Direct Pattern Tracking On Flexible Geometry

Igor Guskov
University of Michigan, Ann Arbor
guskov@eecs.umich.edu

Leonid Zhukov
Caltech
zhukov@gg.caltech.edu

Abstract

We introduce a robust tracking procedure for a regular pattern marked on a flexible moving surface such as cloth. A video of an actor performing a range of motions is processed with our algorithm to yield a dynamic geometric representation. The system is capable of maintaining the tracked grid structure for long periods of time without quality deterioration, and requires minimal user interaction. It has been tested on videos of an actor dressed in a specially marked T-shirt and behaves favorably with the presence of self-occlusions, self-shadowing and folding of the cloth. The focus of this paper is on single camera video sequence processing, even though 3D shape reconstruction with multiple cameras is the motivating goal.

Keywords: Pattern analysis, template matching, dynamic geometry

1 Introduction

Authoring complex secondary motion for animated characters is an important task facing 3D content creators. While physically based simulations are often used to produce such motion [8][17][1][12], they are computationally intensive. The use of acquired data to speed up, replace, or validate such computations is a viable option. A number of methods are available for accurate scanning of static geometry and for motion capture of the character's skeleton structure, while the robust and accurate acquisition of dynamic geometric data for self-occluding flexible moving surfaces remains an interesting open problem.

Our goal is to develop an accurate acquisition system that produces dynamic geometric data suitable for dressing and skinning virtual actors. In this paper our focus is on the processing of a video sequence acquired with a single camera. We use simple computer vision techniques to track a large number of points on a flexible moving surface such as cloth with the assumption that the tracked points form a surface grid whose structure is known and indicated as a painted pattern (see Figure 1). Thus we have much more information about the structure of the object being tracked than is assumed in most of the recent computer vision work [14]. As the result we are able to deduct the structure of the grid marked on the surface and successfully track it through long periods of time, as well as fully recover the correct deformation of that structure even after self-occlusions occur. The initial user interaction needed to setup the tracking system is minimal.

Many current shape acquisition techniques produce several registered point-clouds as their initial result. Such a representation cannot be directly used with animation applications and is therefore transformed into an irregular mesh which is often simplified or further transformed into a parametric surface description through fitting or re-meshing. Such processing can become only more complicated for acquired dynamic geometry. We therefore see an opportunity in the direct acquisition of structured parameterized surface representations.

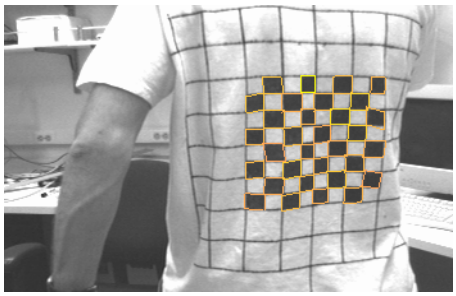


Figure 1: *A frame from the tracking sequence. Tracked grid of squares is overlayed over the image from the video.*

2 Related work

Motion capture data is widely used in computer graphics for animating articulated figures [15]. Motion capture techniques have also been extended to drive facial animation by many computer graphics practitioners in the past ten years [18][7][5][4]. The setups used vary from video tracking useful for the casual user to special marker-based facial motion capture products directed towards professional usage. The latter techniques require an actor to put special markers on her/his face, and track their positions with great precision, producing data suitable for direct integration into facial animation [18][7][9]. The former approaches belong to the class of passive user tracking algorithms employing computer vision techniques such as deformable models [5],

feature tracking [4], and 3D models [11][13]. A good review of computer vision techniques for capturing human motion is presented in [14]; many of these approaches have different objectives, using passively acquired video streams for surveillance and analysis. Being less invasive these techniques do not usually provide the degree of precision needed for production of high quality facial and cloth animation. The capture of quality data for a wide range of flexible motions requires reliable handling of the effects of folding, self-occlusion and self-shadowing. We therefore choose to follow in the steps of [7] and introduce specially marked pattern on the tracked surface.

We pursue the goal of modeling and synthesizing dressed humans which is similar to the motivation behind the work of Jojic et al. [10]. They present an integrated system for reconstructing body geometry as well as parameters for clothing; the system extensively uses physical modeling of cloth. In our work we have not yet used such models (with exception of some simple spatial and temporal heuristics applicable to many types of non-rigid motion). In fact, we believe in postponing such simulation-based efforts to later stages in the processing pipeline, and treat our system as a data acquisition unit whose purpose is to measure motion as precisely as possible for as wide range of materials as possible.

Our shape acquisition is based on tracking a pattern in a video sequence. In the current implementation we track area primitives[2] rather than curve features [3], because area-primitive template-match computations usually give more consistent and predictable results than the corresponding operations for curve primitives. Such consistency is very important when maintaining the correct grid structure during self-shadowing and self-occlusion of the cloth. The area primitives have the ability to appear and disappear during tracking due to occlusion – such functionality was inspired by the work of Tao et al. [16].

3 Tracking grid pattern

In this section we describe our approach to tracking a grid pattern on a flexible moving surface from a video sequence.

3.1 Overview

Our tracking system is designed to acquire the motion of points on a flexible moving surface such as cloth. The result of such acquisition is a polygonal mesh representing the moving surface. It is beneficial to mark this polygonal mesh structure directly on the tracked surface as a pattern. Such a correspondence between the real world and the acquired model helps resolve uncertainties during tracking. In this paper we have chosen a checkerboard pattern to indicate the connectivity of the surface mesh, so the black squares can be matched against a template and thus establish the adjacency relations among the mesh vertices.

While the ultimate goal of the tracking system is a consistent surface representation, we do not track the entire mesh as a whole. Rather the set of polygonal faces (quads) of black color is considered to be a loose collection of individual objects that are tracked individually while trying to enforce overall spatial consistency. Such an approach has several advantages: first of all, individual quads can become active or inactive independent of each other (each quad carries its own confidence value and becomes inactive when this value drops below a certain level). Such flexibility is very important while tracking cloth dynamics where self-occlusions are very common. Secondly, such individual quad treatment simplifies the process of template matching – the number of parameters during any local matching optimization remains six (corresponding to affine transformations of the planar square), rather than grow with the mesh size. Finally, the output of our tracking algorithm is a time-dependent set of active quads with their position and confidence information. Such information can be easily combined with other similar sets of data coming from other cameras, possibly using a physical model of cloth or other underlying material.

3.2 Notation

In this paper we restrict ourselves to working in image coordinates, thus all the positions are represented as pairs of real numbers. The current image in a frame will be denoted as $I(x, y)$, or $I(p)$ where $p = (x, y)$ is the position in the image plane. We also indicate the time dependence of data variables with the integer parameter t (index of the frame).

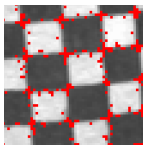
We assume that our squares are small and will not account for perspective distortion or non-rigid flexing of the cloth. Thus a distorted square will be represented by an affine transformation of a planar unit square that is a parallelogram $\omega = (p_0, p_1, p_2, p_3)$, with corners $p_k \in \mathbf{R}^2$ and a constraint: $p_0 + p_2 = p_1 + p_3$. We will use a term *quad* instead of lengthy *parallelogram* to describe an affinely distorted square. The system will track a structured grid of quads, however some quads may become *inactive* (not tracked) for periods of time due to occlusion. We call the quads that are being currently tracked *active* quads.

3.3 Overall structure

The following pseudo-code represents the overall structure of the tracking algorithm:

```
init:
user inputs some active quads
track:
attempt to extend active region
for every frame {
  for each active quad w {
    predict w's corners in time
    improve w's match with local search
  }
  check grid's spatial consistency
  attempt to extend active region
}
```

3.4 Corner/saddle detector

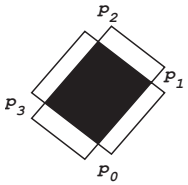


The corners of the tracked squares are saddle points of image intensity. This can be used in a pointwise corner detector that computes the determinant of the Hessian at every pixel of Gaussian filtered image to determine hyperbolic points. [6] The criteria for a saddle point is: $Det(I \star G_\sigma) < -\epsilon$, where $Det(F) := F_{xx}F_{yy} - F_{xy}^2$.

All the pixels detected as saddle points (shown is red in the image on the left) are used as candidates for placing corners of tracked quads. Therefore, it is important to have a few good candidates for every tracked corner (at least one saddle point should be detected at each corner so that the adjacent quads can be reconstructed).

3.5 Black quad detector

We use template matching to obtain the measure of how well a quad approximates a distorted black square surrounded by white lapels.



The distorted black quad detector is a function that for a given quad ω calculates its confidence level $c(\omega)$. We extend a given quad into a cross-like figure as follows: $\kappa(\omega) := \cup_{k=0}^{3} E_k(\omega) \cup \omega$, where $E_k(\omega)$ is a parallelogram defined by the corner points $p_k, p_{k+1}, (1+\alpha)p_{k+1} - \alpha p_{k+2}, (1+\alpha)p_k - \alpha p_{k-1}$ with width parameter α. We use $\alpha = 0.25$.

The template function $\mu : \kappa(\omega) \to \{-1, 1\}$ is defined as

$$\mu_\omega(p) := \begin{cases} -1, & \text{if } p \in \omega \\ 1, & \text{otherwise.} \end{cases}$$

Finally, we define the confidence level as an L_0 norm correlation between the template function and the difference between the image and average image intensity within the template region:

$$c(\omega) := \frac{1}{|\kappa(\omega)|} \sum_{p \in \kappa(\omega)} sign(I(p) - \bar{I}_\omega)\mu_\omega(p).$$

Here $|\kappa(\omega)|$ is the number of pixels within the template region and

$$\bar{I}_\omega := \frac{1}{|\kappa(\omega)|} \sum_{p \in \kappa(\omega)} I(p)$$

is the average image intensity within the template region. The defined confidence measure will increase when the relative distribution of image pixel values within the cross region $\kappa(\omega)$ is close to the template - most image pixels within the quad ω fall below the average intensity and the pixels in the "lapels" are above the average intensity.

Note that the confidence level $c(\omega)$ is guaranteed to be between -1 and 1 and that the values close to 1 indicate a good match. We have experimented with other correlation functions and found that the L_0 correlation defined above gives the most consistent results.

3.6 Temporal prediction

The active quad point positions in the current frame are predicted from the previous two frames using central difference $p_k(t+1) = 2p_k(t) - p_k(t-1), k = 0, \ldots, 3$ if it was active during both frames $t-1$ and t, otherwise the previous frame quad positions are copied $p_k(t+1) = p_k(t)$.

3.7 Optimizing local match via local search.

The optimization takes a given quad and improves its match with the template by searching for better positions for *three* of its corners among close pixels identified as candidate saddle points. The fourth corner is then determined by forming a parallelogram. For each of three corners the candidate saddle-point pixels are enumerated in the order of increasing distance from the initial position, and then the triples of possible corner positions are considered in the order determined by the sum of their indices from the previous stage. The search stops either after a given threshold value c_{high} for the match measure $c(\omega)$ is achieved or after a given maximum number of variants is considered. If after the search the best match is below a preset threshold value c_{low} the parallelogram is dismissed as not matched and becomes inactive.

N_c	$N_i^{max}(N_c)$
0	0
1	1
2	1
3	2
4	2

Table 1: Spatial grid consistency threshold.

In our experiments, we used the values $c_{low} = 0.62$ and $c_{high} = 0.75$.

3.8 Checking spatial grid consistency

The grid consistency check is needed to ensure that the tracked configuration corresponds to a valid grid structure. Every active quad is checked against its active neighbors whose current confidence level is above c_{high}: denote the number of such confident quad-neighbors as N_c. A neighboring quad is consistent with the given quad if their common vertex position lies within a preset threshold. We are using a distance threshold equal to 40% of the maximal quad size in the current implementation. We then count the number N_i of inconsistent pairs for a given quad: note that N_i is always less or equal than N_c. If N_i is greater or equal than an empirically found value $N_i^{max}(N_c)$ that is given in Table 1 then the current quad is deactivated.

3.9 Extending active region via spatial prediction

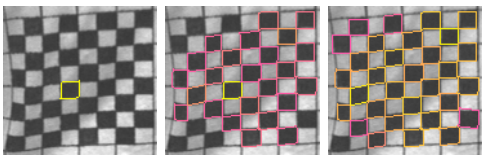


Figure 2: *Extending the active region during initialization. Left: original quad. Center: after one extension step. Right: one optimization step for all the active quads was followed by another extension step. The colors of active quads indicate their confidence levels with yellow corresponding to high confidence close to 1 and purple corresponding to low confidence close to* c_{low}.

The currently inactive parallelogram's vertex positions are predicted by linearly extrapolating the vertex positions of a neighboring active parallelogram whose confidence level is above c_{pred}. After initial extension, the parallelogram parameters are optimized for better matching. If the match or spatial consistency criteria are not satisfied the parallelogram is not activated. We set c_{pred} between the values c_{high} of high and c_{low} of low confidence. In our experiments, we use $c_{pred} = 0.7$.

3.10 Activating and deactivating quads

To summarize the previous discussion in 3.8 and 3.9 there are two places during a tracking cycle where a quad can become inactive: first, if after a temporal prediction the local search fails to improve the confidence level above c_{low} then the quad becomes inactive; another routine that deactivates quads is the spatial consistency check. The spatial prediction, is on the other hand, the only procedure that can activate quads. In order to assure that the active region does not grow too quickly we damp the confidence level of quads just introduced by spatial extension by a factor of 0.9. Since only the quads with sufficiently high confidence can predict their neighbors that restricts the growth of active region only to places were the match confidence is high.

4 Results

We have tested our algorithm on two video sequences acquired with B/W CCD camera at 30 frames/sec. The image resolution was 376 by 240 and the length of both video sequences was 20 seconds. An 8 x 8 checkerboard grid pattern with 32 black squares was painted with black markers on a white t-shirt. A person wearing the marked t-shirt performed various motions before the camera. The acquired video was then processed by our tracking system.

User interaction consisted of roughly delineating one square in the middle of the pattern (four mouse clicks). After this initialization, the grid structure was automatically extended to the entire connected visible region of the pattern (this can be done either before starting the tracking procedure in the first frame, or this extension happens automatically over the first few frames during tracking). Consequently the grid pattern structure was tracked through the video. The tracker has successfully restored the grid structure after self-occlusions occurred. Figure 3 shows several tracked frames from the video sequence around occlusion time. There was no deterioration of the tracking quality through time.

The current procedure takes about five seconds per frame for simple frames with no occlusions, and about ten seconds for complicated frames with significant occlusions and self-shadowing due to increased amount of active region maintenance operations. The system was tested on a PC workstation with 1.7GHz Xeon processor.

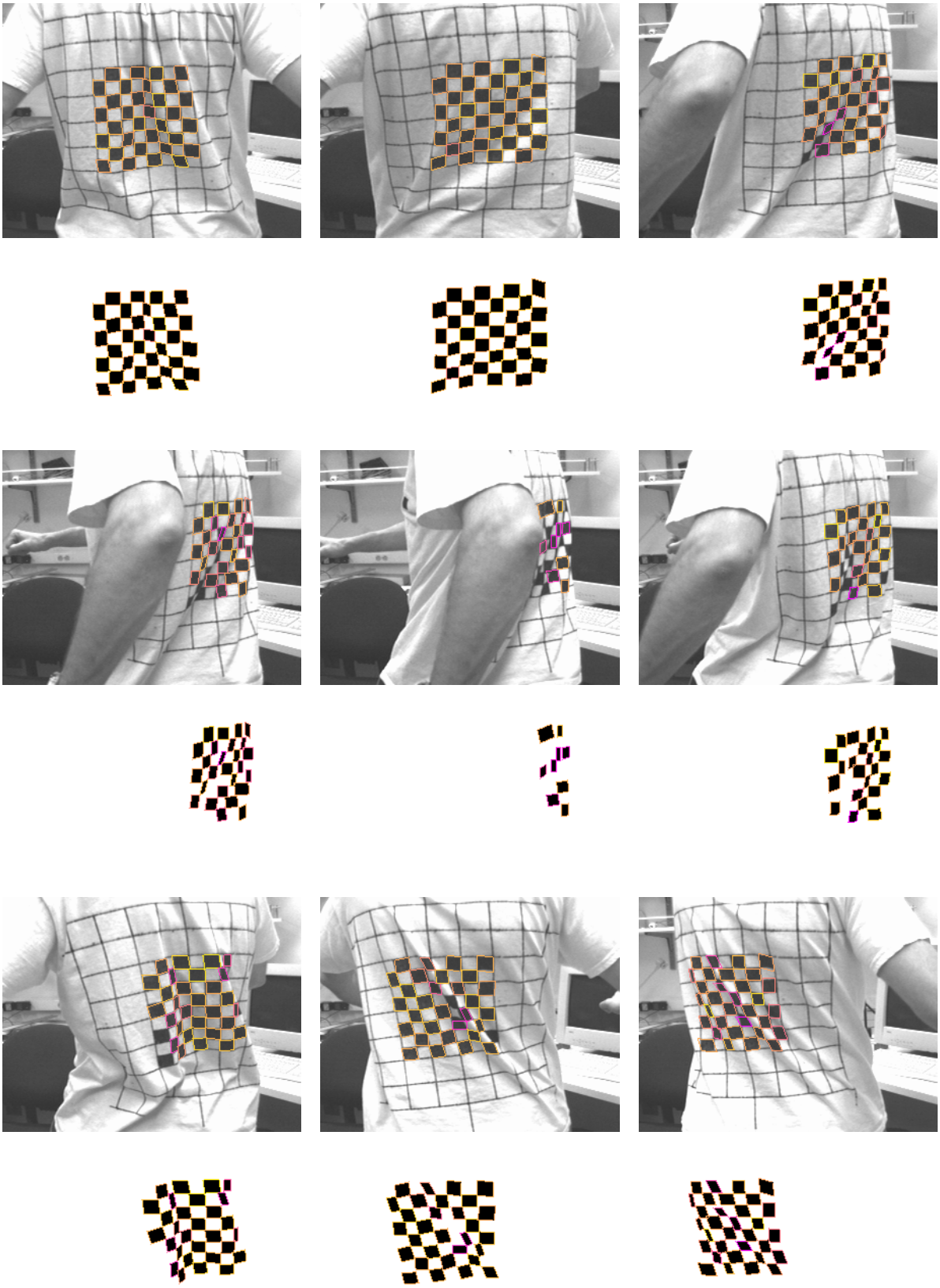


Figure 3: *Nine frames from the tracking sequence together with extracted grid of quads. Note that there are only nine quads properly tracked in second frame of the second row. The structure of the grid is being correctly recovered in the following frames. See note on colors in the caption to Figure 2*

5 Conclusions

We have introduced the concept of direct tracking of marked grid patterns on flexible moving surfaces, and shown that we can track such grids reliably using simple vision techniques. More future work is required with multiple cameras and full 3D surface descriptions.

In this work we have not addresed the problem of acquiring fast abrupt surface motion that may result in tracker's confusion due to the invariance of the pattern to some local translations. This is a fundamental problem and it can be alleviated by careful design of color-coded patterns.

Acknowledgments We would like to thank Pietro Perona and Xiaolin Feng for their help with acquiring the data, and Peter Schröder and David Breen for encouragement and support. This work was supported in part by NSF (DMS-9874082, DMS-9872890, ACI-9982273), Alias|Wavefront, and Microsoft.

References

[1] D. Baraff and A. Witkin. Large steps in cloth animation. In *Proceedings of SIGGRAPH'98*, pages 43–54, 1998.

[2] P.-L. Bazin and J.-M Vezien. Tracking geometric primitives in video streams. In *Proceedings 4th IMVIP*, 2000.

[3] Andrew Blake and Michael Isard. *Active Contours*. Springer, 1998.

[4] I. Buck, A. Finkelstein, C. Jacobs, A. Klein, D.H. Salesin, J. Seims, R. Szeliski, and K. Toyama. Performance-driven hand-drawn animation. In *NPAR 2000*, Annecy, France, June 2000.

[5] D. DeCarlo and D Metaxas. Deformable model-based shape and motion analysis for images using motion residual error. In *Proceedings ICCV'98*, pages 113–119, 1998.

[6] R. Deriche and G. Giraudon. A computational approach for corner and vertex detection. *IJCV*, 10(2):101–124, 1993.

[7] B. Guenter, C. Grimm, D. Wood, H. Malvar, and F. Pighin. Making faces. In *Proceedings SIGGRAPH'98*, pages 55–66, 1998.

[8] Donald H. House and David E. Breen, editors. *Cloth Modeling and Animation*. A K Peters Ltd, 2000.

[9] http://www.metamotion.com/. MetaMotion face tracker.

[10] N. Jojic, J. Gu, H. Shen, and Huang T. Computer modeling, analysis, and synthesis of dressed human. *IEEE Trans. on Circuits and Systems for Video Technology*, 2:378–388, 1999.

[11] I. Kakadiaris and D. Metaxas. Vision-based animation of digital humans. In *Proceedings of Computer Animation 98*, pages 144 – 152, 1998.

[12] Yuencheng Lee, Demetri Terzopoulos, and Keith Walters. Realistic modeling for facial animation. In *Proceedings of SIGGRAPH 95*, pages 55–62, 1995.

[13] N. Magnenat-Thalmann and D. Thalmann, editors. *Modelling and Motion Capture Techniques for Virtual Environments*, Lecture Notes in Artificial Intelligence, 1998. International Workshop, CAPTECH'98.

[14] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81:231–268, 2001.

[15] Gordon Cameron (organizer). Motion capture and CG character animation (panel). In *Proceedings of SIGGRAPH 1997*, 1997.

[16] Hai Tao, Harpeet Sawhney, and Rakesh Kumar. A sampling algorithm for tracking multiple objects. In Bill Triggs, Andrew Zisserman, and Richard Szelinski, editors, *International Workshop on Vision Algorithms*, pages 53–69. Springer, September 1999.

[17] Pascal Volino and Nadia Magnenat-Thalmann. *Virtual Clothing*. Springer Verlag, 2000.

[18] L. Williams. Performance-driven facial animation. In *Proceedings of SIGGRAPH'90*, volume 24, pages 235–242, 1990.