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 versus object edges.

We validate our approach by presenting results on a variety of scene types and directly incorporating our augmented edge map into an existing optical flow-based application, showing that our method can trivially suppress the detrimental effects of strong texture edges. Our approach is simple to implement and has the potential to improve a wide range of image and video-based applications.

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 levels. 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.

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

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 image patch corresponds to a texture edge if over time its appearance can be *well modeled* by a set of basic 2D transformations such as translation, rotation and scale while patches on object boundaries, on the other hand, cannot be as easily modeled. This is because of disocclusions, occlusions, and multiple object motions that occur within a single window can 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 not all object boundaries exist where motion is present. But without additional sources of information, stationary objects in video suffer from the same "photograph ambiguity" as single images, and such nothing can be done in these regions. However, as it is possible to detect regions where our assumption cannot be resolved (textureless or motionless), we propose an optional refinement step that propagates estimates from more confident regions.

In practice, we quantify edges by computing patch correspondences over time by leveraging a recent fast nearest-neighbor search method called PatchMatch [2], and evaluate a gradient-based difference metric to quantify how the video content changes within these patches. We apply our method to a variety of scene types, and provide additional validation by incorporating our output in an existing state-of-the-art optical-flow application.

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.

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. 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 local image derivatives correlates to object edge likelihood, 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], or by looking at texture patches directly [8]. These methods assume that object edges are restricted to texture-texture boundaries. However, natural images can have object edges where no such boundaries exist, and texture edges between textures as well.

Machine learning has also been proposed as a means to try to differentiate edge types, for example using boosting [9]. 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 assumptions suffer from the single image "photograph-ambiguity", which we address using video information. Other methods try to overcome this ambiguity by leveraging additional information, such as structured light [10], or a modified camera with physically offset light sources [11]. 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 edges would be to 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 [12], or active illumination [13]. 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 [14], which attempt to group regions with similar flow vectors, for example using normalized cuts [15].

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 [16], 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.

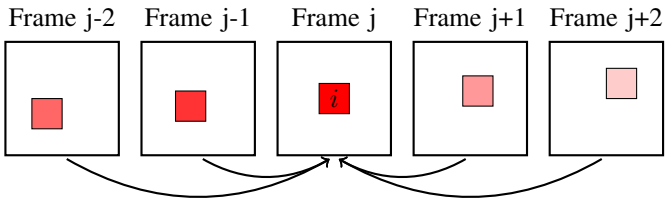


Fig. 1: First, we find best-match patch correspondences in a temporal window. Here we show possible matches for one patch I_i^j over a window of 5 frames.

Unlike motion segmentation methods, our approach can be computed from local, low-level information, without regularization, and does not suffer from the chicken-egg problem. Instead, we leverage a recent approach called *PatchMatch* [2], which efficiently computes a dense patch-based nearest-neighbor field between two images. This approach has several advantages; 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 [17]. 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. As evidence to the difference in approach, we show that the additional information gained by our approach can be used trivially, to improve the quality of state-of-the-art optical flow methods.

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 a map that corresponds to the likelihood of a pixel belonging to an object boundary P given an input video sequence I. We define a patch on at frame j centered at pixel i 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 match I_b^k for all k neighboring frames of j, given a set of restricted transformations; shift, scale, and rotation (Figure 1). 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 (Figure 2).

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 3, we show an example demonstrating this assumption.

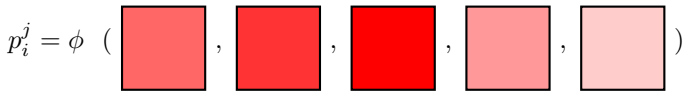


Fig. 2: Second, we apply a difference metric ϕ on these patches to estimate an object contour probability p_i^j. The computed probabilities are assembled into a probability map P, which is the final output of our algorithm.

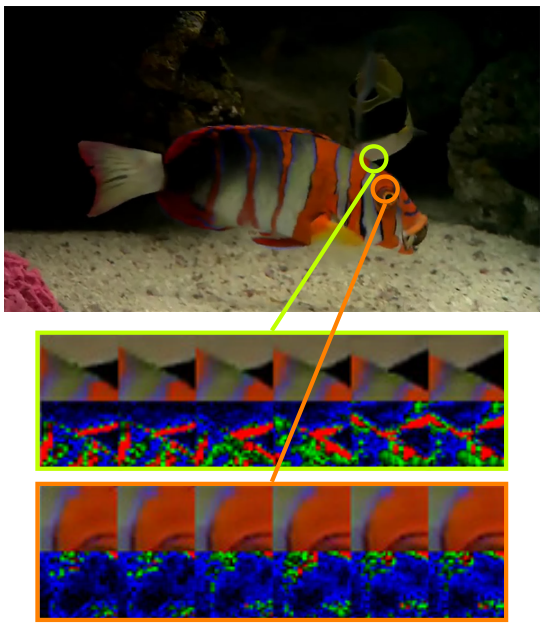


Fig. 3: Best-matched patches at object boundaries exhibit different properties than at texture edges. The orange circle corresponds to a texture edge, which remains more similar over time when compared to a patch at an object boundary (lime). This is visible in the color-mappped (blue to red) patch differences shown each set of matched patches.

Two patches are shown from a frame of video, along with their best matched patches in a temporal window. Below each row is a color-mapped patch-difference visualization.

C. Finding Patch Correspondences

We use the PatchMatch approach to find correspondences over frames [17]. Because we expect the offset for patches to be small when matching between in consecutive frames, we restrict the search window to a maximal of 20 pixel in order to improve matches and reduce running time. We compute, for all patches $i \in I^j$, the best matched patch b in neighboring frames (we note that b is also a function of k as it is different in each frame, but we leave the notation off for clarity).

$$I_b^k = \text{PATCHMATCH}_{k \in N(j)}(I_i^j, I^k) \tag{1}$$

where $N(j)$ are neighbor frames in a temporal window around j, excluding the current frame j.

D. Object Boundary Metric

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

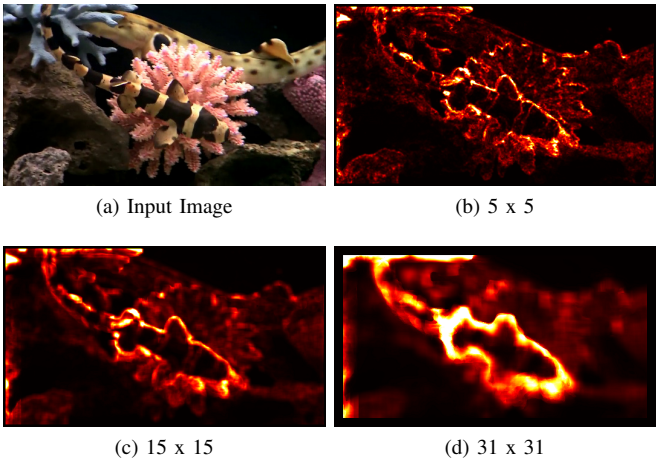
$$p_i^j = \phi(I_i^j, I_b^{(k-n)}, \ldots, I_b^{(k+n)}) \tag{2}$$

where n is the temporal window width. The goal of this metric is to distinguish a set of patches that correspond to texture edges from those that belong to object boundaries. We experimented with different functions for ϕ, including common difference metrics such as normalized cross correlation, histogram difference, bimodality, mean/median L1 and L2 differences. In the end, we use a combination of color and gradient differences, as this approach matches structure details well and is somewhat invariant to lighting changes (although we note that a simple mean-of-means approach also yields good results). In other words,

$$\phi(I_i^j, I_b^{(k-n)}, \ldots, I_b^{(k+n)}) = \frac{1}{||N(j)||} \sum_{k \in N(j)} (||I_i^j - I_b^k|| + .5|\nabla I_i^j - \nabla I_b^k|) \tag{3}$$

where ∇ is the gradient operator, and patch differences are averaged over all pixels. 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 4. Large patches give more reliable measurements, but lose detail, as all patches that overlap object edges validate our assumption. 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 15x15 pixels and a temporal window of $n = 2$.



(a) Input Image (b) 5 x 5

(c) 15 x 15 (d) 31 x 31

Fig. 4: The effect of patch size on P. Larger patch sizes exhibit less noise, but lose edge resolution.

E. Edge Refinement

We note that P becomes visibly fuzzy, as a result of the chosen patch size. While this P may be sufficient for many applications, if more accurate edges are required, we present an optimization-based method to refine the initial

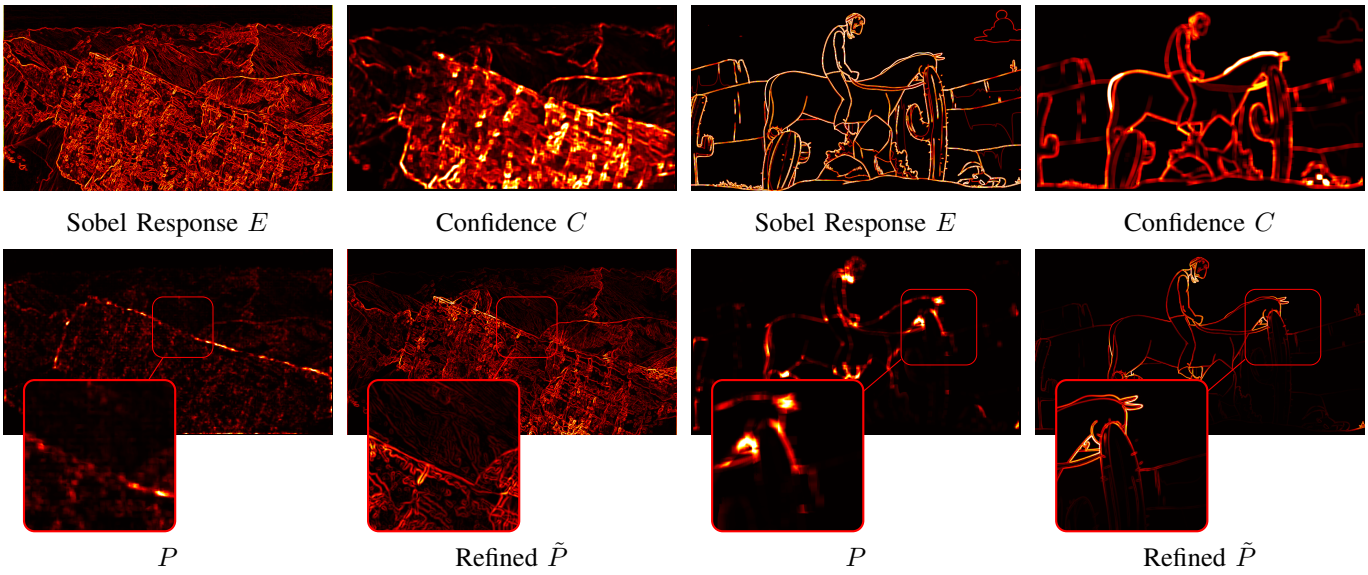


Fig. 5: By refining P, we obtain $\tilde{P}$. This refinement is driven by the confidence map C and the original image edges E. It has the effect of aligning P to image edges, while filling in gaps that arise due to textureless or static regions (inset). In both examples, high $\tilde{P}$ values are 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.

object boundary. A straightforward approach would be to simply modulate the current edge map E by P, e.g.,

$$\hat{P}_i = P_i E_i \tag{4}$$

However, we observe that our patch-consistency assumption is invalidated in several cases; when patches have no image texture, and when no motion has occurred. We can therefore compute a confidence map C, that is derived from these two assumptions;

$$C_i = \frac{1}{||N(j)||} \sum_{k \in N(j)} \left(\delta(i,b) + var(I_b^k)\right) \tag{5}$$

where $\delta(i,b)$ is a function that is proportional to the Euclidean distance between locations i and b (i.e., the amount of motion between the pixel and its match), and $var(I_b^k)$ computes the variance of the patch. We use

$$\delta(i,b) = \frac{1}{\tau} min(||i-b||_2, \tau). \tag{6}$$

This implies that motions greater than τ pixels are considered equally confident. In all results we use a small value of $\tau = 3$ to penalize only nearly static regions. Additionally, $var(I_b^k)$ is low when the texture content of the patch is low.

By using C to propagate $\hat{P}$, low confidence regions can "borrow" information from more confident neighbors, and propagate this along image edges. If we consider images to be vectors with length $l = width * height$, we can write this concept in vector form as:

$$E(\tilde{P}) = (\hat{P} - \tilde{P})^T G (\hat{P} - \tilde{P}) + \lambda \tilde{P}^T L \tilde{P} \tag{7}$$

where G is a diagonal matrix of size l by l whose elements are equal to C, L is the Matting Laplacian [18], and λ is a regularization parameter. The weighted data term $(\hat{P}-\tilde{P})^T G(\hat{P}-\tilde{P})$ preserves the similarity of the refined map $(\tilde{P})$ to the modulated one $(\hat{P})$ in regions where the confidence (C) is high. The elements of L are computed from the *edge* image E, such that the smoothness term $\tilde{P}^T L \tilde{P}$ enforces that neighboring values in $\tilde{P}$ are similar *only* if they are also similar in E.

This can be solved as a sparse system of equations, generating a refined mapping $\tilde{P}$, with values that are similar to the original estimates, but are *propagated* along image edges in regions of low confidence. In Figure 5, we show some results from this refinement step.

IV. Application

As further validation, we embed $\tilde{P}$ into an existing state-of-the-art optical flow approach for which Matlab code was publicly available [16]. This work computes optical flow by minimizing the standard data-plus-smoothness term $E = E_{data} + \lambda E_{smooth}$. It proposes a non-local smoothness weighting where $\lambda = w_{i,u}$ is a spatially varying weight for each pixel i, summed over a window (u is the motion vector). We modify this method by directly using our modified edge map $\tilde{P}$ to modulate the weighted non-local term, e.g. $\lambda = \tilde{P}_i w_{i,u}$.

This has the effect of increasing the weight of the edge-based smoothness term where color differences corresponds to *texture* edges, while decreasing it at object boundaries.

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, computing P for a 1-Megapixel image with a temporal window of 5 frames requires roughly 30 seconds, however as newer methods for computing nearest neighbor fields have been shown to operate up to 30 times faster than PatchMatch [19], a much faster version should be theoretically possible. The refinement step requires solving a large linear system, which again using a direct solve in Matlab takes 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 visual features in the scene for differences in motion and occlusions to be detected. When objects are uniform in color, motion cannot be correctly determined.

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 [20], and account for additional geometric and color transformations [21]. 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] L. W. X. H. I. Martin and D. Metaxas, "Patch-based texture edges and segmentation," in *Proceedings of the 9th European conference on Computer Vision*, 2006, pp. 481 – 493.

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

[10] 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.

[11] 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.

[12] 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.

[13] ——, "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.

[14] 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.

[15] 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.

[16] 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.

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

[18] 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.

[19] 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.

[20] 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.

[21] 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.

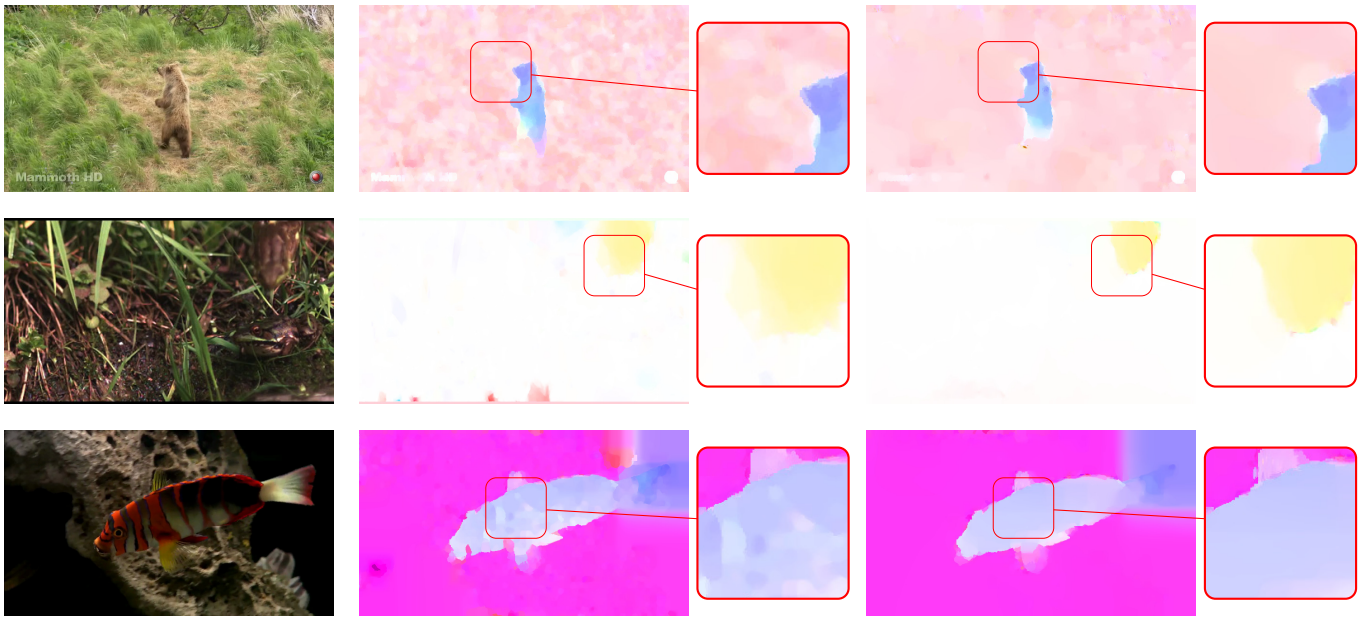


Image Optical flow Optical flow using $\tilde{P}$

Fig. 6: Here we show the result of applying a state-of-the-art optical flow method [16] 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.

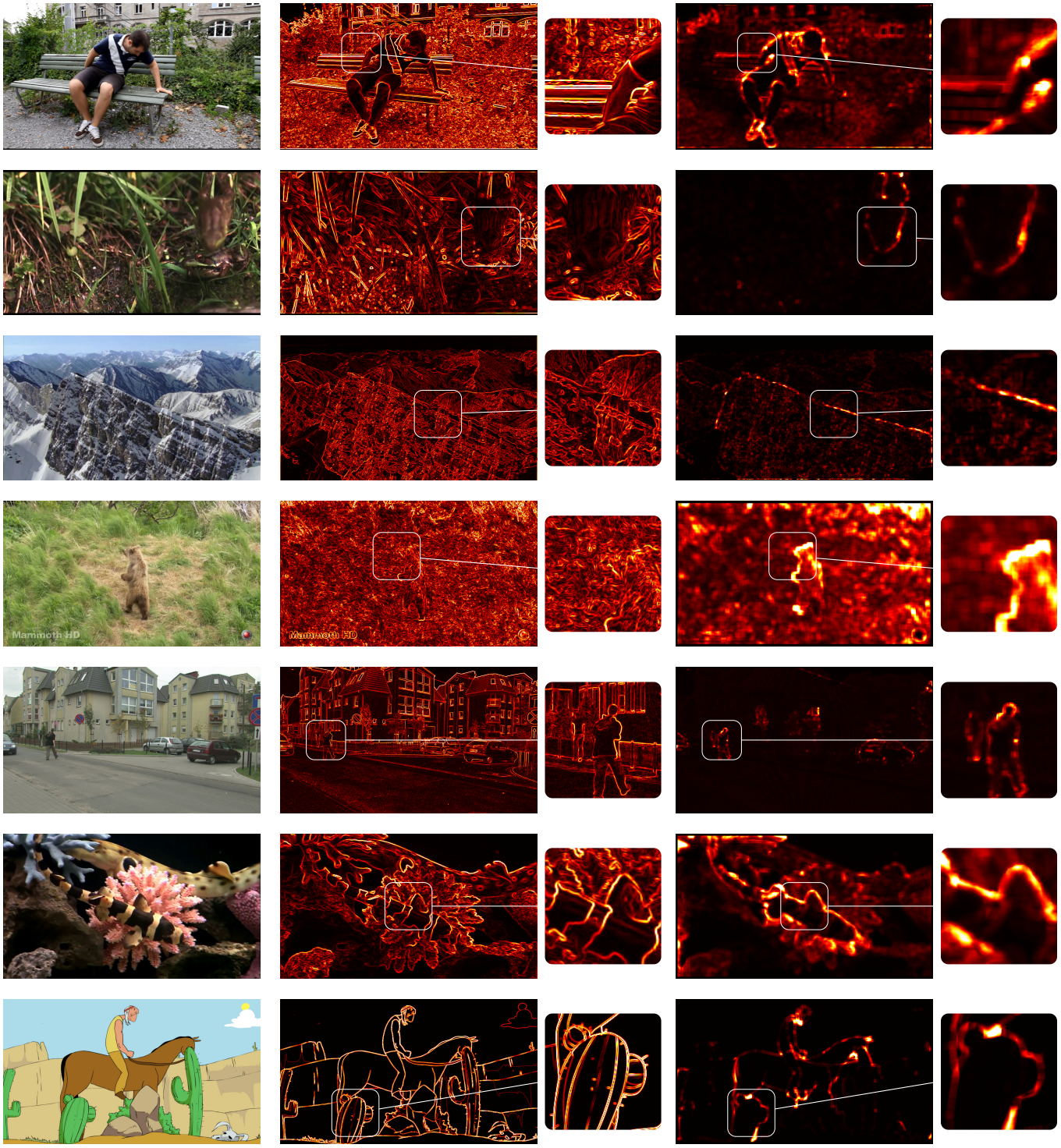


Fig. 7: Comparisons between Sobel edge magnitudes and our object boundary probabilities P. Edge maps are color-mapped for visibility, but may lose detail in the printing process. We recommend readers view images on-screen and zoomed-in.