LABEL PROPAGATION THROUGH EDGE-PRESERVING FILTERS

*Richard Rzeszutek** *Dimitrios Androutsos*

Department of Electrical and Computer Engineering
Ryerson University
Toronto, Ontario, Canada.
{rrzeszut, dimitri}@ee.ryerson.ca

ABSTRACT

In this paper we investigate methods for propagating automatically generated or user-defined labels through an image using edge-preserving filters. We focus on the domain transform filter as it has been used for propagation purposes in the past. The method we present addresses some of the numerical issues that arise with using the filter directly and also improve on the results by better respecting the underlying image structure during the label propagation. Finally we also demonstrate how a filter-based approach is preferable to using global optimization for interpolating automatically generated sparse features.

Index Terms— Domain Transform, 2D to 3D Conversion, Label Propagation, Sparse Feature Interpolation

1. INTRODUCTION

Label propagation is the process by which pixels in an image are assigned membership, either exclusionary or through some weighted mixture, to a set of labels $\mathcal{L}$. The propagation process is done in such a way that the underlying image content is respected. For instance, in the context of image segmentation [1, 2], this corresponds to assigning a pixel to the "most similar" label in such a way that the object boundaries are respected. This would be an example of an exclusionary labelling: each pixel can only be related to one label. Alternatively, colourization [3] is an example of a "soft" labelling where the labels are mixed together during the propagation. The process can be best described as applying anisotropic diffusion [4] to the labels. This ensures that the labels will be mixed (diffused) in homogeneous regions, regions where there are no textures or edges, but will not go beyond object boundaries. Typically, but not always, these correspond to strong image edges.

The 2D to 3D conversion process takes an existing image and generates novel views that simulate as if the image was taken with another camera. In the absence of any other information, a user-guided approach can produce very good results [5, 6, 7, 8]. This is essentially the same problem as colourization where a user provides sparse labels to add colours to a monochromatic image or to modify the colours in a colour image. The difference is that rather than diffusing user-provided colour values, depth values are diffused instead.

Recently, fast edge-preserving filters have been developed that can, in principle, be used to perform label propagation. Lang et al used such a technique to propagate labels through video sequences [9]. By processing the user labelling with the domain transform (DT) filter [10] both spatially and temporally, Lang et al were able to propagate the labelling over long duration sequences (they referred to it as "scribble propagation"). Specifically, they jointly estimate the inter-frame optical flow, also with the DT filter, along with the label propagation. The filtering is done in an iterative manner: first spatially (propagating the labels through the image) and then temporally using the optical flow (determining where the labels will be in the next frame). The number of iterations they have to perform is relatively small; they report using four in their paper.

By processing video Lang et al essentially mitigate a common problem in label propagation: edge localization. Their method utilizes an occlusion penalty that, over several iterations, helps to identify where object edges actually are as even weak edges will occlude the background if the object moves. This avoids propagating the labels into regions where the object is not.

Unfortunately this information is simply not available for label propagation in single images. For methods to generate depth maps for images, directly applying an edge-preserving filter to the user-provided strokes will produce results inferior to those from methods based on global optimization. Furthermore, filter-based methods are highly sensitive to the filter parameters and the appropriate parameters are dependent on the image content. However there are certain cases where an edge-preserving filter is *preferable* to a global optimization approach such as [6, 8].

In this paper we present a method for interpolating potentially noisy sparse features using edge-aware filters. We focus on the domain transform filter as we have found that it is well-suited to this task and address some of the numerical issues inherent with using such an interpolation technique. Because we are only changing how the labelling is interpolated, the basic interface does not change.

Please note that we provide full-resolution versions of all of our figures at `http://www.ee.ryerson.ca/~rrzeszut/icassp2014`.

1.1. Relation to Prior Work

In [11] we presented a method, similar to the Depth Director system [12] by Ward et al, whereby we could automatically extract depth maps from a sequence and allow for user-guided corrections. As part of our method, we presented a simple interpolation scheme to convert the sparse depth estimates into a depth map using the domain transform filter. Because the labels were being processed directly, we were able to also add user corrections in regions, for instance, where the incorrect depth was estimated.

The basic interpolation method was similar to what was already presented in [10] and [9] but in this paper we better expand on how that method can be applied and how to improve its results. More specifically we address how to improve the edge localization since

*Corresponding author.

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

this is especially important when generating depth maps. Please see [6] and [8] for more information.

2. DOMAIN TRANSFORM FILTERING

We begin by providing a brief overview of the Domain Transform (DT) filter. We ask the reader to refer to [10] for a more in-depth discussion. The premise of the filter is that given some domain n, there is a transform $T[n]$ that maps $n \rightarrow m$ such that similar features in n are close together in m. The transform itself is defined as

$$T[n] = \sum_{i=0}^{n} \left\{ 1 + \frac{\sigma_s}{\sigma_r} \sum_{c=0}^{D-1} \left| I_c[i] - I_c[i-1] \right| \right\}, \quad (1)$$

where I_c is the c-th colour channel of a D-colour image $I(x, y)$. The transform is defined purely in one dimension and so $I_c[i]$ refers to either a row or column of the image $I(x, y)$. The transform is such that the distance between samples in the transformed domain is proportional to the L_1-norm in the original domain. When constructing a filter from the transform σ_s defines its spatial extent while σ_r defines its extent with respect to the values of $I(x, y)$.

Filtering with $T[n]$ can be performed in a number of ways and Gastal and Oliviera provide three methods, two[1] of which will be discussed in this paper. The normalized convolution filter (DTNC) produces a filtered output $y[n]$ from a discrete signal $x[n]$ by

$$y[n] = \frac{1}{j-i} \sum_{k=i}^{j} x[k], \quad (2)$$

where i and j are chosen such that $(T[j] - T[i]) = 2r$ and $i < n < j$ for some filter radius r. This is a standard FIR moving average filter whose width is controlled by $T[n]$. The recursive form filter (DTRF) is a first-order IIR filter defined as

$$y[n] = (1 - a^d)x[n] + a^d y[n-1], \quad (3)$$

where $d = T[n] - T[n-1]$ and $0 \leq a < 1$ is a feedback coefficient. Since a is defined to be less than 1, the filter is stable (the pole remains inside of the unit circle). This filter is not symmetric and so Gastal and Oliviera apply this in two passes: first in the forward direction and then in the reverse direction. Please note that both filter variants are linear; the domain transform effectively produces a different filter at each value of n.

Because the filter is only defined for one-dimensional signals, filtering a two-dimensional image is done iteratively in multiple horizontal/vertical passes. This implicitly assumes that the DT is a separable filter when in fact it is not. Therefore after each pass the filter variance is halved so that filtering becomes progressively weaker to avoid unbounded smoothing. Gastal and Oliviera define a parameter $\sigma_H(i)$ such that

$$\sigma_H(i) = \frac{\sigma_s}{\sigma_r} \sqrt{3} \frac{2^{N-i}}{\sqrt{4^N - 1}}, \quad (4)$$

where $1 \leq i \leq N$ is the current filter iteration. The DTNC filter radius is then defined as $r = \sigma_H(i)\sqrt{3}$ and the DTRF feedback coefficient is defined as $a = \exp\left(-\sqrt{2}/\sigma_H(i)\right)$. The result of this is that as $N \rightarrow \infty$ the DT filter, regardless of its variant, will have the same response as a Gaussian filter if $\sigma_r \rightarrow \infty$.

[1]The third, interpolated convolution, has similar properties to normalized convolution and will not be discussed.

(a) DTNC (b) DTRF

Fig. 1: A comparison between the DTNC and DTRF when applied to an image. In both images the filter parameters were $\sigma_s = 100$, $\sigma_r = 1$ and $N = 3$.

A comparison between the two DT filter types is shown in Figure 1. Generally the DTRF filter is more "blurry" than the DTNC filter due to the IIR filter propagating information throughout the image. This can be best seen on the left side of the image where the leaves are more visible Figure 1b. The fact that the DTNC is essentially a "local" filter has important implications when using it for label propagation as it better respects image edges.

3. LABEL PROPAGATION THROUGH FILTERING

In this section we describe how the DT filter is used for label propagation. We are interested in this for the purpose of creating depth maps from sparse depth labels. This is a summary of what was originally proposed by Gastal and Oliviera in [10] and elaborated upon by Lang et al in [9]. Both papers drew on the work of Fattal et al [13] for image colourization. How this approach works is important in understanding some of its shortcomings, which we address in Section 4.

Consider the simple case of a one-dimensional "image" signal $g[n]$. The content of the signal is inconsequential as each DT filter is essentially uniquely generated for a particular $g[n]$. This remains the same throughout our analysis and so the same filter is used throughout. Let the stroke signal be $s[n] = L\delta[n]$ and the confidence signal be $c[n] = \delta[n]$, where $\delta[n]$ is the Dirac delta function and L is the label value. Therefore, the filtered confidence signal $c'[n] = DT\{c[n]|g[n]\}$ is simply the impulse response of the DT filter.

Similarly the label signal can be filtered so that $s'[n] = DT\{s[n]|g[n]\}$. However the label signal is really a scaled version of $c[n]$ and therefore $s'[n] = Lc'[n]$. Dividing $s'[n]$ by $c'[n]$ results in the final interpolated result of $s_{\text{int}}[n] = L$. This is expected as there is a single impulse (label) and so it should be applied to all values of n.

In the preceding example it was assumed that there was only a single label. However it is simple to generalize to the case of multiple impulses (labels), i.e $s[n] = \sum_i L_i \delta[n - P_i]$, where P_i is the position of the i-th label. This in turn results in $c[n] = \sum_i \delta[n - P_i]$. If we define $c'_i[n] = \text{DT}\{\delta[n - P_i]|g[n]\}$ then the interpolated label signal can be found as

$$s_{\text{int}}[n] = \frac{1}{\sum_i c'_i[n]} \sum_i L_i c'_i[n]. \quad (5)$$

Each label is therefore distributed through out the image by the filter and then normalized by the relative contribution of the label itself. This is why it was referred to as a "normalization image".

It is possible to extend the use of $c[n]$ so that it also indicates the relative *confidence* of a particular label. Consider the case were

the labelling is automatically generate by some process, such as sparse feature matching. In this case, each label will also have an associated "quality" or confidence value that indicates how reliable that label is. Now the confidence signal is scaled such that $s[n] = \sum_i (L_i C_i)\delta[n - P_i]$ and $c[n] = \sum_i C_i \delta[n - P_i]$, where C_i is the relative confidence value between 0 and 1. This modifies (5) so that it becomes

$$s_{\text{int}}[n] = \frac{1}{\sum_i C_i c'_i[n]} \sum_i (L_i C_i) c'_i[n]. \tag{6}$$

This weights each label by the confidence value so that a poor quality label will have less impact than a good one. More can be done to change what labels are, and are not, included in the final interpolation and much of [9] is devoted to this.

Extending (6) to images is straightforward. We assume that the user has provided a sparse labelling image

$$\mathcal{L}(x, y) = \begin{cases} L(x, y) & \text{label exists at pixel } (x, y) \\ L_{\text{un}} & \text{otherwise} \end{cases}, \tag{7}$$

where $L(x, y)$ is the label's value at (x, y) and L_{un} is a special value used to indicate that the pixel is unlabelled. From this we define the confidence (i.e. normalization) map

$$\mathcal{C} = \begin{cases} 1 & \mathcal{L}(x, y) \neq L_{\text{un}} \\ 0 & \text{otherwise} \end{cases}. \tag{8}$$

The final, interpolated value is then be obtained by

$$\mathcal{L}_{\text{int}}(x, y) = \frac{\text{DT}\{\mathcal{C}(x, y)\mathcal{L}(x, y)|I(x, y)\}}{\text{DT}\{\mathcal{C}(x, y)|I(x, y)\}}, \tag{9}$$

where $\text{DT}(\cdot)$ is the domain transform filter operator. Please note that the multiplication and division are performed *per-pixel*. Because we desire to use the image to guide the label propagation, the domain transform in (1) is obtained from the original image $I(x, y)$. This in turn blurs the labelling based on the content of the original image. A problem with this approach is that it is possible for there to be regions of the image where the labels were not propagated, i.e. the numerator and denominator of (9) are both zero. This effects the interpolation and we address this in the proceeding section.

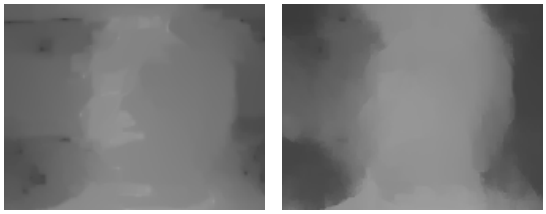
4. SPARSE FEATURE INTERPOLATION

The problem of interpolating sparse features is, in principle, the same as interpolating user-provided labels. The difference, however, is that the labelling is now generated by some automated process rather than a user. This changes the requirements for the interpolation in that the labelling can no now longer be assumed to be completely correct.

4.1. Optimization versus Filtering

If the labelling is not correct then using an optimization-based approach can result in artifacts when producing the final output. Figure 2 shows the extracted depths for a frame of a video sequence that was processed using [11]. The size of the features has been exaggerated for visualization purposes (they are the size of individual pixels). The depth of any particular feature is independent of that of any other feature and so two neighbouring features may have completely different depth values.

Fig. 2: Sparse depth labels obtained through automatic processing. Brighter colours indicate features closer to the camera.



(a) Global Optimization (b) DTRF

Fig. 3: A comparison between using global optimization and the DTRF filter to generate the complete depth maps. The filter parameters used were $\sigma_s = 500$, $\sigma_r = 2$ and $N = 4$.

Any small errors in the estimates will not be immediately visible from just examining Figure 2. However, they become evident when generating a dense depth map. Figure 3 a comparison between the depth map generated when using the method in [6] and using (9) with a DTRF filter. Only the region around the head is shown for clarity.

In Figure 3a there are small, slightly darker regions in the map while in 3b these are greatly reduced. This is a result of the optimization-based result respecting the input labelling. The solution it generates will be completely consistent with what it was provided so if there are any errors, they will be part of that solution. The filter, though, will simply smooth out any of the smaller differences. Using a filter for this application is preferable to optimization because it is more tolerant to an erroneous input labelling.

However, the results produced by the DT filter depend on a large part on the parameters used. This can be seen in Figure 4 where the DTNC filter in particular experiences a dramatic change between the two σ_r values. This is unfortunate as the DTNC filter is very good a preserving large-scale image structures and is desirable to use for label propagation. Choosing a larger σ_r will reduce the likelihood of a $0/0$ condition occurring. But, not only does it not remove it entirely, it will produce significantly more blurring in the interpolated map.

4.2. Iterative Refinement

Ideally we would like to choose a small σ_r to minimizing the blurring while still propagating the provided labelling. To do this we propose an iterative approach that progressively fills in the $0/0$ regions to generate the final labelling. This acts as a simplified diffusion procedure to fill in region that the labelling did not reach initially.

Let $\mathcal{L}^{(0)}_{\text{int}}$ be the initial map generated using (9). We define a

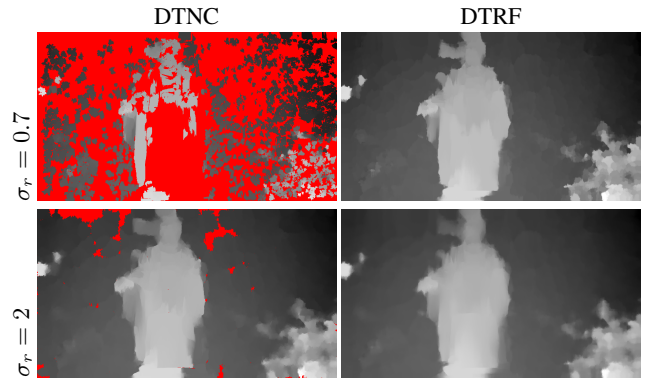


Fig. 4: Comparison between the DTNC and DTRF filters for different values of σ_r with $\sigma_s = 500$ and $N = 4$. Red areas indicate that (9) was $0/0$.

Fig. 5: Depth map produced using DTNC iterative refinement. The filter parameters were $\sigma_s = 500$, $\sigma_r = 0.7$, $N = 4$ and $N_R = 10$.

per-iteration confidence map $\mathcal{C}^{(i)}$ to be

$$\mathcal{C}^{(i)} = \begin{cases} 1 & \mathcal{L}_{\text{int}}^{(i-1)} \text{ is } 0/0 \\ 0 & \text{otherwise} \end{cases}. \tag{10}$$

$\mathcal{C}^{(i)}$ is then used as the confidence map in (9) to generate $\mathcal{L}_{\text{int}}^{(i)}$. We repeat this procedure until no more $0/0$ pixels are detected or until N_R iterations have been reached (we have found 10 iterations to be sufficient in most cases).

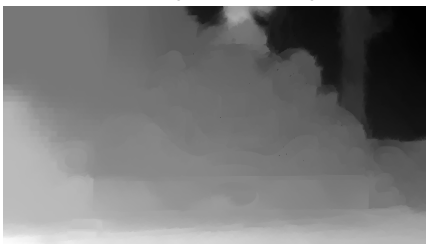
The results of the iterative procedure are shown in Figure 5. They are a significant improvement on the equivalent results produced by the same filter parameters in Figure 4. Because σ_r could be set to a small value, the main edges are well preserved even though the image was filtered multiple times.

4.3. User Correction

The iterative refinement approach has no impact on the ability of the method to accept user corrections. Figure 6 shows an image that has depth values obtained by processing the sequence it is part of using [11]. In this case, there was a failure in the sparse feature tracking and so the left-half of the image has no tracked features and therefore no depth information. The left side of the depth map is completely wrong because of this.

It is trivial for a user to recognize and correct this with a few scribbles to indicate the correct depth values in the untracked regions. This correction is shown in Figure 7. The resulting depth map now has the correct depth and the only modification was to the input labelling, not the label propagation method.

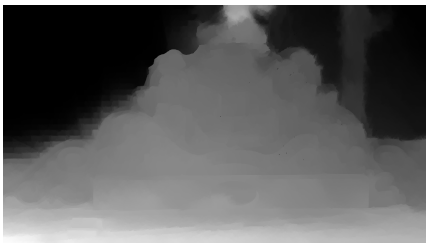
(a) Original Labelling



(b) Depth Map

Fig. 6: Depth map obtained from the original, automatically extracted labelling. Filter parameters are $\sigma_s = 500$, $\sigma_r = 0.7$, $N = 4$ and $N_R = 10$.

(a) Corrected Labelling



(b) Depth Map

Fig. 7: Depth map generated after user-provided correction scribbles. The same filter parameters were used as in Figure 6.

5. CONCLUSION

We have presented a method for interpolating labels, either user-provided or generated automatically using the domain transform filter. Specifically we have addressed the numerical issues inherent with this type of interpolation. While the DTRF is less prone to it than the DTNC, it is useful to be able to use the DTNC variant as it produces more distinct edges than the DTRF. We have also shown how a filter-based approach can be preferable when interpolating potentially erroneous, automatically generated labelling.

6. REFERENCES

[1] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient n-d image segmentation," *International Journal of Computer Vision*, vol. 70, no. 2, pp. 109–131, 2006. [Online]. Available: http://dx.doi.org/10.1007/s11263-006-7934-5

[2] L. Grady, "Random walks for image segmentation," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 11, pp. 1768–1783, 2006.

[3] A. Levin, D. Lischinski, and Y. Weiss, "Colorization using optimization," *ACM Trans. Graph.*, vol. 23, no. 3, pp. 689–694, Aug. 2004. [Online]. Available: http://doi.acm.org/10.1145/1015706.1015780

[4] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 7, pp. 629–639, 1990.

[5] M. Guttmann, L. Wolf, and D. Cohen-Or, "Semi-automatic Stereo Extraction from Video Footage," *Proc. IEEE Intl. Conf. on Computer Vision (ICCV)*, 2009.

[6] R. Phan, R. Rzeszutek, and D. Androutsos, "Semi-automatic 2d to 3d image conversion using scale-space random walks and a graph cuts based depth prior," in *Image Processing (ICIP), 2011 18th IEEE International Conference on*, sept. 2011, pp. 865 –868.

[7] O. Wang, M. Lang, M. Frei, A. Hornung, A. Smolic, and M. Gross, "Stereobrush: interactive 2d to 3d conversion using discontinuous warps," in *Proceedings of the Eighth Eurographics Symposium on Sketch-Based Interfaces and Modeling*, ser. SBIM '11. New York, NY, USA: ACM, 2011, pp. 47–54. [Online]. Available: http://doi.acm.org/10.1145/2021164.2021173

[8] R. Phan and D. Androutsos, "Robust semi-automatic depth map generation in unconstrained images and video sequences for 2d to stereoscopic 3d conversion," *Multimedia, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2013.

[9] M. Lang, O. Wang, T. Aydin, A. Smolic, and M. Gross, "Practical temporal consistency for image-based graphics applications," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 34:1–34:8, Jul. 2012. [Online]. Available: http://doi.acm.org/http://doi.acm.org/10.1145/2185520.2185530

[10] E. S. L. Gastal and M. M. Oliveira, "Domain transform for edge-aware image and video processing," *ACM TOG*, vol. 30, no. 4, pp. 69:1–69:12, 2011, proceedings of SIGGRAPH 2011.

[11] R. Rzeszutek and D. Androutsos, "Efficient automatic depth estimation for video," in *Digital Signal Processing (DSP), 2013 18th International Conference on*, 2013, pp. 1–6.

[12] B. Ward, S. B. Kang, and E. Bennett, "Depth director: A system for adding depth to movies," *Computer Graphics and Applications, IEEE*, vol. 31, no. 1, pp. 36 –48, jan.-feb. 2011.

[13] R. Fattal, "Edge-avoiding wavelets and their applications," *ACM Trans. Graph.*, vol. 28, no. 3, pp. 1–10, 2009.