

Fast Texture-Based Tracking and Delineation Using Texture Entropy *

(EPFL Technical Report ID: IC/2004/89)

Ali Shahrokni[†] Tom Drummond[‡] Pascal Fua[†]

[†] *Computer Vision Laboratory
Swiss Federal Institute of Technology (EPFL)
1015 Lausanne, Switzerland
Email: {ali.shahrokni, Pascal.Fua}@epfl.ch*

[‡] *Department of Engineering, University of Cambridge
Trumpington Street, Cambridge CB2 1PZ, UK
Email: twd20@eng.cam.ac.uk*

November 4, 2004

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

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 [1], we proposed a texture-based approach to finding the edges of highly textured objects, even in the presence of clutter that breaks most gradient-based methods. 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 lose 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 [1] that we refer to as scanstripes. This yields the most likely connected contour(s) of the object.
- Kullback-Leibler Divergence (KLD): The relative entropy between the actual target texture and the prior target texture model. It 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 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 [2, 3] or edge distance transform [4, 5, 6, 7]. 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 use of background subtraction techniques for contour extraction also requires a good estimation of the background image which is not always possible especially if it is not static. Heikkila et. al. [8] for example use local binary pattern (LBP) analysis to extract moving objects. LBP has the advantage of being robust to illumination changes. However, the extracted contours using such techniques are not usually detailed and accurate enough to be used for 3-D tracking.

The main idea of this paper is somewhat similar to interactive texture segmentation problems such as work by Blake et. al. [9] 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 sub-regions. However popular texture segmentation techniques such as methods based on graph cuts [9, 10, 11,

12] 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.

Hanek and Beetz [13] have recently proposed a method for fitting parametric curve models which is based on the EM algorithm. They use local Gaussian distributions to model stripes perpendicular to the curve. In general assuming a Gaussian model for the texture reduces the flexibility of the algorithm. Another example which uses texture for tracking is the work by Ozyildiz et. al. [14] 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 2D 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 [14] 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 [15].

3 Texture Boundary Extraction

In previous work [1] we introduced a real-time texture boundary detection algorithm which uses scanlines to search for texture crossing. below we first review quickly our original scanline approach, 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 $S_1^n = (s_1, \dots, s_n)$. 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 [1].

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.

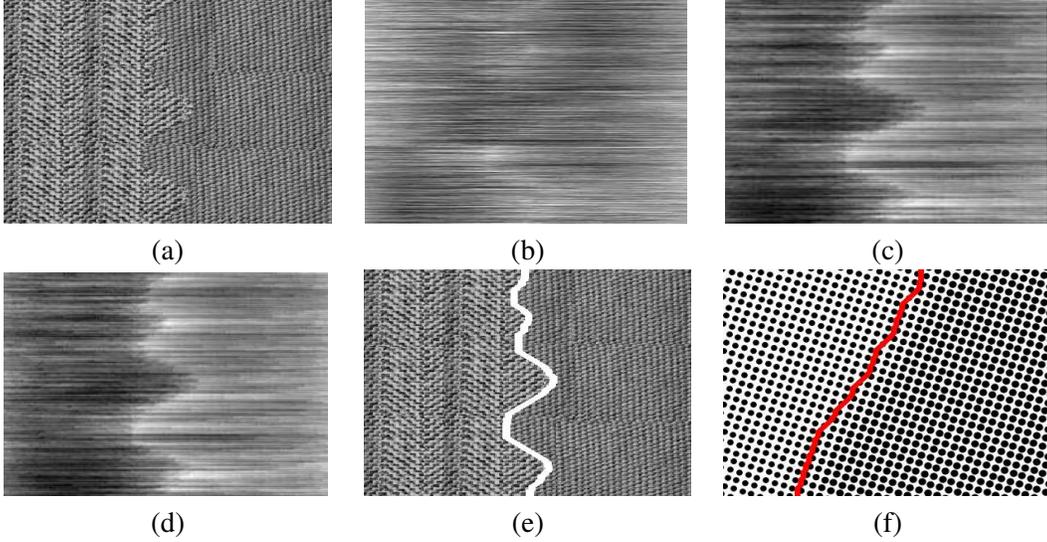


Figure 1: Scanstripe vs. scanline log probabilities for the texture image shown in (a). Individual scanlines and a single transition matrix (b). Parallel and vertical transition matrices on scanstripes made of 5 scanlines (c). One single transition matrix containing both parallel and vertical transitions on scanstripes made of 5 scanlines (d). Finally (e) and (f) show the most likely boundary obtained using an HMM and Viterbi algorithm on complex texture compositions.

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 along 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. Our experiments show that in general it is better to allow the degradation introduced by violation of the non-isotropic assumption rather than increasing the number of degrees of freedom of our model. This is supported by the fact that scan directions are always well oriented wrt the real boundary thanks to the estimated pose in previous frame our the user provided guess.

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

For tracking applications we can generate correspondences between hypotheses in the scanstripe probability distribution domain and the pose of the model directly using RANSAC algorithm whenever such a correspondence is easy to establish. This is referred to as RANSAC linkage of scanstripes and will be explained later in the results 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 the HMM on an image $L_{M \times N}$ where M is the width (length of scanstripes) and N is the height of the image (the total number of scanstripes).

- Each scanstripe L_j corresponds to a state of the model, X_j , i.e. the position of the boundary of the texture. X_j can take M values.
- A likelihood distribution in each of the states, $B = \{b_j(i)\}$ which is given by individual scanstripe probabilities.

$$b_j(i) = p\{L_j \mid X_j = i\}, 1 \leq j \leq N, 1 \leq i \leq M$$

where L_j denotes the j^{th} scanstripe. $b_j(i)$ must satisfy the normal stochastic constraints.

- A set of state transition probabilities $\Lambda = \{a_{kl}\}$.

$$a_{kl} = p\{X_{j+1} = l \mid X_j = k\}, 1 \leq k, l \leq M,$$

where X_j denotes the current state. Transition probabilities should also satisfy the normal stochastic constraints.

- The initial state distribution, $\pi = \{\pi_i\}$ where,

$$\pi_i = p\{X_1 = i\}, 1 \leq i \leq M.$$

Therefore we can use the compact notation $\lambda = (\Lambda, B, \pi)$ to denote the HMM over the scanstripes. In this case we want to find the most likely state sequence for a given set of scanstripes, $L = L_1, L_2, \dots, L_N$ and the model, $\lambda = (\Lambda, B, \pi)$. It can be shown that the posterior distribution for the state probabilities in λ is given by:

$$\begin{aligned} P(X_t \mid L_1, \dots, L_t) &\propto \\ &P(X_t \mid L_t) \sum_i P(X_t \mid X_{t-1} = i) \times \\ &P(X_{t-1} = i \mid L_1, \dots, L_{t-1}) \end{aligned} \quad (1)$$

Eq. 1 is a recursive formula that allows aggregation of the information of each scanstripe in order to calculate the most likely state sequence which corresponds to the most likely texture boundary in the image. The likelihood distribution B gives the term $P(X_t \mid L_t)$ while $P(X_t \mid X_{t-1})$ is determined by state transition probabilities Λ . We use a Gaussian kernel for state transition and the sum in Eq. 1 is calculated by recursive convolution of the posterior probabilities with the state transition probabilities. This is discussed in the following subsection.

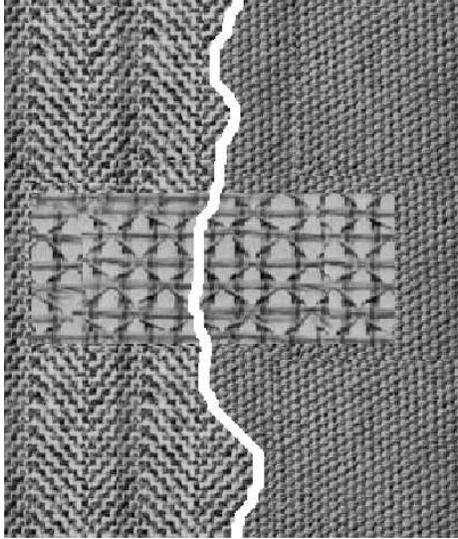


Figure 2: Texture boundary detection on occluded patterns. The HMM linkage of the scanstripe change probabilities gives the optimum path separating the two main textures in the image, excluding the occluding pattern.

3.3.1 Detection of the most likely path

The posterior distribution given by Eq. 1 can readily be used to obtain texture boundaries using the Viterbi algorithm [16]. This algorithm provides an efficient way to extract the highest probability path in the distribution that separates two textures. This is illustrated in Fig. 1-(e) and (f) where the most probable path obtained thus is superposed on the test images. Such a path is the global maximum of the a posteriori solution of the Hidden Markov presentation of the boundary and yields good results even in case of complex and similar textures as shown in the test images or in presence of strong occlusion such as shown in Fig. 2.

As an alternative to Viterbi algorithm, we can draw sample paths directly from posteriori distribution given by Eq. 1. Path sampling is done by first sampling image positions corresponding to the last scanstripe, T according to the distribution $P(X_T | O_T)$. Then we iteratively draw samples from $P(X_t | O_t)$ weighted by the transition probabilities of $a_{t,t+1} = p(X_{t+1} | X_t)$.

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 especially useful in tracking or segmentation of complex texture composition. Unfortunately this is not a trivial issue to deal with. This is because the initial values of the transition matrices on both sides of the scanstripes (corresponding to the exterior and interior textures, see [1] for details) are no longer equal if we use a priori learnt target texture model on one side. 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. This is illustrated in Fig. 3. The average log probability of texture changepoint over several hundred scanstripes running through two different textures are shown for the case where the target model is not learnt a priori (a), we see that this curve is almost symmetric around a peak at the true texture change position, but small ripples can be seen all over this curve. The effect of using a good learnt model is to reduce these ripples and therefore avoiding errors due to changes in the

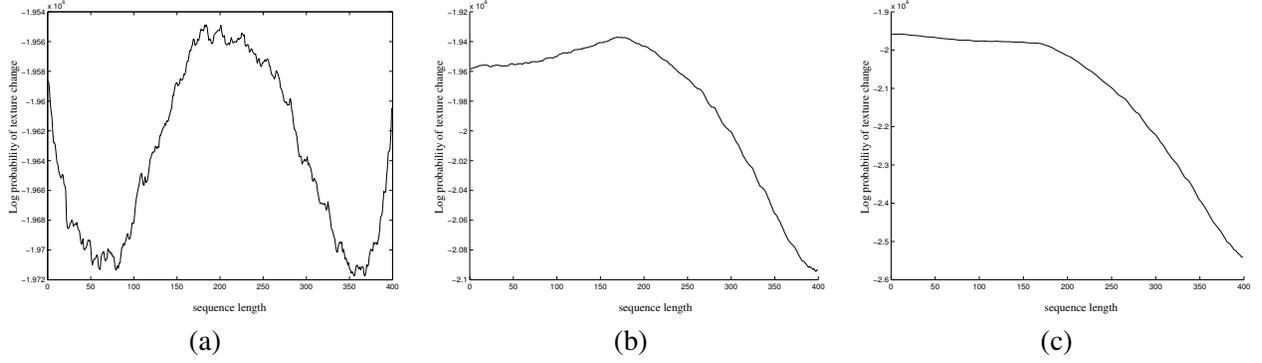


Figure 3: (a) Texture change log probability for the case of using no prior model. Texture change log probability with (b) a good learnt model and (b) a bad learnt model

global peak position. This is shown in (b). On the other hand an incorrect model can lead to a too biased curve which pushes the boundary towards the inside of the learnt target (c). In the following we propose a method to detect and correct a bad prior model.

4.1 Log probability curves and 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. But first we derive the relationship between the log probability curves and entropy for the 0^{th} (histogram) and 1^{st} order (transition matrix) texture distribution, using the equations for estimated probability of texture symbols [1].

0^{th} Order distribution

Imagine that we have C classes in our texture model (bins of intensity histogram for example). It can be shown that the probability of a given sequence of n pixels drawn from an unknown texture process T is given by:

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

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:

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

Simplifications give:

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

Thus:

$$H = -(\ln P(S_1^n) + \hat{H}(n))/n \quad (2)$$

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. 4 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.

1st Order distribution

The probability of a given sequence drawn from a 1st order texture process T governed by a transition matrix instead of a histogram can be obtained by employing Eq. 8 in [1] 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 0th order it's not straight forward to derive a direct formula for the entropy in this case. Letting $O_{ij} = np_j p_{ij}$ 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 1st order entropy of the texture:

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

with

$$\hat{H} = C^2 - C.$$

We can see that unlike the 0th 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.

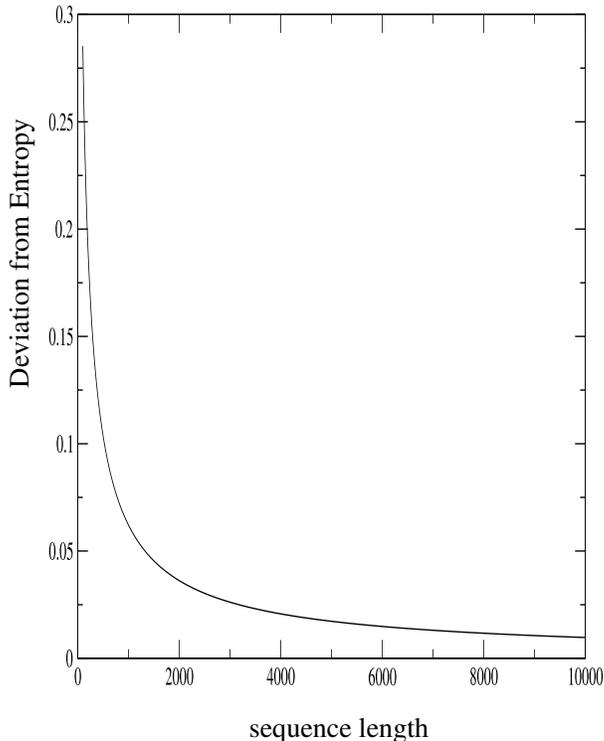


Figure 4: Deviation of the sequence log probability $\ln P(S_1^n)$ from the entropy, $\hat{H}(n)/n$ for $C = 16$.

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:

$$H \approx -\ln P(S_1^n)/n - 1 \quad (3)$$

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

4.2 Updating the Prior Texture Model during tracking

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 : $\tilde{H} = -\ln P(S_1^n|T_1)/n$. The difference $\tilde{H} - H$ is known as Kullback-Leibler 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

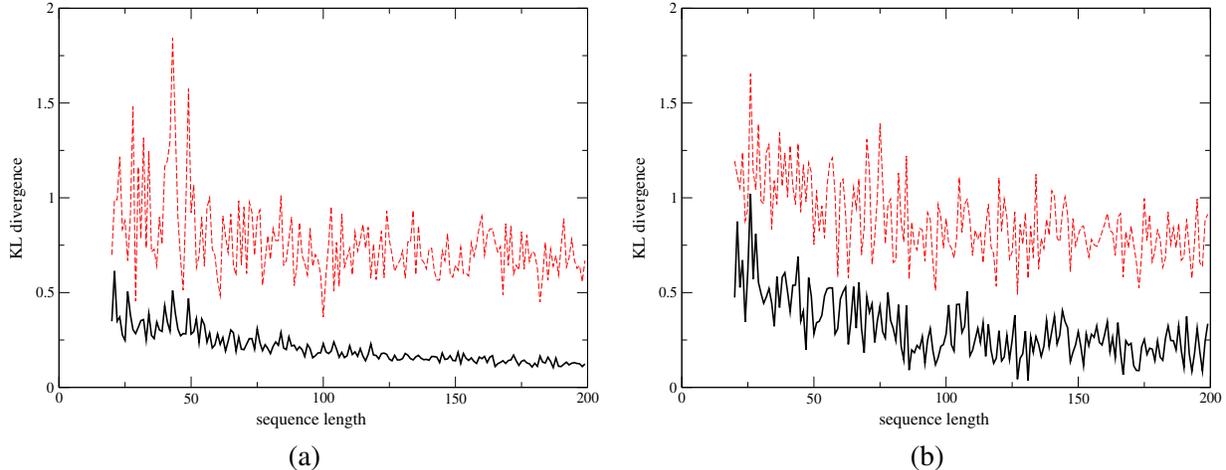


Figure 5: $\tilde{H} - H$ for accurate and inaccurate learnt models. (a) 0^{th} order model using Eq. 2 to calculate H , (b) 1^{st} order model using Eq. 3 to calculate H . Red dashed lines are KL Divergence of the inaccurate model and the thick solid line is the KL Divergence of an accurate model.

of approximations and its 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. 5 shows the curves of $\tilde{H} - H$ for accurate and inaccurate learnt models for $20 < n < 200$ derived using Eqs. 2 and 3 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’s 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 (see Fig. 6). 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, see Fig. 7. In this tracking example, we use the fact that model is made of straight edges and therefore fit straight lines to the scanstripe change probability distribution directly using the RANSAC algorithm instead of HMM linkage.

We have also compared our results with an edge-based tracker [2] 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. The edge-based tracker is easily fooled by the strong edges in the cover of the magazine as can be seen in Fig. 8-(a) for the 90^{th} frame. The scanline tracker also is lost due to poor estimation of texture parameters for some edges 8-(b). Using scanstripes improves the results 8-(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 8-(d)

Delineation. Because it is fast, our method can also serve as an interactive texture segmentation tool. Fig. 9 shows some detected texture boundaries starting from an initial guess—here the circles or straight lines drawn



frame # 73

Figure 6: Tracking without dynamic texture model update. Tracking is lost well before a 100 frames due to changes in target appearance.



frame # 1

frame # 100

frame # 200

frame # 300

Figure 7: 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.

by the users—which provides the scanstripe directions. Scanstripe probabilities are then linked using the HMM described in this paper and the optimum boundary is obtained using the Viterbi algorithm. The zebra result demonstrates that the algorithm works well even on non-isotropic textures. The trees image is an interesting case where there is no obvious and continuous texture boundary but only the imaginary line of tree plantation which has been correctly extracted by our algorithm.



(a)

(b)

(c)

(d)

Figure 8: Frame # 90 for tracking using different methods: (a) edge-based tracker, (b) scanlines only, (c) scanstripes only, and (d) RANSAC linkage of scanstripes.



Figure 9: 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.

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 the 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. 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 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 how long the initial curve is. 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 optimised 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. Shahrokni, T. Drummond, and P. Fua, “Texture Boundary Detection for Real-Time Tracking,” in *European Conference on Computer Vision*, Prague, Czech Republic, May 2004, pp. Vol II: 566–577.

- [2] T. Drummond and R. Cipolla, "Real-time visual tracking of complex structures," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 932–946, July 2002.
- [3] C.J. Taylor, Jitendra Malik, and Joseph Weber, "A real-time approach to stereopsis and lane-finding," in *Intelligent Vehicles*, 1996, pp. 207–212.
- [4] A. Agarwal and B. Triggs, "3-D Human Pose from Silhouettes by Relevance Vector Regression," in *Conference on Computer Vision and Pattern Recognition*, 2004.
- [5] H. Gupta, A. Roy-Chowdhury, and R. Chellappa, "Contour based 3d face modeling from a monocular video," in *British Machine Vision Conference*, Kingston University, England, 2004, pp. 367–376.
- [6] A. Thayananthan, P.H.S. Torr, and R. Cipolla, "Likelihood models for template matching," in *British Machine Vision Conference*, 2004, pp. 949–958.
- [7] Vassilis Athitsos and Stan Sclaroff, "Estimating 3d hand pose from a cluttered image," in *Conference on Computer Vision and Pattern Recognition*, Madison, WI, June 2003, pp. 432–439.
- [8] M. Heikkila, M. Pietikainen, and J. Heikkila, "A texture-based method for detecting moving objects," in *British Machine Vision Conference*, Kingston University, England, 2004, pp. 187–196.
- [9] 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*, 2004, vol. 1, pp. 428–441.
- [10] 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*, Vancouver, Canada, July 2001, pp. Vol I:105–112.
- [11] V. Kolmogorov and R. Zabih, "What energy functions can be minimized via graph cuts?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 2, pp. 147–159, 2004.
- [12] M. Rivera and J. C. Gee, "Two-level mrf models for image restoration and segmentation," in *British Machine Vision Conference*, Kingston University, England, 2004, pp. 809–818.
- [13] R. Hanek and M. Beetz, "The contracting curve density algorithm: Fitting parametric curve models to images using local self-adapting separation criteria," *Int. J. Comput. Vision*, vol. 59, no. 3, pp. 233–258, 2004.
- [14] E. Ozyildiz, N. Krahnstoeber, and R. Sharma, "Adaptive texture and color segmentation for tracking moving objects," *Pattern Recognition*, vol. 35, no. 10, pp. 2013–2029, 2002.
- [15] 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*, Prague, Czech Republic, May 2004, pp. Vol II:430–441.
- [16] G. David Forney, "The Viterbi algorithm," in *Proceedings of IEEE*, March 1973, vol. 61, pp. 268–278.