Distinguishing Texture Edges from Object Boundaries in Video

Oliver Wang, Martina Dümcke, Aljoscha Smolic, Markus Gross

***Abstract*—One of the most fundamental problems in image processing and computer vision is the inherent ambiguity that exists between texture edges and object boundaries in real-world images and video. Despite this ambiguity, many applications in computer vision and image processing often use image edge strength with the assumption that these edges approximate object depth boundaries. However, this assumption is often invalidated by real world data, and this discrepancy is a significant limitation in many of today's image processing methods.**

We address this issue by introducing a simple, low-level, patch-consistency assumption that leverages the extra information present in video data to resolve this ambiguity. By analyzing how well patches can be modeled by simple transformations over time, we can obtain a indication of which image edges correspond to texture edges versus object boundaries.

Our approach is simple to implement and has the potential to improve a wide range of image and video-based applications by suppressing the detrimental effects of strong texture edges on regularization terms. We validate our approach by presenting results on a variety of scene types and directly incorporating our augmented edge map into existing image segmentation and optical flow applications, showing results that better correspond to object boundaries.

I. Introduction

We introduce a simple method to distinguish texture edges from object boundaries by analyzing patch-based changes over time. Methods that benefit from our approach can be found in all areas of image processing. Some examples are applications that use object boundaries to propagate sparse information, such as colorization, markup, and image editing, applications that incorporate edge-based regularization, such as optical flow, stereo vision, and image stitching, and applications that segment objects in video, including object tracking, pedestrian detection, and gesture recognition.

All of the above approaches are similar in that they rely on image edges as a form of scene understanding. However, image edges alone cannot distinguish between object boundaries and texture edges, as often times the type of edge depends on the *context* of the edge, rather than its *appearance*. Consider a photograph of a room with multiple objects at different depths. In this image, both texture and object boundaries exist. However, a photograph *of this photograph*, can look exactly identical to the original image, but in the second case *all* image edges correspond to texture edges. Clearly a single image cannot tell us what the source of the image edges are, and more information is needed in order to differentiate the two.

To overcome this inherent "photograph-ambiguity" in single images, we present an approach that leverages extra information available in video data. We create a restricted definition of *texture* and *object* edges, and propose a simple but effective metric to quantify edge type in terms of this definition. Our method is based off of what we refer to as the *patch consistency* assumption. This assumption states that an edge is a texture edge if its appearance can be well modeled over time by a set of basic 2D transformations (translation, rotation and scale). Image edges that arise at object boundaries on the other hand, cannot be as easily modeled. The motivation for this assumption is that difficult to model phenomena such as disocclusions, occlusions, and different foreground/background motions all occur at object boundaries. The patch consistency assumption is motivated in part by how we as humans perceive object separation, and has been known for a long time as the 'common fate" Gestalt principle [1], which states that a coherent object is one that *moves* together.

We note that our assumption is a simplification of real world object boundaries, as object boundaries can exist in video footage without changing over time (e.g., a fixed camera filming a static scene). But without additional sources of information, stationary objects in video suffer from the same "photograph ambiguity" as single images. However, as long as these regions are spatially rare, we show that it is possible to explicitly detect where our patch consistency assumption fails (textureless and motionless regions), and then perform a refinement step that propagates estimates from more confident regions along image edges.

In practice, we compute object boundary probabilities by first finding patch correspondences over time using a recent fast nearest-neighbor search method called PatchMatch [2]. We then evaluate how well our patch consistency assumption matches the change in content over time to quantify texture edge likelihood. We apply our method to a variety of scene types, and provide additional validation by incorporating our output in existing image segmentation and optical-flow applications.

In summary, the main contribution of our work is a simple and novel assumption that we show can help differentiate object boundaries from texture edges in a video sequence. Our method is easy to implement, requires no additional sources of data, and can directly be used in a large number of image-processing algorithms.

The paper is laid out as follows. Section II describes

O. Wang and A. Smolic are with Disney Research Zurich, M. Dümcke is with ETH Zurich, M. Gross is with both ETH Zurich and Disney Research Zurich

related work, Section III looks at more detail into the patch consistency assumption and explains how we compute the probability that an image edge corresponds to an object boundary, Section IV provides example applications, and Section V discusses limitations and directions for future work.

II. Related Work

A. Edge Detection

Much effort has been spent in the last five decades on the detection of edges in images, and it remains one of the most fundamental image understanding tasks today. Early algorithms focus on finding local maxima in the image first derivative, such as the Prewitt [3] and Sobel operators, or zero-crossings of the second derivative [4]. Despite being among the oldest techniques in image processing, many of these algorithms are still widely used today. These methods often make the assumption that the *strength* of the image derivatives correlates to how likely that edge corresponds to an object boundary, which is often not the case.

To remedy this problem, alternative strategies have been introduced, such as defining edges as borders between regions with *different repeated* textures. These regions can be found by convolving images with filter banks based on Gabor filters [5]–[7], by extending the compass operator to filter responses [8], or by looking at local descriptors of texture patches directly [9]. These methods all solve a fundamentally different problem, which is locating edges between texture-texture regions, rather than finding object boundaries. As a result, when such a separation is needed, these methods make the implicit assumption that object boundaries are equivalent to texture-texture edges. However, in natural images, object boundaries can exist without such edges, and texture edges do not necessitate object boundaries.

Machine learning has also been proposed as a mean to try to differentiate edge types, for example using boosting [10]. This method derives features from multiscale pixel patches and shows promising results, but is applicable only to scenes with edges that share similar features to those used for training.

All of the above methods operate only on single images, and therefore suffer from the "photograph-ambiguity", which we address using video information. Other methods try to overcome this ambiguity by leveraging additional information, such as structured light [11], or a modified camera with physically offset light sources [12]. While both of these methods yield high quality results, they require extra hardware and modification of the original scene, which is not possible in all use cases.

An obvious approach to find depth boundaries would be simply to compute a depth map for the scene and then look for local discontinuities. However, computing high quality depth maps from a single video source is a highly under-constrained problem, and no robust solutions exist today. Most successful methods rely again on additional scene information, such as multiple cameras [13], or active illumination [14]. Our method requires no additional sources of information other than what is available in a short temporal window around a frame.

B. Motion Estimation

We are not the first to propose motion as a cue to resolve edges. Many methods have leveraged motion fields computed by optical-flow methods to assist in segmenting objects. This class of approaches is often known as motion segmentation [15], which attempt to group regions with similar flow vectors, for example using normalized cuts [16].

If perfect motion fields were known, then indeed finding object boundaries would be possible. However, in practice finding motion fields requires solving optical flow. The problem with these approaches, is that optical flow methods inherently suffer from a serious chicken-and-egg problem in regards to isolating object boundaries. Because optical flow (which models the motion of pixels from one frame to the next), is explicitly undefined in areas that contain occlusions and disocclusions, no meaningful correspondences can exist at these locations. Therefore, methods must rely on *regularization* to fill in information, and this regularization step is most commonly driven by image edges [17], again under the assumption that they match object boundaries. Therefore, high-level cues, such as motion fields, already assumes some knowledge of object borders in their computation, and in fact, we show that our metric can be used to improve the quality of a state-of-the-art optical flow method.

Unlike motion segmentation methods, we present a method that can be computed from local, low-level information, and does not suffer from the chicken-egg problem above. Instead, we leverage a recent approach called *PatchMatch* [2], which efficiently computes dense patch-based nearest-neighbors between two images. This approach, described in further detail in Section III, has several advantages over optical flow; it does not require a regularization step, which greatly simplifies computation and reduces prediction errors at boundary regions, and (as a direct result), is able to model additional degrees of patch transformations, such as rotation and scale [18]. These additional degrees of freedom are an important part of our patch consistency assumption, and are currently *not* modeled by motion fields, which describe only translation.

III. Method

In this work, we refer to "edge-maps" in the sense of a real-value per pixel edge magnitude, rather than a binary edge descriptor, however performing thresholding operators on these edge maps is a straightforward extension.

A. Algorithm Overview

The goal of our method is to compute P, the per-pixel likelihood that each pixel corresponds to an object boundary, given an input video sequence I. A patch is specified by a four-element vector i that consists of an x, y position, scale factor s, and rotation angle θ. The contents of a source patch at frame j is written as I_i^j. Our algorithm consists of two main steps. The first step is to find, for every patch I_i^j, its closest set of matches $I_{b_k}^k$, for all $k \in K = \{j-n, \ldots, j+n\}$ neighboring frames excluding j (Figure 1). Here, $2n$ is the temporal window width, and $b_k \in \mathbb{R}^4$ is searched over the set of possible patch transformations; shift (x, y), relative scale

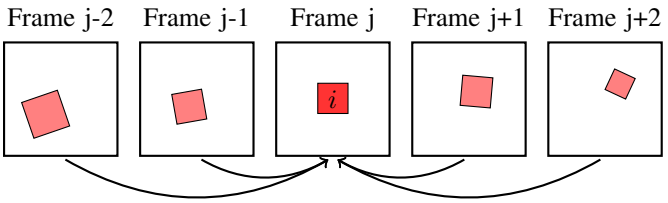


Fig. 1: We find best-match patch correspondences in a temporal window. Here we show possible matches for one patch I_i^j over a window of 5 frames.

(s), and rotation (θ). As $I_{b_k}^k$ is not necessarily rectangular in the target image, we warp and re-sample it into the reference frame defined by patch i, such that direct pixelwise comparisons can be made between I_i^j and $I_{b_k}^k$. We use bilinear interpolation, but found that the choice of interpolation function did not have a significant impact on results. Furthermore, source patches are always defined to have a fixed scale ($s = 1$), and rotation ($\theta = 0$). Next, we determine how well the patch transformation models the data by applying a difference metric ϕ on I_i^j and the set of $I_{b_k}^k$, $\forall k \in K$.

B. Patch Consistency Assumption

Our main assumption is that patches on object boundaries cannot be modeled by simple patch transformations over time, while texture edges can (within a short time-frame). In Figure 2, we show an example demonstrating this assumption. Two patches are shown from a frame of video, along with their best matched patches in a temporal window. Below each row is a visualization of patch consistency over time, computed using ϕ, which we define later Section III-D. This shows areas with high consistency (blue), medium consistency (green) and low consistency (red). Note that the first patch corresponds to an object boundary and exhibits lower patch consistency, while the second patch corresponds a texture edge and has much higher patch consistency.

C. Finding Patch Correspondences

We use the Generalized PatchMatch approach to find correspondences over frames [18]. We give a short description of PatchMatch here, but refer the reader to the original work for more details. The goal of the Generalized PatchMatch method is to find the best match of every patch from a source image to a target image. This can be defined by a function $f_{\text{patchmatch}}$, such that $f_{\text{patchmatch}}(I_i^j, I^k) \in \mathbb{R}^4$ is the four-element vector that gives the position, scale, and rotation that define the nearest neighbor patch $I_{b_k}^k$ for a given patch I_i^j, e.g.,

$$b_k = f_{\text{patchmatch}}(I_i^j, I^k) \quad \forall\, k \in K, i \in I^j \tag{1}$$

The brute force solution to compute $f_{\text{patchmatch}}$ would require $O(mM^2|s||\theta|)$ running time, for images with m pixels, patches of size M, and $|s|, |\theta|$ are the cardinalities of the discretized scale space and rotation space respectively. PatchMatch accelerates this computation using the following algorithm. First, the nearest neighbor field is randomly initialized. Then, a *Propagation* step takes advantage of spatial coherence in images by sequentially updating every patch (top to bottom, left to right) with the best offset vector of itself and the offset vectors of patches immediately to the left and above of it. Secondly, a *Random Search* step avoids local minima by checking a number of random patch parameters to see if any would provide better matching. These two steps are alternated a fixed number of iterations (5 is suggested in the original work).

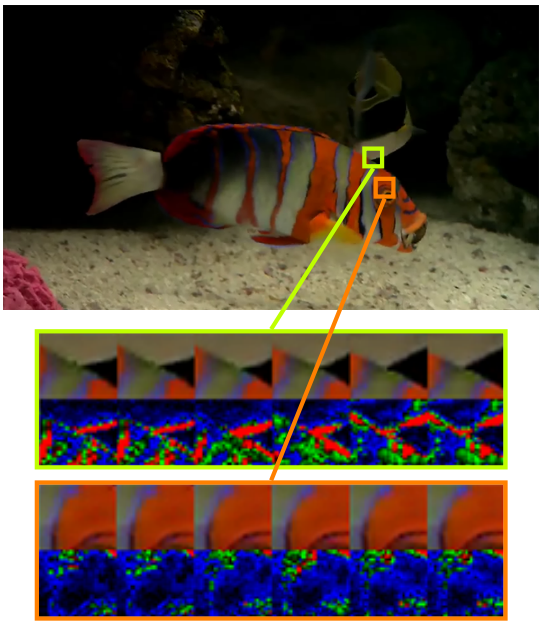


Fig. 2: Best-matched patches at object boundaries exhibit different properties than at texture edges. The orange region corresponds to a texture edge, which is better modeled over time when compared to a patch at an object boundary (lime).

D. Object Boundary Metric

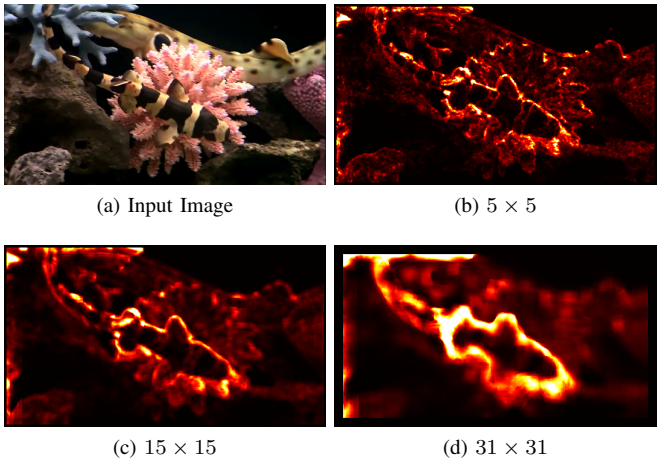
We then define a metric ϕ that computes an object boundary probability value p_i^j for each patch as a function of these matches:

$$p_i^j = \phi(I_i^j, I_{b_k}^k, \ldots) \;\; k \in K \tag{2}$$

In this paper, we will use P as a shorthand for all probabilities p_i^j assembled into an image. The goal of this metric is to distinguish a set of patches that correspond to texture edges from those that belong to object boundaries. As our patch assumption states that patches should be similar if the content corresponds to a texture edge, we define ϕ as a commonly used patch difference metric that combines color and gradient information [19]:

$$\phi(I_i^j, I_{b_k}^k, \ldots) = \frac{1}{|K|} \sum_{k \in K} ||I_i^j - I_{b_k}^k|| + \alpha ||\nabla I_i^j - \nabla I_{b_k}^k|| \tag{3}$$

where ∇ is the gradient operator. We choose this metric as it has been shown to have good discriminative properties as well as some degree of invariance to lighting changes due to the



(a) Input Image (b) 5×5

(c) 15×15 (d) 31×31

Fig. 3: The effect of patch size on P. Larger patch sizes exhibit less noise, but lose edge resolution.

gradient term [19]. For all experiments shown here, we use a value of $\alpha = .5$, which gives roughly equal weighting to the two terms. For empirical validation, we show results of P in Figure 7 on a variety of datasets. These include scenes with a fixed camera and moving objects, moving cameras and fixed objects, and both moving cameras and objects.

We provide an analysis of different patch sizes in Figure 3. Large patches give more reliable measurements, but lose detail, as all patches that overlap object boundaries will validate our assumption to some degree. Similarly, while larger temporal windows could in theory give more reliable results, beyond a few frames the quality of matching degrades significantly so as to introduce additional noise. In all results shown here, we use a patch size of 15×15 pixels and a temporal window of $k = 2$.

E. Refinement

We note that P becomes visibly fuzzy as a result of the patch size. Furthermore, our patch-consistency assumption can be invalidated when no motion has occurred. This can lead to low P values in places with potentially strong image edges. Therefore, in order to address both the fuzziness as well as the limitations of our assumption, we propose an optimization-based method to refine the initial object boundary probability P and propagate information along image edges where our patch-consistency assumption cannot be relied upon.

To accomplish this, we first define a confidence map C. We know our patch-consistency assumption is invalidated when there is no motion, or when patches have no texture, so we can directly use the two terms to compute C as follows,

$$C_i = \frac{1}{|K|} \sum_{k \in K} min(||i(x,y) - b_k(x,y)||, \tau) + var(I_i^j). \quad (4)$$

i and b_k are again four-element vectors defining a patch, and $i(x,y), b_k(x,y)$ refer to only the position components of i and b_k respectively. The norm of their difference gives us the image-space motion between frames. $var(I_i^j)$ computes the variance of the source patch. The constant τ is used to avoid biasing the confidence map to only the largest motions. It should be chosen as small as possible such that some degree of motion is still detectable by ϕ, in all results we use a $\tau = 3$.

Our goal is to now use C to compute a new object boundary probability map $\tilde{P}$, in which low confidence regions can "borrow" information from more confident neighbors, while the result is refined to image edges. To do this, we use an approach similar to image matting [20]. Let us consider images to be vectors with length $l = width \times height$, we can define a quadratic energy term

$$E_{refined} = (P - \tilde{P})^T G (P - \tilde{P}) + \lambda \tilde{P}^T L \tilde{P} \quad (5)$$

where G is a $l \times l$ matrix whose diagonal is C and all other entries are zero. To enforce coherence to the original image edges, we use the Matting Laplacian [20], L. λ is a regularization parameter. The elements of L are derived from the classic edge strength map S, computed as the magnitude of the Sobel response of image I^j (Figure 4). The smoothness term $\tilde{P}^T L \tilde{P}$ forces neighboring values in $\tilde{P}$ to be similar *only* if they are also similar in S. The confidence weighted data term $(P-\tilde{P})^T G (P-\tilde{P})$ preserves the similarity of the refined map $\tilde{P}$ to the original P in regions where the confidence C is high. Combining these two terms means that an optimal $\tilde{P}$ will have values that are similar to the original estimates P, but have been *propagated* along image edges in regions of low confidence. The optimal refined probability map $\tilde{P}$ can be found by minimizing $E_{refined}$. As $E_{refined}$ is linear with respect to $\tilde{P}$, we can perform this minimization by solving the following system of equations in the form $Ax = b$,

$$(L + \lambda G U)\tilde{P} = \lambda G P \quad (6)$$

where $\tilde{P}$ is our vector of unknown refined probabilities, and U is a identity matrix the same size as L. In Figure 4, we show some results from this refinement step.

IV. Applications

Segmentation One class of applications whose performance is often hindered by the confusion between texture and object edges is image segmentation. Segmentation involves assigning object class labels to pixels, and is sometimes known as image labeling. It is often solved by minimizing energy in the form of a pairwise MRF composed of color similarity (*unary*) and local smoothness (*pairwise*) terms. Many methods exist to minimize these pairwise MRFs, and some of the more common are graph-theoretic approaches, such as normalized cuts [22], or graph cuts [21].

We integrate the object boundary probability maps into an example graph-cut segmentation approach due to its simplicity, but we note that any number of more modern methods that minimize unary and pairwise terms could be similarity augmented. In this application, a labeling l is found that assigns each pixel $p \in \mathbb{R}^2$ in I^j a label l_p. A unary constraint sets the class labels for a sparse set of initial seeds, and a pairwise constraint enforces labels to be similar between neighboring pixels of similar color. Consider an image I, the pairwise term can be defined as follows:

$$E_{pairwise} = \sum_{p \in I} \sum_{n \in N(p)} \rho(l_p, l_n) e^{-||I_p - I_n||} \quad (7)$$

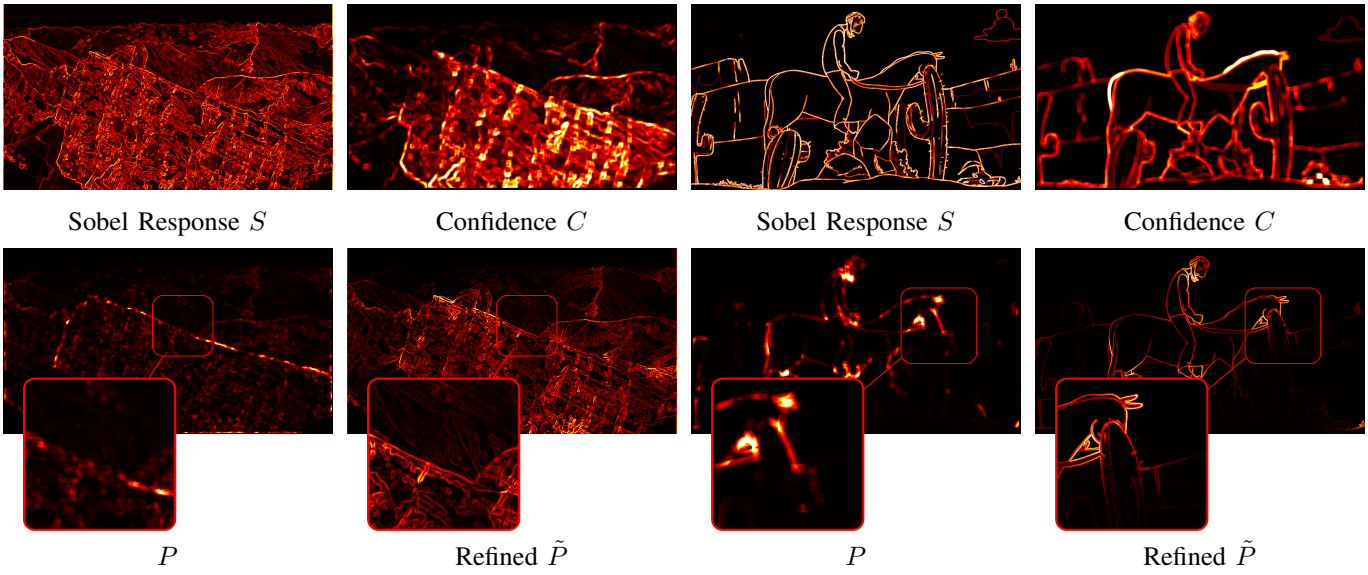


Fig. 4: By refining P, we obtain $\tilde{P}$. This refinement is driven by the confidence map C and the original image edges S. It has the effect of aligning P to image edges, while filling in gaps that arise due to textureless or static regions (inset). In both examples, high $\tilde{P}$ values are correlated to image edges, while the texture edges (the distant mountains in both left and right images, and cactus lines in the right image) are suppressed.

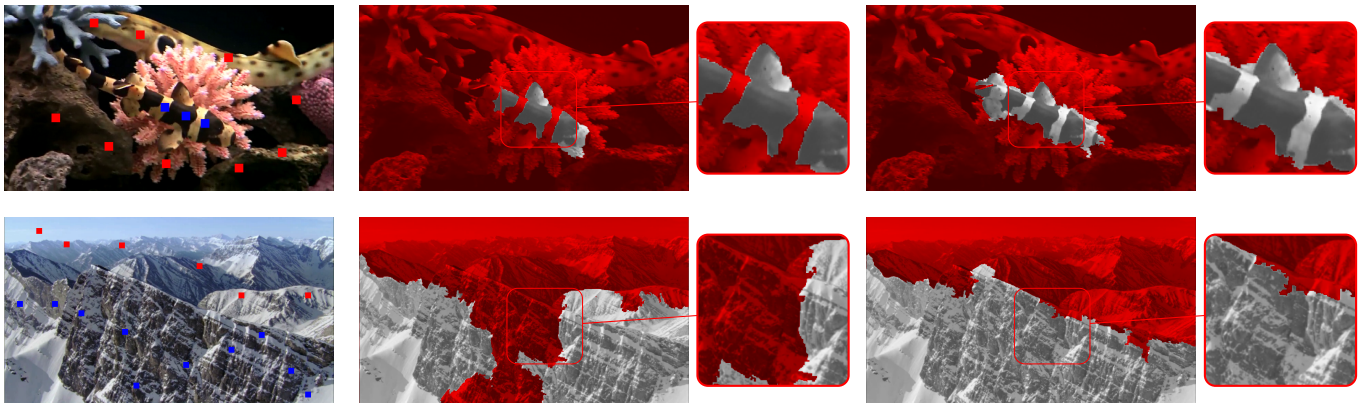


Fig. 5: Graph cut image segmentation [21] from seed points with and without our object boundary probability map. Using $\tilde{P}$ with the same seed points, we achieve a better object segmentation, more successfully isolating the body of the fish (above), and the front facing mountain range (below).

where $N(i)$ are the four pixel neighbors of p, and $\rho(l_p, l_n)$ is a comparison function that is 0 when $l_p = l_n$, and 1 otherwise. This can be understood as an error that occurs when neighboring pixels are assigned a different label, and that is inversely related to the color similarity the two pixels. The optimal labeling l is computed by constructing a graph, and solving a max-flow min-cut problem.

In order to integrate $\tilde{P}$, modify the pairwise term ($E_{pairwise}$) to include a notion of edge type, by weighting the energy with $\tilde{P}$.

$$E_{pairwise'} = \sum_{p \in I} (1 - \tilde{P}_p) \sum_{n \in N(p)} \rho(l_p, l_n) e^{-||I_p - I_n||} \quad (8)$$

This has the effect of increasing the cost of assigning difference labels to neighbors that are *not* at object edges, while reducing the costs of assigning different labels to neighbors that *are* at texture edges.

Figure 5 shows some results using our method. We manually place some seed points as hard constraints (pictured), and perform the segmentation with and without $\tilde{P}$. We can see that because of the strong texture edges, a naive segmentation method has a difficult time locating object boundaries. Using our approach, we can suppress these edges without removing real object depth boundaries.

Optical Flow As further validation, we embed $\tilde{P}$ into an existing state-of-the-art optical flow approach for which

Matlab code was publicly available [17]. This work computes optical flow by minimizing the standard data-plus-smoothness term

$$E_{flow} = E_{data} + \lambda E_{smooth}. \quad (9)$$

It proposes a non-local smoothness weighting where $\lambda = W \in \mathbb{R}^2$ is a spatially varying weight for each pixel. We modify this method by directly including our modified edge map $\tilde{P}$ into the weight map W, producing $\tilde{W}$, e.g.

$$\tilde{W}_p = \tilde{P}_p W_p \ \ \forall p \quad (10)$$

This has the effect of *increasing* the weight of the edge-based regularization term where color differences correspond to *texture* edges, forcing the flow to be more similar, while *decreasing* it at object boundaries, allowing the flow to change.

We show the results using a color-coded optical flow visualization in Figure 6. Using $\tilde{P}$, we can see that lowering the regularization effect of texture edges relative to object edges reduces noise and improves coherence to object boundaries. For example, optical flow vectors of the frog blur into the background in the original implementation, while the shape is better maintained using our method. Additionally, texture edges inside the fish confuse the regularization, causing incorrect motion artifacts, while using our edges, the motion of the whole object is more consistent.

V. Conclusion

We have introduced a simple, easy to implement patch-consistency assumption that allows for texture/object boundary separation. In addition, we proposed an optimization based refinement step, and showed how our approach can trivially improve a sample optical-flow application.

The running time of our method is largely dependent on the speed of PatchMatch, as computing the difference metric is trivial. Using our non-optimized Matlab implementation, on a 2.6ghz 4-core Intel laptop, computing P for a 1-Megapixel image with a temporal window of 5 frames requires roughly 25 seconds , however as newer methods for computing nearest neighbor fields have been shown to operate up to 30 times faster than PatchMatch [23], a much faster version should be theoretically possible. The refinement step requires solving a large linear system of equations, which on the same system requires roughly an additional 30 seconds.

Our method is not without limitations. While we make use of the available information in video sequences, often times this cannot be sufficient to truly determine all object boundaries. For example, when no motion exists, we again have the photograph-ambiguity, in that edges cannot be distinguished from video alone. In addition, we require there to be some degree of visually distinguishing features in the scene for differences in motion and occlusions to be detected. For example, objects edges that move over solid colored backgrounds cannot be used to reliably compute texture edges probability.

To overcome these issues, we presented a confidence-based refinement step, but note that better treatment of these limitations is area for future research. One possible area could be to more extensively use the available video information. Our method operates on a sliding temporal window, producing P independently per-frame. A method that jointly estimates object boundary probabilities over multiple frames would have much more information to work with, but at the cost of computational complexity.

Finally, we note that the performance of our algorithm is strongly dependent on the quality of output from PatchMatch. Fortunately, this approach has shown to be remarkably robust in a number of different applications. Furthermore, PatchMatch is an actively developing area, and since our implementation, more recent versions have been proposed that generate faster and more reliable matches [24], and account for additional geometric and color transformations [25]. These properties should greatly help with tracking patches over time, and future work includes analyzing the performance gains from these extensions.

In summary, we have presented a very simple method that is capable of distinguishing object boundaries edges from texture edges in video. We believe that this simple idea has the potential to help a wide range of applications. It is simple to implement, and performs well on a variety of datasets.

Acknowledgment

The authors would like to thank Mammoth HD, Inc and Poznan University of Technology for allowing us to use samples from their video datasets, and Henning Zimmer for helpful conversations about optical flow.

References

[1] M. Wertheimer, "Untersuchungen zur lehre von der gestalt," *Psychological Research*, vol. 1, no. 1, pp. 47–58, 1922.

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correpondence algorithm for structural image editing," in *Proceedings of ACM SIGGRAPH 2009*, vol. 28, no. 3. ACM Press, 2009.

[3] J. M. S. Prewitt, "Object enhancement and extraction," *Picture processing and Psychopictorics*, 1970.

[4] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986.

[5] M. R. Turner, "Texture discrimination by gabor functions," *Biological Cybernetics*, vol. 55, no. 2-3, pp. 71 – 182, 1986.

[6] I. Fogel and D. Sagi, "Gabor filters as texture discriminator," *Biological Cybernetics*, vol. 61, no. 2, pp. 103 – 113, 1986.

[7] A. K. Jain and F. Farrokhnia, "Unsupervised texture segmentation using gabor filters," *IEEE International Conference on Systems, Man and Cybernetics*, pp. 14 – 19, 1990.

[8] B. A. Maxwell and S. J. Brubaker, "Texture edge detection using the compass operator," in *Proceedings British Machine Conference 2003*. Citeseer, 2003, pp. 9–11.

[9] L. Wolf, X. Huang, I. Martin, and D. Metaxas, "Patch-based texture edges and segmentation," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 481–493.

[10] P. Dollar, Z. Tu, and S. Belongie, "Supervised learning of edges and object boundaries," in *IEEE Conference on Computer Vision and Pattern Recognition,*, 2006.

[11] J. Park, C. Kim, J. Yi, and M. Turk, "Efficient depth edge detection using structured light," *Image and Computer Vision*, vol. 26, no. 11, 2004.

[12] R. Raskar, K.-H. Tan, R. Feris, J. Yu, and M. Turk, "Non-photorealistic camera: Depth edge detection and stylized rendering using multi-flash imaging," in *Proceedings of ACM SIGGRAPH 2004*. ACM Press, 2004.

[13] D. Scharstein and R. Szeliski, "A taxonomy and evaluation of dense two-frame stereo correspondence algorithms," *International journal of computer vision*, vol. 47, no. 1, pp. 7–42, 2002.

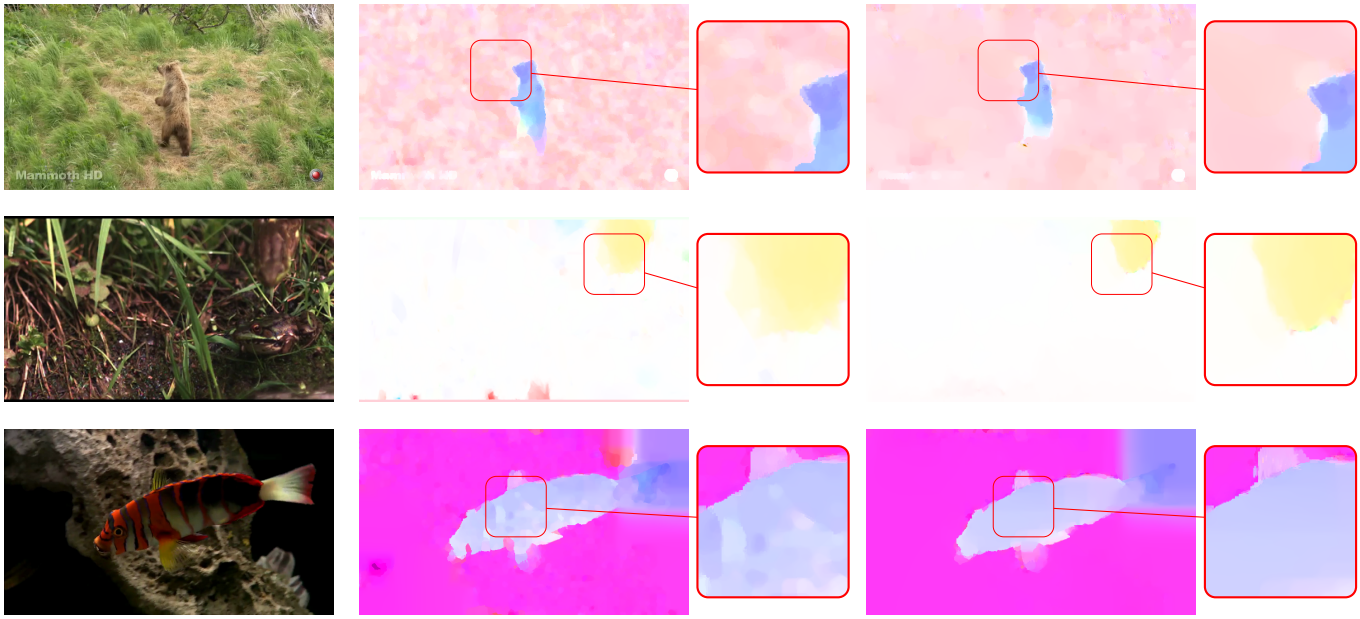


Image | Optical flow | Optical flow using $\tilde{P}$

Fig. 6: Here we show the result of applying a state-of-the-art optical flow method [17] with and without our object boundaries. Using $\tilde{P}$, we can see less noise from textured edges, while motion vectors adhere more to object edges.

[14] ——, "High-accuracy stereo depth maps using structured light," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1. IEEE, 2003, pp. I–195.

[15] L. Zappella, X. Lladó, and J. Salvi, "Motion segmentation: A review," in *Proceedings of the 2008 conference on Artificial Intelligence Research and Development: Proceedings of the 11th International Conference of the Catalan Association for Artificial Intelligence*. IOS Press, 2008, pp. 398–407.

[16] J. Shi and J. Malik, "Motion segmentation and tracking using normalized cuts," in *Computer Vision, 1998. Sixth International Conference on*. IEEE, 1998, pp. 1154–1160.

[17] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2432–2439.

[18] C. Barnes, E. Shechtman, D. B. Goldman, and A. Finkelstein, "The generalized patchmatch correspondence algorithm," in *ECCV*, vol. 3, 2010, pp. 29–43.

[19] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Computer Vision-ECCV 2004*. Springer, 2004, pp. 25–36.

[20] A. Levin, D. Lischinski, and Y. Weiss, "A closed-form solution to natural image matting," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 30, no. 2, pp. 228–242, 2008.

[21] Y. Y. Boykov and M.-P. Jolly, "Interactive graph cuts for optimal boundary & region segmentation of objects in nd images," in *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, vol. 1. IEEE, 2001, pp. 105–112.

[22] J. Shi and J. Malik, "Normalized cuts and image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 22, no. 8, pp. 888–905, 2000.

[23] K. He and J. Sun, "Computing nearest-neighbor fields via propagation-assisted kd-trees," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2012, pp. 111–118.

[24] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image melding: Combining inconsistent images using patch-based synthesis," in *Proceedings of ACM SIGGRAPH 2012*. ACM Press, 2012.

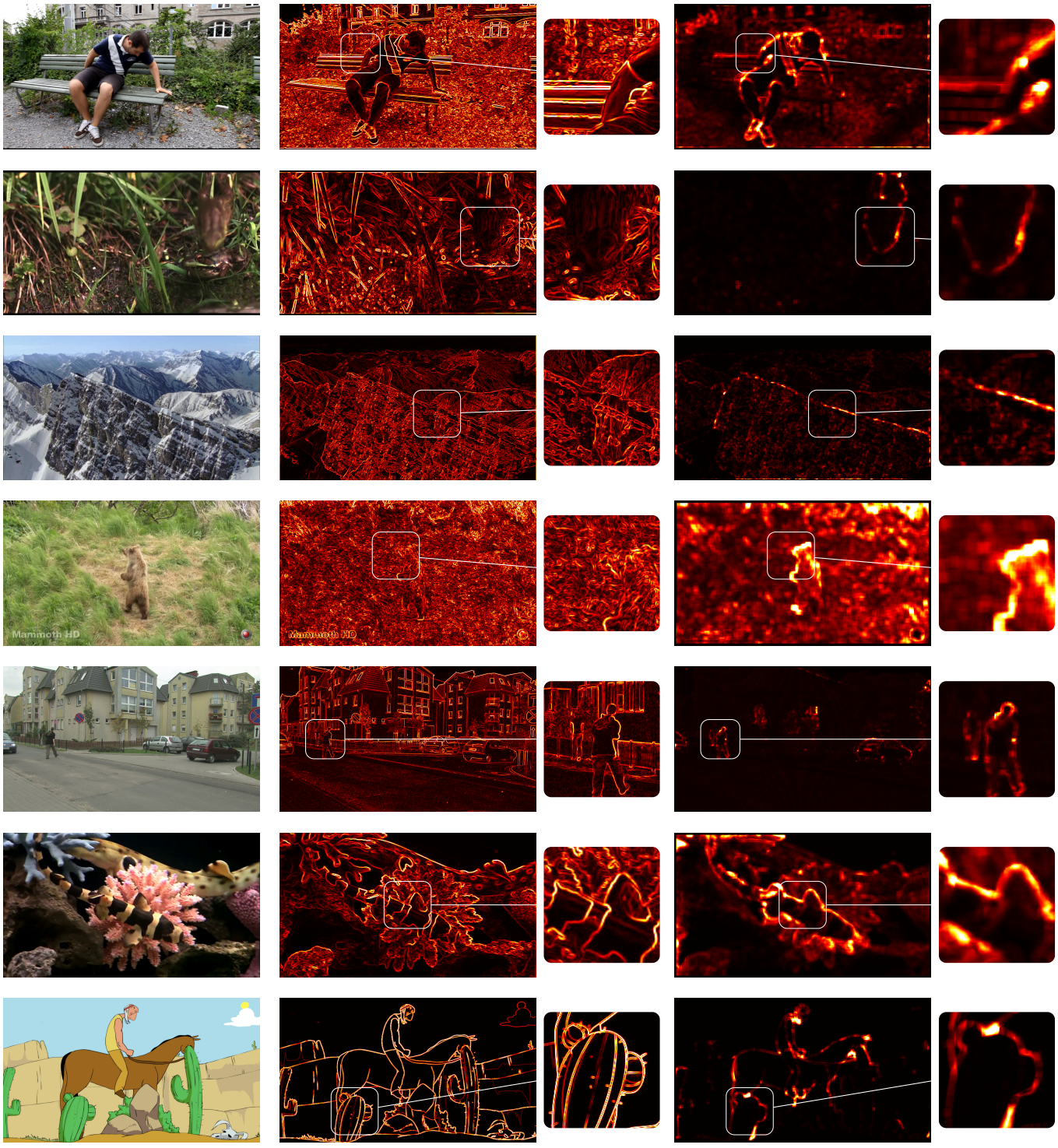
[25] Y. HaCohen, E. Shechtman, D. B. Goldman, and D. Lischinski, "Non-rigid dense correspondence with applications for image enhancement," in *Proceedings of ACM SIGGRAPH 2011*. ACM Press, 2011.

Oliver Wang is currently an Associate Research Scientist with Disney Research Zurich. He received his PhD in Computer Science in 2010 from the University of California, Santa Cruz in the area of computer graphics and image processing. Prior to that, he worked in various research positions at HP Labs, Industrial Light and Magic, and the Max Planck Institut-Informatik.

Martina Dümcke received the B.S. in digital media in 2010 from the University of Bremen, Germany, and her M.S. in computer science in 2013 from ETH Zrich. From 2008 to 2009 she has been a research student in image processing at the University of Electro-Communications, Japan. She is currently a member of the IBM Software Group in Dublin.

Aljoscha Smolic received his Dipl.-Ing. degree in electrical engineering from the Technical University of Berlin, Germany, in 1996, and his Dr.-Ing. degree in electrical and information engineering from Aachen University of Technology (RWTH), Germany, in 2001. Prior to Disney Research, Josh was the Scientific Project Manager at the Fraunhofer Institute for Telecommunications, Heinrich-Hertz-Institut, Berlin, heading a small research group.

Markus Gross is a Professor of Computer Science at the Swiss Federal Institute of Technology Zürich (ETH), head of the Computer Graphics Laboratory, and the Director of Disney Research, Zürich. Before joining Disney, Gross was director of the Institute of Computational Sciences at ETH. He received a master of science in electrical and computer engineering and a PhD in computer graphics and image analysis, both from Saarland University in Germany in 1986 and 1989. He is a fellow of the ACM and of the EUROGRAPHICS Association and a member of the German Academy of Sciences Leopoldina as well as the Berlin-Brandenburg Academy of Sciences and Humanities.



Input Frame I Sobel Response E Object boundary probability P

Fig. 7: Comparisons between Sobel edge magnitudes and our object boundary probabilities P. Edge maps are color-mapped for visibility, but may lose detail in the printing process. We recommend readers view images on-screen and zoomed-in.