synthesis with texture patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2

[55] Tianhan Xu and Tatsuya Harada. Deforming radiance fields with cages. In *ECCV*, pages 159–175, 2022. 2

[56] Ziyi Yang, Xinyu Gao, Wen Zhou, Shaohui Jiao, Yuqing Zhang, and Xiaogang Jin. Deformable 3d gaussians for high-fidelity monocular dynamic scene reconstruction. *arXiv preprint arXiv:2309.13101*, 2023. 3

[57] Yu-Jie Yuan, Yang-Tian Sun, Yu-Kun Lai, Yuewen Ma, Rongfei Jia, and Lin Gao. Nerf-editing: Geometry editing of neural radiance fields. In *CVPR*, pages 18332–18343, 2022. 2

Retargeting Visual Data with Deformation Fields

Supplementary Material

6. Architecture

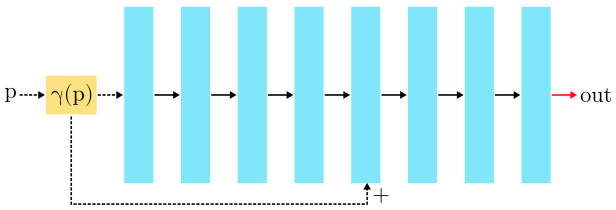

Figure 10. The simple architecture we apply for learning the neural image field, the deformation field(s), energy network and cumulative energy network. We use positional encoding (yellow), then apply linear layers (cyan) with a residual connection, LeakyReLu (black arrow), and an output depending on context (red, with sigmoid for images, otherwise LeakyReLu).

We use simple components for our architecture, with a basic MLP with residual connection being the core of all our used networks. We use the Adam optimiser[22] in all instances, without weight or learning rate decay, and the default learning rate of 0.001 if not specified otherwise.

6.1. Architecture for Images

For the neural field holding the images as described in Sec. 3.2, we apply the network depicted in Fig. 10, a simple MLP with 192 channels, a residual connection, and positional encoding that outputs 3 scalars for the RGB value. For the deformation and the cumulative gradient network, we use the same network with 64 channels and a single output scalar value. We use a learning rate of 0.0001 for expanding the image. We train the network learning the image itself and the network learning the cumulative gradient for 250 epochs with 100 iterations each, while we initialise the deformation network for 50 epochs with a uniform transformation that we then train on our loss defined in Sec. 7. Our slim architecture allows optimisation of all of the image data in one single batch.

6.2. Architecture for Neural Radiance Fields

We use a pre-trained, customised iNGP [31] as our learned 3D scene (both vanilla [29] and mip-NeRF [3] worked just as well in our early tries), then use the same setup as for images. We train initial deformation, energy, and cumulative energy for 5000 iterations each, always using 10000 samples. We optimise the deformation field for 10000 random points from the surface and 10000 random points from the boundary for 50 epochs with 100 iterations each, and always choose the network parameters from the epoch with the best average loss.

7. Exact Loss Formulations

7.1. Loss Formulation for Images

Due to out light weight network we use for our deformation field, we can apply our loss terms directly to all points p in the output domain (*i.e.* the shrunken image), training one single batch for an image.

For L_e, we define:

$$L_e = \frac{|E_{cnet}(p) - E_{cnet}(p_\epsilon)| \cdot |D(p) - D(p_\epsilon)|}{\epsilon} \tag{12}$$

where E_{cnet} is the MLP for cumulative energy and p_ϵ are all pixels p moved by one pixel in direction v. For shearing, for energy net E and $p_\epsilon^\perp$ being all points p moved by one pixel in the direction orthogonal to v, we define:

$$L_s = \frac{E(p) \cdot |D(p) - D(p_\epsilon^\perp)|}{\epsilon} \tag{13}$$

For the boundary, we build our loss around the pixels on the one end of the image, p_l, and the other, p_r, regularising their deformation to be 0 at the one, and $1.0 - \alpha$ at the other end:

$$L_b = mean(|D(p_l)|) + mean(|D(p_l) - (1 - \alpha)|) \tag{14}$$

For monotonicity in images, we use:

$$L_m = \frac{mean(max(0, D(p) - D(p_\epsilon)))}{\epsilon} \tag{15}$$

As weights for our loss terms, we use $\lambda_e = 10000$, $\lambda_s = 250$ $\lambda_b = \lambda_m = 10000$. For the cumulative energy net, we accumulate energy over all pixels, then train the MLP to minimise mean squared error between input (2D position) and output (cumulative energy for that position) for 10000 iterations, always for all pixels.

7.2. Loss Formulation for Neural Radiance Fields

We first define a mixture of surface points and uniform random points p' in source space we run our optimisation on, then finding those points p in target space that are deformed to become p':

$$p = U(p') \tag{16}$$

For L_e, we define:

$$L_e = \frac{mean(|E_{cnet}(p) - E_{cnet}(p_\epsilon)| \cdot |D(p) - D(p_\epsilon)|)}{\epsilon} \tag{17}$$

where E_{cnet} is the MLP for cumulative energy and p_ϵ are our points p moved by an offset in direction v. For shearing, for energy net E and $p^\perp_{\epsilon^1}$, $p^\perp_{\epsilon^2}$ being all points p moved by an offset in the directions orthogonal to v, we define:

$$L_s = \frac{E(p) \cdot |D(p) - D(p^\perp_{\epsilon^1})|}{\epsilon} + \frac{E(p) \cdot |D(p) - D(p^\perp_{\epsilon^2})|}{\epsilon} \tag{18}$$

For the boundary, we build our loss around the pixels on the one end of the image, p_l, and the other, p_r, regularising their deformation to be 0 / $1.0 - \alpha$:

$$L_b = mean(|D(p_l)| + mean(|D(p_l) - (1 - \alpha)|) \tag{19}$$

For monotonicity in images, we use:

$$L_m = \frac{mean(max(0, D(p) - D(p_\epsilon)))}{\epsilon} \tag{20}$$

As weights for our loss terms, we use $\lambda_e = 10$, $\lambda_s = 0.1$ $\lambda_b = 100, = \lambda_m = 1$. After every iteration, we also update U with all points p, *i.e.* minimise $(D(U(p)) - p)^2$ for one step. For the cumulative energy net, we accumulate energy by shooting random rays along the deformation direction with 100 uniformly distributed samples. We gather and accumulate energy from E_{net}, then train the cumulative energy MLP to minimise mean squared error between input (3D position) and output (cumulative energy for that position on that ray) for 10000 iterations, always for 10000 points.

8. Expansion

Expansion of visual data follows the same idea as for seam carving[2]: Instead of compressing (in seam carving: removing) low energy parts, we stretch (in seam carving: double) them. Examples can be seen in Sec. 10. To expand data, *e.g.* to increase its width to 150 percent, we can use the exact same losses as for retargeting to a smaller size, but with one additional loss term and a minor modification:
We require a limit on the change in the deformation to avoid only duplicating a single seam, meaning to punish any change in the deformation that is too large, hence would repeat the same part of the image over and over again. Seam carving prevents this for image expansion by only doubling a seam once. As a new loss term, this gives us:

$$L_{cap} = mean(max(0, \frac{(p - p_\epsilon)}{\epsilon} - 1))) \tag{21}$$

Preventing to more than double an input. In addition, we modify L_e to not punish any information content skipped between two points (as we no longer skip over anything), but instead simply punish the energy directly, *i.e.* replacing $\hat{E}$ from Eq. (6) with E applied to only one point.

9. Editing

For the editing procedures describe in Sec. 3.5, we can perform the same concept for every domain. For this, we always retain the same image size.

Removal To remove an object, we punish every point that maps to the content that we want to remove, *i.e.* take all points of the input, deform them, and look up if the deformed point lies within the marked area. For continuity, we do this by training another network net_{mask} to learn a mask, *i.e.* 0 for regions outside, 1 for regions inside the area to remove. We then simply add $mean(net_{mask}(p))$ for all points p to the loss function, punishing any deformation that maps inside the target area. To allow both sides of the removed content to fill in the space, we disable the monotonicity loss L_m. The remaining loss terms then keep the rest of the image in place.

Moving an object To move an object, we first remove the monotonicity loss L_m to allow more complex deformations. As this disables protection against repetition, we then actively punish our target object occuring anywhere else, *i.e.* using a network net_{mask} containing a binary mask of the object to move, then add $net_{mask}(p)$ for all points p in the output to the loss function except for the target coordinates of the object. We then add a term to the loss that enforces the exact offset value that would place the target at the right location for all parts of it. Lastly, to avoid any other unwanted edits and make the optimisation more stable, we add a last loss term scaled by 0.01 that punishes the absolute mean of deformation, *i.e.* punishes applying any deformation at all. While this avoids unnecessary deformations, it is neglectable compared to the additional loss caused by only the part relevant for the deformation.
To move an object along multiple axis, we decompose the movement to the target position into two axis, then apply our approach for each direction.

10. Additional Results

(see following pages)

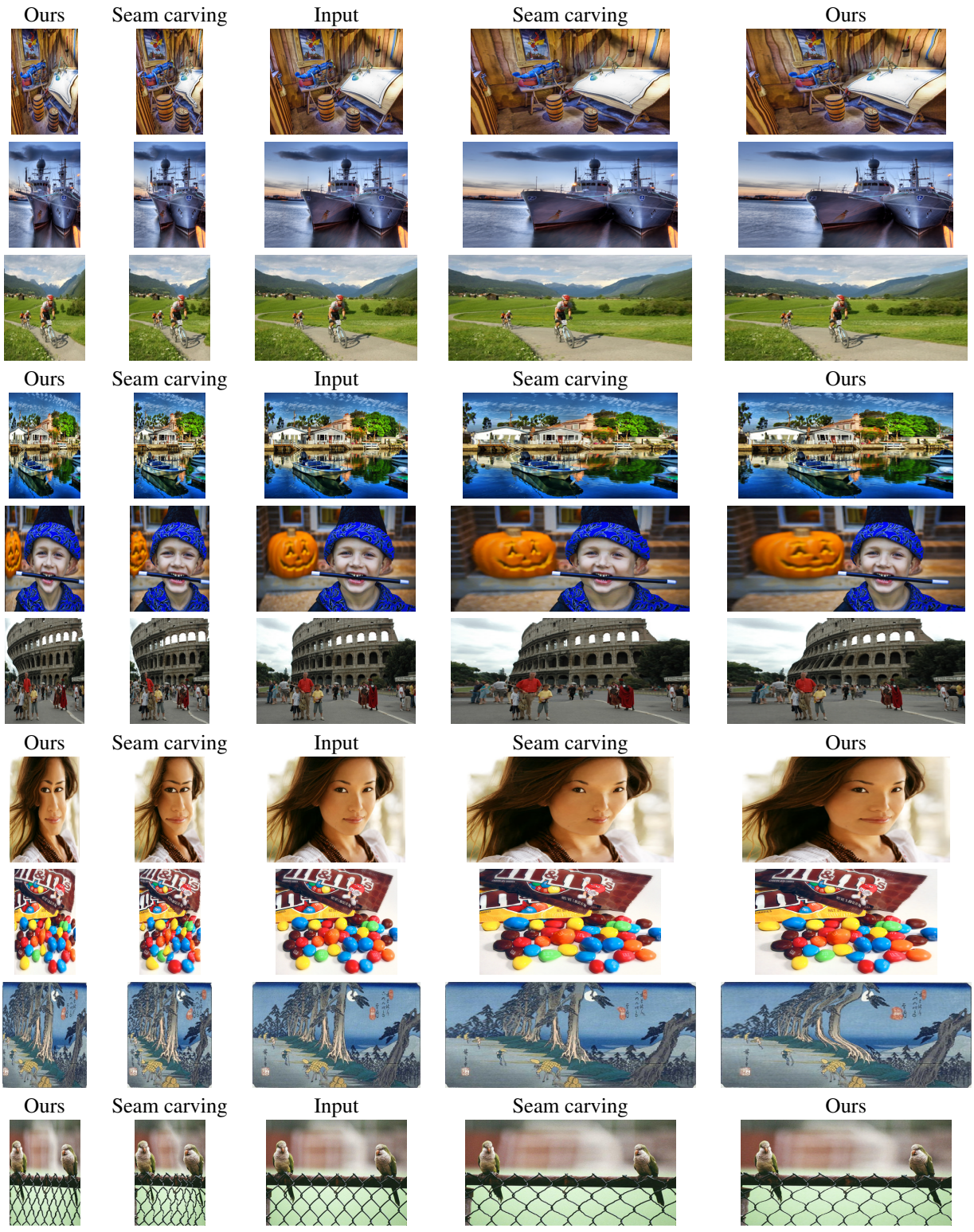

Table 4. More examples of our deformation approach on images, all from the RetargetMe dataset [44].

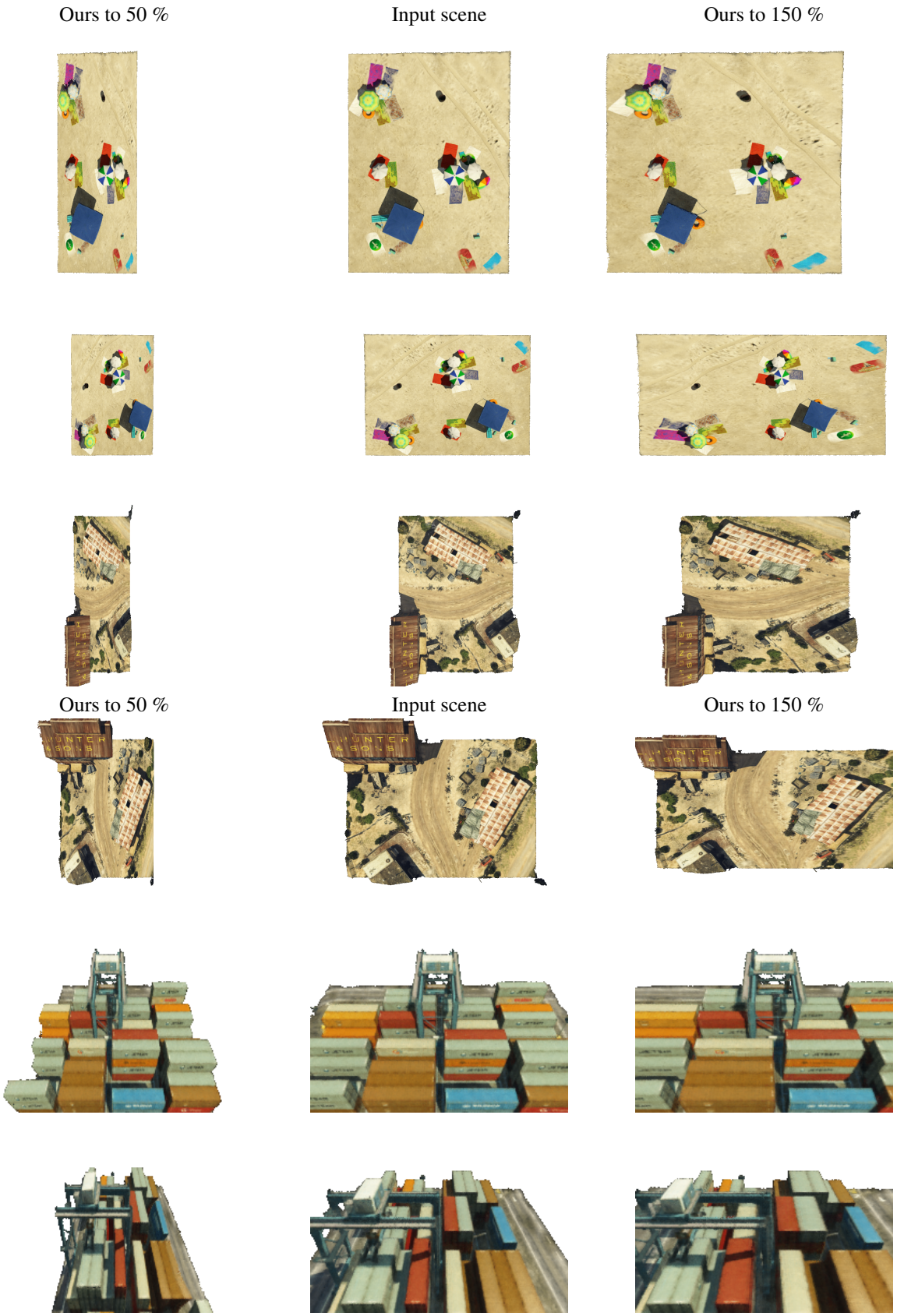

Table 5. More examples of our deformation approach on 3D NeRF scenes.