Video Foreground Segmentation Based on Sequential Feature Clustering

Mei Han  Wei Xu  Yihong Gong
NEC Laboratories America
Cupertino, CA, USA
meihan,xw,ygong@sv.nec-labs.com

Abstract

Segmentation of videos into layers of foreground objects and background has many important applications, such as video compression, human computer interaction, and motion analysis. Most existing methods work on image pixels or color segmentations which are computation expensive. Some methods require extensive manual input, static cameras, and/or rigid scenes. In this paper we propose a fully automatic segmentation method based on sequential clustering of sparse image features. The sparseness makes the method computation efficient. We use both edge and corner features to capture the outline of the foreground objects. Sequential linear regression is applied to the movement sequences of image features in order to compute the motion parameters for foreground objects and background layers, and consider the temporal smoothness simultaneously. Foreground layer is then extracted by a pyramidal Markov Random Field (MRF) model taking into account the spatial smoothness constraint. Experimental results on videos taken by webcams are shown and discussed.

1. Introduction

Segmentation of foreground objects from background has a lot of applications in video compression, human-computer interface, and object tracking. In order to get reliable and visually pleasant results, spatial and temporal information fusion is important. However, more information means heavy computation cost and/or extensive manual input which are not appropriate to be applied directly to cell phones, video conferences, etc. where the computation power is limited.

To date, there have been several research areas that are closely related to the task of foreground object segmentation. Video matting is a classic inverse problem in computer vision that involves the extraction of foreground objects and the alpha mattes that describe their opacity from image sequences [13, 1, 10]. Most of these methods require manual input. Some other matting methods assumed that a trimap was given by users which segmented the image into three regions: foreground, background and unknown [10, 3, 11].

Motion-based segmentation approaches perform motion estimations, and cluster pixels or color segments into regions of coherent motions. Many approaches formulated the problem as a Expectation-Maximization (EM) process to estimate the parametric motion models and the supporting regions [5, 2, 4]. There is also much work done to group pixels or segments into layers based on the affinity of local measurements [12, 6, 15].

However, spatial or temporal clues alone are not sufficient to distinguish objects because different objects may share similar colors or motions. Many approaches have been proposed to take advantage of the joint spatio-temporal segmentation [8, 7, 14]. Ke and Kanade [7] described a factorization method to perform rigid layer segmentation in a subspace because all the layers share the same camera motion. Wang and Ji [14] presented a dynamic conditional random field model to combine both intensity and motion cues to achieve segmentations. Most of these methods work on pixel or color segment level where the computation cost is high.

In this paper we propose a fully automatic segmentation method based on sequential clustering of sparse image features. The sparseness makes the method computation efficient. We work on both edge and corner features to capture the outline of the foreground objects. The feature clustering is built on motion models which work on any type of object and moving/static cameras. We assume there are two motion layers due to camera and/or foreground motion and the depth difference between the foreground and the background. Sequential linear regression is applied to the sequences of the instantaneous replacements of image features in order to compute the affine motion parameters for foreground and background layers, and consider the temporal smoothness simultaneously. The computation cost of this method is limited since it is based on clustering of sparse image features while most other methods work on pixels or color segments. Foreground layer is then extracted.
through a pyramidal Markov Random Field (MRF) model.

2. Sequential Feature Clustering

The foreground segmentation method is built on sparse features for the sake of computation cost. We assume there are only two layers: foreground and background. The sparse features are clustered into two classes based on their motion information. We firstly compute optical flows of the sparse features between consecutive frames, then we apply the linear regression technique to compute affine parameters of the two layers. To take advantage of the temporal information, we perform sequential linear regression on sequences of optical flow values to achieve more reliable and temporally smoother clustering results.

2.1. Sparse Features

We extract both corner and edge features to cover the areas which do not have good textures, but have clear outlines, such as human faces. The edge features provide information about the outline of the object but their optical flows have the foreshortening problem which we have to deal with in the linear regression computation.

We use canny edge detection to extract features from the image, as shown in Figure 1(b). The covariance matrix is computed for each feature to measure if the feature is edge or corner feature:

\[
\text{feature} \begin{cases} 
\text{edge,} & \text{if } \text{eig}_1 > \alpha \text{eig}_2 \text{ and } \text{eig}_2 < \beta \\
\text{corner,} & \text{otherwise} 
\end{cases}
\]

where eig1 and eig2 are the eigenvalues of the covariance matrix, and \( \alpha \) and \( \beta \) are parameters. We use Lucas and Kanade’s method [9] to compute the optical flow values of the features. For edge features we compute its normal direction \((dx, dy)\) from the covariance matrix and project its optical flow to this direction, i.e., we only keep the normal optical flow in affine parameter computation.

2.2. Linear Regression

Given a set of features and their optical flow values between two frames: \((\delta x_i, \delta y_i), i = 1, \ldots, n\) where \( n \) is the number of features, we apply linear regression technique to compute two sets of affine parameters, and classify the features to each set. The algorithm is summarized as follows:

1. Randomly cluster the features into two sets;
2. Compute the least square solutions of the affine parameters for each set of features, use the normal optical flow for edge features;

\[
\begin{align*}
\ a_l x_j + b_l y_j + c_l &= \delta x_j \\
\ d_l x_j + e_l y_j + f_l &= \delta y_j
\end{align*}
\]

where \( l \in \{1, 2\} \) denotes two layers, \( j \in \{1, \ldots, n\} \) and \((x_j, y_j) \in \text{Layer}_l\). Each edge feature only contributes one equation which is the dot product of its normal direction \((dx_j, dy_j)\) and its corresponding two equations.

3. Fit each feature into both affine motion models and compare the residuals;
4. Classify each feature to the affine model with smaller residual, if the smaller residual is above a threshold, it is put into a garbage set which would skip next iteration of computation;
5. Go back to step 2 until the clustering process converges, which means that none of the features would change assigning labels.

2.3. Sequential Clustering

We extend the feature clustering by linear regression between two frames to a few frames so that we can take advantage of the temporal consistence and achieve smoother and more reliable results. Since our feature clustering is based on affine motion models which works best when the camera is moving, and/or the foreground objects and the background objects have independent motion. This is not always true between two frames, but a few frames (such as 5-7 frames when the video frame rate is 6 frames per second) usually provide enough motion information to distinguish the foreground and background layers.

We incorporate the temporal information by performing linear regression on a few consecutive frames jointly. Given \( m \) consecutive frames, we solve \( 2(m - 1) \) affine parameters together where there are a pair of affine parameters to solve between two consecutive frames: \((a_{kl}, b_{kl}, c_{kl}, d_{kl}, e_{kl}, f_{kl}), k = 1, \ldots, m - 1\) to represent the affine motion between frame \( k \) to \( k + 1 \) and \( l \in \{1, 2\} \).
Figure 2. Feature clustering: (a) results of sequential feature clustering (using 3 frames), (b) clustering results without considering temporal constraints (simple linear regression between two frames).

denotes one of the two layers. The connection between the sets of parameters is built upon the feature correspondences which can be achieved through optical flow computation. When a new frame $k$ is available, corner/edge features $(x_i, y_i), i = 1, \cdots, n$ are detected first, then the optical flow $(\delta x_i, \delta y_i)$ between frame $k$ and $k - 1$ is computed for each feature. The corresponding feature in frame $k - 1$ for feature $i$ is searched over the features detected in frame $k - 1$ to find the closest one to the warped feature point $(x_i + \delta x_i, y_i + \delta y_i)$, and if the distance between the closest one and the warped one is below some threshold, the correspondence is established. Otherwise, the feature $i$ is labelled as “no match”. Connection is built for corresponding feature points who share the same layer label. The initialization label for feature $i$ is copied from the label of its corresponding point in frame $k - 1$. As for features with “no match”, the initialization label takes the label of its nearest neighbor in frame $k - 1$.

During the iterations of linear regression for each pair of frames, a joint residual is computed for corresponding features:

$$r_{ti} = \sum_{k=1, \cdots, m=1} r_{kli}$$

for feature $i$. Comparison of $r_{t1}$ and $r_{t2}$ would determine which layer feature $i$ belongs to. For “no match” points, the clustering is the same as in the simple linear regression algorithm between two frames.

The joint solution of sequences of linear regression problems naturally takes into account the temporal consistency which makes the clustering results more reliable and smoother. Figure 2 shows the results of sequential clustering (m=3) (Figure 2(a)) and independent clustering (only linear regression between two frames) (Figure 2(b)).

3. Foreground Extraction Using MRF

Foreground extraction is to get the dense output, i.e., layer labelling of each pixel given the sparse feature clustering. We first determine which layer is the foreground based on the following observations:

1. Foreground layer is closer to the camera, therefore, for most cases the affine parameters of the foreground layer have larger values. In our implementation, we only check the absolute values of the translation parameters: $|c_i| + |f_i|$. The larger this value, the more likely that the layer is foreground. However, special cases exist when the camera is following the foreground object where the foreground barely moves. We could either compensate by calculating the camera motion, which is time consuming, or we could let the other clues lead the decision.

2. Foreground layer is rarely cut into pieces, that is, the foreground layer is one or a few connected areas.

3. Background layer is scattered around the boundaries of the images.

4. If human exists in the foreground, most likely the foreground has more skin color pixels.

In Section 2, we generate two sparse motion layers using sparse feature points extracted from video frames. Because motion layer generation requires the most computational efforts within our foreground segmentation framework, our choice of performing this task in a sparse feature space dramatically reduces the computational cost.

Once the sparse motion layers are obtained, our next task is to turn them into dense motion layers where all the pixels in the frame get their labels. We adopt the Markov Random Field (MRF) model to propagate the labels provided by the sparse motion layers to the rest of the pixels in the frame. MRF, also called undirected graphical model, is commonly used in the statistical machine learning domain to model data sets that possess strong dependencies among individual data points. To utilize the model, we have to first define the following two model components: (1) construct a graph $G$ and a set of cliques $C(G)$ representing the dependencies among individual data points, (2) define a set of potential functions $\psi(c; \Lambda)$ over the cliques $c \in C(G)$ of the graph $G$, where $\Lambda$ is the parameter set of the MRF model.

Given the above model component definitions, the joint probability distribution over a frame $X$ and its label set $L$ is defined by:

$$P_{\Lambda}(X, L) = \frac{1}{Z_{\Lambda}} \prod_{c \in C(G)} \psi(c; \Lambda)$$

where $Z_{\Lambda} = \sum_{X, L} \prod_{c \in C(G)} \psi(c; \Lambda)$ is the partition function that normalizes the probability distribution. Generally, a potential function takes the form

$$\psi(c; \Lambda) = \exp[\lambda_c f(c)]$$
where \( f(c) \) is some real-valued feature function over clique values and \( \lambda_c \) is the weight given to that particular feature function. Substituting Eq.(5) back into Eq.(4), we have

\[
P_\Lambda(X,L) = \frac{1}{Z_\Lambda} \exp \left( \sum_{c \in C(G)} \lambda_c f(c) \right)
\]

In our MRF model, the graph \( G \) is constructed in such a way that each node in \( G \) corresponds to a pixel, each edge connects only two immediate neighboring nodes, and there are no other types of edges in the graph. Two types of cliques are defined on the graph: single node clique that is composed of each node \( i \) only, and 8-neighbor clique that is composed of eight neighbors \( N(i) \) of a node \( i \). Furthermore, each node \( i \) (i.e. pixel \( i \)) is represented by the following six attributes: (1) the coordinates \( X_i \) in frame \( X \), (2) the color triplet \( c_i \), (3) the feature point indicator \( e_i \) which equals one if pixel \( i \) is a feature point, but equals zero otherwise, (4) the normal direction \( d_i \) if pixel \( i \) is an edge feature point, (5) the regression residue \( r_i \) which is defined as follows:

\[
r_i = \begin{cases} 0, & \text{if } e_i = 0 \\ \|\nabla X_i - A_{li} \cdot [X_i 1]\|^2, & \text{otherwise} \end{cases}
\]

where \( l_i \) is the motion layer label for \( X_i \), \( \nabla X_i = [\delta x_i, \delta y_i]^T \) be \( X_i \)'s optical flow and,

\[
A_{li} = \begin{bmatrix} a_{li} & b_{li} & c_{li} \\ d_{li} & e_{li} & f_{li} \end{bmatrix}
\]

which represents the matrix of affine parameters for motion layer \( l_i \).

The definition of the potential functions \( \psi \) for graph \( G \) is the most important part for the MRF model design, because these potential functions play a very important role in determining the accuracy of our approximation to the true joint distribution. These functions can be thought of as compatibility functions. Therefore, a good potential function assigns high values to the clique setting that are highly compatible with each other under the given distribution.

With the above discussion in mind, we introduce three potential functions that strive to accomplish the following objectives:

1. Minimize the total regression residue of the two motion layers, which can be achieved by minimizing

\[
f_1(i) = r_i
\]

that is defined on single node cliques.

2. Assign the same label to the pixels with similar colors. This can be accomplished by minimizing the following potential function defined on 8-neighbor cliques.

\[
f_2(i) = \sum_{j \in N(i)} \delta(l_i, l_j) \exp(-\|c_i - c_j\|^2/\sigma^2)
\]

where \( \delta(l_i, l_j) \) is the delta function, and \( \sigma \) is a constant.

3. Assign the same label to a pair of neighboring edge points having the same normal direction that is perpendicular to the line connecting the two edge points. That can be realized by minimizing the following potential function defined on 8-neighbor cliques.

\[
f_3(i) = \sum_{j \in N(i)} \delta(l_i, l_j)e_i e_j \|d_{ij} \times d_{ij} \|d_{ij} \times d_{ij}
\]

where \( d_{ij} = \frac{X_i - X_j}{\|X_i - X_j\|}, d_i, d_j \) denote the normal edge directions at \( X_i \) and \( X_j \), and \( \times \) is the cross product between vectors.

Substituting the above potential functions into Eq.(6), we have

\[
P_\Lambda(X,L) = \frac{1}{Z_\Lambda} \exp \left( \sum_{c \in C} [\lambda_1 f_1(i) + \lambda_2 f_2(i) + \lambda_3 f_3(i)] \right)
\]

We use the Gibbs sampling method to iteratively obtain the label set \( L \) that maximizes the joint probability \( P_\Lambda(X,L) \). We obtain the weight set \( \Lambda \) through experiments, and set \( \lambda = 2, \lambda_2 = 3, \lambda_3 = 2 \) in our implementation.

4. Experimental Results

The proposed foreground segmentation method has been tested on simulated and real videos taken under different lighting and camera motions. In this section we show two examples captured by a lightweight Creative webcam. The resolution of the images is 640 by 480. Frame rate is 6 frames per second. The quality of the webcam is close to many cell phone video cameras. We allow the webcam move during taking videos and do not require either the foreground or the background is known or static.

The first sequence was taken of a rigid scene while the camera was moving. The scene is composed of a box of tapes which is closer to the camera as the foreground object, and a flat background. Due to the low quality and limited view angle of the webcam, the object was very close to the camera when the video was taken, therefore, there existed some distortions, as shown in Figure 3(a), which made feature tracking and motion modelling difficult.

Figure 3(b) demonstrate the foreground layer extracted by our method. Since we work on edge features, there exist some errors at the background edges where the error in optical flow is large which can be seen in the third frame of the results frames.
The second sequence was taken of a person moving and talking in front of the camera while holding the camera himself. The camera was shaking randomly with the person’s movement. Most features on the face were undergoing non-rigid motions. There were blurred areas in the video where feature tracking has large errors. However, since we work on sequential feature clustering, the temporally local blurring could be fixed over time. Figure 4(b) show the foreground layers extracted by our method.

References