Fast Texture-Based Tracking and Delineation Using Texture Entropy*

Ali Shahrokni† Tom Drummond‡ Pascal Fua†

ali.shahrokni@epfl.ch twd20@eng.cam.ac.uk pascal.fua@epfl.ch

† Computer Vision Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), CH-1015 Lausanne, Switzerland

‡ Department of Engineering, University of Cambridge, Cambridge CB2 1PZ, UK

Abstract

We propose a fast texture-segmentation approach to the problem of 2-D and 3–D model-based contour tracking, which is suitable for real-time or interactive applications.

Our approach relies on detecting texture boundaries in the direction normal to the contour boundaries and on using a Hidden Markov model to link these boundary points in the other direction. The probabilities that appear in this computation closely relate to texture entropy and Kullback-Leibler divergence, a property we use to compute and update dynamic texture models.

We demonstrate results both in the context of interactive 2-D delineation and fast 3-D tracking.

1. Introduction

In earlier work [10], we proposed a texture-based approach to finding the edges of highly textured objects in the presence of clutter. We demonstrated that, unlike most texture-based techniques, it is fast enough for real-time tracking. However, the estimated texture distribution becomes more accurate as we get more observations on each scanline. Moreover, because it relies on learning a texture model inside the object, it may loose track as the object's appearance changes. Here we overcome this problem by incorporating

HMM and scanstripes: A Hidden Markov Model is defined to bind and exploit local texture information in form of texture crossing probabilities on a bundle of scanlines [10] that we refer to as scanstripes. This yields the most likely connected contour(s) of the object.

*This work was supported in part by the Swiss National Science Foundation.

Kullback-Leibler Divergence (KLD): The relative entropy between the actual target texture and the prior target texture model is measured in the course of tracking and used to dynamically update the texture model.

We will show that this yields more robust tracking behavior, for example under lighting changes. Furthermore, this opens the door for other applications that require near real-time performance, such as the interactive drawing of image boundaries that are becoming increasingly popular in image-processing systems such as Photoshop. In the remainder of this paper, we first discuss related work. We then introduce scanstripes and their advantages over scanlines. In Section 3.3 we go through the Hidden Markov Model that aggregates the scanstripe information. Updating the prior target texture model and the relationship between texture changepoint probabilities and the texture entropy is discussed in Section 4. Finally Section 5 reports our experiments on tracking with KLD-based update and interactive segmentation of different objects.

2. Related work

The contours used for most tracking applications are extracted either based on local gradient information [3, 11] or edge distance transform [6, 12]. These methods are appealing due to their simplicity and speed, but their application is restricted to the cases where the contrast is sufficient. Furthermore, these methods tend to fail in the presence of highly textured objects and clutter, which produce too many irrelevant edges. This is due to their failure to exploit the intensity values and structures available in the image. The main reason for not exploiting this information is the complexity of texture patterns and difficulty in adaptation of texture segmentation and analysis techniques to tracking applications.

The main idea here is somewhat similar to interactive texture segmentation problems such as work by Blake *et. al.* [1] where we start from an initial user-provided guess about foreground, background and undetermined regions and the goal is to segment the undetermined regions to foreground and background subregions. However popular texture segmentation techniques such as methods based on graph cuts [2, 7] require computing Markov Random Field (MRF) models from a training set. Moreover they tend to be computationally intensive. These features tend to make such approaches not suitable for tracking. Recently Rother *et. al.* [9] have introduced a technique based on graph cuts which yields impressive interactive segmentation results thanks to improvements in the optimization stage and local post-processing. Their method works on 3-channel images and relies on post-processing to extract details. Here we aim at a method that works on 8-bit images and finds contours given a starting guess and normal directions. Therefore, it can be used in a complementary sense to graph cut methods to refine object contours around estimated positions.

A prototype texture-based tracker is the set forth by Ozyildiz *et. al.* [8] in which a formulation for fusing texture and color is presented in a manner that makes the segmentation reliable while keeping the computational cost low. The texture is modeled by an autobinomial Gibbs Markov Random Field (GMRF) and a 2–D Gaussian distribution is used for modeling the color. However since the MRF model is used to learn the global characteristics of target texture, it can not extract fine contours of the object as is required by 3–D pose tracking algorithms. Instead, this method is used to track the object as a whole in the image. Moreover training MRF's would lead to difficulties in tracking objects with different local textures and patterns.

Updating the prior model is of particular importance in tracking since the initial model of the target changes in the course of tracking due to changes in lighting conditions and object views. In the texture-based tracking scheme proposed in [8] a statistical model for the adaptation over time of the mean and covariance vectors is proposed which uses L previous mean and covariance estimates. This causes rapid accumulation of drift error and gives undesirable tracking results in a long sequence. On the other hand in this work, we exploit the Kullback-Leibler Divergence to update the target model in an adaptive way. The KLD-based update avoids rapid error accumulation thanks to its advantageous flexibility and discriminant characteristics over other distribution distance measure [13].

3. Texture boundary extraction

In this section we first review quickly our original scanline approach to detect texture crossings [10], we then show how to aggregate the information from multiple scanlines to improve the probability distribution of texture boundaries. This is done in two ways. First we use a bundle of neighboring scanlines, scanstripes, in order to better estimate the transition matrices. Second, we link scanstripes in order to aggregate the information on the entire set of scanstripes. The resulting algorithm is still fast and can operate at several frames per second for tracking while being more robust and stable.

3.1. Scanlines

A scanline is a sequence of n pixel intensities $S_1^n = (s_1, ...s_n)$. It is assumed to have been generated by two distinct texture processes each operating on either side of an unknown change point. Thus the observed data is considered to have been produced by the following process: First a changepoint i is selected uniformly at random from the range $[1-n]$. Then the pixels to the left of the changepoint (the sequence S_1^i) are produced by a texture process T_1 and the pixels to the right (S_{i+1}^n) are produced by process T_2. This leads to the probability of texture crossing for point i which can be calculated in closed form for uniform prior for texture distribution [10].

If pixels are drawn from C classes then a 1^{st} order model consists of $C \times (C-1)/2$ parameters, if we assume a symmetric state transition matrix. This implies that using only the observations on a single scanline can be insufficient in order to estimate all these parameters correctly and can cause instable tracking results. In the remainder of this section we consider methods to better estimate these parameters and obtain more reliable probabilities.

3.2. Scanstripes

Scanstripes are a set of parallel scanlines that are scanned simultaneously to count the number of pixel occurrences of a substring S_{i-1}^i in the scanstripe in two perpendicular directions. This improves the estimation of scanline probabilities by basing the estimation on more observations using the neighboring lines around each scanline. In general this results in earlier convergence of the transition matrix.

One might be tempted to use two transition matrices to represent the pixel joint probabilities in two orthogonal directions, one parallel to the scanline, and one normal to it. Although this seem to be the right choice in the case of non-isotropic textures, it has the disadvantage of doubling the number of parameters used to model the texture and can have negligible advantage if the examined textures are rectified according to a fixed reference system using an estimation of their orientation from the pose prior.

To illustrate the effects of scanstripes we have conducted several tests. Fig. 1-(b) shows the individual scanlines tex-

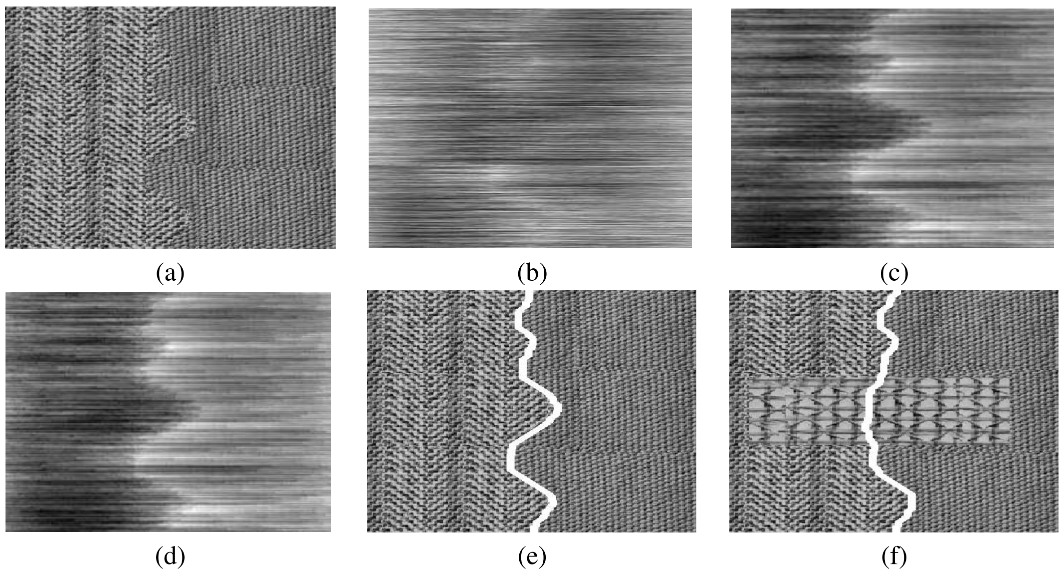


Figure 1. Scanstripe vs. scanline log probabilities for the texture image shown in (a). Individual scanlines and a single transition matrix (b). Parallel and vertical transition matrices on scanstripes made of 5 scanlines (c). One single transition matrix containing both parallel and vertical transitions on scanstripes made of 5 scanlines (d). Finally (e) and (f) show the most likely boundary obtained using an HMM and the Viterbi algorithm on the test image with and without occlusion.

ture crossing log probabilities on the test image shown in Fig. 1-(a) as intensities values. This corresponds to the case that we use only one observation at a time to update the transition matrix as suggested in [10]. Fig. 1-(c) shows the case where we apply a transition matrix corresponding to the direction parallel to the scanstripe as well as another one for the perpendicular direction on a scanstripe made of 5 scanlines. Fig. 1-(d) shows the case where we use the parallel and normal transitions to update one single transition matrix on a scanstripe made of 5 scanlines. The superiority of scanstripes over scanlines can be seen in this example where the textures on both sides are quite similar. The result obtained using scanlines are quite noisy and it is not possible to distinguish a texture boundary from the probability image whereas using scanstripes smoothens out the noise and yields a visible bright boundary (high probability) in the probability image. Moreover, the convergence of the transition matrix happens 3 times faster when using 5 scanlines to update the transition matrix instead of one scanline.

3.3. Linkage of scanstripes

Each scanstripe gives independently the probability of texture change across it. These probabilities can be aggregated to yield a probability distribution for the most probable paths that separates the textures.

Given a geometric (3–D) model of the object, the probability of a hypothesized pose can be calculated from the scanstripe probabilities. Therefore we can use a robust estimator to maximize the pose probability given the scanstripe probabilities. This is explained in detail in section 5.

However, for the cases where the model is not easy to use or for delineation application we present a Hidden Markov Model to link separate scanstripe probabilities. This is the case when the model does not consist of straight or geometrically well defined (such as quadrics) edges. In the remainder of this section we describe the HMM linkage of scanstripes.

We define a HMM with, as observable state, a set of pixel sequences $\mathbb{L} = \{L_i \,|\, i = 1, ..., N\}$ where each sequence L_i is a scanstripe. The HMM is characterized by the visible state L_i which is the sequence of pixel intensities on scanstripe i and the hidden state Z_i which is the location of the real edge along the scanstripe i. The likelihood distribution in each of the states, $P(L_t \,|\, Z_t)$, is given by individual scanstripe probabilities of section 3.2 and the dependency between successive hidden states, $P(Z_{t+1} \,|\, Z_t)$, is modeled by a Gaussian kernel which ensures connectivity and smoothness of the boundary. Finally, the initial state distribution $P(Z_0)$ is assumed to be uniform. We wish to maximize the probability of the hidden states for a given set of sequences, $\mathbb{L}$:

$$\arg\max_{z_1,...,z_N} P(Z_1 = z_1, ..., Z_N = z_N \,|\, L_1, ..., L_N) \quad (1)$$

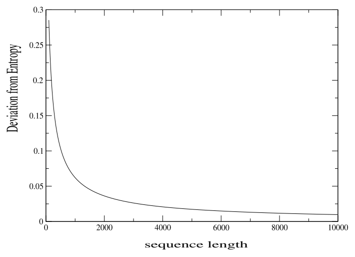


Figure 2. $\hat{H}(n)/n$ or the deviation of the sequence log probability $\ln P(S_1^n)$ from the entropy for $C = 16$.

The posterior distribution given by Eq. 1 can readily be used to obtain texture boundaries using the Viterbi algorithm [5], which provides an efficient way to extract the highest probability path in the distribution that separates two textures. This is illustrated in Fig. 1-(e) where the most probable path obtained thus is superposed on the test image. The marked advantage of such global maximization of the a posteriori distribution of the boundary is its robustness in presence of complex and similar textures or strong occlusion such as shown in Fig. 1-(f).

4. Learning target texture model

It is sometimes helpful to learn the transition matrix of a known target a priori in order to detect its boundary with an arbitrary background. This is particularly useful in tracking or segmentation of complex texture compositions. Unfortunately this is not a trivial issue because the initial values of the transition matrices on both sides of the scanstripes (corresponding to the exterior and interior textures, see [10] for details) would no longer be equal. The final changepoint probability in this case will be biased. If the effects of this bias are too large, they can obscure the true peaks in the scanline probability curves and therefore push the detected boundary towards the inside of the learnt target. Moreover, it is useful to measure the consistency of the learnt texture model and the actual object texture during tracking. This measure can be used to update the model as the object's texture and the scene's ambiance are evolving. In the following we propose a method to detect and correct a bad prior model and adaptively update the texture model.

4.1. Log probability and the entropy of texture

The log probability curves of a scanstripe have an important mathematical interpretation. We show how this curve relates to the entropy of the texture. This is an important issue since it can be exploited to distinguish the prior models which are not in compliance with the current texture and moreover it gives us a distance measure which can be directly used to update them as discussed in the next section. First we derive the relationship between the log probability curves and entropy for the 0^{th} (histogram) and 1^{st} order (transition matrix) texture model, using the estimated probability of texture symbols [10].

0^{th} order model

Imagine that we have C classes in our texture model (bins of intensity histogram for example). The uniform texture distribution prior implies that the probability of a given sequence of n pixels drawn from an unknown texture process T is given by:

$$\begin{aligned} P(S_1^n) &= \tfrac{O_{s_1}}{(1+C-1)} \times \tfrac{O_{s_2}}{(2+C-1)} \times \cdots \times \tfrac{O_{s_n}}{(n+C-1)} \\ &= \frac{\prod_{i=1}^{C} O_{s_i}!}{\frac{(n+C-1)!}{(C-1)!}} \end{aligned} \tag{2}$$

where O_{s_i} is the number of times a class, to which pixel S_i belongs, has appeared in the sequence S_1^i. If n is large enough we have $O_{s_i} = np_i$, with p_i being the probability of class i.

The log probability of this term is therefore:

$$\ln P(S_1^n) = \sum_{i=1}^{C} \ln(np_i)! - \ln(n + C - 1)! + \ln(C - 1)!$$

Using the Stirling's formula, $\ln x! \approx x \ln x - x + 1$, we get:

$$\begin{aligned} \ln P(S_1^n) \;=\; & \textstyle\sum_{i=1}^{C} (np_i \ln(np_i) - np_i + 1) \\ & - (n + C - 1)\ln(n + C - 1) \\ & + n \,+\, (C - 1)\ln(C - 1)\,. \end{aligned}$$

Simplifications give:

$$\begin{aligned} \ln P(S_1^n) \;=\; & -nH - \\ & (n \ln(\tfrac{n+C-1}{n}) + (C-1)\ln(\tfrac{n+C-1}{C-1}) - C) \\ =\; & -nH \,-\, \hat{H}(n)\,. \end{aligned}$$

Thus:

$$H = -(\ln P(S_1^n) + \hat{H}(n))/n \tag{3}$$

with H being the entropy of the texture. $\hat{H}(n)$ determines the amount of drift from the real entropy in the sequence as n changes. The above equation suggests that the entropy can be derived from the probability of the sequence. Fig. 2 shows the deviation of the sequence log probability $\ln P(S_1^n)$ from the entropy, i.e. $\hat{H}(n)/n$ vs. n. As can be seen this deviation approaches zero for large n. While for small sequence lengths it should be taken into consideration.

1^{st} order distribution

The probability of a given sequence drawn from a 1^{st} order texture process T governed by a transition matrix instead of a histogram can be obtained by employing Eq. 8 in [10] and expanding the sequence probability term. This gives:

$$P(S_1^n) = \frac{\prod_{j=1}^{C}\prod_{i=1}^{C}(1/C + O_{ij} - 1)!}{\prod_{j=1}^{C} O_j!}$$

The log probability of this probability term is therefore:

$$\ln P(S_1^n) = \sum_{j=1}^{C}\sum_{i=1}^{C}\ln(1/C + O_{ij} - 1)! - \sum_{j=1}^{C}\ln O_j!$$

where O_{ij} is the number of times symbol j is followed by symbol i in the sequence of n pixels S_1^n drawn from a texture. Unlike the 0^{th} order it is not straight forward to derive a direct formula for the entropy in this case. However, letting $Oij = np_j p_{ij}$ and $1/C + O_{ij} - 1 \approx O_{ij}$ would make appear the entropy term. These assumptions are not accurate for small n. Nevertheless, they allow us to estimate the relationship between the log probability curve and the 1^{st} order entropy of the texture:

$$H = -(\ln P(S_1^n) + \hat{H})/n$$

with

$$\hat{H} = C^2 - C\,. \qquad (4)$$

We can see that unlike the 0^{th} order model, the deviation from the entropy, $\hat{H}$, does not increase with n which indicates the advantage of using a transition matrix instead of a histogram.

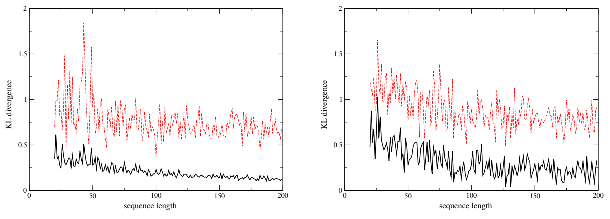
However, as mentioned earlier the approximation $1/C + O_{ij} - 1 \approx O_{ij}$ is not a good one for small n. Instead, our experiments show that we can approximate the entropy using the equation:

$$H \approx -\ln P(S_1^n)/n - 1 \qquad (5)$$

for small n and common choices of number of pixel intensity classes $C < 20$.

4.2. Updating the texture model

As the tracking goes on the appearance of the learnt texture of the target changes due to various lighting conditions. Measuring these changes and updating the learnt model is indispensable for a successful tracking. The predicted position given by the previous tracking stage allows us to calculate the entropy of the current target texture as discussed above from the log probability of a sequence of pixels. A second entropy $\tilde{H}$ can be computed from the sequence by using the prior texture model, T_1: $\tilde{H} = -\ln P(S_1^n|T_1)/n$.



(a) (b)

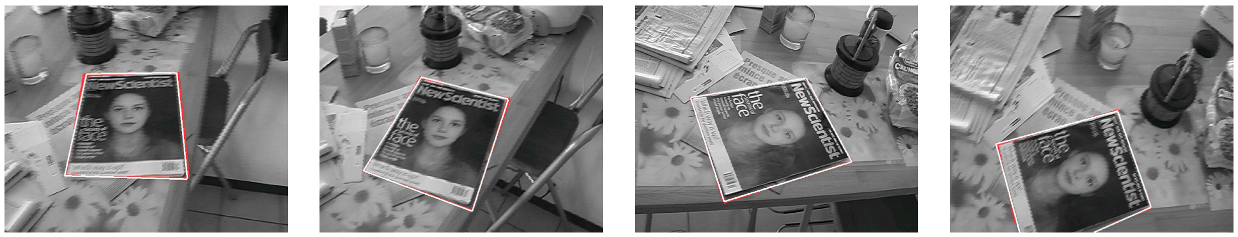
Figure 3. $\tilde{H} - H$ for the true and a lousy learnt model. (a) 0^{th} order model using Eq. 3 to calculate H, (b) 1^{st} order model using Eq. 5 to calculate H. Red dashed lines are KL divergence of the poor model and the thick solid line is the KL divergence of the true model.

The difference $\tilde{H} - H$ is known as Kullback-Leibeler Divergence (KLD) or relative entropy and is always a positive value. The KL Divergence gives a clear measure of how different our calculated model is from the actual texture process. We can use the KL Divergence to mix the current texture and the learnt model in order to update the model. In that case we use a filter with a parameter $\alpha = 1 - \exp(-\frac{\tilde{H}-H-B}{\tau_c})$ which depends on the KLD measure. τ_c is the user-defined time constant that determines the latency of the filter. In practice we use a small constant B to compensate the effects of approximations and it is value is determined manually. We set it to be $B \approx 0.4$. The prior model T_1 is thus updated with $T_1' = \alpha T_2 + (1-\alpha) T_1$ where T_2 is the current texture model. Fig. 3 shows the curves of $\tilde{H} - H$ for the true and a poor learnt model for $20 < n < 200$ derived using Eqs. 3 and 5 for the 0^{th} and 1^{st}order model respectively.

5. Application to tracking and interactive segmentation

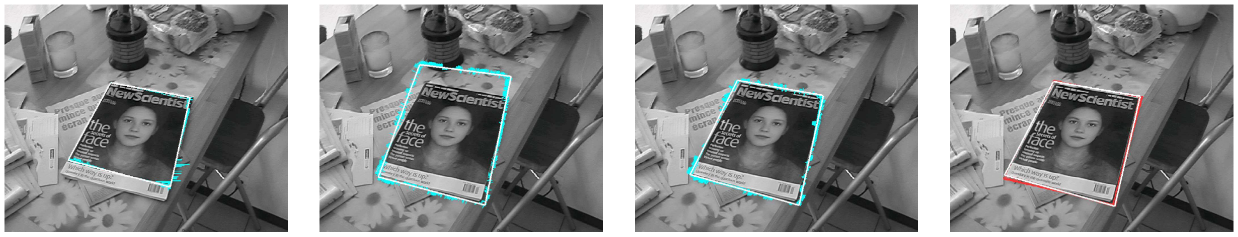
In this section we present our experimental results of applying the proposed method to both tracking and interactive texture segmentation.

Tracking. To show the effect of the KL divergence in updating the prior texture model in case of changes in lighting we tracked a shiny magazine cover. It reflects light differently as the camera moves around it. Moreover, there is a high amount of motion blur due to the motion of the camera and the interlaced images. If we use a constant (including zero, i.e. no update) value for α in the texture update formula $T_1' = \alpha T_2 + (1-\alpha) T_1$ of Section 4, tracking fails after a few seconds (results not shown due to lack of space). On the other hand, adaptive α derived from the KLD of the prior learnt model and the current target texture enables tracking through the whole sequence as shown in Fig. 4. In



frame # 1 frame # 100 frame # 200 frame # 300

Figure 4. Tracking with KLD-based update of the prior model. RANSAC linkage is used to fit straight lines (red) to the scanstripe probabilities directly which are then used to calculate the pose of the model.



(a) (b) (c) (d)

Figure 5. Frame # 90 for tracking using different methods: (a) edge-based tracker, (b) scanlines only, (c) scanstripes only, and (d) RANSAC linkage of scanstripes.

this tracking example, we use the fact that model is made of straight edges and therefore robustly fit straight lines to the scanstripe change probability distribution directly using the RANSAC algorithm [4] instead of HMM linkage.

We have also compared our results with an edge-based tracker [3] in addition to the cases where we use only scanlines and where we use only scanstripes without linkage. We observe that the RANSAC linkage of scanstripes with the KLD-based update outperforms other methods. Moreover the speed of the method remains close to real-time (6-10 fps on a 2.8 GHz Pentium 4). The edge-based tracker is easily fooled by the strong edges in the cover of the magazine as can be seen in Fig. 5-(a) for the 90^{th} frame. The scanline tracker also is lost due to poor estimation of texture parameters for some edges 5-(b). Using scanstripes improves the results 5-(c), while RANSAC linkage of scanstripes which enforces the fact that the searched boundary needs to be a straight lines improves the stability of the tracking results 5-(d)

Delineation. Because it is fast, our method can also serve as an interactive texture segmentation tool. Fig. 6 shows some detected texture boundaries starting from an initial guess –here the circles or straight lines drawn by the users– which provides the scanstripe directions. Scanstripe probabilities are then linked using the HMM described in section 3.3 and the optimal boundary is obtained using the Viterbi algorithm. The zebra result demonstrates that the algorithm works well even on non-isotropic textures. More segmentation results are shown in Fig. 7 where initial guess in obtained by fitting an spline curve on a few user-provided points.

6. Conclusion

We have proposed an approach to contour-based object tracking and texture segmentation that relies on detecting texture boundaries in a given search region and direction.

The reliability of the algorithm lies on the way it aggregates scanline information. Scanstripes are used to estimate texture parameters and assign all pixels in the search region a texture change probability. These probabilities are then further linked using an HMM or RANSAC algorithm depending on the object's geometric model. We have also shown that the probabilities that appear in this computation closely relate to texture entropy and Kullback-Leibler divergence, a property we use to compute and update dynamic texture models.

The HMM posterior can be used to extract the most

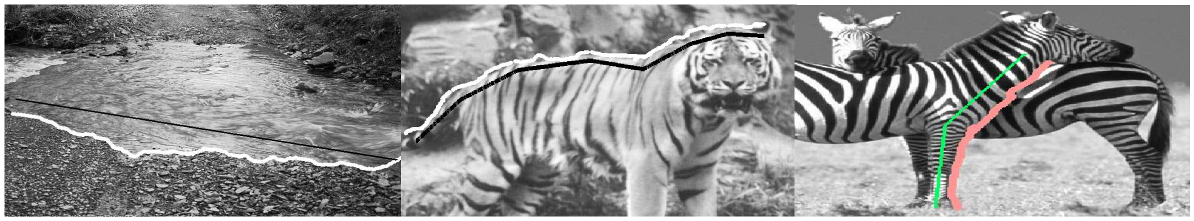


Figure 6. Fast interactive texture segmentation. An initial guess is given by the user (thin circles or straight lines). The HMM is used to link scanstripes and the Viterbi algorithm gives the final texture boundary shown as a thick curve. In the case of the zebras, note that the algorithm finds the boundary between similar textures of different orientations.

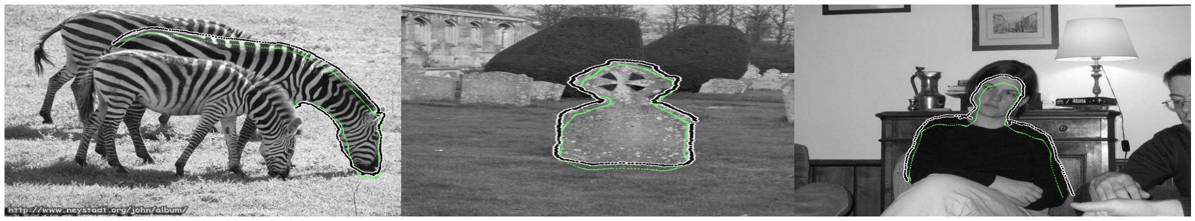


Figure 7. More interactive texture segmentation results with initial curve defined by an spline. The black curve on white band marks the detected outline and the thin gray line is the spline curve.

probable texture boundary efficiently using the Viterbi algorithm. This serves as an interactive texture segmentation tool where the user roughly provides the scanstripe directions and the search region. The time of calculation of the texture boundary is typically less than a second depending on the length of the initial curve. Moreover, this is a generic tool and does not rely on a training database, unlike conventional texture segmentation techniques.

Our implementation for tracking is also fast and can be optimized to process several frames per second. This is partly thanks to the novel formulation that considers uniform prior for all texture distributions and does not assume a prior texture distribution model.

References

[1] A. Blake, C. Rother, M. Brown, P. Perez, and P. Torr. Interactive image segmentation using an adaptive gmmrf model. In *European Conference on Computer Vision*, volume 1, pages 428–441, 2004.

[2] Y. Boykov and M. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in n-d images. In *International Conference on Computer Vision*, pages Vol I:105–112, Vancouver, Canada, July 2001.

[3] T. Drummond and R. Cipolla. Real-time visual tracking of complex structures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):932–946, july 2002.

[4] M. Fischler and R. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications ACM*, 24(6):381–395, 1981.

[5] G. D. Forney. The Viterbi algorithm. In *Proceedings of IEEE*, volume 61, pages 268–278, March 1973.

[6] H. Gupta, A. Roy-Chowdhury, and R. Chellappa. Contour based 3d face modeling from a monocular video. In *British Machine Vision Conference*, pages 367–376, Kingston University, England, 2004.

[7] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, 2004.

[8] E. Ozyildiz, N. Krahnstoever, and R. Sharma. Adaptive texture and color segmentation for tracking moving objects. *Pattern Recognition*, 35(10):2013–2029, 2002.

[9] C. Rother, V. Kolmogorov, and A. Blake. ”GrabCut”: Interactive Foreground Extraction using Iterated Graph Cuts. *ACM Transactions on Graphics*, 23(3):309–314, 2004.

[10] A. Shahrokni, T. Drummond, and P. Fua. Texture Boundary Detection for Real-Time Tracking. In *European Conference on Computer Vision*, pages Vol II: 566–577, Prague, Czech Republic, May 2004.

[11] C. Taylor, J. Malik, and J. Weber. A real-time approach to stereopsis and lane-finding. In *Intelligent Vehicles*, pages 207–212, 1996.

[12] A. Thayananthan, P. Torr, and R. Cipolla. Likelihood models for template matching. In *British Machine Vision Conference*, pages 949–958, 2004.

[13] N. Vasconcelos, P. Ho, and P. Moreno. The kullback-leibler kernel as a framework for discriminant and localized representations for visual recognition. In *European Conference on Computer Vision*, pages Vol II:430–441, Prague, Czech Republic, May 2004.