A Subspace Approach to Layer Extraction, Patch-Based SFM, and Video Compression

Qifa Ke and Takeo Kanade

December 2001

CMU-CS-01-168

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

The author was supported in part by DiamondBack Inc.

The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Carnegie Mellon University or the U.S. Government.

Keywords: subspace, layer extraction, layered representation, structure from motion, patch-based SFM, video compression, video representation, motion segmentation

Abstract

Representing videos with layers has important applications such as video compression, motion analysis, 3D modeling and rendering. This thesis proposes a subspace approach to extracting layers from video by taking advantages of the fact that homographies induced by planar patches in the scene form a low dimensional linear subspace. In the subspace, layers in the input images are mapped onto well-defined clusters, and can be reliably identified by a standard clustering algorithm (e.g., mean-shift). Global optimality is achieved since both spatial and temporal redundancy are simultaneously taken into account, and noise can be effectively reduced by enforcing the subspace constraint. The existence of subspace also enables outlier detection, making the subspace computation robust. Based on the subspace constraint, we propose a patch-based scheme for affine structure from motion (SFM), which recovers the plane equation of each planar patch in the scene, as well as the camera epipolar geometry. We propose two approaches to patch-based SFM: (1) factorization approach; and (2) layer based approach. Patch-based SFM provides a compact video representation that can be used to construct a high quality texture map for each layer.

We plan to apply our approach to generating Video Object Planes (VOPs) defined by MPEG-4 standard. VOP generation is a critical but unspecified step in MPEG-4 standard. Our motion model for each VOP consists of a global planar motion and localized deformations, which has a closed-form solution. Our goals are: (1) combining different low level cues to model VOPs; and (2) extracting VOPs that undergo more complicated motion (non-planar or non-rigid).

1 Introduction

Decomposing an image sequence into layers has been proposed as an efficient video representation for coding, motion and scene analysis, and 3D scene representation [35, 23, 3]. There are two types of layers: 2D layer and 3D layer. A 2D layer consists of 2D sub-images such that pixels within the same layer share some common model. The model could be defined based on low level cues such as motion and texture. Motion model is often used (e.g., 2D parametric transformation [35], or non-parametric model defined by dense smooth flow field [37]). A 3D layer consists of a 3D plane equation, the texture of that plane, a per-pixel opacity map and depth-offset [3]. Extracting 3D layers usually requires the recovery of camera motion, which essentially reduces the problem to a structure from motion (SFM) task, a non-trivial task for computer vision. In fact,2D layers are suffice for the purpose of video compression. In the framework of MPEG-4 (content based video coding standard), a 2D layer is defined as a Video Object Plane (VOP).

In this thesis, we study the problem of 2D layer extraction from uncalibrated image sequence. The three major issues of layer extraction are: (1) determining the number of layers; (2) recovering the model for each layer; and (3) assigning pixels to layers. We first propose a subspace approach to extracting the layers based on 2D parametric motion model. As another application of the subspace approach, we propose a patch-based scheme for structure from motion (SFM). Then we propose a competition approach for extracting VOPs for MPEG-4 video coding. Since 2D parametric motion model might not suffice for modelling some VOPs with complex motion (non-rigid or articulate), we define a motion model with hierarchical complexity, and combine different low level cues to model the VOPs.

1.1 Subspace Approach to Layer Extraction

Various approaches have been proposed for layer extraction based on motion. They include mixture model estimation with Expectation-Maximization (EM) algorithm [20, 2, 38, 37, 32], and pixel or region grouping based on a certain affinity criterion using k-means algorithm [35] or normalized graph cut [27].

Initialization (setting the number of models and the motion for each model) is an important but difficult step for EM approach [27, 32]. Without good initialization, EM algorithm may not converge to desired optimal solutions. A typical initialization method [2] is to divide the image into a fixed number of tiles, and use them as the initial layers for the EM algorithm. Followed by each EM iteration is the application of MDL principle to determine the number of models, which is usually implemented as an exhaustive search [2]. The robust motion estimation in the M-step [2] requires the inclusion of dominant motion inside each initial or intermediate layer[1], which can not be guaranteed by the regular tiling initialization. Moreover, if one real layer is divided into different tiles, and if those tiles have different dominant motions (or without any dominant motion at all), such an unlucky layer becomes hard to be extracted.

Grouping pixels based on local measurement does not have the similar initialization difficulty. However, grouping based on pure local measurement ignores the global constraints. Moreover,

[1] The presence of dominant motion of the whole image is not required.

grouping in a high dimensional space is often unreliable given noisy local measurements.

In this thesis, we present a low dimensional linear subspace approach which can exploit the global spatial-temporal constraints. We formulate the layer extraction problem as clustering in the low dimensional subspace, where clusters become denser, better-defined, and thus more reliably identifiable.

Linear subspace constraints have been successfully used in computer vision. Tomasi and Kanade [31] are the first that used the rank-3 constraint in structure from motion (SFM). Shashua and Avidan [26] derived the linear subspace of planar homographies induced by multiple planes between a pairs of views. Zelnik-Manor and Irani [39, 40] extended the results to multiple planes across multiple views, and applied such constraints to estimate the homographies of small regions.

The subspace constraints to be exploited in this thesis are the ones that are derived from the relative affine transformations collected from homogeneous color regions [22]. Our algorithm assumes that each homogeneous color region is a planar patch. Such assumption is generally valid for images of natural scenes, and has been extensively used in motion analysis and stereo [5, 38, 13, 30].

Our subspace approach to layer extraction has the following advantages:

- Clusters in the subspace become denser and better-defined.
- As it is a multi-frame approach, both spatial and temporal global optimality are achieved by simultaneously taking into account all valid regions across the image sequence. Previously approaches in [34, 2] are essentially two-frame approach.
- Noise in estimated motion is reduced by the process of subspace projection, and global geometry constraint is enforced.
- The existence of subspace enables outlier detection, thus making the subspace computation robust.

1.2 Patch-Based SFM

We propose a patch-based scheme for affine structure from motion based on subspace constraints. When constructing the texture model for layers (e.g., the mosaic image), the recovered structure is useful if the scene is not strictly planar, which is often the case in the real world. For example, when the non-planar effect is compensated by the recovered structure, we can construct a higher quality mosaic image.

Previous approaches to SFM, either feature-based approach [31] or direct approach [19], have been focused on the recovery of the 3D position of each single point. In patch-based SFM, we recover the plane equation of each planar patch in the scene as well as the camera epipolar geometry. Two methods for patch-based SFM will be considered:

- The first method is based on the factorization of a measurement matrix consisting of 2D affine motion of each planar patch in the scene. Such method is a natural extension of [31], where the measurement matrix consists of 2D position of each feature point, i.e., the 2D motion model is translational.

- The second method is layer based. A cloud of correspondences are either computed or hallucinated [29] for each layer. Such approach has the advantage of avoiding the degenerate case (correspondences come from a single plane) or ill-posed case (most of the correspondences come from a single plane).

1.3 Competition Approach for VOP Generation

A VOP can not be extracted using the subspace approach when (1) it can not be approximated by a planar patch, or (2) its motion can not be modelled by 2D parametric motion. To deal with this problem, we propose a competition approach for VOP generation. In such framework, the model of the VOP could combine different low level cues, and its motion model can be more complicated than 2D parametric motion. Since some VOPs are simpler to model and easier to identify than others, we use a progressive scheme to extract the layers. Simple VOPs (such as the layers with 2D parametric motion model) are extracted first, which are then used to construct some intermediate background representation (such as a sprite) to guide the extraction of foreground VOPs that require more complex models.

2 Subspace of Relative Affine Homographies

This section shows that the homographies induced by 3D planar patches in a static scene, each represented by a column vector in the parameter space, reside in a low dimensional linear subspace. Such subspace comes from the fact that multiple planar patches in the scene share the common global camera geometry. The redundancy exploited by the subspace is high since there exists a large number of homogeneous color regions in real images, most of which can be approximated as planar patches.

2.1 Planar Homography

Suppose we are given a plane π [2] and two views of π. A point $\mathbf{X}_\pi$ on π is projected onto $\mathbf{x}$ and $\mathbf{x}'$ on the two views respectively, where $\mathbf{x}$ and $\mathbf{x}'$ are 3-vectors (homogeneous coordinates). There exist a unique non-singular 3×3 matrix $\mathbf{H}$ such that $\mathbf{x}' = \mathbf{H}\mathbf{x}$. This 3×3 matrix $\mathbf{H}$ is called the *homography* induced by the plane π. The explicit expression of $\mathbf{H}$ is given by the following theorem [17]:

Theorem 1 *Given a plane defined by* $\pi^T\mathbf{X} = 0$ *with* $\pi = (\mathbf{v}^\mathbf{T}, 1)$ *, and its two projective views with projection matrix* $\mathbf{P} \cong [\mathbf{I} \mid \mathbf{0}]$ *and* $\mathbf{P}' \cong [\mathbf{A} \mid -\mathbf{a}]$*, then the homography induced by plane* π *is* $\mathbf{x}' \cong \mathbf{H}\mathbf{x}$ *with* $\mathbf{H} \cong \mathbf{A} + \mathbf{a}\mathbf{v}^\mathbf{T}$

If we are given the fundamental matrix $\mathbf{F} = \mathbf{e}'_\times \mathbf{A}$ between two views, then we can choose the two cameras to be $[\mathbf{I} \mid \mathbf{0}]$ and $[\mathbf{A} \mid -\mathbf{e}']$. The homography induced by the 3D plane in the scene can then be described as [17]:

[2] Not containing the camera optical center.

$$\mathbf{H}_{3\times3} \cong \mathbf{A}_{3\times3} + \mathbf{e}'\mathbf{v}^T \tag{1}$$

Here $\mathbf{v} = (v_1, v_2, v_3)^T$ defines the 3D plane[3]. $[\mathbf{e}']_\times \mathbf{A} = \mathbf{F}$ is any decomposition of the fundamental matrix $\mathbf{F}$, where e' is the epipole in the second view and $\mathbf{A}$ is a homography matrix induced by *some* plane ([17], pp.316).

Given k planes in the scene, we have k homography matrices $\mathbf{H}_i, i = 1, 2, ..., k$. Suppose we construct a matrix $\mathbf{W}_{9\times k}$ by reshaping each $\mathbf{H}_i$ into a 9-dimensional column vector. The rank of $\mathbf{W}$ is known to be at most *four* [26]. In other words, all homographies between two projective views span a *four* dimensional linear subspace of $\Re^9$. This result was extended to the case of multiple projective views, and has been used to accurately estimate the homographies for small pre-segmented planar patches [39].

2.2 Relative Affine Homographies

Affine camera [24] is an important model usable in practice. One advantage of affine camera is that it does not require calibration. Moreover, when perspective effect is small or diminishes, using affine camera model avoids computing parameters that are inherently ill-conditioned [25, 16].

Eq.(1) holds for affine camera as well ([17], pp.350, or see our proof in the appendix). Given uncalibrated cameras, it is known that the projective homography can only be determined up to an unknown scale. This is not the case for affine cameras. In affine camera, the 2D affine transformation can be *uniquely* determined, and we can rewrite Eq.(1) as (see the proof in appendix):

$$\mathbf{m}_{2\times3} = \mathbf{m_r} + \mathbf{e}'\mathbf{v}^T. \tag{2}$$

Here $\mathbf{m_r}$ is the affine transformation induced by the reference plane. $\mathbf{e}' = (e_1, e_2)^T$, where $(e_1, e_2, 0)$ is the direction of epipolar lines in homogeneous coordinate in the second camera. A 3-vector $\mathbf{v}$ representing the plane is independent of the second affine camera.

Notice an important difference between Eq.(1) and (2). Eq.(1) has an unknown scale while Eq.(2) does not. Therefore, we can define *relative affine transformation* as:

$$\Delta\mathbf{m} = \mathbf{m} - \mathbf{m}_r = \mathbf{e}'\mathbf{v}^T. \tag{3}$$

where $\mathbf{m}_r$ is the affine transformation induced by the reference plane. The reference plane can be either a real plane or a virtual plane.

2.3 Subspace of Relative Affine Homographies

We will show that the collection of all relative affine transformations across more than two views resides in a three dimensional linear subspace:

Result 1 *Given a static scene with k planar patches, a reference view ψ_r and another $F(F \geq 1)$ views $\{\psi_f | f = 1, ..., F\}$ of this scene, the collection of all relative affine transformations induced by these k planar patches between the reference view ψ_r and any other view ψ_f resides in a three dimensional linear subspace.*

[3] We ignore the degenerate case where a plane is projected into a line in the image.

Proof: between reference view and view f, denote the k affine transformations as $\mathbf{m}_{f,1}, ..., \mathbf{m}_{f,k}$. From Eq.(2) we have $\Delta\mathbf{m}_{f,i} = \mathbf{m}_{f,i} - \mathbf{m}_{f,r} = \mathbf{e}'_f\mathbf{v}_i^T$, where $\mathbf{v}_i = [v_{1,i}, v_{2,i}, v_{3,i}]^T$. Reshape each $\Delta\mathbf{m}_i$ into a 6×1 column vector, and stack them into a matrix $\mathbf{W}^f_{6\times k}$. The following factorization is obvious [26]:

$$\begin{aligned}\mathbf{W}^f_{6\times k} &= \begin{bmatrix} e_{f,1} & 0 & 0 \\ 0 & e_{f,1} & 0 \\ 0 & 0 & e_{f,1} \\ e_{f,2} & 0 & 0 \\ 0 & e_{f,2} & 0 \\ 0 & 0 & e_{f,2} \end{bmatrix}_{6\times 3} * \begin{bmatrix} v_{1,1} & \cdots & v_{1,k} \\ v_{2,1} & \cdots & v_{2,k} \\ v_{3,1} & \cdots & v_{3,k} \end{bmatrix}_{3\times k} \\ &= \mathbf{E}^f_{6\times 3} * \mathbf{V}_{3\times k} \end{aligned} \tag{4}$$

where $\mathbf{V}$ is common to all views. Therefore, we have:

$$\mathbf{W}_{6F\times k} = \begin{bmatrix} \mathbf{W}^1 \\ \mathbf{W}^2 \\ \cdots \\ \mathbf{W}^F \end{bmatrix}_{6F\times k} = \begin{bmatrix} \mathbf{E}^1 \\ \mathbf{E}^2 \\ \cdots \\ \mathbf{E}^F \end{bmatrix}_{6F\times 3} * \mathbf{V}_{3\times k} \tag{5}$$

The matrix dimension on the right-hand side of Eq.(5) implies that the rank of $\mathbf{W}$ is at most 3. $\diamond$

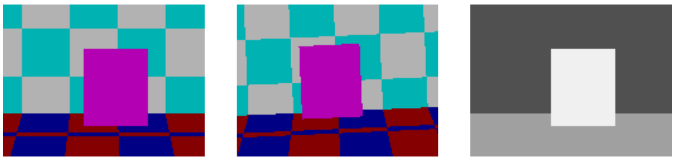
From Eq.(5) we can see that the subspace comes from the fact that multiple planes share the common camera geometry, i.e., the direction of parallel epipolar lines. The matrix $\mathbf{W}$ is built from the motions of planar patches. We can exploit high redundancy by using subspace since there exists a large number of homogeneous color regions in real images, many of which are planar patches. Multiple views have more redundancy. For the special *instantaneous* homography, it is known that there is a similar definition of relative projective homography and its subspace [40].

2.4 Subspace Dimensionality

The actual dimension of the subspace, i.e., the rank of W in Eq.(5), depends on the scene and the camera geometry, and could be *lower* than three. For example, if all planes in the scene are parallel to each other (not necessary front-parallel), or if there is only one plane in the scene, then the subspace dimension is *one* instead of three.

Another important fact is that the assumption of static scenes for deriving Eq.(5) is a sufficient condition but *not a necessary* one. This means that even with moving objects in the scene, we may still have a low dimensional linear subspace.

To verify the above observation, let us consider the following situations. A 3D scene consists of three planes, with the table plane stationary and foreground and background planes moving upward and downward independently. At the same time, a pinhole camera is undergoing simultaneously zooming out, translating horizontally, and rotating about its optical axis. Under such camera motion, each plane in the scene will induce an affine transformation. Fig.(1) shows the two rendered



(a) (b) (c)

449.0	261.2	0.006	0.0007	0.0002	0.0001

(d)

Figure 1: Results on synthetic sequence, where both camera and objects are moving independently: (a) and (b) two frames of the synthetic sequence; (c) the layer map by clustering in the 2-D subspace; (d) the eigenvalues of matrix $\mathbf{W}_{6\times31}$.

frames. Notice each plane is made of many color patches. With two views ($F = 1$), and $k = 31$ patches (1 on foreground plane, 15 on background plane, 15 on table plane), the eigenvalues of $\mathbf{W}_{6\times31}$ are computed and shown in Fig.(1d). They clearly show that the dimension of subspace is two.

2.5 Related Work

Linear subspace constraints have been successfully used in computer vision. Tomasi and Kanade [31] are the first that utilized the rank-3 constraint in Structure from Motion (SFM). Shashua and Avidan [26] derived the linear subspace of projective homographies induced by multiple planes between a pairs of views. Zelnik-Manor and Irani [39, 40] extended the results to multiple planes across multiple views, and applied such constraints to estimate the projective homographies of pre-segmented small regions.

There are several distinctions between our current work and previous work:

- In [39, 40], the subspace constraint is used to estimate the motion of *pre-segmented* regions that are manually specified. We use the subspace to provide a better space to cluster regions (e.g., homogeneous color segments) into layers. The high redundancy among larger number of regions are exploited even these regions come from the same layer/plane, given as few as two views of the scene.

- The approach to prove the existence of subspace in the previous work [39, 40] can not be directly applied to affine camera case, because affine camera has singular projection matrix (i.e., the up-left 3×3 matrix of projection matrix is rank 2). Affine camera is an important model that is widely used in practice. We proved the existence of 3-dimensional subspace for relative affine homographies.

- The affine camera motion can undergo large rotation, while in [40] the camera motion needs to be instantaneous (small rotation and forward translation).

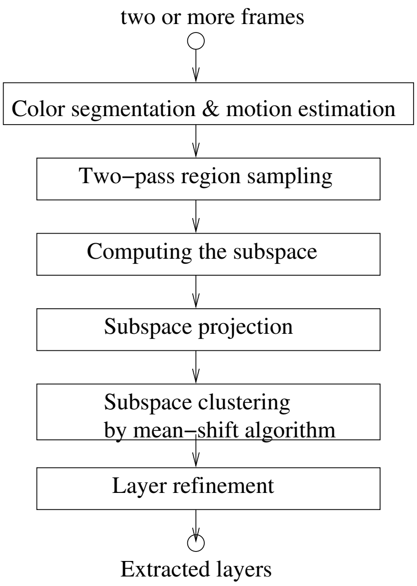


Figure 2: Overview of layer extraction algorithm.

3 Algorithm for Layer Extraction Using Subspace

Fig.(2) shows the steps of layer extraction algorithm. The input is two or more images, with one of them selected as the reference view frame. The reference image is segmented based on static color information. It is in general safer to over-segment, so that each segment corresponds to a single planar patch. Then an affine or translational motion is estimated with respect to each other frame for each color segment. Then the region sampling algorithm will select valid color segments, and the affine motions from these selected color segments are used to compute the linear subspace. Data points in the subspace are then generated by projecting the affine motion into the subspace. We use the mean-shift based clustering technique [7, 9] to derive the initial layers. Finally, the un-selected color segments are assigned to layers in the layer refinement step.

3.1 Color Segmentation and Motion Estimation

Our layer extraction algorithm assumes that pixels inside each color segment belong to the same layer, and the motion of each color segment can be described by a 2D parametric model, such as affine or projective homography[4]. We use the color segmentation algorithm proposed by [8]. Since color segmentation is not our final goal, over-segmentation has been used here in order to assure the validity of the above assumption to the largest extent. Such assumption is generally valid for over-segmented images of natural scenes, and has been successfully used in motion analysis and stereo [5, 38, 30].

For every color segment in the reference frame, we directly estimate a parametric motion using

[4] Note that color segmentation is applied only on the reference image. We directly estimate the motion of each region without doing region correspondence between reference image and other images.

a simple hierarchical model-based approach with robust estimation [4, 2, 5]. In our experiment, translational or affine model is estimated depending on the area support of each color segment.

Large color segments usually still have enough intensity variation to estimate affine motions. For a segment with little intensity variation, a translational motion can still be reliably estimated from the boundaries of color segment, if there is not occlusion.

3.2 Two-Pass Region Sampling

To derive the subspace, we must select regions to be used to build the matrix $\mathbf{W}$ in Eq.(5). Those regions must be the ones for which affine motions are estimated, and in general, they should uniformly distribute over the reference frame, so that each layer in the image domain will have enough samples and form a dense cluster in the feature space where clustering is performed.

A straightforward region sampling method is to divide the reference frame into small $n \times n$ blocks, and then select the blocks where an affine motion can be estimated [35]. Since affine motions are usually not available or erroneous in *small* textureless blocks, a layer containing large homogeneous color regions will not have enough number of samples to become a single dense cluster in the feature space. On the other hand, a layer with rich texture may have much more samples and the clustering algorithm may bias toward such layer.

To deal with the above problems while at the same time uniformly sample the reference image, we design a two-pass sampling approach based on color segmentation, as illustrated in Fig.(3). In the first pass, color segments for which affine motions have been estimated are selected as region samples[5]. The remaining un-selected areas are used in the second pass. Such remaining areas usually have rich texture and contain many small color segments where only translational motions are available. In the second pass, the reference image is divided into $n \times n$ blocks ($n = 20$ in our experiments). For each block containing more than 80% of un-selected pixels, we re-estimate an affine motion using the un-selected pixels inside this block. If the intensity residual of such estimated motion is small, the un-selected color segments inside such block are chosen as region samples.

3.3 Computing Subspace

Computing the subspace of homographies involves building and factorizing the matrix $\mathbf{W}$ in Eq.(5), which has been constructed from the affine transformations of the k selected region samples: $\mathbf{m}_i, i = 1, 2, ..., k$.

There are three important implementation details in building $\mathbf{W}$:

- We can choose one color region with large area support and good motion estimation as the reference plane. In practice, we found the average transformation $\overline{\mathbf{m}} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{m}_i$ serves as a good reference affine transformation induced by some "virtual" plane[6].

[5] A simple outliers detection is applied here. Regions with large registration error are considered as outliers.

[6] Notice that $\overline{\mathbf{m}}$ is induced by some world plane (either real or virtual) if and only if there exists $\mathbf{F} = [\mathbf{e}']_\times \overline{\mathbf{m}}$, where $\mathbf{F}$ is the fundamental matrix[17].

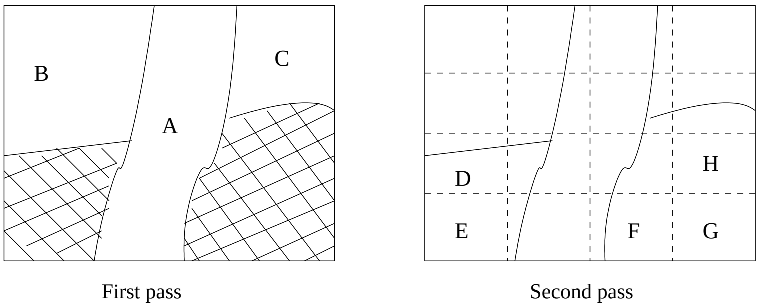


Figure 3: Two-pass sampling. Solid lines in the figures show the boundaries of color segments. In the first pass, color segments A, B, C are selected. The remaining color segments are small. In the second pass, the image is divided into $n \times n$ blocks. Blocks D-H are selected since they contain more than 80% of un-selected pixels. Affine motion for each selected block is estimated based only on the unselected pixels inside it.

- The area of each selected color segment is to be taken into account. For a selected color segment $\mathbf{m}_i$ containing n pixels, we reshape $\Delta\mathbf{m}_i$ into a 6×1 column vector, and then put n columns of $\Delta\mathbf{m}_i$ into $\mathbf{W}$[7]. In other words, regions with larger area have larger weights. Obviously adding such weight does not change the rank of $\mathbf{W}$.

- We scale the different components in the affine transformation, such that a unit distance along any component in the parameter space corresponds to approximately a unit displace at the image boundaries [35]. Such scaling makes the subspace approximately *isotropic*. We use the image width as the scale factor. Specifically, the matrix $\mathbf{W}_{6\times k}$ is left-multiplied by the following scale matrix:

$$\mathbf{S} = \begin{bmatrix} w & 0 & 0 & 0 & 0 & 0 \\ 0 & w & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & w & 0 & 0 \\ 0 & 0 & 0 & 0 & w & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

  Again, such linear transformation does not change the rank of $\mathbf{W}$, or the dimension of the subspace. Let us denote $\tilde{\mathbf{W}} = \mathbf{S}\mathbf{W}$. In practice, we found that $\mathbf{S}$ is not a sensitive parameter. The final results do not change for a wide range of the w in matrix $\mathbf{S}$.

We use SVD algorithm to factorize the matrix $\tilde{\mathbf{W}}$:

$$\tilde{\mathbf{W}}_{6\times k} = \mathbf{U}_{6\times 6}\mathbf{\Sigma}_{6\times 6}\mathbf{V}^T_{6\times k} \tag{6}$$

The diagonal of $\mathbf{\Sigma}$ contains the eigenvalues α_i of $\tilde{\mathbf{W}}$ in decreasing order. The actual rank of $\tilde{\mathbf{W}}$ depends on the camera and the planes in the scene, and is detected by [18]:

$$\frac{\sum_{i=0}^{d} \alpha_i^2}{\sum_{i=0}^{6} \alpha_i^2} > t \tag{7}$$

[7] If we do not use the average transformation $\overline{m}$ as reference, we need to subtract mean from each column.

where d is the rank of $\tilde{W}$, and t determines the noise level we want to tolerate.

The linear subspace is defined by the first d columns of U, which are the bases of the subspace. The motions of the region samples are projected into this subspace as $\Sigma_{d\times d}V_{d\times k}^T$, where each column becomes a feature point in the d-dimensional subspace.

3.4 Layer Initialization by Subspace Clustering

We now apply a clustering algorithm to the data points in the d-dimensional subspace for initial layers. The mean-shift based clustering algorithm, proposed by Commaniciu and Meer [8, 9], has been successfully applied to color segmentation and non-rigid object tracking [8, 10]. We adopt this algorithm because: (1) it is non-parametric and robust; (2) it can automatically derive the number of clusters and the cluster centers. Refer to [8, 9] for a clear description and details on this algorithm.

A critical parameter in this clustering algorithm is the window radius r of mean shift. This parameter determines the resolution of segmentation. We will show results over a range of r.

3.5 Layer Refinement & Post-Processing

Once we have the initial layers given by subspace clustering, we re-estimate an affine motion for each initial layer by using all of the region samples inside that layer. Then we re-assign every color segment[8] to the layer that predicts its motion best. This layer refinement is similar to one EM iteration in its goal, but without the probabilistic notion.

There are some spurious small regions, largely due to outliers. We have an optional post-processing step to remove such regions, by assigning them to their neighbors with similar motions. Such post-processing is desirable since a small number of compact layers are preferable for applications such as video compression.

3.6 Experimental Results

This section presents the experimental results of two real image sequences: *flower garden* and *mobile & calendar*.

There are two parameters that need to be specified. One is the noise level parameter t in Eq.(7) for determining the dimension of the subspace. In the following experiments, both sequences were found to have a two-dimensional subspace with $t = 95\%$. The other parameter is the window radius r. It is a critical parameter in the mean-shift based clustering algorithm. The value of this parameter can be derived from the covariance matrix of $\tilde{\mathbf{W}}$. According to [8], in our experiments it is to be set proportional to $\sigma = \sqrt{trace(cov(\tilde{\mathbf{W}}))}$. We have found by experiments that $r = 0.3\sigma$ produces the desired results. We will also show different layer extraction results by varying r over a wide range of $[0.3\sigma, 1.3\sigma]$.

[8] Including the color segments that are not selected in the two-pass region sampling step.

3.6.1 *Flower Garden* Sequence

Fig.(4a) and Fig.(4b) show two frames of the *flower garden* sequence, where the scene is static and the camera is translating approximately horizontally.

Fig.(4c) shows the color segmentation result on the reference image by applying the color segmentation algorithm with over-segmentation class proposed in [8]. Fig.(4d) shows the region samples selected by the two-pass sampling algorithm, and the initial layers via mean-shift clustering in the subspace. The black regions are un-selected regions. Notice that most of the occlusion regions are not selected, perhaps due to the two-pass sampling algorithm. Four layers (which roughly correspond to tree, branch, house, and flower bed) have been identified by the clustering algorithm, with window radius $r = 0.3\sigma_{garden}$, where $\sigma_{garden} = 4.5$. The tree layer and the branch layer contain large color segments and are easier to extract. Notice that the flower bed and the house consist of mostly small regions. The subspace clustering successfully identifies them as two separate layers.

Fig.(4e) shows the four layers after the layer refinement step but without post processing. Every initially unselected color segments has been assigned to one of the layers.

Fig.(4g-j) shows the four layers where the small spurious regions are assigned to neighbor regions based on motion affinity by the post processing step.

3.6.2 *Mobi* Sequence

The *mobile & calendar* sequence is used to show that static scene assumption in the analysis of Section 2 is a sufficient condition but *not a necessary one*. In this sequence, the train is pushing a rotating ball leftwards, and the calendar is pulled upwards, while camera is panning and tracking the train.

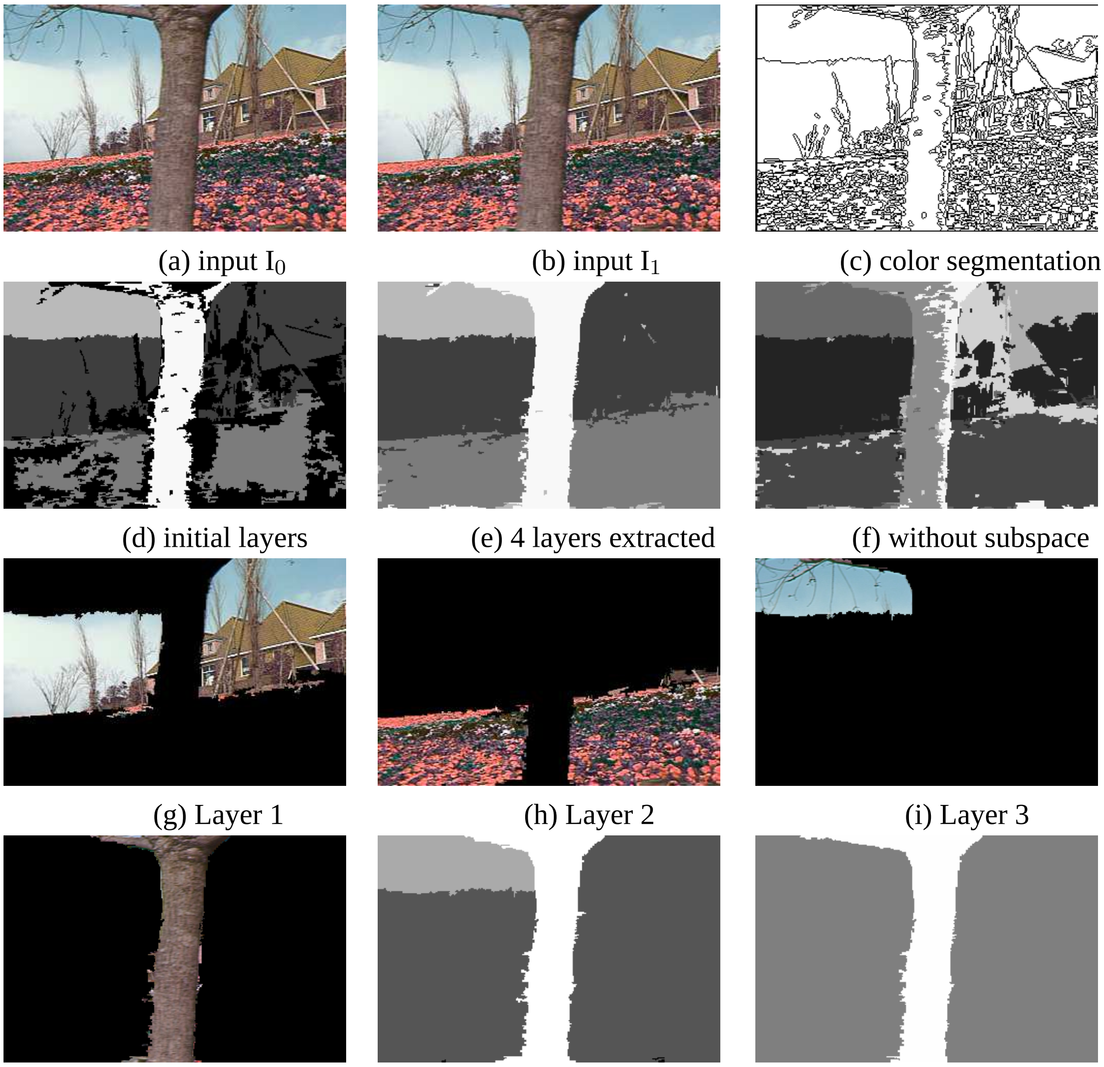
Fig.(5d) shows the region samples and initial layers by mean shift clustering with $r = 0.3\sigma_{mobi}$, where $\sigma_{mobi} = 3.2$. Again we notice that most of the occlusion regions are in the un-selected black regions. Fig.(5e) shows the result of layer refinement but without post processing. Note that the ball (in the lower middle) is extracted successfully. Although its area support is small, its motion is distinct and it forms a separate cluster in the subspace. In previous work of layer extraction on this sequence, for example in [2], the ball layer tends to be missed since its motion is not dominant in any initial or intermediate layer.

3.6.3 Increasing Window Radius

In this experiment, we vary the window radius r to see how the segmentations of different resolutions are derived. Fig.(4k) and (4m) show the layer maps obtained when increasing the window radius to $0.7\sigma_{garden}$ and $1.3\sigma_{garden}$ respectively[9]. Notice that in Fig.(4m), part of the branch layer is erroneously merged into the background layer. Fig.(5k) and (5m) are for *mobi* sequence.

The functionality of parameter r is similar to the "coding length" of MDL [2]. However, r is easier to understand and is more natural to set, in a way similar to the variance of Gaussian in [38].

[9] Further increasing r will eventually results in a layer map with only one layer in it.



(a) input I_0 (b) input I_1 (c) color segmentation

(d) initial layers (e) 4 layers extracted (f) without subspace

(g) Layer 1 (h) Layer 2 (i) Layer 3

(j) Layer 4 (k) (m)

Figure 4: Results of *flower garden* sequence. (a) and (b) Two frames of the sequence; (c) Color segmentation map; (d) Selected regions and initial layer map by clustering in the 2D subspace, where black indicates un-selected regions; (e) Layers after refinement; (f) Noisy layers extracted using the original six dimensional parameter space instead of subspace; (g)-(j) Four layers extracted after post-processing; (k) & (m) Layer maps by increasing the window radius of mean-shift algorithm.

3.6.4 Comparison with Clustering without Using Subspace

To demonstrate the advantages of using subspace, we also show the results of layer extraction without using subspace. To make the window radius comparable in both cases, we have scaled them by the following factor:

$$s = \frac{sqrt(\alpha_0^2 + \alpha_1^2 + ... + \alpha_5^2)}{sqrt(\alpha_0^2 + \alpha_1^2)} \tag{8}$$

where α_i's are the eigenvalues of $\tilde{\mathbf{W}}$.

Fig.(4f) and Fig.(5f) are the results of clustering in the original six-dimensional affine parameter space, with $r = s \times 0.3\sigma$. Some layers are split into two or more layers, possibly due to the fact that in the high dimensional space, the data are sparser and the cluster are not as well defined as in the low dimensional space. Also some regions are assigned to wrong layers.

3.6.5 Three-Frame Post-Processing

In some rare case, a color region may contain multiple motions. Such color region need to be broken into smaller elements to be re-assigned to different layers. Given the reference frame I_f and another two frames I_{f-1}, I_{f+1}, the following post-processing is applied:

- After the layers are derived, we re-compute affine motion for each layer [10].
- A pixel p is marked if its corresponding layer motion can not predict its color in either I_{f-1} or I_{f+1}.
- Re-assign each marked pixel to the layer that predicts its color best in **either** I_{f-1} **or** I_{f+1}.

Note that if a pixel is occluded in frame I_{f-1}, it is usually visible in the other frame I_{f+1}, since the movement of the camera tends to be continuous. Therefore, we assign the pixel to the layer whose motion predicts its color best in one temporal direction, even such layer can not predict its color in the reverse temporal direction.

Figure 6 shows the result of applying the above algorithm. As can be seen in Fig.(6d), part of the background is in the same color region of the tree and is assigned to the tree layer. Such error is corrected by applying the three-frame post-processing algorithm, as shown in Fig.(6e).

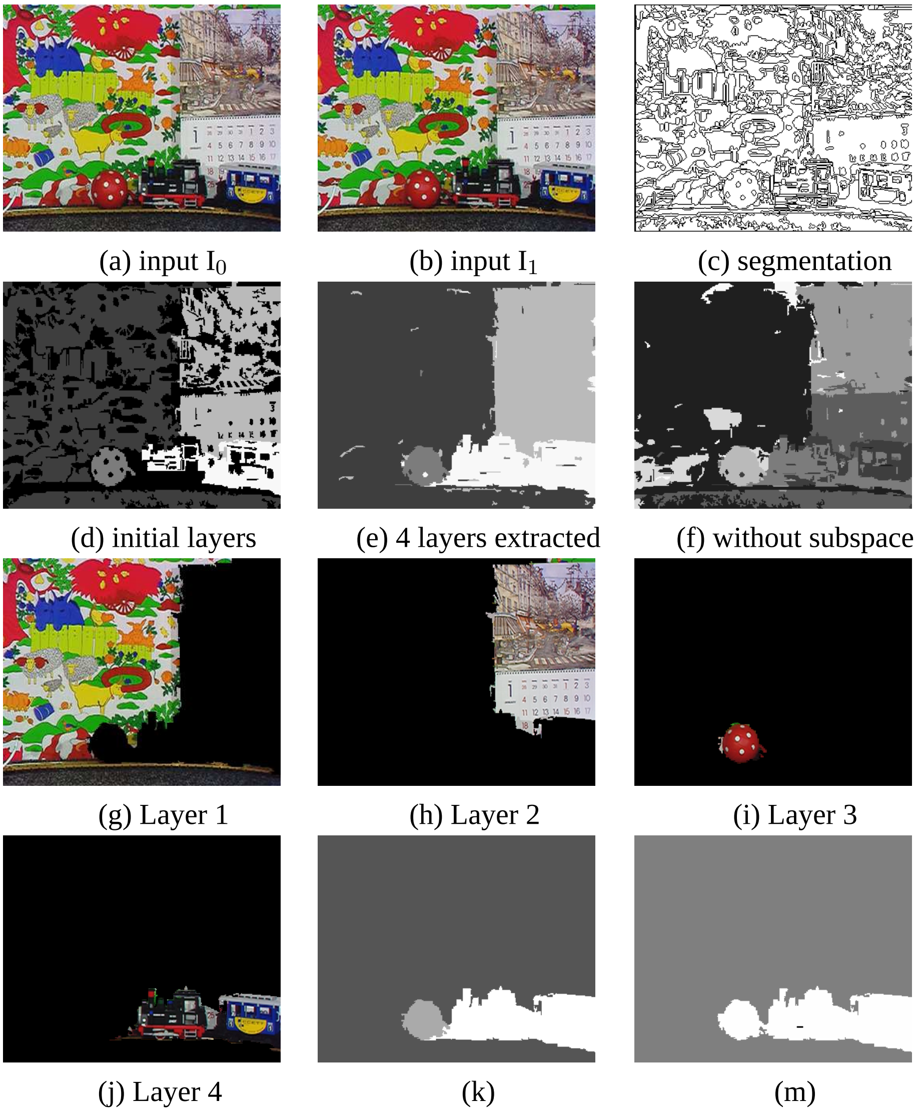
4 Robust Layer Extraction with Subspace

The subspace not only provides a space where data clusters are well defined but also provides a mechanism for outlier detection.

The SVD of the measurement matrix $\tilde{\mathbf{W}}$ is:

$$\tilde{\mathbf{W}}_{6F\times k} = \mathbf{U}_{6F\times 6}\mathbf{\Sigma}_{6\times 6}\mathbf{V}^T_{6\times k}$$

[10] We can compute a projective transformation now because we have enough area support at each layer.



(a) input I_0

(b) input I_1

(c) segmentation

(d) initial layers

(e) 4 layers extracted

(f) without subspace

(g) Layer 1

(h) Layer 2

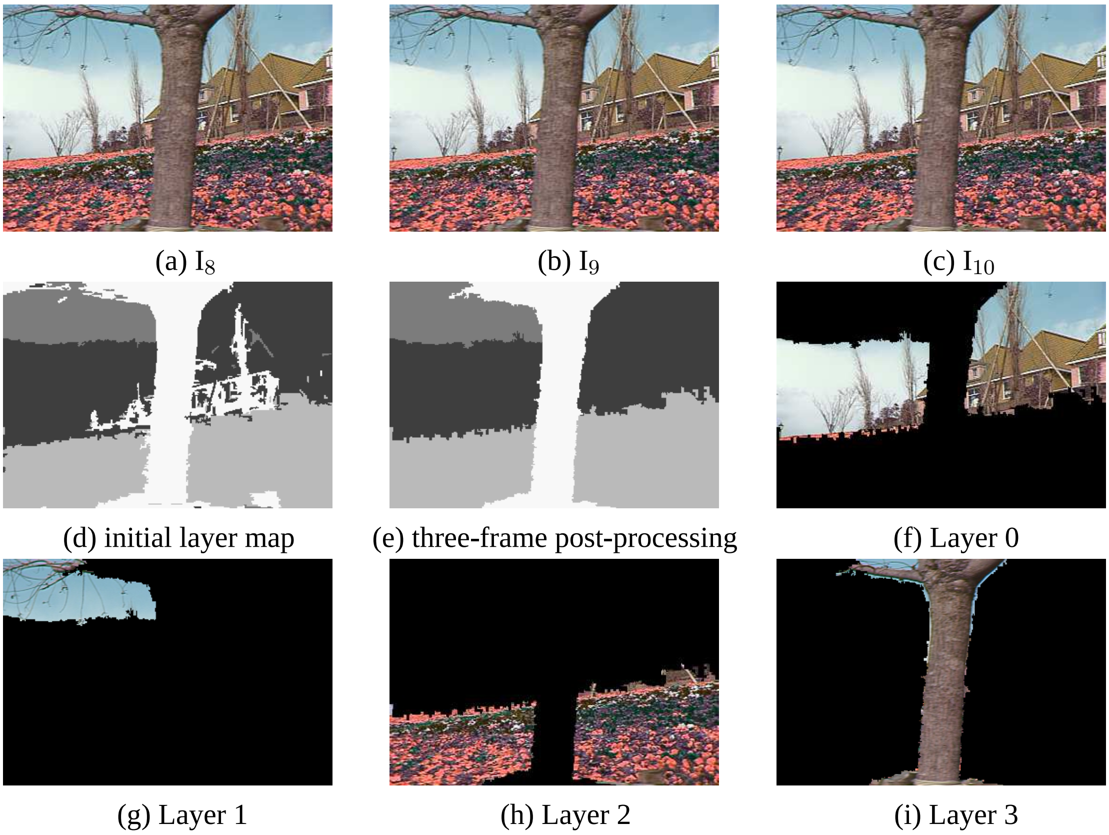
(i) Layer 3

(j) Layer 4

(k)

(m)

Figure 5: Results of *mobile & calendar* sequence. (a) and (b) Two frames of the sequence; (c) Color segmentation map; (d) Selected regions and initial layer map by clustering in the 2D subspace, where black indicates un-selected regions; (e) Layers after refinement; (f) Noisy layers extracted using the original six dimensional parameter space instead of subspace; (g)-(j) Four layers extracted after post-processing; (k) & (m) Layer maps by increasing the window radius of mean-shift algorithm.



(a) I_8

(b) I_9

(c) I_{10}

(d) initial layer map

(e) three-frame post-processing

(f) Layer 0

(g) Layer 1

(h) Layer 2

(i) Layer 3

Figure 6: Three-frame post-processing. (a)-(c) Three input frames, with (b) the reference frame; (d) initial layer map by algorithm in Section 3; (e) layer map with three-frame post-processing; (f)-(i) four layers extracted

The first d columns of $\mathbf{U}$ are the bases of the d-dimensional subspace S. The last three columns form the bases of the residual space, which is orthogonal to S, and is denoted as $S^{\perp}$. The first three columns of $\mathbf{\Sigma}_{6\times 6}\mathbf{V}^T$ are the projections of $\tilde{\mathbf{W}}$ onto the subspace S, while its last three columns are the projections of $\tilde{\mathbf{W}}$ onto the residual space $S^{\perp}$.

There are two kinds of outliers:

- data with extreme values that inflate the covariance matrix of $\tilde{\mathbf{W}}$.
- data that can not be represented by the subspace S, and have large projection values in the residual space $S^{\perp}$.

The detection of outliers is based on the Mahalanobis distance d_i^2 of the i-th data point m_i:

$$d_i^2 = (\mathbf{m_i} - \overline{\mathbf{m}})^T \mathbf{S}^{-1} (\mathbf{m_i} - \overline{\mathbf{m}})$$

where $\mathbf{S}$ is the covariance matrix of $\tilde{\mathbf{W}}$.

Under the assumption that data are sampled from an underlying elliptic normal distribution with covariance matrix $\mathbf{S}$, d_i^2 follows χ^2 distribution with N degrees of freedom [21].

Since

$$\begin{aligned} \mathbf{S}^{-1} &= \mathbf{U}\Sigma^{-2}\mathbf{U}^T \\ \mathbf{m_i} - \overline{\mathbf{m}} &= \mathbf{U}\Sigma\mathbf{v_i} \\ \overline{\mathbf{m}} &= \mathbf{U}\Sigma\overline{\mathbf{v}} \end{aligned}$$

we have:

$$d_i^2 = \sum_{p=1}^{6} v_{i,p}^2$$

All data samples with d_i^2 lie outside the p-th percentage point of the corresponding χ_N^2 distribution ($N = 6$ is the degrees of freedom) are marked as outliers.

A problem with the above measurement is that it may not give enough weight to the last three bases, which usually identify the outliers that violate the correlation structure imposed by the bulk of data, but not necessarily inflate the covariance matrix. For this reason, we also look at the residual space $S^{\perp}$:

$$o_i^2 = \sum_{p=d+1}^{6} v_{i,p}^2$$

where d is the rank of $\tilde{\mathbf{W}}$; and o_i^2 follows χ_N^2 distribution [14], with degrees of freedom $N = 3$.

Our algorithm for robust subspace computation consists of the following steps:

- Step 1: Use SVD to compute initial subspace.
- Step 2: Compute d_i^2 and o_i^2 for each region. Mark regions whose d_i^2 and o_i^2 are outside the p-th confidence interval of χ^2 distribution as outliers.

- Step 3: Use weighted data to recompute the subspace. The weights vary between 0 and 1 according to d_i^2 and o_i^2. The weight for outliers is 0.

- Step 4: Repeat Step 2 and 3 until the set of inlier stabilizes.

Note that an initial “outlier” may become an inlier later because the subspace is changing during the iterations. We do not use RPCA proposed in [33], where the number of training images is usually much smaller than the dimension of the original space (which is equal to the number of pixels in one image). In such case, weighting or discarding a datum (the whole image) is a big deal. Therefore, they weight each pixel across different images. In our case, the number of measurements is much larger than six, the dimension of original space. We can apply a single weight to each measurement, which is easier and more appropriate for our case.

Computing SVD of a large matrix $\tilde{\mathbf{W}}$ is expensive. In Section 3.3, for each region consisting of N pixels, we put N identical columns of data into $\tilde{\mathbf{W}}$. Each column contains six parameters of the corresponding affine transformation. To reduce the dimension of $\tilde{\mathbf{W}}$, we can just put one column in $\tilde{\mathbf{W}}$, but weigh this column by $\sqrt{N}$. The SVD of the weighted $\tilde{\mathbf{W}}$ will results in same subspace as in Section 3.3. This simple trick greatly reduces the dimension of matrix $\tilde{\mathbf{W}}$, and makes the iteration in robust subspace computation feasible.

Figure 7 shows an example of robust subspace computation. We can see that regions or patches containing few texture or dis-continuous motion are detected as outliers.

4.1 RANSAC

RANSAC is a general and successful robust method. The adaption of RANSAC[11] based on the above algorithm is shown in the following:

- Step 1: Randomly select p samples from the set S of regions and patches, and instantiate the subspace from this subset.

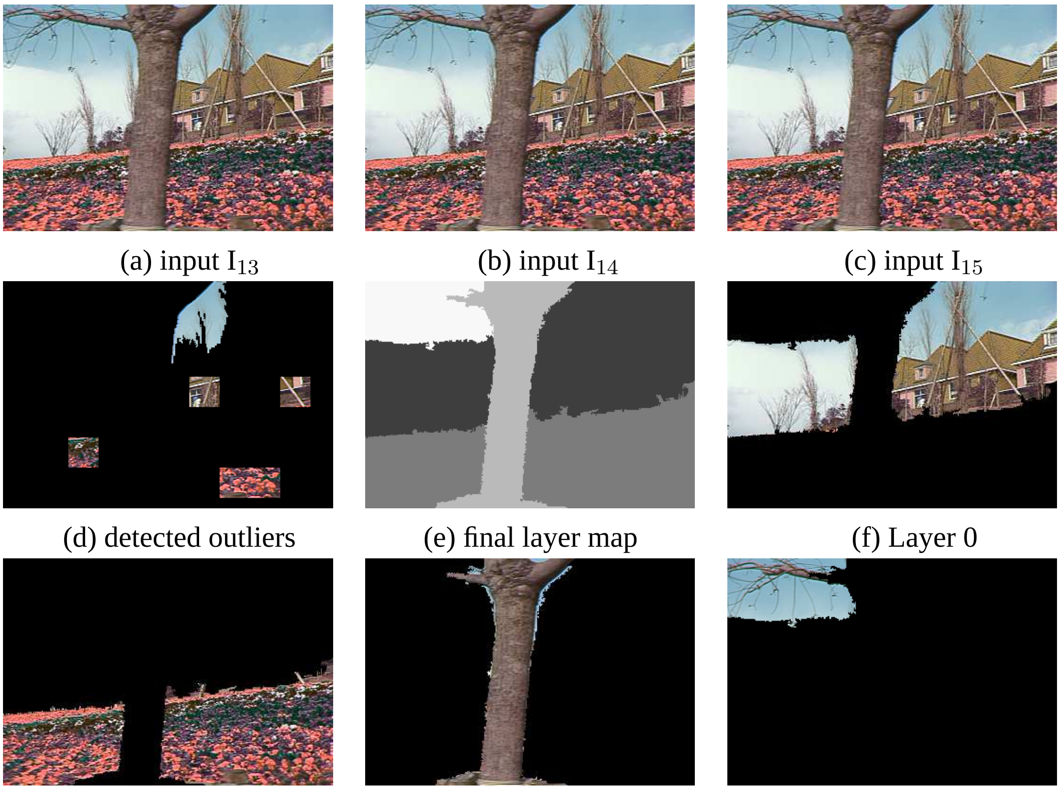
- Step 2: Determine the set of data points S_i which are inlier of this the subspace. A datum point is classified as an inlier if $o_i^2 < t$, where t is the χ_N^2 value corresponding to confidence interval α.

- Step 3: If the number of inliers is larger than a threshold T, recompute the subspace using the inliers, and terminate.

- Step 4: If the number of inliers is less than T, goto Step 1.

- Step 5: After N trials the largest consensus set S_i is selected and the subspace is recomputed using S_i.

The parameters in the RANSAC algorithm are: N, T, p, and t.



(a) input I_{13} (b) input I_{14} (c) input I_{15}

(d) detected outliers (e) final layer map (f) Layer 0

(g) Layer 1 (h) Layer 2 (i) Layer 3

Figure 7: Robust subspace computation. (a)-(c) input frames with (b) the reference frame; (d) detected outliers; (e) final layer map; (f)-(i) four layers extracted.

5 Patch-Based SFM

Previous approaches to Structure from Motion (SFM), either the classical feature-based or direct approach (plane+parallax), have been focusing on the recovery of the 3D positions of points in the scene. We propose patch-based SFM, which simultaneously recovers the plane equation [11] of each planar patch in the scene as well as the global camera epipolar geometry.

There are several motivations for using patch-based SFM:

- Patch-based representation is dense and thus more appropriate for video compression than point-based representation. When constructing the texture model for layers (e.g., the mosaic image), the recovered structure is useful if the scene is not strictly planar, which is often the case in the real world. For example, when the non-planar effect is compensated by the recovered structure, we can construct a higher quality mosaic image.
- It can be classified as "direct" approach. There exist some regions with enough texture variation to compute an affine motion, but without any obvious feature points.

The estimated homography for each region is noisy, especially when the region is small and the texture is not rich enough. We need to utilize both spatial and temporal redundancy to make the algorithm robust to noise. We propose two approach to patch-based SFM: factorization approach and layer-based approach.

5.1 Approach 1: Factorization

This method is based on the factorization of a measurement matrix consisting of relative 2D affine motions of planar patches in the scene. Such method is a natural extension of [31], where the measurement matrix consists of 2D position of each feature point.

Factorization approach can use multiples frames in a batch way to compute epipoles and plane equation simultaneously. From Section 2, we know that each column in the measurement matrix $\mathbf{W}^f_{6\times k}$ is the relative affine motion of the k-th planar patch, between the reference frame and Frame f. By applying SVD to $\mathbf{W}$, we have the following decomposition in multi-frame case:

$$\mathbf{W}_{6F\times k} = \begin{bmatrix} \mathbf{W}^1 \\ \mathbf{W}^2 \\ \ldots \\ \mathbf{W}^F \end{bmatrix}_{6F\times k} = \mathbf{U}_{6F\times 3}\mathbf{\Sigma}_{3\times 3}\mathbf{V}_{3\times k} = \begin{bmatrix} \hat{\mathbf{E}}^1 \\ \hat{\mathbf{E}}^2 \\ \ldots \\ \hat{\mathbf{E}}^F \end{bmatrix}_{6F\times 3} * \hat{\mathbf{V}}_{3\times k} = \hat{\mathbf{E}} * \hat{\mathbf{V}}$$

For any non-singular matrix $\mathbf{Q}_{3\times 3}$, $\hat{\mathbf{E}}\mathbf{Q}$ and $\mathbf{Q}^{\mathbf{-1}}\hat{\mathbf{V}}$ is also a valid factorization of $\mathbf{W}$. We need to find the matrix $\mathbf{Q_{3\times 3}}$ such that $\mathbf{E}^f = \hat{\mathbf{E}}^f\mathbf{Q}$ has the following standard form:

$$\begin{bmatrix} \hat{\mathbf{E}}^1 \\ \hat{\mathbf{E}}^2 \\ \ldots \\ \hat{\mathbf{E}}^F \end{bmatrix}_{6F\times 3} * \mathbf{Q}_{3\times 3} = \begin{bmatrix} \mathbf{B^1} \\ \mathbf{B^2} \\ \ldots \\ \mathbf{B^F} \end{bmatrix}$$

[11] More precisely, we recover the three parameters that describe the plane with respective to the reference plane.

where $\mathbf{B}_{6\times3}^{f} = \left[e_1^f \mathbf{I}_{3\times3}, e_2^f \mathbf{I}_{3\times3}\right]^T$, with $\mathbf{I}$ the identity matrix and $\left[e_1^f, e_2^f\right]$ the epipolar line in the f-th frame. The epipoles are up to an unknown scale. Without loss of generality, we can set $e_1^1 = 1$ or $e_2^1 = 1$ to fix the scale. The above equation set is over-constrained and is linear in $\mathbf{Q}$, which can thus be solved by linear least-squares method.

5.2 Approach 2: Layer-Based Approach

If we are given two or more layers, we can use them to compute the epipolar geometry, and then derive the plane equation for each planar patch in the scene using Equation (5). For each layer, we can either hallucinate some point correspondences inside its support according to its affine motion, as suggested in [29]. Or we can compute real correspondence by dividing the layer into $n \times n$ patches, refine the patch alignment based on the layer affine motion, and then pick up the center of the patches as the corresponding feature points. The epipolar geometry can be solved by the point correspondences using the traditional approaches in [25].

The advantages of using layer-based approach to compute epipolar geometry are:

- Avoid degenerate case where points comes from a single plane.
- We can uniformly distribute points across different planes. In traditional approach, points are usually selected according to texture, which might favor some plane with rich texture, and results in ill-posed configuration, i.e., most points are from the same single plane with rich textures and only a small percentage of the points are from different planes.

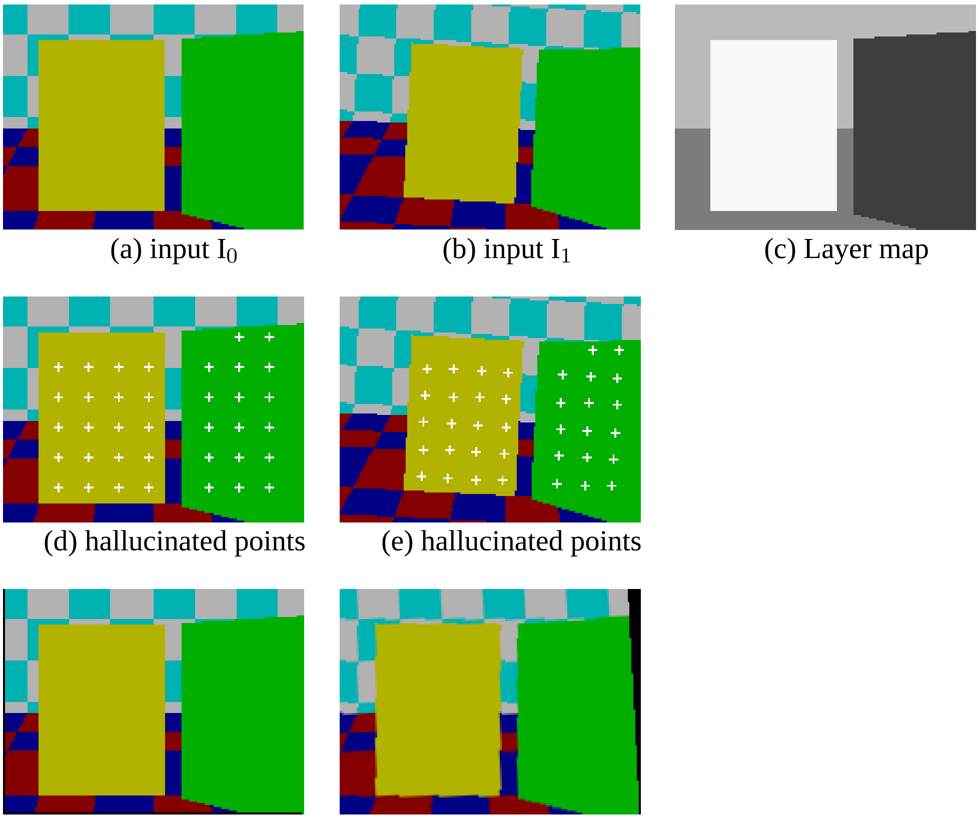
5.3 Projective SFM Using Subspace Constraints

If we scale the projective transformations associated with each planar patch, then we have a linear algorithm for projective SFM, which recovers the camera geometry and plane equations.

5.4 Experiments with Synthetic Data

We have verified the above two approaches in two-frame case with synthetic data. Fig (8) shows the results using layer-based approach. Fig (8a) is the reference image. In Fig (8b), the camera is rotated $\theta = 3^\circ$ degrees around its optical axis, zoomed out by a factor of $s = 0.9$, and translated to the left by $0.2f$, where f is the focal length. Fig (8d, 8e) shows the hallucinated point correspondences on two of the layers. The epipolar equation derived from the correspondences is: $ax + by + cx' + dy' + e = 0$, with $(a, b, c, d, e) = (0, 0.6689, 0.0389, -0.7422, 1.213)$. The direction of the epipolar line computed from these correspondences are $(0, 1)$ and $(0.0523, -0.9986)$. The rotation angle and scale factor derived from the epipolar equation are $\theta' = 3^\circ$, and $s' = 0.9$, which verifies the correctness of the computed epipolar equation. We show the rectified images according to the derived epipolar equation. We can see that the rectified images have only horizontal parallax.

For the same data, we test the factorization approach and get the same results.



(a) input I_0

(b) input I_1

(c) Layer map

(d) hallucinated points

(e) hallucinated points

(f) rectified I_0

(f) rectified I_1

Figure 8: Layer-based approach. (a) and (b) two synthetic image; (c) layers extracted; (d) and (e) hallucinated point correspondence on two layers; (f) and (g) I_0 and I_1 rectified by the derived epipolar equation.

6 A Competition Approach for Video Object Plane Generation

In this section we introduce the *Video Object Plane* (VOP) defined by MPEG-4 standard. We classify the VOPs into background VOPs and foreground VOPs. We then introduce the competition framework for the extraction of both background and foreground VOPs.

6.1 VOPs in MPEG-4

MPEG-4, an ISO/IEC standard developed by MPEG committee, is an object-based video coding standard. In MPEG-4, the scene is viewed as a composition of some *video objects* (VO) with intrinsic properties such as texture, shape and motion. A *video object plane* (VOP) is formed by projecting a video object in the scene onto the image plane. A video frame is thus composed by layers of VOPs. Each VOP is encoded/decoded independently. Four types of VOPs are defined in MPEG-4 according to how the VOPs are coded:

- Intra-coded VOP (I-VOP): coded using information only from itself.
- Predictive-coded VOP (P-VOP): coded using motion compensation from a past reference VOP.
- Bidirectionally predictive-coded VOP (B-VOP): coded using motion compensation from a past and/or future reference VOP(s).
- Sprite VOP (S-VOP): A VOP for a sprite object, or a VOP coded using prediction based on global motion compensation from a past reference VOP. A sprite is a large object accumulated from pixels in many frames of a video segment. For example, in the *mobile calendar* video segment, the mosaic image of the background (wall) layers forms a sprite.

We should notice that the above VOPs are defined according to their coding method. In general, the definition of semantically meaningful VOPs is vague. Moreover, the extraction of such VOPs from videos of natural scene, a hard problem that has attracted many researchers, is not specified in MPEG-4.

6.2 Sprite VOPs

When the scene is static and the camera optical center does not move (only pan/tilt/zoom), a single Sprite VOP can be used to represent the static scene. In more general cases, each rigid 3D plane in the scene can be represented by a sprite, no matter how the camera or the plane moves.

Two types of sprites are used in sprite coding: (1) off-line static sprites, and (2) on-line dynamic sprites.

Off-line sprites, sometimes called static sprites, are built off-line prior to encoding. A classical example is the mosaic of background images taken by a pan-tilt camera. Such sprite is formed by warping and then blending corresponding VOPs in the video frames under a reference view. The motion between the sprite and the VOPs in each frame is described by global 2D parametric transformations, such as affine or projective transformations. In the decoder, such sprites are warped and cropped to form the background VOP in each video frame. Since off-line sprites need to be transmitted to the receiver only once, the transmitted bit rate is reduced enormously. Note that a VOP created from an off-line sprite does not have an associated residual error signal.

On-line sprites are generated on-line during coding both in the encoder side and the decoder side. In the encoder side, the existing sprite is warped to the current frame using the global motion estimated between successive VOPs. Then the VOPs from the current frame are blended into the warped sprite. On-line sprite is dynamically updated such that the reference view of the sprite is always the view of current frame, and is therefore also called dynamic sprite. In the decoder side, dynamic sprites are used for predictive coding.

6.3 Background and Foreground VOPs

In many real video sequences, the background scene can be approximated by one or more rigid planes [12]. Their motions between two video frames can be represented by global 2D parametric motions. The foreground objects are usually closer to camera and sometimes undergo non-rigid or articulate movements. Simply approximating such foreground objects with planes and describing their motions with 2D parametric motion may result in higher error signals, and therefore higher bit rates.

In general, background objects are usually simpler to model and have larger areas in the images. Therefore they are easier to extract than foreground objects. Based on such observation, we classify VOPs into background VOPs and foreground VOPS. A background VOP can be modelled by a plane and represented by a static sprite (mosaic images of layers). A foreground VOP might not be modelled as rigid planar objects, and need to be represented as ordinary VOPs.

The observation that background VOPs are easier to extract than the foreground VOPs indicates a progressive scheme for VOP extraction. The background VOPs are extracted first, which are then used to construct some intermediate background representation (such as a mosaic) to guide the extraction of foreground VOPs.

6.4 Modelling VOPs

Given the data (video sequence) $\mathcal{D}$, our task is to compute:

- the pixel labels $\mathcal{L}$ specifying the mask of each VOP.
- the model $\mathcal{M}$, one for each VOP. $\mathcal{M}$ contains the model parameters as well as a flag to indicate if it is a background or foreground object. $\mathcal{M}$ may vary across time.

It is hard to define a semantic model for a VOP. We define $\mathcal{M}$ using low level features such as motion, texture, edge information, and spatial relationship between pixels.

[12] Static scene under pan/tilt/zoom camera corresponds to the plane in infinite.

6.4.1 Motion Model for VOP

Motion is the most important cue for modelling the VOPs. 2D parametric motion (such as projective transformation) is sufficient for planar object. In order to model more complex foreground objects, we define a hierarchical motion models with increasing complexity:

$$u(x, y) = T(x, y) + \sum_{i=1}^{L} c_i B_i(x - x_i, y - y_i) \tag{9}$$

where:

- $u(x, y)$ is the 2D motion at pixel (x, y).
- $T(x, y)$ is a global 2D motion model.
- $B_i(x - x_i, y - y_i)$ is the i_{th} basis function centered at (x_i, y_i). c_i is the weight for B_i. The second term in Eq.(9) describes the local non-rigid or articulate motion.
- L is the number of basis used in the motion model.

There are two alternatives in designing the format of the basis function $B_i(x - x_i, y - y_i)$:

1. Use global basis, where (x_i, y_i) is located at the center of VOP (x_c, y_c), and the support of $B_i(x - x_c, y - y_c)$ covers the whole VOP. Similar representation was used in [12] to represent optical flow fields, and in [6] to represent non-rigid 3D shapes.
2. Use local basis based on kernel regression. The bounding box of VOP is subdivided into $n \times n$ blocks. A local basis B_i is then placed at the center of each block. The support of B_i is concentrated inside its block, but could extends outside of the block, depending on a scale parameter σ.

We prefer the second scheme since it is closer to the definition of Macro Block in MPEG-4. The basis B_i could be a constant function (corresponding to translational motion in each block), a spline function [28], or a Gaussian function [37].

The number of basis L indicates the complexity of motion model. $L = 0$ gives the simplest model, where the motion model is a 2D global motion model ($T(x, y)$). The most complex model is when L is equal to the number of pixels in the interested VOP. In such case, the above motion model becomes the *plane* + *parallax* model, where the first term $T(x, y)$ is the model for the *plane* and the second term indicates the *parallax* of every pixel. In general, the bounding box of VOP is divided into $n \times n$ blocks (for example, $n = 16$ in the macro block of MPEG-4 standard). One basis function is then assigned at the center of each $n \times n$ block.

6.4.2 Texture Model for VOP

For each background VOP, the texture is represented by a static off-line sprite accumulated over time. To represent the texture of a foreground VOP, we use either dynamic sprite or just the texture from the reference frame (ordinary VOPs).

Reconstructing sprite/mosaic image from multiple views usually involves the following two steps:

1. Warp each source image to the destination view where mosaic will be constructed.

2. Blend the warped images to form the mosaic. Temporal median filter is often used in this step.

We need to address several issues in order to construct a high visual quality mosaic image:

- Using bi-linear interpolation in Step 1 will blur the image. Instead, we should take into account the texture structure in determining the interpolation weights. For example, one would expect the interpolation happens along the edge but not across it, in order to maintain the sharpness of edges in the image [1].

- The scene is not exactly planar, and the global 2D parametric transformations used to warp the images may not be perfect. To compensate such error, in the temporal filter we may need to consider an element larger than a single pixel, such as a color region.

- We should consider the geometry of the warping function [36], as well as the fact that some frames may have higher resolution or SNR than the others (e.g., some frames may be out of focus). The temporal weighting should favor frames with higher resolution or SNR.

Step 1 and 2 should be done in a batch way instead of sequentially. For a pixel p (or an element such as a color region) in the destination mosaic image, warp it to each of the input views, denoted as p'_i in the i_{th} view. Stack the neighbor pixels of p'_i from each view into a 3D volume. We want to design an optimal MMSE (minimum mean square error) interpolation filter to determine the value of pixel p, based on its 3D neighborhood .

We also need to choose an optimal reference view under which the mosaics/sprites are constructed (such reference view is not necessary appeared in the video). We need to take into the following two issues into account:

- image resolution, especially in zooming sequence.

- data amount to represent the mosaic/sprite. For example, if there are few frames containing high resolution information, we do not need to construct the mosaic at the frame view with highest resolution. Instead, we should construct the mosaic at a reasonable resolution plus some enhancement layers.

6.5 Competition Framework

The competition framework consists of the following three iterative steps:

- Model regression: each VOP computes its model based on its current support.
- Model prediction and competition: models compete for supporting pixels for their corresponding VOPs [13].
- Model merging/elimination according to some global criteria function (e.g., MDL).

The above steps are iterated until a global energy functional reaches its local minima. The global energy is defined as:

$$E = \sum_i (E_{D,i} + E_{M,i}) \tag{10}$$

$E_{D,i}$ is the residual error between original VOP$_i$ and the synthesized/decoded VOP$_i$:

$$E_{D,i} = \sum_{x \in \text{VOP}_i} \rho(I(x) - I'(x)) \tag{11}$$

where I is the original texture image, and I' is the texture image synthesized using the model of VOP$_i$.

$E_{M,i}$ is the amount of data used to encode the model of VOP$_i$, including motion, texture, and mask. For example, the following energy function favors smooth boundary and simple motion model for each VOP:

$$E_{M,i} = \frac{1}{2}\lambda \oint_{\partial R_i} ds + N_i \times f(L_i) \tag{12}$$

where ∂R_i is the contour of VOP$_i$; λ is the coding length of unit arc length of the contour; N_i is the number of pixels inside VOP$_i$; L_i is the number of basis used in representing the motion, as in Eq.(9).

Under the competition framework, existing multiple models compete for pixels in the given data, in a way that each pixel is assigned to the model so that E will decrease. One advantage of using competition approach is that the competition process is localized and independent of the complexity of the global energy formulation. Therefore, we can design a complicated energy function that combines different low level cues (e.g., motion, texture, edge information, and spatial coherence), by adding a new energy term in Eq.(12) for each cue.

6.5.1 Model Initialization

Both semi-automatic and full-automatic have been proposed for VOP generation. In semi-automatic approach (e.g., [15]), the user roughly segments the first frame. Then the algorithm will automatically segment the following frames in the video segment. Sometimes semi-automatic approach is desirable because only user can define semantic Video Objects (VO).

[13] This could happen either along the boundaries of VOPs or inside the VOPs, or both.

In our semi-automatic scheme, we only require the minimum user interaction by drawing boxes in the desired VOPs in the first frame, which is much simpler to do than specifying the VOP boundary [15]. Then our competition approach will automatically segment all other frames.

We propose two approaches for full automatic initialization. The first scheme use RPCA in Section 4. The second scheme uses a simple planar patch detection method based on rank 1 constraint.

- In RPCA scheme, we use a stringent threshold to classify regions/patches as inliers. Inliers are initialized using the approach in Section 3. The outlier regions/textures will then be initialized as foreground VOPs.

- In plane detection scheme, we divide the image into $n \times n$ overlap blocks. The 2D affine motion between the reference image and other images are then estimated. For each group of nine nearby blocks, we compute the rank of $\tilde{\mathbf{W}}_{6F\times 9}$, where F is the number of frames (except the reference) used. If $rank(\tilde{\mathbf{W}}) = 1$, then these 9 blocks are grouped into one block, and the motion are re-estimated using these 9 blocks. The resulted blocks, including grouped or original blocks, are considered to be planar patches, which are then input to a survivability test:

  - A block B derives its support by growing/shrinking around its boundary. A pixel p at B's boundary is assigned to B if the B's motion predicts p's motion. A block can survive only if it has enough continuous support (both spatially and temporally).

  The blocks that pass the test becomes the initial VOPs participating in the competition process.

6.5.2 Model Regression

In model regression step, each VOP computes its model based on the pixels currently assigned to it, such that $E_{D,i}$ is minimized. More specifically, it includes motion estimation and mosaic (sprite) construction if it is a sprite VOP.

To estimation the motion in Eq.(9), we first estimate $T(x, y)$ using robust multi-scale approach [4, 2]. Then if $L_i > 0$, we divide the bounding box of VOP into $n \times n$ blocks, and then estimate c_i for each B_i:

- If B_i is a constant with support only inside its block, we simply compute a bounded translation or affine motion.

- If B_i is a spline function [28] or Gaussian function [37], the resulted motion is a smooth optical flow field.

Typically L is equal to zero for background VOPs. For foreground VOPs, there is a choice between using more VOPs, or fewer VOPs with larger L for each VOP.

6.5.3 Model Prediction and Competition

Given a pixel or region, its appearance is predicted by each VOP using synthesizing/decoding based on the VOP model. By using synthesis prediction, we achieve the goal of using information accumulated from multiple frames, and being robust to outliers such as occlusion.

Different VOPs compete for pixels/regions along shared boundaries. A pixel/region is assigned to the VOP that can best predict its appearance in the sense that the global energy E will decrease most.

In order to construct the texture model for the background VOPs, we use a progressive scheme in VOP generation. The assumption is that background VOPs are simpler to model and easier to extraction. At this point, we only require a rough segmentation that identify the background pixels. The texture model of each background VOP is then computed by constructing a mosaic/sprite. Foreground VOPs are then initialized or refined by competing with the background VOPS.

6.5.4 Model Elimination, Merging and Splitting

A model is eliminated or merged with other model(s) if doing so decreases the global energy significantly. Similarly, if a VOP's model is too complicated, it is split into two VOPs with different models, if doing so decreases the global energy significantly.

6.5.5 Related Work

Competition approach was used in color segmentation in [41], where a color segment is modelled by a Gaussian distribution. The scheme in [41] can not be used in motion segmentation, because pixels in a textureless region can be assigned to any model, and the competition process may result in error segmentation. Moreover, Gaussian model is not an appropriate model for VOP.

In motion segmentation using EM algorithm [2, 38]), the E-step (pixel labelling step) also used competition in that a pixel is assigned to a most likely model, usually a 2D parametric model. The problems are:

- Use a single model formulation for all objects/layers. In general, for a foreground object (with more complex motion) to be able to compete with the background objects, its model should be more complex than that of the background object.
- Prediction is done by warping pixels from reference frame to other frame, which is subject to error due to occlusion.
- Do not use segmentation information accumulated from multi-frames.
- Do not combine all available low level cues.

6.6 Preliminary Results

We show the preliminary results on two image sequences: the *mobile & calendar* (50 frames) and *MPEG flower garden* (28 frames). Currently, the implementation is based on affine motion model, and the VOPs are defined based on motion only.

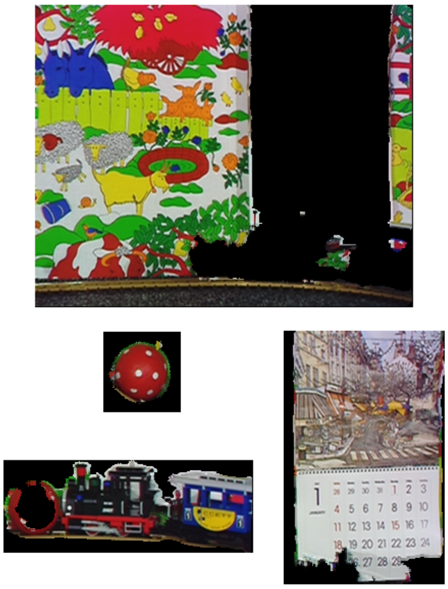


Figure 9: Sprite VOPs of mobile & calendar sequence.

The robust subspace algorithm (full-automatic) works consistently on the *flower garden* sequence. On the *mobile & calendar* sequence, the wall and the calendar are sometimes mixed with each other using the current model initial approach (two-pass region sampling). The main reason is that both layers contain similar white background color, and the color segmentation algorithm outputs those backgrounds as a common region. The semi-automatic model initialization works well with both image sequences.

Please see `http://www.cs.cmu.edu/~ke/proposal/` for the original video sequence and the masks of VOPs on *mobile & calendar* sequence (50 frames) and *MPEG flower garden* sequence (28 frames).

7 Application: Video Compression

MPEG-4 has defined how to encode the extracted VOPs. In this section we give some initial experimental results.

7.1 VOP Encoding & Decoding

In MPEG-4, each VOP is encoded/decoded independently.

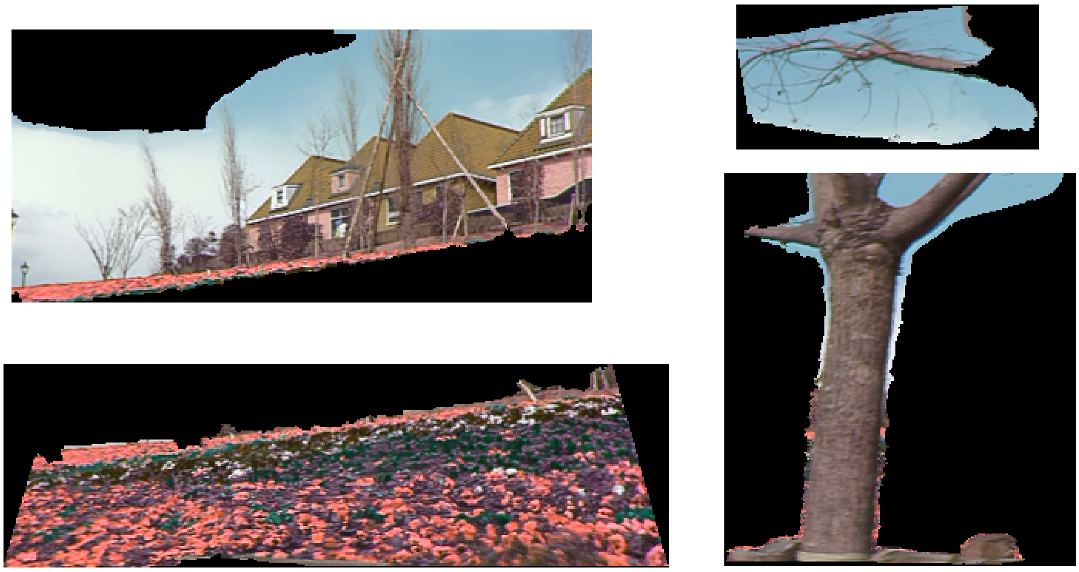


Figure 10: Sprite VOPs of flower garden sequence.

7.2 Preliminary Results

7.2.1 Mobile and Calendar

Fig. (9) shows the static sprites constructed using 28 frames from the *mobile & calendar* sequence. Note that each sprite is padded at its boundaries to avoid black breakages in the decoded frames. Please see
`http://www.cs.cmu.edu/˜ke/proposal/` for the original video sequence and decoded video sequence.

7.2.2 Flower Garden

Fig. (10) shows the static sprites constructed using 28 frames from the *MPEG flower garden* sequence. Again each sprite is padded at its boundaries to avoid black breakages in the decoded frames. Please see
`http://www.cs.cmu.edu/˜ke/proposal/` for the original video sequence and decoded video sequence.

References

[1] J.P. Allebach and P.W. Wong. Edge-directed interpolation. In *ICIP96*.

[2] S. Ayer and H. Sawhney. Layered representation of motion video using robust maximum-likelihood estimation of mixture models and mdl encoding. In *ICCV95*.

[3] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *CVPR98*, 1998.

[4] J.R. Bergen, P. Anandan, K.J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV92*.

[5] M. J. Black and A. Jepson. Estimating optical flow in segmented images using variable-order parametric models with local deformations. *PAMI*, 18(10), 1996.

[6] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *CVPR00*, pages II:690–696, 2000.

[7] Y.Z. Cheng. Mean shift, mode seeking, and clustering. *PAMI*, 17(8), 1995.

[8] D. Comaniciu and P. Meer. Robust analysis of feature spaces: color image segmentation. In *CVPR97*.

[9] D. Comaniciu and P. Meer. Distribution free decomposition of multivariate data. *Pattern Analysis and Applications*, 2(1), 1999.

[10] D. Comaniciu, V. Ramesh, and P. Meer. Real-time tracking of non-rigid objects using mean shift. In *CVPR00*.

[11] M.A. Fishler and R.C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach.*, 224(6):381–395, 1981.

[12] D.J. Fleet, M.J. Black, Y. Yacoob, and A.D. Jepson. Design and use of linear models for image motion analysis. *IJCV*, 36(3):169–191, February 2000.

[13] M. Gelgon and P. Bouthemy. A region-level graph labeling approach to motion-based segmentation. In *CVPR97*.

[14] R. Gnanadesikan and J.R. Kettenring. Robust estimates, residuals, and outlier detection with multiresponse data. *Biometrics*, 28:81–124, March 1972.

[15] C. Gu and M.C. Lee. Semiautomatic segmentation and tracking of semantic video objects. *CirSysVideo*, 8(5):572–584, September 1998.

[16] C. Harris. Structure-from-motion under orthographic projection. In *ECCV90*.

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[18] M. Irani. Multi-frame optical flow estimation using subspace constraints. In *ICCV99*.

[19] M. Irani, P. Anandan, and Meir Cohen. Direct recovery of planar-parallax from multiple frames. In *ICCV'99 Workshop: Vision Algorithms 99*.

[20] A.D. Jepson and M.J. Black. Mixture models for optical flow computation. In *CVPR93*.

[21] I. T. Jolliffe. *Principal Components Analysis*. Springer, 1986.

[22] Qifa Ke and Takeo Kanade. A subspace approach to layer extraction. In *CVPR 2001*.

[23] M.C. Lee, W.G. Chen, C.L.B. Lin, C. Gu, T. Markoc, S.I. Zabinsky, and R. Szeliski. A layered video object coding system using sprite and affine motion model. *CirSysVideo*, 7(1), 1997.

[24] J.L. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[25] L.S. Shapiro. *Affine Analysis of Image Sequences*. Cambridge University Press, 1995.

[26] A. Shashua and S. Avidan. The rank 4 constraint in multiple (over 3) view geometry. In *ECCV96*.

[27] J. Shi and J. Malik. Motion segmentation and tracking using normalized cuts. In *ICCV'98*.

[28] R. Szeliski and H.Y. Shum. Motion estimation with quadtree splines. *PAMI*, 18(12):1199–1210, December 1996.

[29] R. Szeliski and P. Torr. Geometrically constrained structure from motion: Points on planes. In *European Workshop on 3D Structure from Multiple Images of Large-Scale Environments (SMILE)*, June 1998.

[30] H. Tao and H. S. Sawhney. Global matching criterion and color segmentation based seereo. In *WACV2000*.

[31] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *IJCV*, 9(2), 1992.

[32] P.H.S. Torr, R. Szeliski, and P. Anandan. An integrated bayesian approach to layer extraction from image sequences. In *ICCV99*.

[33] F. Torre and M. J. Black. Robust principal component analysis for computer vision. In *ICCV2001*.

[34] J.Y.A. Wang and E.H. Adelson. Layered representation for motion analysis. In *CVPR93*.

[35] J.Y.A. Wang and E.H. Adelson. Representing moving images with layers. *IEEE Trans. on Image Processing*, 3(5), 1994.

[36] L. Wang, S.B. Kang, R. Szeliski, and H.Y. Shum. Optimal texture map reconstruction from multiple views. In *CVPR01*, pages I:347–354, 2001.

[37] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR97*.

[38] Y. Weiss and E.H. Adelson. A unified mixture framework for motion segmentation: Incorporating spatial coherence and estimating the number of models. In *CVPR96*.

[39] L. Zelnik-Manor and M. Irani. Multi-view subspace constraints on homographies. In *ICCV99*.

[40] L. Zelnik-Manor and M. Irani. Multi-frame estimation of planar motion. *PAMI*, 22(10), 2000.

[41] Song Chun Zhu and Alan L. Yuille. Region competition: Unifying snakes, region growing, and bayes/MDL for multiband image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(9):884–900, 1996.

Appendix

(1) Planar affine:

Given a static 3D plane π and a pair of affine cameras (or equivalently, a single static camera with the 3D plane undergoes 3D affine transformation), the two images of plane π is related by a 2D affine transformation $\mathbf{m}_{2\times3} = \{\mathbf{a}_{2\times2}, \mathbf{t}_{2\times1}\}$.

Proof: We only need to show that for any point $P \in \pi$, its imaging points p and p' in two cameras observes $p' = \mathbf{a}p + \mathbf{t}$, where $\mathbf{m}_{2\times3} = \{\mathbf{a}_{2\times2}, \mathbf{t}_{2\times1}\}$ is the affine transformation induced by π.

Let P_1, P_2, P_3 denote three non-collinear points on plane π. Let p_1, p_2, p_3 and p'_1, p'_2, p'_3 denote the image points of P_1, P_2, P_3 under affine camera $\{\mathbf{M}_{2\times3}, \mathbf{T}_{2\times1}\}$ and $\{\mathbf{M}'_{2\times3}, \mathbf{T}'_{2\times3}\}$ respectively. We have:

$$\begin{aligned} [p_1, p_2, p_3] &= \mathbf{M}\,[P_1, P_2, P_3] + \mathbf{T_1} \\ [p'_1, p'_2, p'_3] &= \mathbf{M}'\,[P_1, P_2, P_3] + \mathbf{T'_1} \end{aligned}$$

A 2D affine transformation $\mathbf{m}_{2\times3}$ is *uniquely* determined by these three non-collinear matched pairs such that $[p'_1, p'_2, p'_3] = \mathbf{a}[p_1, p_2, p_3] + \mathbf{t}$.

For any point $P \in \pi$, we have:

$$P = \alpha P_1 + \beta P_2 + \gamma P_3,$$

where $\alpha + \beta + \gamma = 1$. The image of P under camera $\{\mathbf{M}', \mathbf{T}'\}$ is:

$$\begin{aligned} p' &= \mathbf{M}'P + \mathbf{T}' \\ &= \mathbf{M}'(\alpha P_1 + \beta P_2 + \gamma P_3) + (\alpha + \beta + \gamma)\mathbf{T}' \\ &= \alpha p'_1 + \beta p'_2 + \gamma p'_3 \\ &= \mathbf{a}(\alpha p_1 + \beta p_2 + \gamma p_3) + \mathbf{t} \\ &= \mathbf{a}(\alpha \mathbf{M}P_1 + \beta \mathbf{M}P_2 + \gamma \mathbf{M}P_3 + (\alpha + \beta + \gamma)\mathbf{T}) + \mathbf{t} \\ &= \mathbf{a}p + \mathbf{t} \end{aligned}$$

Therefore, the image of any point on plane π undergoes the same 2D affine motion $\mathbf{m}$. $\diamond$

(2) Parametric representation of affine transformation:

Given a pair of affine cameras ψ_r, ψ', and a reference plane π_r, we can represent any other affine transformation $\mathbf{m}_{2\times3}$ induced by a plane π_m by:

$$\mathbf{m} = \mathbf{m_r} + \mathbf{e}'\mathbf{v}^T,$$

where $\mathbf{m_r}$ is the affine transformation induced by reference plane π_r, $\mathbf{e}' = (e_1, e_2)^T$, and the homogeneous coordinates $(e_1, e_2, 0)$ is the direction of epipolar lines in camera ψ'. $\mathbf{v}^T = (v_1, v_2, v_3)$ is a 3-vector independent of camera ψ'.

Proof: Without loss of generality, let us choose three non-collinear points $[P_0, P_1, P_2]$ on 3D plane π_r. We ignore the degenerate case where a plane projects onto a line in the camera imaging plane. $[P_0, P_1, P_2]$ projects onto three non-collinear points $[p_0, p_1, p_2]$ in camera ψ_r, and $[p'_0, p'_1, p'_2]$ in camera ψ', where $p_i = (x, y)^T$ and $p'_i = (x', y')^T$ are 2D image coordinates. There exist three non-collinear points $[P'_0, P'_1, P'_2]$ on plane π_m that will also project onto $[p_0, p_1, p_2]$ in camera ψ_r. Denote the image points of $[P'_0, P'_1, P'_2]$ in camera ψ' as $[p''_0, p''_1, p''_2]$, as shown in Fig.(11).

Since an affine transformation is uniquely determined by three pairs of non-collinear corresponding points, we have:

$$\begin{bmatrix} p'_0 & p'_1 & p'_2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_r \\ 0\,0\,1 \end{bmatrix} * \begin{bmatrix} p_0 & p_1 & p_2 \\ 1 & 1 & 1 \end{bmatrix} \tag{13}$$

$$\begin{bmatrix} p''_0 & p''_1 & p''_2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_{2\times3} \\ 0\,0\,1 \end{bmatrix} * \begin{bmatrix} p_0 & p_1 & p_2 \\ 1 & 1 & 1 \end{bmatrix} \tag{14}$$

Since affine camera has parallel projection, $\left[\overline{P_0P'_0}, \overline{P_1P'_1}, \overline{P_2P'_2}\right]$ are three parallel line segments. Parallelism is preserved by affine camera. Therefore, $\left[\overline{P_0P'_0}, \overline{P_1P'_1}, \overline{P_2P'_2}\right]$ will project onto parallel line segments $\left[\overline{p'_0p''_0}, \overline{p'_1p''_1}, \overline{p'_2p''_2}\right]$ (epipolar lines) in affine camera ψ' whose projection matrix is $\{\mathbf{M'}_{2\times3}, \mathbf{T}'\}$. Denote $\overline{p_ip_j} = p_j - p_i$. We have:

$$\begin{aligned} \left[\overline{p'_0p''_0}, \overline{p'_1p''_1}, \overline{p'_2p''_2}\right] &= \mathbf{M}' * \left[\overline{P_0P'_0}, \overline{P_1P'_1}, \overline{P_2P'_2}\right] \\ &= \mathbf{M}' * \mathbf{D} * [k_0, k_1, k_2]\,, \end{aligned} \tag{15}$$

where $\mathbf{D}$ (unit 3-vector) denotes the direction of parallel lines $\overline{P_0P'_0}, \overline{P_1P'_1}, \overline{P_2P'_2}$, and $\left[\overline{P_0P'_0}, \overline{P_1P'_1}, \overline{P_2P'_2}\right] = \mathbf{D} * [k_0, k_1, k_2]$, with k_i denoting the length of line segment $\overline{P_iP'_i}$. $[k_0, k_1, k_2]$ is independent of camera ψ'.

Denote $\mathbf{e}' = [e_1, e_2]^T = \mathbf{M}' * \mathbf{D}$ (It is obvious that $[e_1, e_2, 0]^T$ is the direction of epipolar lines in homogeneous coordinates in camera ψ'). From Eq.(15) we have:

$$\begin{aligned} [p''_0, p''_1, p''_2] &= [p'_0, p'_1, p'_2] + \left[\overline{p'_0p''_0}, \overline{p'_1p''_1}, \overline{p'_2p''_2}\right] \\ &= [p'_0, p'_1, p'_2] + \mathbf{e}' * [k_0, k_1, k_2] \end{aligned} \tag{16}$$

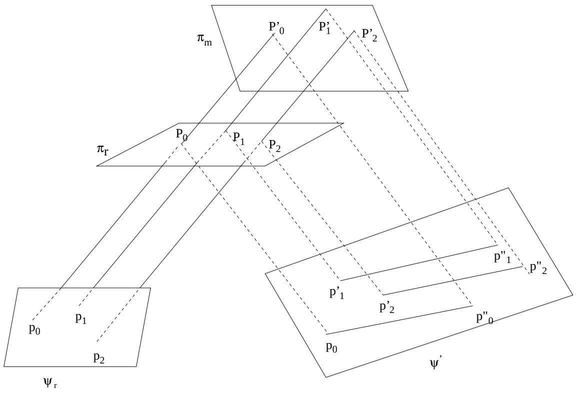


Figure 11: The relationship between 3D planes and affine cameras.

Substitute Eq.(16) and Eq.(13) into Eq.(14), we have:

$$\begin{aligned}&\begin{bmatrix}\mathbf{m}_{2\times 3}\\ 0\ 0\ 1\end{bmatrix} * \begin{bmatrix}p_0 & p_1 & p_2\\ 1 & 1 & 1\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{m}_r\\ 0\ 0\ 1\end{bmatrix} * \begin{bmatrix}p_0 & p_1 & p_2\\ 1 & 1 & 1\end{bmatrix} + \begin{bmatrix}\mathbf{e}'\\ 0\end{bmatrix} * [k_0, k_1, k_2]\end{aligned} \tag{17}$$

Since $[p_0, p_1, p_2]$ are non-collinear points, the matrix $P_{3\times 3} = \begin{bmatrix}p_0 & p_1 & p_2\\ 1 & 1 & 1\end{bmatrix}$ is non-singular and $P_{3\times 3}^{-1}$ exists. Therefore, from Eq.(17), we have:

$$\mathbf{m} = \mathbf{m}_r + \mathbf{e}' * [v_0, v_1, v_2] \tag{18}$$

Here $\left[\mathbf{e}'^T, 0\right]$ is the direction of epipolar lines in homogeneous coordinate in camera ψ', and

$$\mathbf{v}^T = [v_0, v_1, v_2] = [k_0, k_1, k_2] * P_{3\times 3}^{-1}$$

. It is obvious that the 3-vector $\mathbf{v^T}$ is independent of the second camera ψ'. $\diamond$