Active Surfaces for Video Tracking and 3-D Segmentation Based on a New Method for Multidimensional Optimization

Nidhal Bouaynaya and Dan Schonfeld

Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, USA.

ABSTRACT

We propose an optimal framework for active surface extraction from video sequences. An active surface is a collection of active contours in successive frames such that the active contours are constrained by spatial and temporal energy terms. The spatial energy terms impose constraints on the active contour in a given frame. The temporal energy terms relate the active contours in different frames to preserve desired internal and external properties of the active surface. For computational efficiency, we reduce the 3-D active surface ((x, y, t) coordinates) optimization problem to a 2-D model ((ϕ, t) coordinates) by considering only point indices along normal lines ϕ of each contour and define the energy terms in a causal way. We develop an n-D dynamic tree programming algorithm to find the optimum of n-D semi-causal functions. We prove that the n-D dynamic tree programming algorithm converges to the global optimum. In particular, the classical 1-D dynamic programming algorithm is a special case of the n-D dynamic tree programming algorithm. The optimal active surface is subsequently obtained by using the 2-D dynamic tree programming algorithm. Simulation results show the efficiency and robustness of the proposed approach in active surface extraction for video tracking and segmentation of the human head in real-world video sequences.

Keywords: tracking, segmentation, active surface, active contour, n-D dynamic programming.

1. INTRODUCTION

Robust object tracking in video sequences requires a parametric state representation of the target[1] .[2] In many applications, however, it is important to extract the object contour by precisely delineating its boundary. For example, in virtual reality, we need an accurate elastic contour delineating the object's boundary in order to create virtual scenes.[3] Accurate contour delineation of objects is also necessary to refine the tracking or segmentation process.[4]

Active contours (also known as snakes) have been proven to provide a powerful approach in elastic contour extraction in still and moving images[5] [6] .[7] In this approach, a compound energy function, taking into account the contour properties and the image forces, is minimized. The curve properties are incorporated into an internal energy function, which measures the desired properties of the contour such as smoothness and continuity. The image forces are incorporated into an external energy term, which is derived from a selected image functional to measure desired features such as edges, lines, regions, etc. The technique uses an initial guess of the contour and fixed scalar parameters to weigh the different energy terms. The solution is highly dependent upon the weight parameters which define the energy function as well as the initial contour due to the many local minima of the total energy function. Many extensions and variations have been suggested to the original active contour by Kass et. al.,[5] referring to the definition of the energy function as well as the optimization technique. The balloon model[8] and the dual active contour model[9] are two different techniques proposed to reduce the sensitivity of the active contour to the initial condition and prevent the solution from getting stuck at local minima. However, the balloon model increases the number of parameters and the dual active contour model needs two initial contours: one contracts from outside and the other expands from inside. Dynamic programming[10] and simulated annealing[11] algorithms have subsequently been proposed to reach the global minimum. A comparison of different active contour models has also been investigated.[12]

In contour extraction of objects in video sequences, tracking is achieved on each frame separately except for the propagation of the result from one frame to the next[13] [14] .[4] Active contours are then used as a post-processing or refinement procdedure. The initial condition of the active contour is given by the parametric

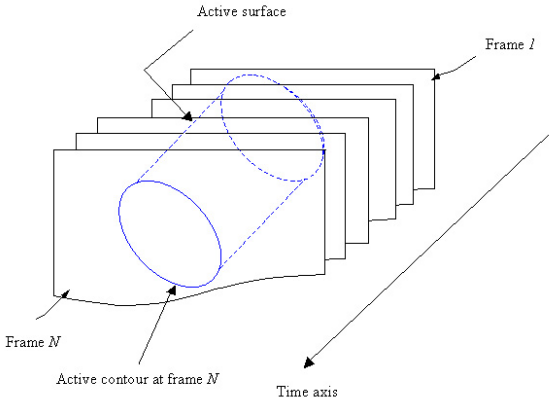

Figure 1. An active surface is formed by a collection of active contours in successive frames.

contour of the tracker.[4] The sensitivity of the active contour to the initial condition and the weight parameters at each frame, translates to an extremely jittery and unstable contour in video sequences. In other words, the contour in the video sequence is not smooth over time. Hence the idea of incorporating temporal information for contour extraction in video sequences. Chalana et. al. included a temporal energy function to detect cardiac boundaries from a sequence of echocardiograms.[15] However, their model is constrained by a monotonic motion assumption. Moreover, the temporal energy term is applied to points that do not necessarily correspond to each other. In,[16] a spatio-temporal energy function is used to track the tongue contour in ultrasound images. The proposed technique, however, requires two initial contours specified at two different depths. A motion based contour extraction technique has been proposed in.[17] In this approach, the authors pursue a sub-optimal solution by decomposing the optimization with respect to the spatial and temporal energy terms.

We propose to use active surfaces to provide a stable and less jittery contour extraction from video sequences. An *active surface* is formed by a collection of active contours in successive video frames (see Fig. 1). We simplify the active surface optimization problem from a 3-D model ((x, y, t) dimensions) to a 2-D model((ϕ, t) dimensions) by considering only observations along normal lines ϕ of each contour (see Fig. 2). Moreover, we define the energy terms in a semi-causal way to allow propagation of the total energy from the the first normal line in the first frame to the last normal line in the last frame of the surface. More generally, we extend the notion of causality to n-D spaces as follows: a semi-causal cost function J from $\mathbb{R}^n$ to $\mathbb{R}$ evaluated at point $x_{k_1,\cdots,k_n}$ depends only on the cost of the set $\{x_{i_1,\cdots,i_n} : i_1 \leq k_1, \cdots, i_n \leq k_n \text{ and } (i_1, \cdots, i_n) \neq (k_1, \cdots, k_n)\}$. We propose an n-D dynamic tree programming algorithm to find the optimal solution of semi-causal n-D cost functions. The dynamic tree programming algorithm has an intuitive geometric interpretation using the inclusion-exclusion principle. The 1-D dynamic programming algorithm is a special case of the n-D dynamic tree programming algorithm. We obtain the optimal solution of the 2-D active surface optimization problem by using the 2-D dynamic tree programming algorithm.

This paper is organized as follows: In Section 2, we present the 2-D active surface model and define the spatial energy compound function in a causal way. In Section 3, we motivate and define the different temporal energy terms. Finally, we present a 2-D optimization problem whose solution yields the active surface. In Section 4, we formulate the optimization problem of semi-causal n-D cost functions. We define the principle of optimality, which states that the 'tail' of the optimal state is optimal for the corresponding 'tail' subproblem. Based on the principle of optimality, we derive the n-D dynamic tree programming algorithm and prove, by induction, that it converges to the optimal solution. The active surface optimization problem is then optimally solved in Section 5 using the 2-D dynamic tree programming algorithm. Section 6 presents some simulation results for active surface extraction for video tracking and segmentation of the human head in video sequences. The temporal jitter of the active surface is drastically reduced compared to an active contour extraction without any temporal constraints. Conclusions and brief discussion of future work is presented in Section 7.

2. 1-D ACTIVE CONTOUR REPRESENTATION

Based on the 1-D active contour model presented in[18] , we consider a 2-D active surface model. At frame t, only observations along the normal lines of the contour are detected. Let $\phi = 1, \cdots, M$ be the index of normal lines and $\lambda_{\phi,t} = -N, \cdots, N$ be the index of pixels along the normal line ϕ at time t. Each normal line has $2N+1$ pixels, which are indexed from $-N$ to N. The center point of each normal line is placed on the initial contour and indexed as 0. $I(\lambda_{\phi,t})$ denotes the intensity value pixel index λ_ϕ. The 2-D contour is then represented by the set of its M normal lines. Figure 2 depicts a 1-D active contour model at frame t. We consider a sequence of

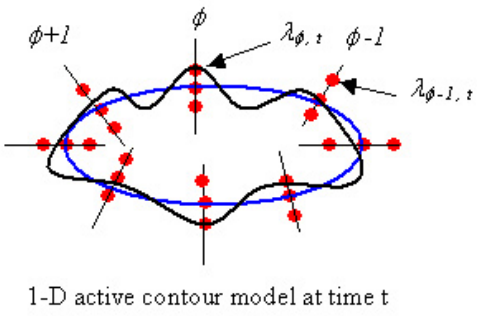

Figure 2. Illustration of the 1-D active contour model. We choose the ellipse to be the initial parametric contour given by the tracking module of the human head. The elastic curve is the best local contour that we want to find. A set of measurements are collected along the M normal lines of the initial contour. The 2-D contour is then represented by the set of its M normal lines.

T video frames. The T active contours form the *active surface*. The video tracking and segmentation problems can be formulated as the determination of the active surface represented by pixel locations $\lambda_{\phi,t}$ for each normal line $\phi = 1, \cdots, M$ and time $t = 1, \cdots, T$. Hence we reduced the 3-D model of the active surface optimization problem to a 2-D model.

Like tractional contour models, we need to define energy functions, which impose constraints on the active surface. We classify the energy terms into two categories: spatial and temporal. The spatial energy terms impose constraints on the contour in a given frame independently of other contours in the other frames. The temporal energy terms relate the active contours in different frames to impose desired properties on the active surface. Spatial energy terms are depicted by a superscript s whereas temporal energy terms have the superscript t. To do the optimization efficiently instead of the slow iterative search, we define the spatial and temporal energy terms in a causal way. The optimal active surface can therefore be found by a single iteration of dynamic tree programming as we will describe in Section 5. We begin by defining the spatial energy terms.

Spatial external energy The external forces push the contour towards the image features such as edges. The external energy is then defined as a function of the image gradient along the direction of the line[5] [6]:

$$E^{\mathrm{s}}_{\mathrm{ext}}(\lambda_{\phi,t}) = g(-|\frac{d}{d\lambda_{\phi,t}} I(\lambda_{\phi,t})|^2) \quad (1)$$

$$\approx g(-|I(\lambda_{\phi,t}+1) - I(\lambda_{\phi,t})|^2), \quad (2)$$

where g is a monotonically increasing function.

Spatial internal energy The spatial internal energy imposes smoothness of the contour by penalizing rough contour points. In the traditional active contour model, the roughness is defined by the first and second derivatives

of the contour. This definition is not causal because the first and second derivatives depend on both the pixels before and after the current pixel on the contour. This leads to an iterative solution of the optimization problem ,which is not adequate for real-time applications. Therefore, we redefine the spatial internal energy term in a causal manner as follows:

$$E^{\mathrm{s}}_{\mathrm{int}}(\lambda_{\phi,t}, \lambda_{\phi-1,t}) = |\lambda_{\phi,t} - \lambda_{\phi-1,t}|^2. \tag{3}$$

Spatial shape energy Taking into account a shape information will avoid the snake to lock onto unwanted background features despite of the smoothness energy term. Assuming that the initial contour is relatively accurate, we can penalize the points that are far away from the center of the normal line by fitting a zero-mean Gaussian at the center point. For example, this situation can occur if the initial contour at each time frame is the output of an accurate tracking module. The shape energy term is then defined by

$$E^{\mathrm{s}}_{\mathrm{shape}}(\lambda_{\phi,t}) = \frac{\lambda^2_{\phi,t}}{\sigma^2}, \tag{4}$$

where σ is the standard deviation of the Gaussian. If the tracking is not accurate enough for some frames *, then σ should be large to reflect the inaccuracy of the initial contour.

The total spatial energy function is a weighted sum of the different spatial energy terms, i.e.,

$$E^{\mathrm{s}}(\lambda_{\phi,t}, \lambda_{\phi-1,t}) = \alpha^s_{\mathrm{ext}}\ E^{\mathrm{s}}_{\mathrm{ext}}(\lambda_{\phi,t}) + \alpha^s_{\mathrm{int}}\ E^{\mathrm{s}}_{\mathrm{int}}(\lambda_{\phi,t}, \lambda_{\phi-1,t}) + \alpha^s_{\mathrm{shape}}\ E^{\mathrm{s}}_{\mathrm{shape}}(\lambda_{\phi,t}), \tag{5}$$

where $\alpha^s_{\mathrm{ext}}, \alpha^s_{\mathrm{int}}$ and $\alpha^s_{\mathrm{shape}}$ are the weight coefficients.

3. TEMPORAL CONSTRAINTS

We propose the incorporation of temporal energy terms in the total energy function. Similar to the spatial energy terms, we choose the temporal energy terms to maintain a causal relation between adjacent normal lines. Extension of the total energy to include both spatial and temporal energy terms is critical to provide a stable and less jittery contour extraction in video sequences.

Temporal external energy The external energy term relates the active surface to the video features. In traditional active contour models, the spatial external energy term relates the active contour to the current frame features (e.g., edges, lines, texture) without taking into account the features of the neighboring frames. If the current frame happens to have weak features due to noise or blurring, then the active contour will not lock into the desired boundaries. Incorporating adjacent frames, which potentially have stronger features, will influence the active contour to lock to the correct boundaries. Since we are considering intensity and gradients as the image features, correlation is a suitable measure to find the best feature match in a given neighborhood in adjacent frames. In order to maintain the causal relation between indices, we define the temporal external energy as

$$E^{\mathrm{t}}_{\mathrm{ext}}(\lambda_{\phi,t}, \lambda_{\phi,t-1}) = -|I(\lambda_{\phi,t})I(\lambda_{\phi,t-1}) + I(\lambda_{\phi,t}+1)I(\lambda_{\phi,t-1}+1)|^2. \tag{6}$$

Temporal internal energy In order to reduce the jitter of the active surface, we need to impose smoothness constraints on the active surface. Similarly to the definition of the spatial internal energy function, we define the temporal internal energy function as the causal approximation of the first derivative of the surface with respect to time.

$$E^{\mathrm{t}}_{\mathrm{int}}(\lambda_{\phi,t}, \lambda_{\phi,t-1}) = |\lambda_{\phi,t} - \lambda_{\phi,t-1}|^2. \tag{7}$$

The total temporal energy function is a weighted sum of the temporal internal and external energy functions, i.e.,

$$E^{\mathrm{t}}(\lambda_{\phi,t}, \lambda_{\phi,t-1}) = \alpha^t_{\mathrm{ext}}\ E^{\mathrm{t}}_{\mathrm{ext}}(\lambda_{\phi,t}, \lambda_{\phi,t-1}) + \alpha^t_{\mathrm{int}}\ E^{\mathrm{t}}_{\mathrm{int}}(\lambda_{\phi,t}, \lambda_{\phi,t-1}). \tag{8}$$

*The accuracy of a tracking algorithm can be measured by a performance index, e.g., number of effective particles in Particle filter trackers.[19]

Let $c(\phi,t)$ denote the optimal index point at normal line ϕ at time t and let $S(\phi,t)$ denote the optimal active surface, which is a function of normal lines $\phi = 1,\cdots,M$ and time $t = 1,\cdots,T$. The total objective function of the active surface $S(\phi,t)$ is given by

$$\begin{aligned} E(S(\phi,t),S(\phi-1,t),S(\phi,t-1)) &= \sum_{t=1}^{T}\sum_{\phi=1}^{M} \alpha^s_{\text{ext}}\ E^{\text{s}}_{\text{ext}}(S(\phi,t)) + \alpha^s_{\text{int}}\ E^{\text{s}}_{\text{int}}(S(\phi,t),S(\phi-1,t)) \\ &+ \alpha^s_{\text{shape}}\ E^{\text{s}}_{\text{shape}}(S(\phi,t)) + \ \alpha^t_{\text{ext}}\ E^{\text{t}}_{\text{ext}}(S(\phi,t),S(\phi,t-1)) \\ &+ \alpha^t_{\text{int}}\ E^{\text{t}}_{\text{int}}(S(\phi,t),S(\phi,t-1)). \end{aligned} \tag{9}$$

The optimal active surface $S(\phi,t)$ is the one that minimizes the total energy E for $\phi = 1,\cdots,M$ and $t = 1,\cdots,T$. A naive approach to this optimization problem would take $(2N+1)^{MT}$ to find the optimal active surface. Fortunately, the semi-causal definitions of the energy terms allows a dynamic tree programming algorithm, which finds the optimal active surface in one single iteration as explained in the following sections.

4. N-D DYNAMIC TREE PROGRAMMING ALGORITHM

In this section, we consider the general optimization problem of a semi-causal cost function defined on $\mathbb{R}^n$.

Problem Formulation Consider the following discrete-time semi-causal n-dimensional system

$$x_{k_1+1,k_2+1,\cdots,k_n+1} = f_{i_1,i_2,\cdots,i_n}(x_{i_1,i_2,\cdots,i_n}, u_{i_1,i_2,\cdots,i_n}), \tag{10}$$

such that $i_1 \le k_1+1, i_2 \le k_2+1,\cdots,i_n \le k_n+1$ and $(i_1,i_2,\cdots,i_n) \ne (k_1+1,k_2+1,\cdots,k_n+1)$, $k_i = 0,\cdots,N_i$ for all $i = 1,\cdots,n$, and where

- $x_{i_1,i_2,\cdots,i_n}$ is the n-D state at point $(i_1,i_2,\cdots,i_n)$.
- $u_{i_1,i_2,\cdots,i_n}$ is the control or decision to be selected from a given set.

Figure 3(a) illustrates the semi-causality condition in 2-D spaces. The control $u_{k_1,k_2,\cdots,k_n}$ is related to the state $x_{k_1,k_2,\cdots,k_n}$ by the policy $\mu_{k_1,k_2,\cdots,k_n}$, i.e.,

$$u_{k_1,k_2,\cdots,k_n} = \mu_{k_1,k_2,\cdots,k_n}(x_{k_1,k_2,\cdots,k_n}). \tag{11}$$

Let π be the policy vector

$$\pi = (\mu_{0,0,\cdots,0}, \mu_{1,0,\cdots,0},\cdots,\mu_{N_1,N_2,\cdots,N_n}). \tag{12}$$

To simplify the notations, we will remove the subscript $(k_1,k_2,\cdots,k_n)$ from the considered functions whenever there is no confusion about the meaning. The cost function starting at $x_{0,0,\cdots,0}$ is assumed to be additive, i.e.,

$$J_\pi(x_{0,0,\cdots,0}) = \sum_{k_1=0}^{N_1}\sum_{k_2=0}^{N_2}\cdots\sum_{k_n=0}^{N_n} g_{k_1,k_2,\cdots,k_n}(x_{k_1,k_2,\cdots,k_n}, \mu_{k_1,k_2,\cdots,k_n}(x_{k_1,k_2,\cdots,k_n})) \tag{13}$$

In the 1-D case, i.e., $n = 1$, the cost function in Eq. (13) reduces to

$$J_\pi(x_0) = g_N(x_N) + \sum_{k=1}^{N-1} g_k(x_k). \tag{14}$$

This is the additive cost function of the classical 1-D dynamic programming algorithm[20] .[21]

The Optimization problem is performed over the policy vector π, i.e.,

$$J^*(x_{0,0,\cdots,0}) = \min_\pi J_\pi(x_{0,0,\cdots,0}). \tag{15}$$

The optimal policy satisfies

$$J_{\pi^*}(x_{0,0,\cdots,0}) = J^*(x_{0,0,\cdots,0}), \tag{16}$$

and π^* is independent of $x_{0,0,\cdots,0}$.

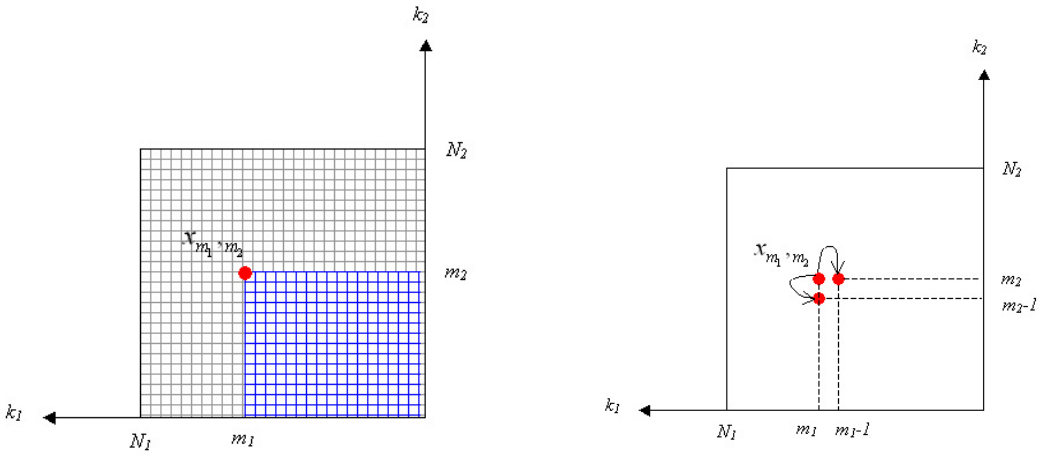

Figure 3. Semi-causality and strict semi-causality in 2-D spaces: (a) Semi-causal system: the state x_{m_1,m_2} depends on all the states with dimensions (k_1, k_2) enclosed in the blue grid except for the dimension (m_1, m_2); (b) Strict semi-causal system: the state x_{m_1,m_2} depends only on the states x_{m_1-1,m_2} and x_{m_1,m_2-1}.

Principle of optimality Let $\pi^* = (\mu^*_{0,0,\cdots,0}, \mu^*_{1,0,\cdots,0}, \cdots, \mu^*_{N_1,N_2,\cdots,N_n})$ be an optimal policy. Consider the 'tail subproblem' whereby we are at $x_{i_1,i_2,\cdots,i_n}$ and wish to minimize the cost from point $(i_1, i_2, \cdots, i_n)$ to point $(N_1, N_2, \cdots, N_n)$ with respect to the tail policy $\pi = (\mu_{i_1,i_2,\cdots,i_n}, \cdots, \mu_{N_1,N_2,\cdots,N_n})$. The principle of optimality states that the tail policy is optimal for the tail subproblem, i.e., $\pi^* = (\mu^*_{i_1,i_2,\cdots,i_n}, \cdots, \mu^*_{N_1,N_2,\cdots,N_n})$ minimizes the cost from point $(i_1, i_2, \cdots, i_n)$ to point $(N_1, N_2, \cdots, N_n)$.

Theorem 1.(n-D Dynamic Tree Programming)
Start with $J(x_{k_1,\cdots,k_{i-1},N_i,k_{i+1},\cdots,k_n}) = g_{k_1,\cdots,k_{i-1},N_i,k_{i+1},\cdots,k_n}(x_{k_1,\cdots,k_{i-1},N_i,k_{i+1},\cdots,k_n})$ for all $k_i = 1, \cdots, N_i$, for all $i = 1, \cdots, n$ and go backwards using

$$J(x_{k_1,k_2,\cdots,k_n}) = \min_{u_{k_1,k_2,\cdots,k_n}} \left\{ g(x_{k_1,k_2,\cdots,k_n}, u_{k_1,k_2,\cdots,k_n}) + \sum_{i_1=k_1}^{N_1} \cdots \sum_{\substack{i_n=k_n \\ (i_1,\cdots,i_n)\neq(k_1,\cdots,k_n)}}^{N_n} g(x_{i_1,\cdots,i_n}) \right\} \quad (17)$$

$$= \min_{u_{k_1,k_2,\cdots,k_n}} \left\{ g(x_{k_1,k_2,\cdots,k_n}, u_{k_1,k_2,\cdots,k_n}) + \sum_{i=1}^{n} J(x_{k_1,\cdots,k_{i-1},k_i+1,k_{i+1},\cdots,k_n}) \right. \quad (18)$$

$$- \sum_{\substack{(i_1,i_2)=(1,1) \\ i_1\neq i_2 \\ (i_1,i_2)=(i_2,i_1)}}^{(n,n)} J(x_{k_1,\cdots,k_{i_1}+1,\cdots,k_{i_2}+1,\cdots,k_n}) + \cdots\cdots$$

$$+ (-1)^{m-1} \sum_{\substack{(i_1,i_2,\cdots,i_m)=(1,1,\cdots,1) \\ i_1\neq i_2\neq\cdots i_m \\ (i_1,i_2,\cdots,i_m)=P((i_1,i_2,\cdots,i_m))}}^{(n,n,\cdots,n)} J(x_{k_1,\cdots,k_{i_1}+1,\cdots,k_{i_m}+1,\cdots,k_n}) + \cdots\cdots$$

$$\left. + (-1)^{n-1} J(x_{k_1+1,k_2+1,\cdots,k_n+1}) \right\},$$

where $P((i_1, i_2, \cdots, i_m))$ denotes a permutation of $(i_1, i_2, \cdots, i_m)$. Equation (18) is obtained from Eq. (17) using the inclusion-exclusion principle. $J(x_{0,0,\cdots,0})$, generated at the last step, is equal to the optimal cost $J^*(x_{0,0,\cdots,0})$. Also, the policy $\pi^* = (\mu^*_{0,0,\cdots,0}, \cdots, \mu^*_{N_1,\cdots,N_n})$ is optimal.

Corollary 1.(2-D Dynamic Tree Programming)
Start with $J(x_{N_1,k_2}) = g_{N_1,k_2}(x_{N_1,k_2})$ for all $k_2 = 1, \cdots, N_2$ and $J(x_{k_1,N_2}) = g_{k_1,N_2}(x_{k_1,N_2})$ for all $k_1 = 1, \cdots, N_1$. Then go backwards using

$$J^*(x_{k_1,k_2}) = \min_{\mu_{k_1,k_2}} \{g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + J(x_{k_1+1,k_2}) + J(x_{k_1,k_2+1}) - J(x_{k_1+1,k_2+1})\}. \tag{19}$$

Proof of Theorem 1. We prove the dynamic tree programming algorithm in the 2-D case. The proof of the n-D case (where $n \geq 3$ and n is finite) follows by induction from the 2-D case. From Eq. (13), the cost function starting at state $x_{0,0}$ is

$$J_\pi(x_{0,0}) \quad = \quad \sum_{k_1=0}^{N_1} \sum_{k_2=0}^{N_2} g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) \tag{20}$$

We prove by induction that $J(x_{k_1,k_2}) = J^*(x_{k_1,k_2})$ defined as the optimal cost of the tail subproblem that starts at state x_{k_1,k_2}. Let $\hat{\pi}_{k_1,k_2}$ denote a tail policy from point x_{k_1,k_2} onwards excluding the point x_{k_1,k_2}, i.e., $\hat{\pi}_{k_1,k_2} = \{\mu_{k_1+1,k_2}, \mu_{k_1,k_2+1}, \mu_{k_1+1,k_2+1}, \cdots, \mu_{N_1,N_2}\}$. Initially, $J(x_{N_1,j}) = J^*(x_{N_1,j})$ and $J(x_{i,N_2}) = J^*(x_{i,N_2})$ for all $i = 0, \cdots N_1$ and $j = 0, \cdots N_2$ are assumed to be known. Assume at stage (i_1, i_2) that $J(x_{i_1,i_2}) - g(x_{i_1,i_2}, \mu_{i_1,i_2}(x_{i_1,i_2}))$ is optimal for the tail problem from point (i_1, i_2) onwards excluding the point (i_1, i_2). Then

$$\begin{aligned} J^*(x_{k_1,k_2}) = \min_{\mu_{k_1,k_2}, \hat{\pi}_{k_1,k_2}} & \Bigg\{ g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + \sum_{i_1=k_1+1}^{N_1} \sum_{i_2=k_2+1}^{N_2} g_{i_1,i_2}(x_{i_1,i_2} + \mu_{i_1,i_2}(x_{i_1,i_2})) \\ & + \sum_{j=k_2+1}^{N_2} g_{k_1,j}(x_{k_1,j}, \mu_{k_1,j}(x_{k_1,j})) + \sum_{i=k_1+1}^{N_1} g_{i,k_2}(x_{i,k_2}, \mu_{i,k_2}(x_{i,k_2})). \Bigg\} \qquad (21) \\ = \min_{\mu_{k_1,k_2}} & \Bigg\{ g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + \min_{\hat{\pi}_{k_1,k_2}} \Bigg[\sum_{i_1=k_1+1}^{N_1} \sum_{i_2=k_2+1}^{N_2} g_{i_1,i_2}(x_{i_1,i_2} + \mu_{i_1,i_2}(x_{i_1,i_2})) \\ & + \sum_{j=k_2+1}^{N_2} g_{k_1,j}(x_{k_1,j}, \mu_{k_1,j}(x_{k_1,j})) + \sum_{i=k_1+1}^{N_1} g_{i,k_2}(x_{i,k_2}, \mu_{i,k_2}(x_{i,k_2})) \Bigg] \Bigg\}. \qquad (22) \end{aligned}$$

Figure 4(a) illustrates graphically the different summation terms in Eq. (21). Observe that

$$\begin{aligned} & \sum_{i_1=k_1+1}^{N_1} \sum_{i_2=k_2+1}^{N_2} g_{i_1,i_2}(x_{i_1,i_2} + \mu_{i_1,i_2}(x_{i_1,i_2})) + \sum_{j=k_2+1}^{N_2} g_{k_1,j}(x_{k_1,j}, \mu_{k_1,j}(x_{k_1,j})) + \sum_{i=k_1+1}^{N_1} g_{i,k_2}(x_{i,k_2}, \mu_{i,k_2}(x_{i,k_2})) \\ = \quad & J(x_{k_1+1,k_2}) + J(x_{k_1,k_2+1}) - J(x_{k_1+1,k_2+1}) \qquad (23) \\ = \quad & J(x_{k_1,k_2}) - g_{k_1,k_2}(x_{i_1,i_2} + \mu_{i_1,i_2}(x_{i_1,i_2})). \qquad (24) \end{aligned}$$

Equations (23) and (24) can be seen graphically from Fig. 4(b). Therefore, Eq. (22) becomes

$$\begin{aligned} J^*(x_{k_1,k_2}) \quad &= \quad \min_{\mu_{k_1,k_2}} \{g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + \min_{\hat{\pi}_{k_1,k_2}} [J(x_{k_1,k_2}) - g_{k_1,k_2}(x_{i_1,i_2} + \mu_{i_1,i_2}(x_{i_1,i_2}))]\} \qquad (25) \\ &= \quad \min_{\mu_{k_1,k_2}} \{g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + \left(J(x_{k_1,k_2}) - g_{k_1,k_2}(x_{i_1,i_2} + \mu_{i_1,i_2}(x_{i_1,i_2}))\right)^*\} \qquad (26) \end{aligned}$$

By inductive assumption, $J(x_{k_1,k_2}) - g_{k_1,k_2}(x_{i_1,i_2} + \mu_{i_1,i_2}(x_{i_1,i_2}))$ is optimal for the corresponding sub problem. Therefore, $\left(J(x_{k_1,k_2}) - g_{k_1,k_2}(x_{i_1,i_2} + \mu_{i_1,i_2}(x_{i_1,i_2}))\right)^* = J(x_{k_1,k_2}) - g_{k_1,k_2}(x_{i_1,i_2} + \mu_{i_1,i_2}(x_{i_1,i_2}))$. Finally, using Eqs. (23) and (25), we obtain

$$J^*(x_{k_1,k_2}) = \min_{\mu_{k_1,k_2}} \{g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + J(x_{k_1+1,k_2}) + J(x_{k_1,k_2+1}) - J(x_{k_1+1,k_2+2})\}. \tag{27}$$

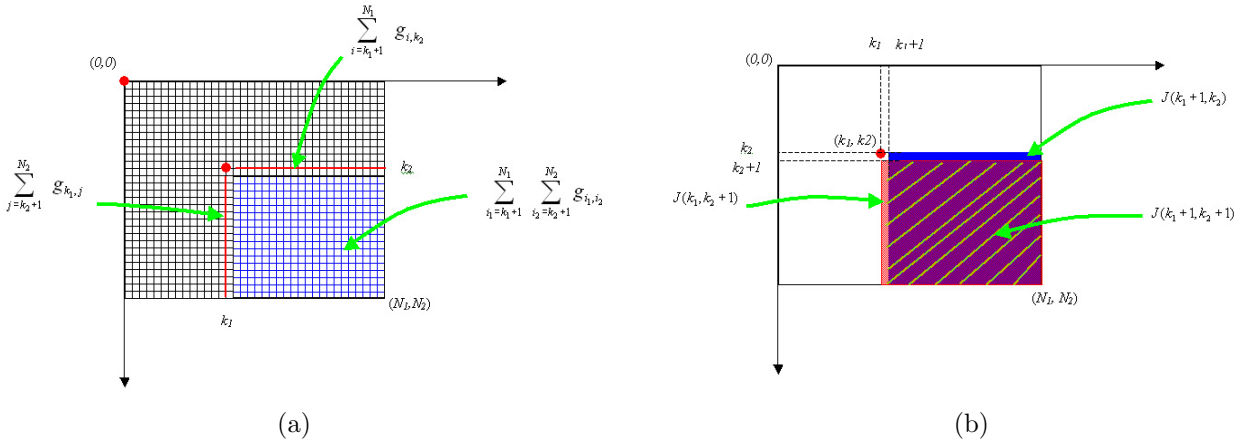

Figure 4. Graphical illustration of the summation terms involved in the proof of Theorem 1: (a) Illustration of the different summation terms in Eq. (21); (b) Illustration of the different cost functions in Eq. (23)

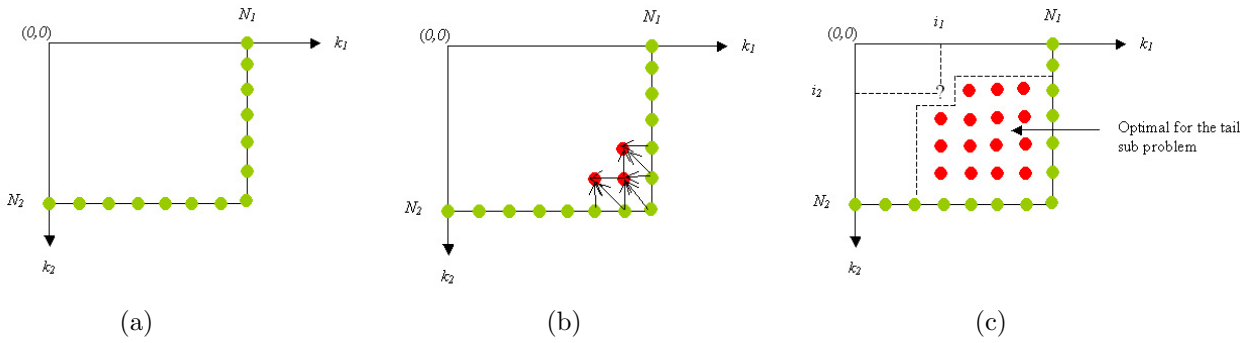

Figure 5. Illustration of the backpropagation process: (a) Initial optimal points; (b) Backpropagation from the initial optimal points; (c) The recurrence step: the area enclosed by the trapeze contains the optimal points for the corresponding sub problem, i.e., $J(x_{i_1+1,i_2}) + J(x_{i_1,i_2+1}) - J(x_{i_1+1,i_2+1})$ is optimal for the subproblem from point (i_1, i_2) onwards excluding the point (i_1, i_2). The goal is to determine the cost of (i_1, i_2), $J(x_{i_1,i_2})$.

We verify that Eq. (27) is exactly Eq. (17) for $n = 2$.

We now specialize the result of Theorem 1 to a subclass of semi-causal cost functions, called strictly semi-causal functions.

Definition 1. A discrete time n-D system is said to be strictly semi-causal if

$$x_{k_1+1,k_2+1,\cdots,k_n+1} = f(\{x_{k_1+1,k_2+1,\cdots,k_{i-1}+1,k_i,k_{i+1}+1,\cdots,k_n+1}, \mu_{k_1+1,k_2+1,\cdots,k_{i-1}+1,k_i,k_{i+1}+1,\cdots,k_n+1} : i = 1, \cdots, n\}). \tag{28}$$

For instance, a 2-D discrete time semi-causal system is defined as

$$x_{k_1+1,k_2+1} = f(x_{k_1,k_2+1}, x_{k_1+1,k_2}, \mu_{k_1,k_2+1}(x_{k_1,k_2+1}), \mu_{k_1+1,k_2}(x_{k_1+1,k_2})). \tag{29}$$

Figure 3(b) illustrates the strict semi-causality condition in 2-D spaces.

Corollary 2. (n-D Dynamic Tree Programming for strictly semi-causal systems)
Start with $J(x_{k_1,\cdots,k_{i-1},N_i,k_{i+1},\cdots,k_n}) = g_{k_1,\cdots,k_{i-1},N_i,k_{i+1},\cdots,k_n}(x_{k_1,\cdots,k_{i-1},N_i,k_{i+1},\cdots,k_n})$ for all $k_i = 1, \cdots, N_i$ and for all $i = 1, \cdots, n$ and go backwards using

$$\mu^*_{k_1,k_2,\cdots,k_n} = \underset{\mu_{k_1,k_2,\cdots,k_n}}{argmin} \left\{ g(x_{k_1,k_2,\cdots,k_n}, u_{k_1,k_2,\cdots,k_n}) + \sum_{i=1}^{n} J(x_{k_1,\cdots,k_{i-1},k_i+1,k_{i+1},\cdots,k_n}) \right\} \tag{30}$$

and
$$J^*(x_{k_1,k_2,\cdots,k_n}) = J(x_{k_1,k_2,\cdots,k_n}, \mu^*_{k_1,k_2,\cdots,k_n}(x_{k_1,k_2,\cdots,k_n})), \tag{31}$$

where J is the cost function inside the minimization brackets in Eq. (18) and $J(x_{k_1,k_2,\cdots,k_n}, \mu^*_{k_1,k_2,\cdots,k_n}(x_{k_1,k_2,\cdots,k_n}))$ is the cost function $J(x_{k_1,k_2,\cdots,k_n})$ evaluated at $\mu_{k_1,k_2,\cdots,k_n}(x_{k_1,k_2,\cdots,k_n}) = \mu^*_{k_1,k_2,\cdots,k_n}(x_{k_1,k_2,\cdots,k_n})$.

Corollary 3. (2-D Dynamic Tree Programming for strictly semi-causal systems)
Start with $J(x_{N_1,k_2}) = g_{N_1,k_2}(x_{N_1,k_2})$ for all $k_2 = 1, \cdots, N_2$ and $J(x_{k_1,N_2}) = g_{k_1,N_2}(x_{k_1,N_2})$ for all $k_1 = 1, \cdots, N_1$. Let $J(x_{k_1,k_2}) = g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + J(x_{k_1+1,k_2}) + J(x_{k_1,k_2+1}) - J(x_{k_1+1,k_2+1})$. Go backwards using

$$\mu^*_{k_1,k_2} = \underset{\mu_{k_1,k_2}}{argmin}\{g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + J(x_{k_1+1,k_2}) + J(x_{k_1,k_2+1})\}, \tag{32}$$

and
$$J^*(x_{k_1,k_2}) = J(x_{k_1,k_2}, \mu^*_{k_1,k_2}). \tag{33}$$

Proof of Corollary 2. It suffices to prove the result for the 2-D case. The n-D case follows then by induction. From Eq. (19), we have

$$\begin{aligned} J^*(x_{k_1,k_2}) &= \min_{\mu_{k_1,k_2}} J(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) && (34)\\ &= \min_{\mu_{k_1,k_2}} \{g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + J(f(x_{k_1,k_2}, x_{k_1,k_2-1}, \mu_{k_1,k_2}, \mu_{k_1,k_2-1})) && (35)\\ &+ J(f(x_{k_1,k_2}, x_{k_1-1,k_2}, \mu_{k_1,k_2}, \mu_{k_1-1,k_2})) - J(f(x_{k_1,k_2+1}, x_{k_1+1,k_2}, \mu_{k_1,k_2+1}, \mu_{k_1+1,k_2}))\}. \end{aligned}$$

$J(f(x_{k_1,k_2+1}, x_{k_1+1,k_2}, \mu_{k_1,k_2+1}, \mu_{k_1+1,k_2}))$ is independent of μ_{k_1,k_2}. Therefore, Eq. (34) becomes

$$\begin{aligned} J^*(x_{k_1,k_2}) &= \min_{\mu_{k_1,k_2}} \{g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + J(f(x_{k_1,k_2}, x_{k_1,k_2-1}, \mu_{k_1,k_2})) && (36)\\ &+ J(f(x_{k_1,k_2}, x_{k_1-1,k_2}, \mu_{k_1,k_2}))\} - J(f(x_{k_1,k_2+1}, x_{k_1+1,k_2})). \end{aligned}$$

Hence,

$$\mu^*_{k_1,k_2} = argmin\{g_{k_1,k_2}(x_{k_1,k_2}, \mu_{k_1,k_2}(x_{k_1,k_2})) + J(x_{k_1+1,k_2}) + J(x_{k_1,k_2+1})\}. \tag{37}$$

and

$$J^*(x_{k_1,k_2}) = J(x_{k_1,k_2}, \mu^*_{k_1,k_2}(x_{k_1,k_2})) \tag{38}$$

5. ACTIVE SURFACE OPTIMIZATION

From Eq. (9), we observe that the energy of index $\lambda_{\phi,t}$ depends only on the energies of $\lambda_{\phi-1,t}$ and $\lambda_{\phi,t-1}$. Therefore, we have a strictly semi-causal total energy function.

5.1. Dynamic tree programming

We first define the *reverse problem* as the original problem in which the direction of all arrows and paths are reversed. So, a reversed active surface is a collection of active contours from frame N to frame 1 (time is reversed) and in which the order of the normal lines ϕ is reversed, i.e., $\phi = M, \cdots, 1$.

The forward dynamic tree programming algorithm is the backward dynamic tree programming for the reverse active surface. The semi-causal definitions of all energy terms allows us to propagate the total energy using the forward dynamic tree programming algorithm and then back track the optimal surface. If the total energies $E(\lambda_{\phi-1,t})$ and $E(\lambda_{\phi,t-1})$ are known, then the total energy $E(\lambda_{\phi,t})$can be propagated to every index point on line ϕ as follows:

$$\begin{aligned} E(\lambda_{\phi,t}) \quad = \quad & \min_{(\lambda_{\phi-1,t},\lambda_{\phi,t-1})} \{E(\lambda_{\phi-1,t} + E(\lambda_{\phi,t-1}) + \alpha^s_{\text{int}}\ E^{\text{s}}_{\text{int}}(\lambda_{\phi-1,t}) + \alpha^t_{\text{ext}}\ E^{\text{t}}_{\text{ext}}(\lambda_{\phi,t-1}) + \alpha^t_{\text{int}}\ E^{\text{t}}_{\text{int}}(\lambda_{\phi,t-1})\} + \\ & \alpha^s_{\text{ext}}\ E^{\text{s}}_{\text{ext}}(\lambda_{\phi,t}) + \alpha^s_{\text{shape}}\ E^{\text{s}}_{\text{shape}}(\lambda_{\phi,t}). \end{aligned} \tag{39}$$

The initial conditions are set for time $t = 1$ and for normal lines $\phi = 1$ as follows:

- At time $t = 1$,
  - $E(\lambda_{1,1}) = \alpha^s_{\text{ext}}\ E^{\text{s}}_{\text{ext}}(\lambda_{1,1}) + \alpha^s_{\text{shape}}\ E^{\text{s}}_{\text{shape}}(\lambda_{1,1})$, for all $\lambda_{1,1} = -N, \cdots, N$;
  - For $\phi = 2, \cdots, M$,
    $E(\lambda_{\phi,1}) = \min_{(\lambda_{\phi-1,1})}\{E(\lambda_{\phi-1,1}) + \alpha^s_{\text{int}}\ E^{\text{s}}_{\text{int}}(\lambda_{\phi-1,1})\} + \alpha^s_{\text{ext}}\ E^{\text{s}}_{\text{ext}}(\lambda_{\phi,1}) + \alpha^s_{\text{shape}}\ E^{\text{s}}_{\text{shape}}(\lambda_{\phi,1})$.
- At normal line $\phi = 1$ and time $t = 2, \cdots, T$,
  $E(\lambda_{1,t}) = \min_{(\lambda_{1,t-1})}\{E(\lambda_{1,t-1}) + \alpha^t_{\text{ext}}\ E^{\text{t}}_{\text{ext}}(\lambda_{1,t-1}) + \alpha^t_{\text{int}}\ E^{\text{t}}_{\text{int}}(\lambda_{1,t-1})\} + \alpha^s_{\text{ext}}\ E^{\text{s}}_{\text{ext}}(\lambda_{1,t}) + \alpha^s_{\text{shape}}\ E^{\text{s}}_{\text{shape}}(\lambda_{1,t})$.

For each pixel index $\lambda_{\phi,t}$, record its spatial minimum argument, $\lambda_{\phi-1,t}$, and its temporal minimum argument, $\lambda_{\phi,t-1}$. After the energy is propagated to the last normal line in the last frame, we backtrack to find the best local active surface as follows:

1. In the last frame, T, let $c(M, t)$ be the pixel index with the minimum total energy on line $\phi = M$. Backtrack through the recorded spatial minimum arguments; that is only spatial information is used to find the last active contour because no temporal information is available at the last frame.
2. For frames $t = T - 1, \cdots, 1$,
   a. The optimal index point in the last normal M is the one that has the minimal energy $E(\lambda_{M,t})$.
   b. For $\phi = M - 1, \cdots, 1$, there are two candidates for the optimal index point: the minimum spatial argument of $c(\phi + 1, t)$ and the minimum temporal argument of $c(\phi, t + 1)$. Choose $c(\phi, t)$ to be the minimum of the spatial and temporal arguments.

6. SIMULATIONS

We use the 2-D dynamic tree programming algorithm to find the optimal active surface of the human head from cluttered office sequences. There are 499 frames in this sequence, with 30 frames per second. The tracking was performed using a particle filter described by Bouaynaya and Schonfeld in.[4] The tracker uses an ellipse as the parametric state of the human head. Our goal is to obtain an elastic contour delineating the head's boundaries. At each frame, the ellipse is the initial contour. We consider an active surface composed of 8 successive frames. We compare our simulation results with the 1-D randomly perturbed active contour model described in.[4] Figure 7 shows that the active surface yields a much more stable and less jittery active contours of the target in the video sequences.

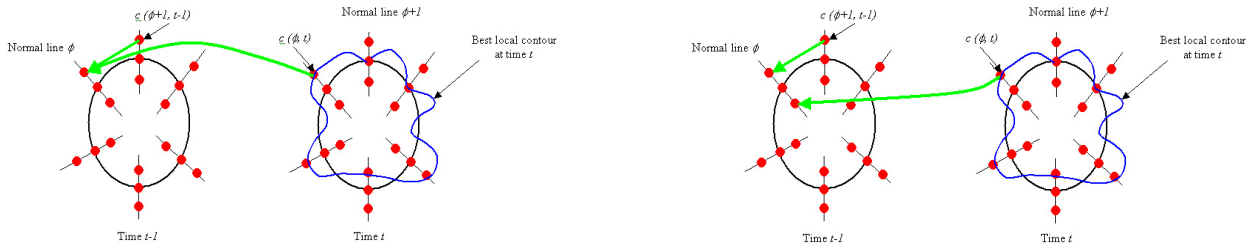

Figure 6. Illustration of the backpropagation process. We want to determine the optimal index point $c(\phi, t-1)$ at normal line ϕ at time $t-1$ given that all optimal index points $c(\varphi, \tau)$ are known for every $\varphi \geq \phi$ and every $\tau \geq t-1$. We distinguish 2 cases: (a) The minimum spatial argument of $c(\phi+1, t-1)$, $argmin_{spatial} E(\lambda_{\phi+1,t-1})$, and the minimum temporal argument of $c(\phi, t), argmin_{\text{temporal}} E(\lambda_{\phi,t})$, are equal. Then, set $c(\phi, t-1) = argmin_{\text{spatial}} E(\lambda_{\phi+1,t-1})$; (b) The minimum spatial argument of $c(\phi+1, t-1)$ and the minimum temporal argument of $c(\phi, t)$ are different. Then, set $c(\phi, t-1)$ to be the argument (spatial or temporal) with the minimal total energy.

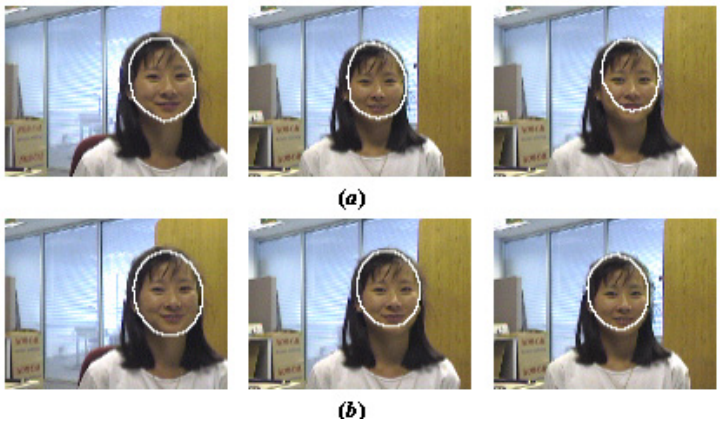

Figure 7. Object boundary delineation: (a) Active contours; (b) Active surfaces. (The video is courtesy of Stan Birchfield of Stanford University).

7. CONCLUSION AND FUTURE WORK

Using the concept of active surfaces, we have developed a general framework for tracking and 3-D segmentation based on a new method for multidimensional optimization. An active surface is a generalization of an active contour to higher dimensions. Active surfaces appear naturally in image processing applications if we imbed the problem in a higher dimensional space. For instance, adding the time dimension to a set of still images forms a video. Thus, the problem of p contour extraction from a video sequence composed of p frames can be seen as the problem of one surface extraction from a 3-D frame. We proposed the use of active surfaces to provide a stable and less jittery contour extraction from video sequences. Efficient implementation of the proposed active surface algorithm is provided by extension of dynamic programming to tree structures. We obtain a binary tree as opposed to a one-dimensional trellis. By selecting the branch that minimizes the total energy cost we generalized the Viterbi algorithm to minimization of semi-causal multidimensional cost functions. Future work can investigate the sensitivity of the active surface to the initial surface and to the spatial and temporal weight coefficients.

ACKNOWLEDGMENTS

The authors wish to express their gratitude to Sunil Peechara for conducting some of the simulations in the paper.

REFERENCES

1. B. Anderson and J. Moore, *Optimal Filtering*, Prentice Hall, New Jersy: Englewood Cliffs, 1979.
2. M. Isard and A. Blake, "Condensation conditional density propagation for visual tracking," *International Journal of Computer Vision* **29**(1), pp. 5–28, 1998.
3. E. Marchand and F. Chaumette, "Virtual visual servoing: a framework for real-time augmented reality," in *Proc. Eurographics*, p. 289298, 2002.
4. N. Bouaynaya and D. Schonfeld, "A complete system for head tracking using motion-based particle filter and randomly perturbed active contour," in *Proceedings of SPIE Image and Video Communications and Processing*, A. Said and J. G. Apostolopoulos, eds., **5685**, pp. 864–873, March 2005.
5. M. Kass, A. Witkin, , and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision* **1**(4), p. 321331, 1988.
6. D. Terzopoulos and R. Szeliski, "Tracking with kalman snakes," in *Active Vision*, A. Blake and A. Yuille, eds., p. 320, MIT Press, 1992.
7. C. Chesnaud, P. Refregier, and V. Boulet, "Statistical region snake-based segmentation adapted to different physical noise models," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**, p. 11451157, November 1999.
8. L. Cohen and I. Cohen, "Finite-element method for active contour models and balloons for 2-d and 3-d images," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**(11), pp. 1131–1147, 1993.
9. S. Gunn and M. Nixon, "Robust snake implementation: A dual active contour," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**, pp. 63–68, January 1997.
10. A. Amini, T. Weymouth, and R. Jain, "Using dynamic programming for solving variational problems in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**(9), p. 855867, 1990.
11. Storvik, "A bayesian approach to dynamic contours through stochastic sampling and simulated annealing," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **16**(10), pp. 976–986, 1994.
12. J. Denzler and H. Niemann, "Evaluating the performance of active contour models for real-time object tracking," in *Second Asian Conference on Computer Vision*, p. 341345, 1995.
13. F. Leymarie and M. Levine, "Tracking deformable objects in the plane using an active contour model," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **15**, pp. 617–634, June 1993.
14. Y. Akgul, C. Kambhamettu, and M. Stone, "Automatic extraction and tracking of the tongue contours," *IEEE Transactions on Medical Imaging* **18**, p. 10351045, October 1999.
15. V. Chalana and D. Linker, "Model for cardiac boundary detection on echocardiographic sequences," *IEEE Transactions on Medical Imaging* **15**, pp. 290–298, June 1996.
16. Y. Akgul, C. Kambhamettu, , and M. Stone, "A task-specific contour tracker for ultrasound," in *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis*, p. 135142, (South Carolina), 2000.
17. M. Li and C. Kambhmettu, "Motion-based post processing of deformable contours," in *Indian Conference on Computer Vision, Graphics and Image Processing*, December 2002.
18. Y. Chen, Y. Rui, and T. Huang, "Mode-based multi-hypothesis head tracking using parametric contours," in *Proc. Automatic Face and Gesture Recognition*, p. 112117, May 2002.
19. A. Doucet, "On sequential simulation-based methods for bayesian filtering," Tech. Rep. CUED/FINFENG/TR.310, Signal Processing Group, Department of Engineering, University of Cambridge,, CB2 1PZ Cambridge, 1998.
20. D. P. Bertsekas, *Dynamic Programming and Optimal Control: Vol. I*, Prentice-Hall, 2001.
21. D. P. Bertsekas, *Dynamic Programming and Optimal Control: Vol. II*, Prentice-Hall, 2005.