Texture Boundary Detection for Real-Time Tracking

Ali Shahrokni[1], Tom Drummond[2], and Pascal Fua[1*]

[1] Computer Vision Laboratory
EPFL
CH-1015 Lausanne, Switzerland
`ali.shahrokni@epfl.ch`
`http://cvlab.epfl.ch`
[2] Department of Engineering, University of Cambridge
Trumpington Street, Cambridge CB2 1PZ
`http://www-svr.eng.cam.ac.uk/~twd20/`

Abstract We propose an approach to texture boundary detection that only requires a line-search in the direction normal to the edge. It is therefore very fast and can be incorporated into a real-time 3–D pose estimation algorithm that retains the speed of those that rely solely on gradient properties along object contours but does not fail in the presence of highly textured object and clutter.
This is achieved by correctly integrating probabilities over the space of statistical texture models. We will show that this rigorous and formal statistical treatment results in good performance under demanding circumstances

1   Introduction

Edge-based methods have proved very effective for fast 3–D model-driven pose estimation. Unfortunately, such methods often fail in the presence of highly textured objects and clutter, which produce too many irrelevant edges. In such situations, it would be advantageous to detect texture boundaries instead. However, because texture segmentation techniques require computing statistics over image patches, they tend to be computationally intensive and have therefore not been felt to be suitable for such purposes.

To dispel this notion, we propose a texture-based approach to finding the projected contours of 3–D objects while retaining the speed of standard edge-based techniques. We demonstrate its effectiveness for real-time tracking while using only a small fraction of the computational power a modern PC. Our technique is inspired by earlier work [1] on edge-based tracking that starts from the estimated projection of a 3–D object model and performs a line search in the direction perpendicular to the projected edges to find the most probable boundary location. Here, we replace conventional gradient-based edge detection by a method which can directly compute the most probable location for a texture boundary on the search line. To be versatile, the algorithm is designed to work even when neither of the textures on either side of the boundary are known *a priori*. Our technique is inspired by the use of Markov processes for texture description and segmentation. However, our requirements differ from those of classical texture segmentation methods in the sense that we wish to find the optimal pose parameters rather than arbitrary region boundaries.

This is challenging because speed requirements compel us to restrict ourselves to computing statistics along a line, and therefore a fairly limited number of pixels. We achieve this by correctly integrating probabilities over the space of statistical texture models. We will show that this rigorous and formal statistical treatment has allowed us to reach our goal under these demanding circumstances, which we regard as the main contribution of this paper. It is also worth noting that our implementation results in a real-time 3–D tracker that uses only a fraction of the computational resources of a modern PC, thus opening the possibility to simultaneously track many objects on ordinary hardware.

In the remainder of this paper, we first discuss related works. We then introduce our approach to detecting texture boundaries along a line. Finally, we integrate it into a contour-based 3–D tracker and demonstrate its effectiveness on real video-sequences.

2   Related Work and Background

We first briefly review the state-of-the art in real-time pose estimation and then discuss existing techniques for texture segmentation and show why they are not directly applicable to the kind of real-time processing we are contemplating in this paper.

2.1   Real-Time 3–D Tracking

Robust real-time tracking remains an open problem, even though offline camera registration from an image sequence [2,3,4] has progressed to the point where commercial solutions have become available. By matching natural features such as interest points between images these algorithms achieve high accuracy even without *a priori* knowledge. For example, in [3], the authors consider the image sequence hierarchically to derive robust correspondences and to distribute error over whole of it. Speed not being a critical issue, these algorithms take advantage of time-consuming but effective techniques such as bundle adjustment.

Model-based approaches, such as those proposed in [5,6], attempt to avoid drift by looking for a 3–D pose that correctly re-projects the features of a given 3–D model into the 2–D image. These features can be edges, line segments, or points. The best fit is found through least-squares minimisation of an error function, which may lead to spurious results when the procedure becomes trapped in erroneous local minima.

In earlier work [1], we developed such an approach that starts from the estimated projection of a 3–D object model and performs a line search in the direction perpendicular to the projected edges to find the most probable boundary location. Pose parameters are then taken to be those that minimise the distance of the model's projection to those estimated locations. This process is done in terms of the SE(3) group and its Lie algebra. This formulation is a natural choice since it exactly represents the space of poses that form the output of a rigid body tracking system. Thus it provides a canonical method for linearizing the relationship between image motion and pose parameters. The corresponding implementation works well and is very fast when the target object stands out clearly against the background. However it tends to fail for textured objects whose

boundaries are hard to detect unambiguously using conventional gradient-based techniques, as the chair shown in Fig. 1. We will argue that the technique proposed here remedies this failing using only small fraction of the computational power of a modern PC. This is in contrast to other recent model-based techniques [7] that can also deal with textured objects but require the full power of the same PC to achieve real-time performance.
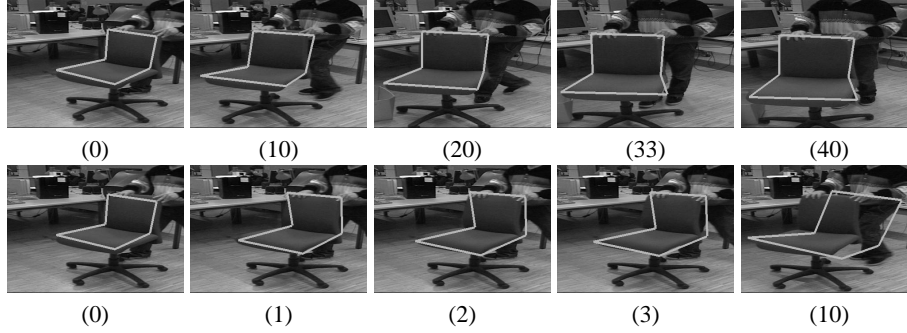

Figure1. Tracking a chair using a primitive model made of two perpendicular planes. Top row: Using the texture-based method proposed in this paper, the chair is properly tracked throughout the whole sequence. Bottom row: Using a gradient-based method to detect contours, the tracker starts being imprecise after the 3rd frame and fails completely thereafter.

2.2 Texture-Based Segmentation

Many types of texture descriptors have been developed to characterize textures. Gabor filters [8,9] have proved as an excellent descriptive tool for a wide range of textures [10] but are computationally exhaustive for real time applications. Multi-resolution algorithms offer some speed-up without loss of accuracy [11,12,10,13,14]. However, while this approach is effective for global segmentation of images containing complex textures, it is not the optimal solution for industrial applications in which the search space is limited. In the context of object tracking, the structure of the object to be tracked is already known, and there is a strong prior on its whereabouts. Hence, where real-time performance is required, classical texture segmentation is not the best choice, but a fast technique which localizes the target boundaries is desired.

However, the approach presented in this paper does borrow ideas from the texture segmentation literature. Hidden Markov random fields appear naturally in problems such as image segmentation, where an unknown class assignment has to be estimated from the observations at each pixel. Statistical modeling using Hidden Markov Models are very rich in mathematical structure and hence can form the theoretical basis for use in a wide range of applications. A comprehensive discussion about Markov Random Fields (MRF) and Hidden Markov Models (HMM) is given in [15,16]. [17] uses a Gibbs Markov Random Field to model texture which is fused with a 2D Gaussian distribution model for color segmentation for real time tracking. In contrast to HMM, MRF methods

are non causal and therefore not desirable for line search methods for which a statistical model would be constructed during the search.

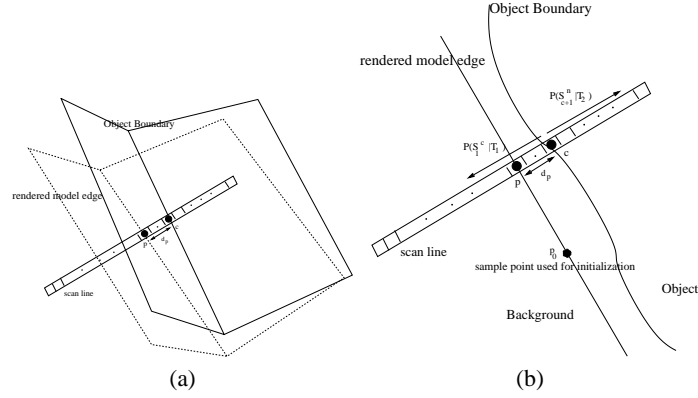## 3   Locating Texture Boundaries Using 1–D line Search

Figure2. Contour-based 3-D tracking. (a) Search for a real contour in the direction normal to projected edge.(b) Scanning a line through model sample point p for which a statistical model associated to model point p_0 is retrieved and used to find the texture crossing point c. Notice that the statistical model of p_0 is made offline and conforms with the real object statistics in the neighbourhood.

As discussed in Section 2.1, we use as the basis for this work a tracker we developed in earlier work. Successive pose parameters are estimated by minimizing the observed distances from the rendered model to the corresponding texture crossing point, that is the point where the underlying statistics change, in the direction normal to the projected edge. This search is illustrated by Fig. 2. In this section, we formalize the criteria we use in this search and derive the algorithms we use to evaluate them.

A texture is modeled as a statistical process which generates a sequence of pixels. The problem is then cast as follows:A sequence of n pixel intensities, $S_1^n = (s_1, ... s_n)$, is assumed to have been generated by two distinct texture processes each operating on either side of an unknown change point, as shown in Fig. 2(b). Thus the observed data is considered to have been produced by the following process: First a changepoint c is selected uniformly at random from the range [1-n]. Then the pixels to the left of the changepoint (the sequence S_1^c) are produced by a texture process T_1 and the pixels to the right (S_{c+1}^n) are produced by process T_2. The task is then to recover c from S_1^n. If both T_1 and T_2 are known then this corresponds to finding the c that maximises:

$$P(\text{changepoint at } c|S_1^n, T_1, T_2) = KP(S_1^c|T_1)P(S_{c+1}^n|T_2) \ . \tag{1}$$

where K is a normalisation constant. If one of the textures, for example, T_1 is unknown, then the term $P(S_1^c|T_1)$ must be replaced by the integral over all possible texture pro-

cesses:

$$P(S_1^c) = \int P(S_1^c|T)P(T) \ \mathrm{d}T \ . \tag{2}$$

While it may be tempting to approximate this by considering only the most probable T to have generated S_1^c, this yields a poor approximation for small data sets, such as are exhibited in this problem. A key contribution of this paper is that we show how the integral can be solved in closed form for reasonable choices of the prior $P(T)$ (e.g. uniform).

In this work we consider two kinds of texture processes: first, one in which the pixel intensities are independently drawn from a probability distribution and second, one in which they are generated by a 1st order Markov process, which means that the probability of selecting a given pixel intensity depends (only) on the intensity of the preceding pixel. We refer to these two processes as 0 and 1st order models.

3.1   Solving for the 0th order model

The 0th order model states that the pixel intensities are drawn independently from a probability distribution over I intensities ($T = \{p_i\}; i = 1..I$). If such a texture is known *a priori* then $P(S_1^c|T) = \prod_i p_{s_i}$. If the texture is unknown then:

$$P(S_1^c) = \int P(S_1^c|T)P(T) \ \mathrm{d}T = \int P(s_c|T)P(S_1^{c-1}|T)P(T) \ \mathrm{d}T \tag{3}$$

$$= P(S_1^{c-1}) \int p_{s_c} P(T|S_1^{c-1}) \ \mathrm{d}T \tag{4}$$

The integral in (4) is $E(p_{s_c}|S_1^{c-1})$: i.e. the expected value of the probability p_{s_c} in the texture given the observed sequence S_1^{c-1}. If we assume a uniform prior for T over the I-1 simplex of probability distributions, then this integral becomes:

$$E(p_{s_c}|S_1^{c-1}) = \frac{\int_0^1 \int_0^{1-p_1} \cdots \int_0^{1-\sum_{i=1}^{I-2} p_i} \ p_{s_c} \prod_{j=1}^I p_j^{o_j} \ \mathrm{d}p_{I-1} \cdots \mathrm{d}p_2 \mathrm{d}p_1}{\int_0^1 \int_0^{1-p_1} \cdots \int_0^{1-\sum_{i=1}^{I-2} p_i} \prod_{j=1}^I p_j^{o_j} \ \mathrm{d}p_{I-1} \cdots \mathrm{d}p_2 \mathrm{d}p_1} \tag{5}$$

where there are o_j occurrences of symbol j in the sequence S_1^{c-1}. Note that both of these integrals have the same form, since the additional p_{s_c} in the numerator can be absorbed into the product by adding one to o_{s_c}. Substituting $p_I = 1 - \sum_{i=1}^{I-1} p_i$ and repeatedly integrating by parts yields:

$$E(p_{s_c}|S_1^{c-1}) = \frac{o_{s_c}+1}{c+I-1} \tag{6}$$

This result states that if an unknown probability distribution is selected uniformly at random and a set of samples are drawn from this distribution, then the expected value of the distribution is the distribution obtained by adding one to the number of instances of each value observed in the sample set.

For example, if a coin is selected with a probability of flipping heads randomly drawn from the uniform distribution over [0,1], and it is flipped 8 times, giving 3 heads and 5 tails, then the probability that the next flip will be heads is (3+1)/(8+2) = 0.4.

This result can be applied recursively to the whole sequence to give Algorithm 1.

Algorithm 1 Rapid 0th order computation of $\int P(S_1^c|T)P(T) \; dT$

---

sequence_probability (S[], c)
  dim Observations[NUM_CLASSES]
  // seed Observations[] with 1 sample per bin
  for i=1..NUM_CLASSES **do**
    Observations[i]=1
  end for
  Probability=1
  for i=1..c **do**
    Probability = Probability * Observations[S[i]] / $\sum$ Observations[]
    Observations[S[i]] = Observations[S[i]]+1
  end for
  return Probability

3.2   Solving for the 1st order model

This idea can be immediately extended to a 1st order Markov process in which the intensities are drawn from a distribution which depends on the intensity of the preceding pixel ($T = \{p_{i|j}\}; i, j = 1..I$, where $p_{i|j}$ is the probability of observing intensity i given that the previous pixel had intensity j). These $p_{i|j}$ can be considered as a transition matrix (row i, column j). Again, the probability of a sequence given a known texture is easy to compute:

$$P(S_1^c|T) = P(s_1|T) \prod_{i=2}^{c} p_{s_i|s_{i-1}} \text{ where } P(s_1|T) \text{ is using the 0th order model.} \quad (7)$$

For a first order Markov process, the 0th order statistics of the samples must be an eigenvector of $p_{i|j}$ with eigenvalue 1. Unfortunately, this means that a uniform prior for T over $p_{i|j}$ is inconsistent with the uniform prior used in the 0th order case. To re-establish the consistency, it is necessary to choose a 1st order prior such that the expected value of a column of the transition matrix $p_{i|j}$ is obtained by adding $1/I$ rather than 1 to the number of observations in that column of the co-occurrence matrix before normalising the column to sum to 1. This means that the transition matrix is

$$E(p_{i|j}|S_1^c) = \frac{C_{ij} + 1/I}{1 + \sum_i C_{ij}} = \frac{C_{ij} + 1/I}{1 + o_j} \quad , \quad (8)$$

where C_{ij} is the number of times that intensity i follows intensity j in the sequence S_1^c. And hence the expected 0th order distribution (which is the vector $\frac{(o_j+1)}{(c+I)}$) has the desired properties since

$$\sum_j E(p_{i|j}|S_1^c)\frac{(o_j + 1)}{(c + I)} = \frac{\sum_j C_{ij} + 1/I}{c + I} = \frac{o_i + 1}{c + I} \quad . \quad (9)$$

This modification is equivalent to imposing a prior over $p_{i|j}$ that favours structure in the Markov process and is proportional to $\prod_{ij} p_{i|j}^{(1/I-1)}$). This gives Algorithm 2.

Algorithm 2 Rapid 1st order computation of $\int P(S_1^c|T)P(T)\ \mathrm{d}T$

---

sequence_probability (S[], c)
  dim CoOccurrence[NUM_CLASSES][NUM_CLASSES]
  // seed CoOccurrence[][] with 1/NUM_CLASSES samples per bin
  for r=1..NUM_CLASSES **do**
    for c=1..NUM_CLASSES **do**
      CoOccurrence[r][c]=1/NUM_CLASSES
    end for
  end for
  Probability=1/NUM_CLASSES   *// probability of the first symbol*
  for i=2..c **do**
    Probability = Probability * CoOccurrence[S[i]][S[i-1]] / $\sum$ CoOccurrence[][S[i-1]]
    CoOccurrence[S[i]][S[i-1]] = CoOccurrence[S[i]][S[i-1]]+1
  end for
  return Probability

3.3 Examples applied to a scanline through Brodatz textures

We illustrate these ideas by considering the problem of locating the boundary between two Brodatz textures.
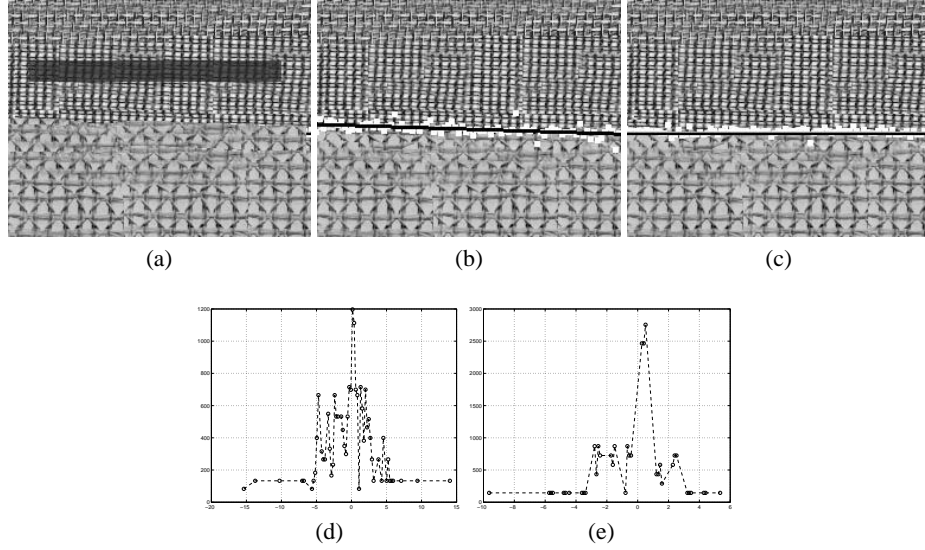

(a)   (b)   (c)

(d)   (e)

Figure3. Brodatz textures results. (a) texture patch used to learn the target texture model(the dark stripe). This model is used to detect the boundary of the target texture with another texture. (b) and (c) detected boundary using 0th and 1st order model respectively. White dots are the detected change point and the black line is the fitted texture boundary. (d)distances from edge in 0th order model (e)distances from edge in 1st order model
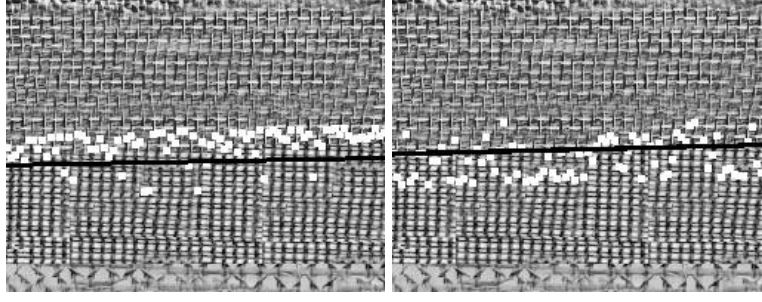
Figure4. In this case, the 0th order model (left) yields a result that is less precise than the 1st order one. As before, white dots are the detected change point and the black line is the fitted texture boundary.

Fig. 3 shows detection results in a case where the texture is different at the top and bottom of the image. In Fig. 3 (a) the region that is used to generate model statistics is marked by a dark stripe. In (b) the boundary between the upper and lower textures is correctly found using 0th order Markov model for both sides. (c) shows the boundary found using 1st order Markov model for both sides. While the model for the lower points is built in an autoregressive manner, the model of the upper points is the one created during the initialization phase (a). White dots show the exact detected point on the boundary along vertical scanlines, to which we robustly fit a line shown in black. The distribution of the distances from white points to the black line are shown in (d) and (e) for the 0th order and 1st order respectively. As can be noticed the distribution peak is less sharp for the 0th order model than the 1st order. This can hamper the detection of boundaries when the distributions are similar as shown in Fig. 4.

An example of texture segmentation for a piece of texture for which we have a geometric model is shown in Fig. 5. This is a especially difficult situation due to the neighbouring texture mixture. Nevertheless, both 0th and 1st order models detect the boundary of the target object (black lines) accurately by robust fitting of the polygonal model to the detected changepoints (white dots). However it was observed that 0th order model is more sensitive to initial conditions, which can be explained by the above statement that the 0th order observations are less accurate.

Our final test on textures involves the case where there is no a priori model for the texture on either side. Some results of application of our method are shown in Fig. 6. In the Fig. 6(a) the boundary is accurately detected. On the other hand for more complicated test of Fig. 6(b) the results are less accurate due to high outlier ratio.

4   Real-Time Texture Boundary Detection

We now turn to the actual real-time implementation of our texture-boundary detection method using the 0th an 1st order algorithms of Section 3.

Assuming a 1st order Markov model for the texture on both sides of the points along the search line, we wish to find the point for which the exterior and interior statistics best match their estimated model. These model can be built online or learned a priori. In the latter case, we build up and register a local 0th and 1st order probability distribution
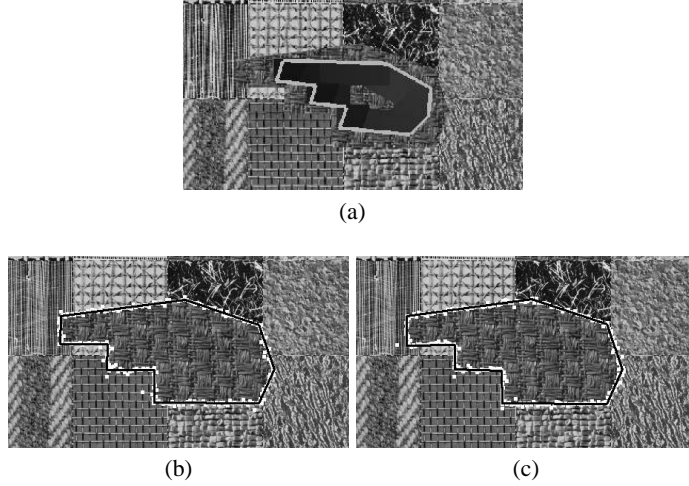
(a)

(b)   (c)

Figure5. Segmentation of a polygonal patch. (a) Initialization: texture patch used to learn the target texture model(the dark stripe). This model is used to detect the boundary of the target texture with another texture. (b) and (c) detected boundary using 0 and 1st Markov model respectively. 0th order model is more sensitive to the initial conditions. As before, white dots are the detected change point and the black line is the fitted texture boundary.
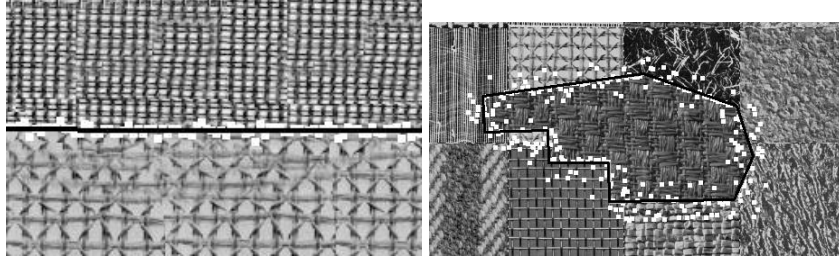

Figure6. Texture boundary detection with no a priori model assumption. In the case of polygonal texture, the white dots (detected changepoints are more scattered but robust fitting yields an accurate result for the boundary (black lines).

(intensity histograms and pixel intensity co-occurrence matrices) for a set of model sample points M_0 during the initialization where the 0 subscript indicates time. Our system uses graphical rendering techniques to dynamically determine the visible edges and recognize the exterior ones. Our representation of the target model provides us with inward pointing normal vectors for its exterior edges. These normal vectors are used to read and store stripes of pixel values during initialization. The stripes need not be very large. These stripes are used to calculate local intensity histograms and pixel intensity co-occurrence matrices for each sample point in M_0. Having local information allows us to construct a more realistic model of texture in the cases where the target object contains patches of different texture.

During tracking, at time i, for each sample point p in the set of sample M_i, we find the closest point in initialization sample set, M_0, and retrieve its associated histogram and co-occurrence matrix. Then, as depicted by Fig. 2(b), for each pixel c along the

scanline associated to rendered point p, two probabilities $P(S_{c+1}^n|T_2)$ and $P(S_1^c|T_1)$ are calculated using algorithm 2 of Section 3, as the line is being scanned inwards in the direction of the edge normal. $P(S_{c+1}^n|T_2)$ is the probability that the successive pixels are part of the learned texture for the target object and $P(S_1^c|T_1)$ is the probability that the preceding pixels on the scanline are described by the the autoregressive model that is built as we move along the scan line and compute the integral of Eq. (2). This process is depicted by Fig. 2(b). This autoregeressive model is initially uniform and is updated for each new visited pixel and includes both the zeroth order distribution and a co-occurrence probability distribution matrix. Having calculated $P(S_{c+1}^n|T_2)$ and $P(S_1^c|T_1)$ for all points c on the scanline, the point c for which the total probability as given by Eq. (1) is maximum, is said to be the texture boundary and the distance d_p between changepoint c and sample point p is passed to the minimiser to compute the pose parameters which minimises the total sum of distances.

5   Experimental results

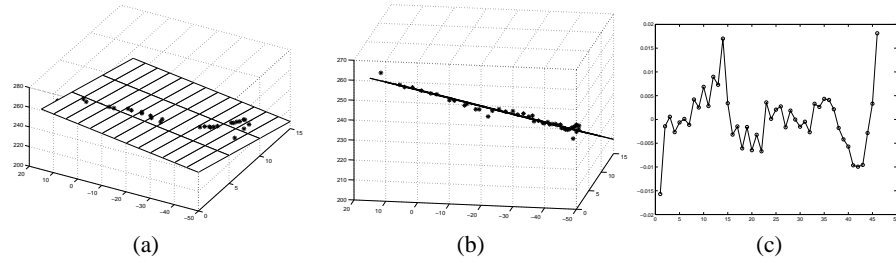| (a) | (b) | (c) |

Figure7. Plotting the motion of the center of gravity of the chair of Fig. 1. (a,b) Top and side view of the plane fitted to the recovered positions of the center of gravity. (c) Deviations from the plane, which are very small (all measurements are in mm).

In the case of the chair of Fig. 1, everything else being equal, detecting the boundaries using the texture measure we propose appears to be much more effective that using gradients. To quantify this, in Fig. 7, we plot the motion of the center of gravity of the model recovered using the texture-based method. Since the chair remains on the ground, its true motion is of course planar but our tracker does not know this and has six degrees of freedom, three rotations and three translations. The fact that the recovered motion is also almost planar is a good indication that the tracking is quite accurate.

Figs. 8, 9, and 10 show the stable result of tracking different textured objects and an O2 computer against a cluttered background. Results of Fig. 9 are obtained without using prior models for the texture on either side of the model. Note that the algorithm works well on the O2 even though it is not particularly textured, showing its versatility.

Our current implementation can process up to 120 fps on a 2.6 GHz machine using a dense set of samples on our CAD model set and a 1st order statistical model. This low computational cost potentially allows the tracking of multiple textured and non-textured objects in parallel using a single PC.
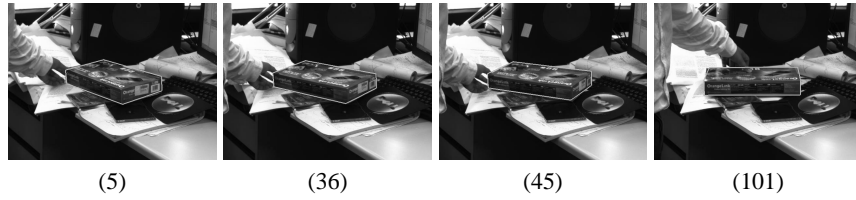
<center>(5)   (36)   (45)   (101)</center>

Figure8. Tracking a textured box against a cluttered background
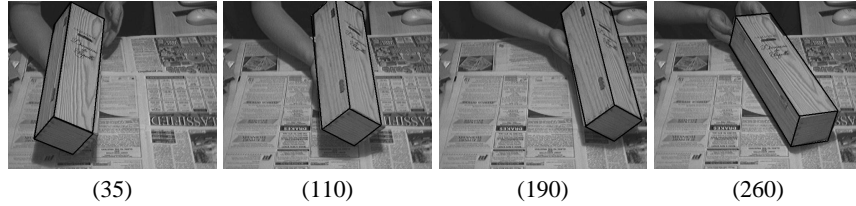

<center>(35)   (110)   (190)   (260)</center>

Figure9. Tracking a textured box against a cluttered background without recording prior models. The model is materialised by black lines.

6 Conclusion

In this paper, we have shown that a well formalized algorithm based on a Markov model lets us use a simple line search to detect the transition from one texture to another that occurs at object boundaries. This results in a very fast technique that we have validated by incorporating it into a 3–D model-based tracker, which unlike those that rely on edge-gradients to detect contours, succeeds in the presence of texture and clutter.

We have demonstrated our technique's effectiveness to track man-made objects. However, all objects whose occluding contours can be estimated analytically are subject to this treatment. For example, many body tracking algorithms model the human body
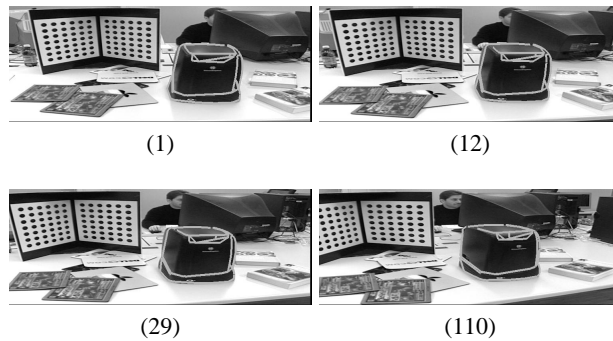

<center>(1)   (12)</center>

<center>(29)   (110)</center>

Figure10. Tracking of a much less textured O2 computer against a cluttered background

as a set of cylinders or ellipsoids attached to an articulated skeleton. The silhouettes of these primitives have analytical expressions as a function of the pose parameters. It should therefore be possible to use the techniques proposed here to find the true outlines and deform the body models accordingly. This will be the focus of our future work in that area. Another issue for future work is the behaviour of this method when lighting conditions change. This will be handled by replacing the current stationary statistical models by dynamic ones that can evolve.

References

1. Drummond, T., Cipolla, R.: Real-time tracking of highly articulated structures in the presence of noisy measurements. In: International Conference on Computer Vision, Vancouver, Canada (2001)
2. Tomasi, C., Kanade, T.: Shape and Motion from Image Streams under Orthography: A Factorization Method. International Journal of Computer Vision **9** (1992) 137–154
3. Fitzgibbon, A., Zisserman, A.: Automatic Camera Recovery for Closed or Open Image Sequences. In: European Conference on Computer Vision, Freiburg, Germany (1998) 311–326
4. Pollefeys, M., Koch, R., VanGool, L.: Self-Calibration and Metric Reconstruction In Spite of Varying and Unknown Internal Camera Parameters. In: International Conference on Computer Vision. (1998)
5. Marchand, E., Bouthemy, P., Chaumette, F., Moreau, V.: Robust real-time Visual Tracking Using a 2D-3D Model-Based Approach. In: International Conference on Computer Vision, Corfu, Greece (1999) 262–268
6. Lowe, D.G.: Robust model-based motion tracking through the integration of search and estimation. International Journal of Computer Vision **8(2)** (1992)
7. Vacchetti, L., Lepetit, V., Fua, P.: Fusing Online and Offline Information for Stable 3–D Tracking in Real-Time. In: Conference on Computer Vision and Pattern Recognition, Madison, WI (2003)
8. Jain, A.K., Farrokhnia, F.: Unsupervised texture segmentation using gabor filters. Pattern Recognition **23(12)** (December 1991) 1167–1186
9. Bovik, A., Clark, M., Geisler, W.: Multichannel Texture Analysis Using Localized Spatial Filters. IEEE Transactions on Pattern Analysis and Machine Intelligence **12** (1990) 55–73
10. Puzicha, J., Buhmann, J.M.: Multiscale annealing for grouping and unsupervised texture segmentation. Computer Vision and Image Understanding: CVIU **76** (1999) 213–230
11. Bouman, C., Liu, B.: Multiple Resolution Segmentation of Textured Images. IEEE Transactions on Pattern Analysis and Machine Intelligence **13(2)** (1991) 99–113
12. Pietikinen, M., Rosenfeld, A.: Image Segmentation Using Pyramid Node Linking. IEEE Transactions on Systems, Man and Cybernetics **12** (1981) 822–825
13. Schroeter, P., Bigün, J.: Hierarchical Image Segmentation by Multi-dimensional Clustering and Orientation Adaptive Boundary Refinement. Pattern Recognition **28(5)** (1995) 695–709
14. Rubio, T.J., Bandera, A., Urdiales, C., Sandoval, F.: A hierarchical context-based textured image segmentation algorithm for aerial images. Texture2002 (http://www.cee.hw.ac.uk/ texture2002) (2002)
15. Li, S.Z.: Markov Random Field Modeling in Computer Vision. Springer-Verlag, Tokyo (1995)
16. Rabiner, L.R.: A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In: IEEE. Volume 77(2). (1989) 257 – 286
17. Ozyildiz, E.: Adaptive texture and color segmentation for tracking moving objects. Master's thesis, Pennsylvania State University (1999)