A Robust Subspace Approach to Extracting Layers from Image Sequences

Qifa Ke

CMU-CS-03-173

August 1, 2003

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Thesis Committee:
Takeo Kanade, Chair
Robert Collins
Jianbo Shi
Thomas Strat, DARPA

Keywords: layer extraction, layered representation, subspace, clustering, robust, video segmentation, video analysis, ego-motion

To my parents.

Abstract

A layer is a 2D sub-image inside which pixels share common apparent motion of some 3D scene plane. Representing videos with such layers has many important applications, such as video compression, 3D scene and motion analysis, object detection and tracking, and vehicle navigation. Extracting layers from videos involves solving three subproblems: 1) segment the image into sub-regions (layers); 2) estimate the 2D motion of each layer; and 3) determine the number of layers. These three subproblems are highly intertwined, making the layer extraction problem very challenging. Existing approaches to layer extraction are limited by 1) requiring good initial segmentation, 2) strong assumptions about the scene, 3) unable to fully and simultaneously utilize the spatial and temporal constraints in video, and 4) unstable clustering in high dimensional space. This thesis presents a subspace approach to layer extraction which does not have the above limitations. We first show that the homographies induced by the planar patches in the scene form a linear subspace whose dimension is as low as two or three in many applications. We then formulate the layer extraction problem as clustering in such low dimensional subspace. Each layer in the input images will form a well-defined cluster in the subspace, and a simple mean shift based clustering algorithm can reliably identify the clusters thus the layers. A proof is presented to show that the subspace approach is guaranteed to increase significantly the layer discriminability, due to its ability to simultaneously utilize spatial and temporal constraints in the video. We present the detailed robust algorithm for layer extraction using subspace, as well as experimental results on a variety of real image sequences.

Acknowledgements

I am deeply grateful to my advisor Takeo Kanade. Without his support and guidance, this thesis would not be possible. His wisdom and optimism guided me through my graduate study in CMU. His insight, creative thinking and patience greatly increased the quality of my research.

I would like to thank other members in my thesis committee - Robert Collins, Jianbo Shi and Thomas Strat, for their valuable comments and suggestions on both my thesis proposal and thesis. Their expertise and sound advice has help improve my thesis. I would like to thank Simon Baker and Martial Hebert for their valuable comments on my papers that became part of this thesis. I would also like to thank Harry Shum, Zhengyou Zhang, and Richard Szeliski in MSR for a fruitful internship and their comments on my papers.

Thanks to Sharon Burks for making my life as a graduate student simpler and more delightful.

Thanks to my fellow students Jing Xiao, Yanghai Tsin, Stella Yu,Jinxiang Chai, Lie Gu, Yan Li, and in CS - Yanghua Chu and Jun Gao, for making my student journey in CMU so enjoyable.

I thank my wife Yinglian Xie for her love, encouragement, and support. She patiently listened to all of my practice talks, and gave great feedbacks that improved not only the talks, but also this thesis. I thank my mother-in-law for helping take care of my baby daughter Emily. I thank my sisters and my bother for their love and long term support of my pursuits. I thank my parents for everything. This thesis is dedicated to my parents!

Contents

List of Figures

Chapter 1

Introduction

In this thesis, we develop a novel *subspace* approach to derive a mid-level video representation, the *layer(ed)* representation. Centering around subspace, we will present what subspace is, why use subspace, how to compute the subspace, and how to use subspace to extract layers from video.

(a) (b)

Figure 1.1: Layer segmentation of a sample frame in a short video sequence where there persons are walking on a road in front of a building. (a): One frame in the walking-people sequence. (b) Layer segmentation.

1.1 Layer representation as a mid-level video representation

An image is worth a thousand words. Fig 1.1(a) shows one single video frame of a scene. Human beings are able to understand the basic content in this image, i.e., three *persons walking* on a *road* in *front* of a *building*. Given only one image, such image understanding task seems easy for human beings, yet it actually involves high level processes that are usually very difficult for computers to perform, such as object recognition (person, road, building) and action recognition (walking).

Towards the ultimate goal of image understanding, the central issue is the image or video representation problem. In computer vision community, image representation has been divided into three levels: low-level, mid-level, and high-level [66]. Low-level representation focuses on local aspects of images, such as pixels, corners or edges output by local operators, the well-known multi-scale image representation (i.e., pyramid), optical flows, etc. High-level representation, on the other hand, fully describes the image content including objects and events. While low-level representation has been well defined and studied, the suitable "computer vocabulary" for general high-level representation remains unsolved except in some very constrained domain (e.g., face detection [75, 78]). Mid-level representation is somewhere between low-level and high-level representation. It contains information more global than low-level representation, such as global motion, lighting, camera motion, etc. Perhaps one of the most important task of mid-level representation is to bridge the gap between the computer-capable low-level representation and the human-capable high-level representation.

Motion information is one of the most important information to be extracted and represented in mid-level representation. An important characteristic of video is that it contains temporal redundant information by recording multiple frames of the same scene at different time instances, i.e., a scene point in the scene is imaged at multiple frames across time. Such redundant information contains the motion information of camera and the objects in the scene. When given more consecutive frames of the same scene in Fig 1.1(a), human beings do not gain much more information from the additional frames. However, the motion information provided by the additional frames can be utilized by computers to derive an efficient mid-level video representation. Such efficient mid-level video representations are not only useful for video coding and

editing, but also important for high level vision tasks, since such high level tasks would require efficient information storage/retrivel and communication between processing units. In the following we will review two such compact mid-level representations: mosaic representation and layer representation. The mosaic representation can be considered as a special case of layer representation with only one layer.

1.1.1 Mosaic representation

A popular example of mid-level video representation is video mosaic [86]. Given the video of a static scene, if the camera optical center does not move during recording the video, or if the camera is far away from the scene (e.g., airborne video), then the image motion between video frames can be modelled by a single 2D transformation $\mathbf{H}$ (homography):

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \cong \mathbf{H}_{3\times 3} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \tag{1.1}$$

where $(x', y', 1)$ and $(x, y, 1)$ are the homogeneous image coordinates in two different images, and $\mathbf{H}$ is a 3×3 non-singular matrix. The eight parameters in $\mathbf{H}$ capture the motion information of all pixels between two images.

The video mosaic is obtained by warping video frames to a common coordinate frame using the 2D homography, and then stitching the warped images together. Video mosaic is a compact representation since the temporal redundancy in the video has been identified and modelled by the homography, and removed by the image stitching. Examples of important applications of video mosaic are video coding [44], virtual reality (Apple quicktime® VR), environmental texture mapping [87], etc.

1.1.2 Layer representation

In real scenarios, we often can not meet the requirements for constructing video mosaic. The camera optical center may translate; the distance between camera and scene may not be far away enough; the objects may move in the scene. In such cases, parallax information presents in the video, and a single homography $\mathbf{H}_{3\times 3}$ is no longer enough to describe the motion of all pixels between two video frames. We

Figure 1.2: Two frames ($t = 10$ and $t = 15$) in the flower garden sequence. The camera is translating to the right. Notice the occlusion relationship between the tree and the background objects (garden and house).

are no longer able to build a single mosaic representation for the given video. Fig. 1.2 shows two frames of the *flower garden* sequence, where the scene is static, and the camera is translating to the right. The tree appears moving to the left and occludes the background scene (garden and house). A single homography can not capture the motion information for both the tree and the background objects.

Layer representation [1, 28] extends the mosaic representation in an important way that explicitly represents the parallax information. A layer consists of one or more 2D regions in the image domain, inside which the pixels share the same 2D motion model. When 2D homography is used as the motion model, a layer will correspond to a 3D plane in the scene. An image sequence containing parallax information can be decomposed into multiple layers, where each layer can be considered as a single planar mosaic with other information such as depth ordering. These layers may overlapped or occluded with each other. According to [1], a layer representation consists of several layers with depth ordering. Each layer contains a set of registered maps. The three basic maps are:

- intensity map $E_i(x, y)$: the color of each point (x, y) in the i-th layer;
- alpha map $\alpha_i(x, y, t)$: transparency of each point (x, y) in the i-th layer at time t;
- velocity map $V_i(x, y, t)$: the motion of each point (x, y) in the i-th layer at time t.

In many applications, $V_i(x, y, t)$, the motion at time instance t, can be represented by a few parameters, such as the 2D affine transformation with 6 parameters.

Fig. 1.3 and 1.4 show a layer representation example. Fig. 1.3 shows the layer intensity map of the *flower garden* sequence. Four layers are used to represent this sequence: tree, tree-branch, garden, and house. Fig. 1.4 shows two layer masks at $t = 10$ and $t = 15$. It is a $\{0, 1\}$ alpha map, where for each layer, $\alpha = 1$ for the pixels assigned to it, otherwise $\alpha = 0$. The motion model used in this example is 2D affine transformation.

Ideally, the layer representation should contain all information in the original video, and the original video frame can be re-synthesized from the layer representation, in a way similar to the cel animation. Indeed, layer representation borrows the ideas from cel animation (also known as 2D animation, cartoon animation). In cel animation, characters are painted on sheets of transparent plastics. These painted sheets are placed over a background to build up the scene, which is then photographed in succession to give the illusion of movement in the final film. More specifically, given layer representation, we can synthesize the video frame at time t by painting the layers one over another in a back-to-front order using the composition operator [15]:

$$F \odot B = \alpha F + (1 - \alpha_F)B$$

where F and B are foreground layer and background layer, and α_F is the alpha map of the foreground layer. Denote $L_i(x, y, t) = E_i(x, y, t) * \alpha_i(x, y, t)$, where $E_i(x, y, t)$ is obtained by warping $E_i(x, y)$ using the motion $V_i(x, y, t)$. Then the image composition from multiple layers can be represented by:

$$I(x, y, t) = \bigodot_i L_i(x, y, t) \tag{1.2}$$

For example, using the layer representation in Fig. 1.3 and 1.4, we can re-synthesize the two frames in Fig. 1.2. The synthesis results are shown in Fig. 1.5.

1.1.3 Why layer representation?

Just like mosaic representation, layer representation is compact and has important applications in video coding and video editing. As shown in the flower garden example, the size of layer representation (layer intensity maps, layer masks, and layer motion)

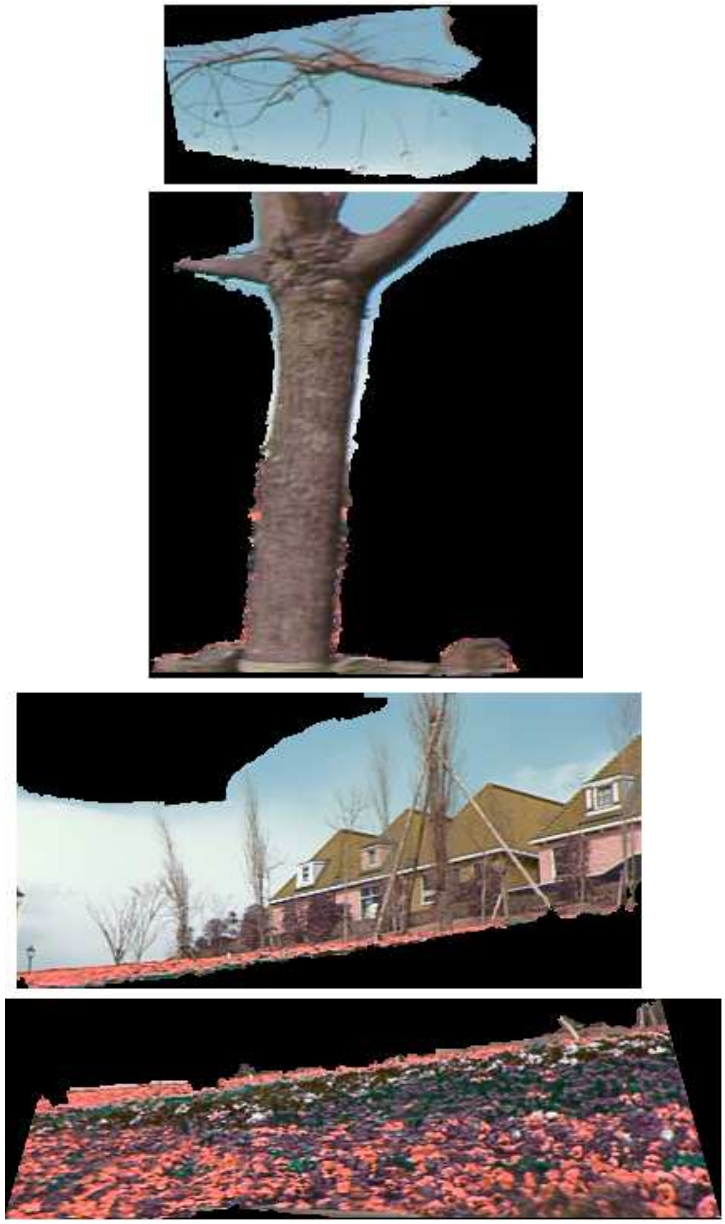


Figure 1.3: The layer intensity map for the flower garden sequence. Four layers are used to represent the flower garden sequence.

Figure 1.4: The masks of layers for the frames shown in Fig. 1.2. The masks show the pixel assignment. It represents the $\{0, 1\}$ alpha map, where for each layer, $\alpha = 1$ for the pixels assigned to it, otherwise $\alpha = 0$.

Figure 1.5: Re-synthesize the video frames from layer representation.

is about 40KB, which can used to re-synthesize the original 30 frames that is 7MB in file size.

Yet the implications of layer representation are far beyond the mosaic representation. In layer representation, each layer is an intermediate"object" that moves coherently according to some layer motion model. A single object, especially rigid object, tends to move together in a coherent way, and is likely to become one single layer in the layer representation. Therefore, layer representation gives a first cut in scene segmentation which facilitates following higher level tasks. The applications of layer representation to higher level vision tasks are recently attracting more and more researchers. In the following we give some examples:

- In motion analysis [102, 51, 43], layer representation explicitly represents the occlusion relationship, which is the hardest part in motion analysis. Image motion estimation is inherently an ill-posed problem [92] due to the *aperture problem*, which requires additional constraints in order to estimate the motion, such as regularization [71, 72], or parametric model that assumes some pixels share a common model with a few parameters [63]. However, it is undesirable to apply such constraint across motion boundaries, which are not known prior to the motion estimation. In layer representation, we can enforce such smoothness constraint only inside each layer, and explicitly represent the non-smoothness at the boundaries among layers.

- In object tracking [89], layer representation enables modelling both foreground objects (being tracked) and background objects. By maintaining both the targets and background objects in a layer representation, the often encountered drift problem in tracking can be reduced. Also by enforcing constraints (color model, smooth motion model, etc) inside each layer, the tracking is more reliable.

- In scene analysis, layer representation provides a powerful scheme similar to "plane+parallax" [61]. For example, layer representation was used in stereo reconstruction [9]. Layer representation can also be used in structure from motion (SFM) analysis, and camera self-calibration. While traditional SFM algorithms use feature points and compute a sparse scene representation, layer based SFM algorithm uses image regions as features and computes a dense scene representation. Moreover, due to the fact that regularity (smoothness)

constraints can be safely applied, SFM based on layer representation is expected to be more robust.

- In object detection and recognition, layer representation gives the first cut to locate the desirable objects. For example, the layer segmentation in Fig 1.1(b) lends guidelines for the task of detecting humans in video.
- In visual navigation, layer representation can be used to extract and represent the ground layers, and objects (cars, pedestrians, etc). Ground layer is useful in estimating the car ego-motion [58] and navigating the vehicles.

In summary, the world is regular to some extent instead of purely chaotic. When captured in video, such regularity leads to the regularity in image motion. Layers are powerful mid-level primitives to capture such regularity, and therefore provide efficient mid-level representations that facilitate higher level vision tasks.

1.2 Layer extraction problem

This thesis focuses on deriving the layer representation from video by utilizing the temporal and spatial redundant information in the video.

The core of layer extraction task is to segment the images into some number of *sub-images*, in such a way that pixels within each sub-image share some common 2D parametric transformation (or nonparametric model defined by dense smooth flow field [104]). The three major issues of layer extraction are:

- Segmentation: which portion of the image corresponds to one layer?
- Motion: what motion does a layer undergo?
- Number of layers: how many layers exist in the given image?

Layer extraction is a non-trivial task since the above three sub-problems are coupled together. On one hand, spatial layer supports (including number of layers) are required to estimate the motion model for each layer. On the other hand, assigning pixels to layers requires the knowledge of layer motion model.

In this thesis, the layer motion is modelled by 2D homography (e.g., 2D affine transformation or projective transformation). Given such motion model, a layer in

the image domain corresponds to a plane in the 3D scene. Therefore, segmenting the images into 2D layers can be considered as approximating the scene with some number of 3D planes.

1.3 Previous approaches to layer extraction

Various approaches have been proposed for layer extraction. In this section, we review the most often used approaches including EM approach, dominant motion approach, and grouping approach.

1.3.1 EM approach

A natural approach to solve the coupled problems in layer extraction is the Expectation-Maximization (EM) algorithm [51, 8, 105, 104, 96, 59]. In such approach, the likelihood of the video data is formulated as some mixture model, such as the mixture of Gaussians, with each mixture component represents a layer. The EM algorithm iteratively performs the E-step and M-step to maximize the likelihood function, where the E-step assigns pixels to layers, and the M-step computes the motion of each layer given the result of E-step. Followed by each EM iteration is the application of MDL principle [73] to determine the number of layers, which is realized as an exhaustive search in [8].

Initialization (the number of models and the motion for each model) is an important but difficult step for EM approach [81, 96]. Without good initialization, EM algorithm may not converge to desired optimal solutions. A typical initialization method [8] is to divide the image into a fixed number[1] of tiles, and use them as the initial layers for the EM algorithm. However, the initial regular tiling does not guarantee the existence of dominant motion inside each initial or intermediate layer[2], which is required for the robust motion estimation of each intermediate layer in the M-step [8]. Moreover, if one real layer is divided into different parts, each part belonging to a different initial tile with different dominant motion (or without any dominant motion at all), then it becomes hard to extract such unlucky layer.

[1] This number can not be too large due to the expensive computation in EM iteration and MDL step

[2] Unlike dominant motion approach, the presence of dominant motion of the whole image is not required.

1.3.2 Dominant motion approach

The dominant motion approach is a top-down approach. It assumes that the current image contains a dominant layer, which means the majority of the image pixels are from the same plane in the scene. The approach consists of multiple passes. Each pass determines the current dominant layer through dominant motion estimation [47, 11, 77] using robust estimator [74, 14]. The dominant layer is then removed from the current remaining image. The remaining portion of the image becomes the input of the next pass where another dominant layer will be sequenced out. The algorithm stops until all image pixels have been assigned to layers.

The dominant motion approach is effective only if the dominant layer assumption is held at each pass. However, such assumption is often violated in real scenarios. For example, in the walking-people sequence (Fig 1.1(a)) and *flower garden* sequence (Fig 1.2) there is not any clear dominant layer.

1.3.3 Grouping approach

The grouping approach is a bottom-up approach that avoids the dominant layer assumption. The image is first divided into small blocks (segments). For each block, its 2D homographies (local measurements) from the reference frame (being segmented into layers) to other frames in the video are estimated. Since a plane in the scene undergoes a unique 2D homography, image blocks belonging to the same layer should share similar homography parameters, and therefore form a cluster in the motion parameter space. By grouping local measurements into clusters in the parameter space, we can identify the 2D layers in the image domain. Methods used to group these local blocks into layers include the k-means algorithm [103, 5] and normalized graph cut [81].

Grouping local measurements (pixels or local blocks) does not have the initialization difficulty encountered in EM approach. However, grouping purely based on local measurements ignores the global spatial-temporal constraints. Moreover, grouping in high dimensional space is often unreliable given noisy local measurements. For example, if we are given a short image sequence containing 11 frames. Suppose we use 2D affine transformation (6 parameters) as the layer motion model. The local measurement of each block contains 6×10 parameters! Grouping in such a high dimensional (60 dimensional) parameter space will be unstable.

1.4 Subspace approach to layer extraction

1.4.1 What is subspace approach?

In this thesis, we present a low dimensional linear subspace approach that can naturally exploit the global spatial-temporal constraints existing in the video simultaneously. Our approach is base on the fact that the parametric motions (local measurements) across multiple frames reside in a low dimensional linear subspace, as a result of scene regularity and the geometry of camera imaging process. We formulate the layer extraction problem as grouping in the low dimensional subspace, where clusters (layers) of local measurements become denser and better-defined. Therefore, such clusters can be more reliably identified in the subspace.

1.4.2 Why subspace approach?

The first stage of our approach is a bottom-up grouping process that avoids both the initialization difficulty in EM-based approach and the dominant motion assumption in top-down approach.

Compared to previous grouping approaches, grouping in the low dimensional space is more stable. We use statistical analysis to show the merits of subspace approach resulting from its ability to naturally exploit the global spatial-temporal constraints existing in videos. Specifically, we prove that 1) layers are more discriminative in the subspace, and 2) increasing the frame number in the input video increases the layer discriminability in the subspace, but not in the original high dimensional space. In summary, our subspace approach has the following advantages:

- Clusters in the subspace become denser and better-defined.
- Global optimality is achieved by simultaneously taking into account all valid regions.
- Noise in estimated motion is reduced by subspace projection, and global geometry constraint is enforced.
- The subspace by itself provides a constraint to detect outliers in the local measurements, therefore making the layer extraction robust.

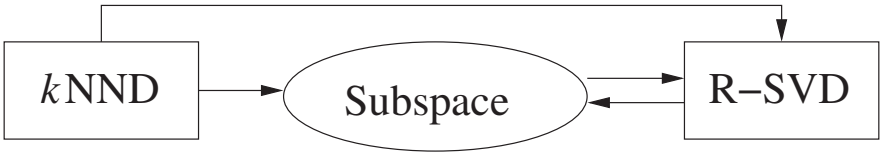


Figure 1.6: Three integrated components in semi-progressive layer extraction algorithm. Subspace is the centering component.

1.4.3 How to robustly compute the subspace?

Outliers in local measurements occur due to multiple motions in one local region, non-rigid motions, lighting changing, occlusions, etc. In layer extraction, good local measurements follow two rules. First, they form clusters in the motion parameter space. Second, they lie in a low dimensional subspace. Data items far away from the clusters are defined as extreme outliers. An extreme outliers may lie in the subspace. Data items violating the subspace constraint are defined as structure outliers. A structure outlier has large projection values in the residual space. The reason we classify outliers into these two types is that extreme outliers are more influential on the subspace computation. Therefore, we need to apply non-parametric method to detect the extreme outliers. Once they are detected and removed, a relatively good initial subspace model will be available. We can therefore apply parametric approach to detect the structure outliers.

Two important issues arise in order to fully exploit the subspace constraint for a reliable layer extraction algorithm: 1) to achieve low dimensional subspace, especially for dynamic scenes; 2) to make subspace computation robust to outliers that often exist in real data. We achieve the above goals by naturally integrating the following three components in our layer extraction algorithm: subspace, kNND, and robust-SVD, with subspace the centering component, as shown in Figure 1.6. The kNND component, a simple and non-parametric approach based on the k-th Nearest Neighbor Distance (kNND) metric, implicitly utilizes the intrinsic cluster structure to detect extreme outliers and small layers (cluster size less than k), *without* knowing the clustering information. The remaining large layers are guaranteed to reside in a low dimensional subspace, which in turn provides constraints for the robust-SVD (RSVD) component to detect the *structure outliers* that violate the correlation structure imposed by subspace. The RSVD component computes an accurate subspace model, due to 1) the removal of influential extreme outliers by kNND and 2) its ability to detect the remaining structure outliers.

1.5 Semi-progressive layer extraction

In our approach, a layer is identified as a cluster in the parameter subspace. A well defined cluster should have enough data items to form an identifiable cluster. For a small foreground layer in the scene, it usually does not form a well-defined cluster, due to: 1) it does not have enough number of local measurements, and 2) its local measurements tend to be un-stable since foreground objects tend to have complex motion. It is therefore much harder to detect the small foreground layers than to detect the large background layers. Moreover, if the small foreground layers into account in the measurement matrix are moving independently, the dimension of the subspace will be increased due to the complex motion of the foreground layers.

It is therefore desirable to first detect the large layers that form dense clusters in low dimensional subspace, then use the information or constraints provided by those large layers to guide the detection of the small layers.

We use semi-progressive layer extraction strategic, where the large layers are simultaneously extracted using subspace approach. The small layers are then progressively extracted against the extracted layers in a layer competition frame work. Excluding the small layers at the very beginning reduces the dimension of the subspace and therefore increases the reliability of the extraction of the large layers in the subspace. Such semi-progressive scheme is more like a feed-back process, where information readily available provides feedback to the subsequent harder tasks.

We use the kNND procedure to exclude small layers. The idea is simple. For a small layer whose number of local measurements is less than k, it either forms a cluster with cluster size less than k, or does not form a cluster at all if its local measurements are not good enough. Therefore for any item in such cluster, its distance to its k-th nearest neighbor will be large since its k-th nearest neighbor is either in another cluster or a sparse item far away. On the other hand, for any data item resides in a cluster with size larger than k, its k-th nearest neighbor is in the same cluster, and therefore its kNND will be small. In the 1D histogram of the kNND of all data items, local measurements from large layers will have small kNND and form the first peak. Other local measurements not in the first peak will be excluded either as outliers or small layers. In summary, kNND can exclude the small foreground layers at the very beginning using the intrinsic cluster structure of the large layers, but without knowing the clustering result.

Our overall semi-progressive layer extraction algorithm is summarized in Fig. 1.7. Its major steps are:

1. Estimate the local measurements (e.g., 2D parametric motions for each local blocks) and construct the local measurement matrix.
2. Robustly compute the low dimensional subspace and project the local measurements onto the subspace. Outliers and small layers are excluded in this step.
3. Cluster the inliers in the subspace into initial layers, which are large layers.
4. Progressively extract the excluded small layers, and refine layers using layer competition.

1.6 Contributions

In this thesis, we present novel approach to layer extraction, with solid theoretical analysis and experiments using both synthetic and real data. We make contributions in both theory and practice. More specifically, this thesis makes the following contributions:

- We present a novel subspace approach to layer extraction problem [55]. Our approach can naturally utilize the spatial-temporal constraints in the video. We prove the merits of subspace approach by both statistical analysis and experiments on real image sequences.
- In layer extraction, we observe that inliers form clusters and lie in the subspace. Robust subspace computation with outlier detection is based on the above observation. We present kNND approach to detect the extreme outliers by implicitly using the intrinsic cluster structure, but without knowing the clustering results. The subspace by itself provides a constraint that enables us to detect the structure outliers through robust SVD algorithm.
- To extract small foreground layers that undergo complex motion, we present the semi-progressive layer extraction algorithm. It guarantees a low dimensional subspace for the extraction of larger layers, which provide feedbacks to guide

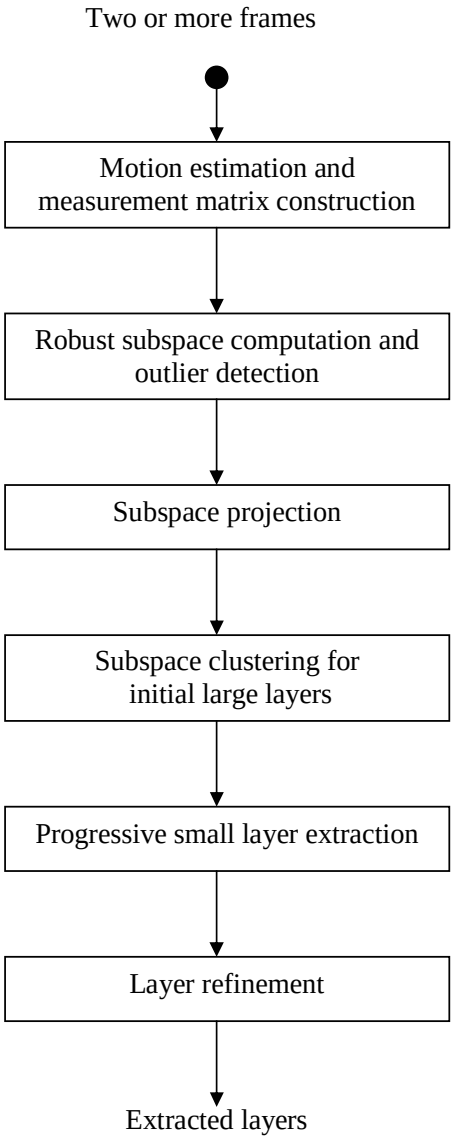


Figure 1.7: The procedure of layer extraction algorithm.

the extraction of small layers that may undergo complex motion and hard to extract before the larger layers are identified.

- In video-based car ego-motion estimation and ground detection where the camera mounted on the car is calibrated, a subspace constraint is explicitly available due to planar motion constraint. We use a virtual downward-looking camera to explicitly exploit such constraint to robustly estimate the car ego-motion and detect the ground plane [58].

1.7 Related work on subspace

Linear subspace constraints have been successfully used in computer vision. The subspace may come from the scene regularity and the geometric relationship between the scene and the camera. In such cases, the subspace dimension is usually predetermined. Tomasi and Kanade [95] used the rank-3 constraint in structure from motion (SFM), where the local measurements are translations of feature points, and the factorization of measurement matrix according to the subspace constraint leads to the structure of the feature points and the motion of the camera. Costeira and Kanade [25] used the subspace constraint to segment the tracked feature points into independently moving objects. Irani [45] used the subspace constraint in Lucas-Kanade [63] method to estimate the optical flows of an image sequence. Bregler [16] extended the above approach to recover the shape and motion of non-rigid moving objects. Shashua and Avidan [80] derived the 4-dimensional linear subspace of planar homographies induced by multiple planes between a pairs of views. Zelnik-Manor and Irani [107, 108] extended such results to multiple planes across multiple views, and applied such constraints to estimate the homographies of *predefined* small regions. Fig 1.8 summarizes some of the related work that utilizes subspace constraints coming from the geometry or physical constraints of the problems under consideration.

In some cases, subspace is represented by the principal components of the input data (training data), where the subspace dimension depends on the statistics of the input training data. Such parametric subspace is often used to represent the shape, appearance, and motion, which can then be used for tasks like object detection and recognition, and tracking [69, 100, 13, 30].

In a wider perspective, subspace constraint has been used to recover the shape and reflectance of static objects, given its images under different lighting directions [76].

Measurement (2D model)	Rigidity	Camera model	Subspace dimension	Applications
Translation	Rigid	Affine	3	SFM [95], Flow estimation [45]
Translation	Rigid	Perspective	9	Flow estimation [45]
Translation	Non	Affine	4N	Motion segmentation [25]
Translation	Non	Affine	3K	Nonrigid object modeling/tracking [16]
Projective	Rigid	Perspective	4	Homography estimation
Color	N/A	N/A	2	Shape/reflectance recovery [76]
Force Sensor	N/A	N/A	DoF	Sensor calibration [101]

Figure 1.8: Summary of previous work using subspace constraints coming from the geometry or physical constraints of the problems.

In [101], subspace constraint is used to calibrate the force sensor without using any known reference.

1.8 Thesis outline

This thesis centers around the concept of subspace. In Chapter 2, we show that the 2D homographies induced by planar patches in the scene reside in a low dimensional linear subspace. The dimension of the subspace is bound by the scene regularity and the geometry of imaging process. We study the subspace dimensionality in different scenarios, including static and dynamic scenes.

We then formulate the layer extraction as clustering in the subspace. In Chapter 3, we use statistical analysis to verify the merits of the subspace approach to layer extraction. Specifically, we prove that 1) layers are more discriminative in the subspace, and 2) increasing the frame number in the input video increases the layer discriminability in the subspace, but not in the original space.

To robustly compute the subspace from noisy local measurements, we need to deal with outliers. In Chapter 4, we classify outliers into extreme and structure outliers. We then present different methods to deal with these two types of outliers. We present kNND (non-parametric metric) to detect the extreme outliers. The subspace by itself provides a constraint that we use to detect the structure outliers through robust SVD

algorithm.

Once we derive the subspace, we can extract the layers by clustering the local measurements in the subspace. Chapter 5 presents the details of layer extraction algorithm. We also present the semi-progressive approach to deal with small layers undergoing complex motion. We present experimental results on several image sequences.

In the typical setup of video-based car navigation, the camera is calibrated and fixed w.r.t. the car. The car is moving on the ground plane, i.e., undergoes planar motion. In such setup, the subspace constraint due to planar motion is explicit. In Chapter 7, we use a virtual downward-looking camera to explicitly utilize such constraint for robust car ego-motion estimation and ground plane detection.

We conclude the dissertation in Chapter 8, and presents future work along and beyond the line of this thesis.

Chapter 2

Subspace of 2D Homographies

2.1 Introduction

In this chapter, we show that the 2D homographies induced by planar patches in the scene reside in a low dimensional linear subspace. The dimension of the subspace is bound by the scene regularity and the geometry of imaging process. We study the subspace dimensionality in different scenarios, including static and dynamic scenes.

In the framework of bottom-up approach to layer extraction, the reference image to be segmented into layers is first divided into small local blocks. For each local block, its 2D parametric motion (e.g., affine transformation) from the reference frame to every other frame is estimated. The local measurement of one block is obtained by reshaping its estimated motion parameters into a column vector, and the *measurement matrix* $\mathbf{W}$ is formed by stacking all local measurements (reshaped column vectors) into a matrix of dimension $6F \times K$, if we are given $F + 1$ frames and use affine transformation model. K is the number of local blocks or local measurements. We show that the measurement matrix $\mathbf{W}$ is rank deficient. In other words, there is a lower dimensional linear subspace of the original $6F$ dimensional column space. Based on the geometry relationship between the scene planes and the camera, we will prove the existence of the linear subspace by factorizing the measurement matrix $\mathbf{W}$:

$$\mathbf{W}_{6F\times K} = \mathbf{U}_{6F\times d}\mathbf{V}^{\top}_{d\times K} \tag{2.1}$$

where d is the dimension of the linear subspace, and the columns of matrix $\mathbf{E}$ forms the bases of the linear subspace.

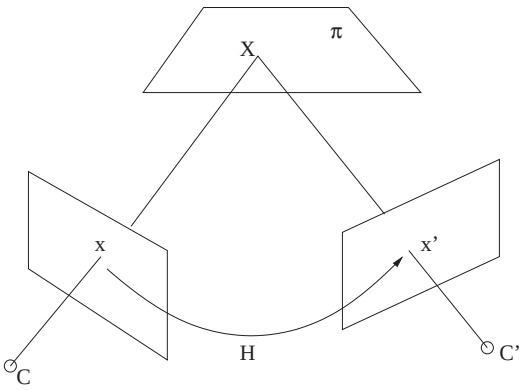


Figure 2.1: The homography induced by a plane. For any point $\mathbf{X}$ on the plane π, its image points $\mathbf{x}$ and $\mathbf{x}'$ in the two camera views are related by a unique and common non-singular 3×3 matrix $\mathbf{H}$ such that $\mathbf{x}' = \mathbf{H}\mathbf{x}$. The matrix $\mathbf{H}$ is called homography.

2.2 Planar homography

Suppose we are given a plane π that does not contain the camera optical center, and two camera views of π. A point $\mathbf{X}$ on the plane π is projected onto $\mathbf{x}$ and $\mathbf{x}'$ on the two views respectively, where $\mathbf{x}$ and $\mathbf{x}'$ are homogeneous coordinates (3-vectors). There exists a unique non-singular 3×3 matrix $\mathbf{H}$ such that $\mathbf{x}' \cong \mathbf{H}\mathbf{x}$, where $\cong$ means equal up to an unknown scale. This 3×3 matrix $\mathbf{H}$ is called the *homography* induced by the plane π. The explicit parametric represnetation of $\mathbf{H}$ is given by the following theorem [37]:

Theorem 2.1. *Given a plane defined by* $\pi^\top \mathbf{X} = 0$ *with* $\pi = (\mathbf{v}^\top, 1)^\top$ *, and its two projective views with projection matrix* $\mathbf{P}_{3\times4} \cong [\mathbf{I}_{3\times3} \mid \mathbf{0}_{3\times1}]$ *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* $\mathbf{H} \cong \mathbf{A} - \mathbf{a}\mathbf{v}^\top$

Proof: Consider a point $\mathbf{X}$ on plane π which projects onto $\mathbf{x}$ and $\mathbf{x}'$ on the image planes of the two cameras, respectively. We have:

$$\lambda \mathbf{x} = \mathbf{X} \tag{2.2}$$

$$\lambda' \mathbf{x}' = \mathbf{A}\mathbf{X} + \mathbf{a} \tag{2.3}$$

where λ and λ' are the projective depth of $\mathbf{X}$ in the two camera coordinate frames, respectively.

Substitute Eq.(2.2) into Eq.(2.3), we have:

$$\frac{\lambda'}{\lambda}\mathbf{x}' = \mathbf{A}\mathbf{x} + \frac{1}{\lambda}\mathbf{a} \tag{2.4}$$

Since $\mathbf{X}$ is on the plane π, it satisfies:

$$\mathbf{v}^\top \mathbf{X} + 1 = 0 \tag{2.5}$$

Substitute Eq.(2.2) into Eq. (2.5), we have:

$$\frac{1}{\lambda} = -\mathbf{v}^\top \mathbf{x} \tag{2.6}$$

Substitute Eq.(2.6) into Eq.(2.4), we have:

$$\begin{aligned} \mathbf{x}' &\cong \mathbf{A}\mathbf{x} - \mathbf{v}^\top \mathbf{x}\mathbf{a} \\ &\cong (\mathbf{A} - \mathbf{a}\mathbf{v}^\top)\mathbf{x} \\ &\cong \mathbf{H}\mathbf{x} \end{aligned}$$

where $\mathbf{H} = \mathbf{A} - \mathbf{a}\mathbf{v}^\top$. □

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 [37]:

$$\mathbf{H}_{3\times 3} \cong \mathbf{A}_{3\times 3} - \mathbf{e}'\mathbf{v}^\top \tag{2.7}$$

Here $\mathbf{v} = (v_1, v_2, v_3)^\top$ defines the 3D plane[1]. $[\mathbf{e}']_\times \mathbf{A} = \mathbf{F}$ 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 ([37], pp.316).

Given K planes in the scene, we have K homography matrices $\{\mathbf{H}_i | i = 1, 2, ..., K\}$, one for each plane. The *measurement matrix* $\mathbf{W}_{9\times K}$ is constructed by reshaping each $\mathbf{H}_i$ into a 9-dimensional column vector, and then stack them together. The rank of $\mathbf{W}$ is known to be at most *four* [80]. 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

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

estimate the homographies for small pre-segmented planar patches [107].

2.3 Parametric representation of affine transformation

Affine camera [68] 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 [79, 36].

In Section 2.2, the derivation of parametric homography representation $\mathbf{H} \cong \mathbf{A} - \mathbf{a}\mathbf{v}^\top$ assumes that the camera projection matrix has the canonical form of $[\mathbf{I}_{3\times3} \,|\, \mathbf{0}_{3\times1}]$ (see Eq. (2.2)), which does not apply to affine camera. The canonical form of affine camera projection matrix is:

$$P = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

As we can see, the left 3×3 matrix in the affine camera matrix has rank two. As a result, non-homogeneous projection matrix is often used in affine cameras:

$$\mathbf{x} = \mathbf{M}_{2\times3}\mathbf{X} + \mathbf{T}_{2\times1} \tag{2.8}$$

where the image point $\mathbf{x}$ is the projection of the scene point $\mathbf{X}$, both are in non-homogeneous coordinates, and $\{\mathbf{M}, \mathbf{T}\}$ is the affine camera projection matrix.

In this section, we first show that a plane in the scene viewed by two affine cameras induces an affine homography. Then we will show the parametric representation of affine transformation.

2.3.1 Planar affine transformation

The following theorem shows that a 3D plane in the scene induces an affine homography between two affine cameras.

Theorem 2.2. *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 $\mathbf{m}_{2\times 3} = [\mathbf{a}_{2\times 2}, \mathbf{t}_{2\times 1}]$.

Proof: Since three non-collinear points uniquely determine an affine transformation, we only need to show that all three-point triplets from the same plane share the same affine transformation. Specifically, we need to show that for *any* point $P \in \pi$, its imaging points p and p' in two cameras is related by $p' = \mathbf{a}p + \mathbf{t}$, where $\mathbf{m}_{2\times 3} = [\mathbf{a}_{2\times 2}, \mathbf{t}_{2\times 1}]$ 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 $[\mathbf{M}_{2\times 3}, \mathbf{T}_{2\times 1}]$ and $[\mathbf{M}'_{2\times 3}, \mathbf{T}'_{2\times 3}]$ respectively. We have:

$$[p_1, p_2, p_3] = \mathbf{M}\,[P_1, P_2, P_3] + \mathbf{T_1} \tag{2.9}$$

$$[p'_1, p'_2, p'_3] = \mathbf{M}'\,[P_1, P_2, P_3] + \mathbf{T'_1} \tag{2.10}$$

A 2D affine transformation $\mathbf{m}_{2\times 3} = [\mathbf{a}_{2\times 1}, \mathbf{t}_{2\times 1}]$ is *uniquely* determined by these three non-collinear matched pairs:

$$[p'_1, p'_2, p'_3] = \mathbf{a}[p_1, p_2, p_3] + \mathbf{t} \tag{2.11}$$

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

$$P = \alpha P_1 + \beta P_2 + \gamma P_3 \tag{2.12}$$

where $\alpha + \beta + \gamma = 1$ [2]. The image of P under camera $\{\mathbf{M}', \mathbf{T}'\}$ is:

$$\begin{aligned}
p' &= \mathbf{M}'P + \mathbf{T}' \\
&= \mathbf{M}'(\alpha P_1 + \beta P_2 + \gamma P_3) + (\alpha + \beta + \gamma)\mathbf{T}' \qquad (Eq.\ 2.12) \\
&= \alpha p'_1 + \beta p'_2 + \gamma p'_3 \qquad (Eq.\ 2.10) \\
&= \mathbf{a}(\alpha p_1 + \beta p_2 + \gamma p_3) + \mathbf{t} \qquad (Eq.\ 2.11) \\
&= \mathbf{a}(\alpha \mathbf{M}P_1 + \beta \mathbf{M}P_2 + \gamma \mathbf{M}P_3 + (\alpha + \beta + \gamma)\mathbf{T}) + \mathbf{t} \qquad (Eq.\ 2.9) \\
&= \mathbf{a}p + \mathbf{t}
\end{aligned}$$

Therefore, the image of any point on plane π undergoes the same 2D affine motion

[2] Let P_1 be the origin and $P_2 - P_1$ and $P_3 - P_1$ the two bases of the plane. Then for any point P, we have $P - P_1 = \beta(P_2 - P_1) + \gamma(P_3 - P_1)$. It turns out $P = (1 - \beta - \gamma)P_1 + \beta P_2 + \gamma P_3$.

$\mathbf{m}$. □

2.3.2 Parametric representation of affine transformation

The affine transformation induce by a plane in the scene can be represented by the parameters of plane equation and camera geometry, as shown in the following theorem:

Theorem 2.3. *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: $\mathbf{m} = \mathbf{m_r} + \mathbf{e'v}^\top$, where $\mathbf{m_r}$ is the affine transformation induced by reference plane π_r, $\mathbf{e'} = (e_1, e_2)^\top$, and the homogeneous coordinates $(e_1, e_2, 0)$ is the direction of epipolar lines in camera ψ'. $\mathbf{v}^\top = (v_1, v_2, v_3)$ is a 3-vector independent of camera ψ'.*

In the following we provide an illustrative geometric-base proof. In Appendix 2.7.1 we provide a general algebraic proof that applies to both perspective camera and affine camera.

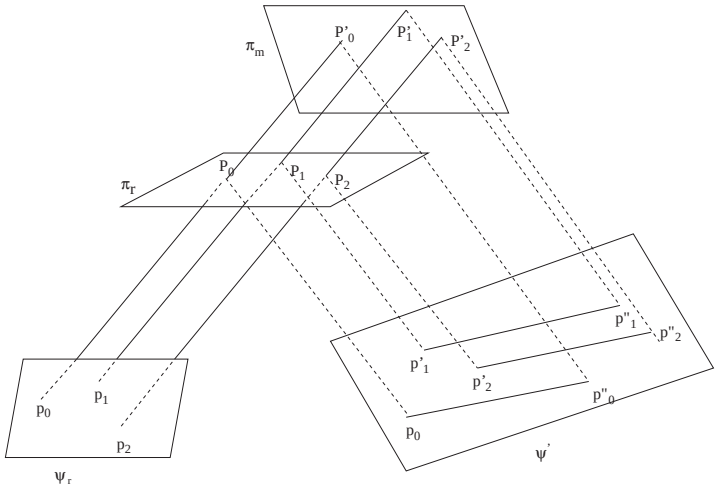


Figure 2.2: The relationship between 3D planes and affine cameras.

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 $p_i = (x, y)^\top$ and $p'_i = (x', y')^\top$ 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.(2.2).

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

$$\begin{bmatrix} p'_0 & p'_1 & p'_2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_r \\ 0\ 0\ 1 \end{bmatrix} * \begin{bmatrix} p_0 & p_1 & p_2 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.13}$$

$$\begin{bmatrix} p''_0 & p''_1 & p''_2 \\ 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{m}_{2\times 3} \\ 0\ 0\ 1 \end{bmatrix} * \begin{bmatrix} p_0 & p_1 & p_2 \\ 1 & 1 & 1 \end{bmatrix} \tag{2.14}$$

Since affine camera has parallel projection, $[\overline{P_0P'_0}, \overline{P_1P'_1}, \overline{P_2P'_2}]$ are three parallel line segments. Parallelism is preserved by affine camera. Therefore, $[\overline{P_0P'_0}, \overline{P_1P'_1}, \overline{P_2P'_2}]$ will project onto parallel line segments $[\overline{p'_0p''_0}, \overline{p'_1p''_1}, \overline{p'_2p''_2}]$ (epipolar lines) in affine camera ψ' whose projection matrix is $\{\mathbf{M'}_{2\times 3}, \mathbf{T'}\}$. Denote $\overline{p_ip_j} = p_j - p_i$. We have:

$$\begin{aligned} [\overline{p'_0p''_0}, \overline{p'_1p''_1}, \overline{p'_2p''_2}] &= \mathbf{M'} * [\overline{P_0P'_0}, \overline{P_1P'_1}, \overline{P_2P'_2}] \\ &= \mathbf{M'} * \mathbf{D} * [k_0, k_1, k_2]\,, \end{aligned} \tag{2.15}$$

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 $[\overline{P_0P'_0}, \overline{P_1P'_1}, \overline{P_2P'_2}] = \mathbf{D} * [k_0, k_1, k_2]$, 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 $\mathbf{e'} = [e_1, e_2]^\top = \mathbf{M'} * \mathbf{D}$ (It is obvious that $[e_1, e_2, 0]^\top$ is the direction of epipolar lines in homogeneous coordinates in camera ψ'). From Eq.(2.15) we have:

$$\begin{aligned} [p''_0, p''_1, p''_2] &= [p'_0, p'_1, p'_2] + [\overline{p'_0p''_0}, \overline{p'_1p''_1}, \overline{p'_2p''_2}] \\ &= [p'_0, p'_1, p'_2] + \mathbf{e'} * [k_0, k_1, k_2] \end{aligned} \tag{2.16}$$

Substitute Eq.(2.16) and Eq.(2.13) into Eq.(2.14), we have:

$$\begin{aligned} &\begin{bmatrix} \mathbf{m}_{2\times 3} \\ 0\ 0\ 1 \end{bmatrix} * \begin{bmatrix} p_0 & p_1 & p_2 \\ 1 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{m}_r \\ 0\ 0\ 1 \end{bmatrix} * \begin{bmatrix} p_0 & p_1 & p_2 \\ 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} \mathbf{e'} \\ 0 \end{bmatrix} * [k_0, k_1, k_2] \end{aligned} \tag{2.17}$$

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 $P_{3\times3}^{-1}$ exists. Therefore, from Eq.(2.17), we have:

$$\mathbf{m} = \mathbf{m}_r + \mathbf{e}' * [v_0, v_1, v_2] \tag{2.18}$$

Here $\left[\mathbf{e}'^{\top}, 0\right]$ is the direction of epipolar lines in homogeneous coordinate in camera ψ', and

$$\mathbf{v}^{\top} = [v_0, v_1, v_2] = [k_0, k_1, k_2] * P_{3\times3}^{-1}$$

It is obvious that the 3-vector $\mathbf{v}^{\top}$ is independent of the second camera ψ'. □

2.4 Relative affine homographies

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.(2.7) as (see Theorem 2.3):

$$\mathbf{m}_{2\times3} = \mathbf{m_r} + \mathbf{e}'\mathbf{v}^{\top}. \tag{2.19}$$

Here $\mathbf{m_r}$ is the affine transformation induced by the reference plane. $\mathbf{e}' = (e_1, e_2)^{\top}$, where $(e_1, e_2, 0)$ is the direction of epipolar lines in homogeneous coordinate in the second camera. The 3-vector $\mathbf{v}$ representing the plane is independent of the second affine camera.

Notice an important difference between Eq.(2.7) and (2.19). Eq.(2.7) has an unknown scale while Eq.(2.19) does not. Therefore, we can define *relative affine transformation* as:

$$\Delta\mathbf{m} = \mathbf{m} - \mathbf{m}_r = \mathbf{e}'\mathbf{v}^{\top}. \tag{2.20}$$

where $\mathbf{m}_r$ is the affine transformation induced by the reference plane, which can be either a real plane or a virtual plane. A convenient choice of the reference affine transformation is the average affine transformation:

Result 2.1. *If* $\{\mathbf{m}_i \,|\, i = 1, ..., N\}$ *are* N *affine transformations induced by* N *plane in the scene, then the average affine transformation* $\mathbf{m}_r = \frac{1}{N}\sum_i^N \mathbf{m}_i$ *is induced by some real or virtual world plane.*

Proof: An affine transformation $\mathbf{m}$ is induced by some world plane *if and only if*

$\mathbf{m}$ is consistent with the fundamental matrix. Since each $\mathbf{m}_i$ corresponds to a world plane, we have $\mathbf{m}_i^\top \mathbf{F} + \mathbf{F}^\top \mathbf{m}_i = 0$ where $\mathbf{F}$ is the fundamental matrix defined by the two cameras. It follows $\mathbf{m}_r^\top \mathbf{F} + \mathbf{F}^\top \mathbf{m}_r = 0$, which means that $\mathbf{m}_r$ is consistent with the fundamental matrix. Therefore, $\mathbf{m}_r$ is induced by some (*virtual*) world plane. □

2.5 Subspace of relative affine homographies: static scene

In this section, we study the subspace dimension of relative affine homographies induced by planar patches in the scene, where either the scene is static and the camera is moving, or vise versa.

Result 2.2. *We are given two views of a static scene consisting of K planar patches. The collection of all relative affine transformations induced by these K planar patches resides in a linear subspace with dimension $d = \min(3, L-1)$, where L is the number of different planes in the scene.*

Proof: From Eq.(2.19) we have $\Delta\mathbf{m}_i = \mathbf{m}_i - \mathbf{m}_r = \mathbf{e}'\mathbf{v}_i^\top$. Here $\mathbf{m}_i$ is the affine transformation induced by the i-th planar patch; $\mathbf{m}_r$ is the reference affine transformation; and $\mathbf{v}_i = [v_{1,i}, v_{2,i}, v_{3,i}]^\top$ defines the i-th planar patch. Reshape each $\Delta\mathbf{m}_i$ into a 6×1 column vector, and stack them into a matrix $\mathbf{W}_{6\times K}$. The following factorization is obvious [80]:

$$\begin{aligned}\mathbf{W}_{6\times k} &= \begin{bmatrix} e_1 & 0 & 0 \\ 0 & e_1 & 0 \\ 0 & 0 & e_1 \\ e_2 & 0 & 0 \\ 0 & e_2 & 0 \\ 0 & 0 & e_2 \end{bmatrix}_{6\times 3} * \begin{bmatrix} v_{1,1} & \dots & v_{1,K} \\ v_{2,1} & \dots & v_{2,K} \\ v_{3,1} & \dots & v_{3,K} \end{bmatrix}_{3\times K} \\ &= \mathbf{E}_{6\times 3} * \mathbf{V}_{3\times K} \end{aligned} \tag{2.21}$$

We have

$$\text{rank}(\mathbf{W}) \leq \min(\text{rank}(\mathbf{E}), \text{rank}(\mathbf{V})) = \min(3, \text{rank}(\mathbf{V}))$$

where $\text{rank}(\mathbf{V})$ depends on the configuration of the planes in the scene, and is no

more than L, the number of different planes in the scene.

If one of the plane in the scene is selected as the reference plane, then columns in $\mathbf{V}$ from patches in the reference plane will become zero, and $\text{rank}(\mathbf{V}) \leq L-1$. If the average affine transformation is selected as the reference motion, then summing all columns in $\mathbf{V}$ together will result in a zero column vector, which means that $\text{rank}(\mathbf{V}) \leq L-1$. In the following of this chapter regarding to $\text{rank}(\mathbf{W})$, the above argument applies, and we always have $\text{rank}(\mathbf{W}) \leq L-1$. □

We now show that the collection of all relative affine transformations across more than two views still resides in a low dimensional (no more than 3) linear subspace.

Result 2.3. *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 linear subspace with dimension $d = \min(3, L-1)$, where L is the number of different planes in the scene.*

Proof: Denote the K affine transformations between reference view and view f as $\mathbf{m}_{f,1}, ..., \mathbf{m}_{f,k}$. From Eq.(2.19) we have $\Delta\mathbf{m}_{f,i} = \mathbf{m}_{f,i} - \mathbf{m}_{f,r} = \mathbf{e}'_f \mathbf{v}_i^\top$, where $\mathbf{v}_i = [v_{1,i}, v_{2,i}, v_{3,i}]^\top$. Reshape each $\Delta\mathbf{m}_i$ into a 6×1 column vector, and stack them into a matrix $\mathbf{W}^f_{6\times K}$. We have:

$$\begin{aligned}\mathbf{W}^f_{6\times K} &= \begin{bmatrix} e_{f,1} & 0 & 0 \\ 0 & e_{f,1} & 0 \\ 0 & 0 & e_{f,1} \\ e_{f,2} & 0 & 0 \\ 0 & e_{f,2} & 0 \\ 0 & 0 & e_{f,2} \end{bmatrix}_{6\times 3} * \begin{bmatrix} v_{1,1} & ... & v_{1,K} \\ v_{2,1} & ... & v_{2,K} \\ v_{3,1} & ... & v_{3,K} \end{bmatrix}_{3\times K} \\ &= \mathbf{E}^f_{6\times 3} * \mathbf{V}_{3\times K} \end{aligned} \tag{2.22}$$

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

$$\mathbf{W}_{6F\times K} = \begin{bmatrix} \mathbf{W}^1 \\ \mathbf{W}^2 \\ ... \\ \mathbf{W}^F \end{bmatrix}_{6F\times K} = \begin{bmatrix} \mathbf{E}^1 \\ \mathbf{E}^2 \\ ... \\ \mathbf{E}^F \end{bmatrix}_{6F\times 3} * \mathbf{V}_{3\times K} \tag{2.23}$$

The matrix dimension on the right-hand side of Eq.(2.23) implies that

$$\text{rank}(\mathbf{W}) \leq \min(3, L-1)$$

□

From Eq.(2.23) we can see that the subspace comes from both the spatial and temporal redundancy in the image sequence. The spatial redundancy is due to the fact that multiple planes share the common camera geometry, i.e., the direction of parallel epipolar lines. The temporal redundancy is due to the fact that multiple cameras share the same scene. The actual dimension of the subspace, i.e., the rank of W in Eq.(2.23), depends on the scene and the camera geometry, and could be *lower* than three.

Result 2.4. *If all of the planes in the scene different only in depth, i.e., they are parallel to each other, then* $\text{rank}(\mathbf{W}) = 1$.

Proof: We just need to show that any two of the relative affine motions are linear dependent, i.e., $\Delta\mathbf{m}_i = \lambda\Delta\mathbf{m}_j$.

Since all planes are parallel to each other, we denote the i-th plane equation as $\pi_i = (\mathbf{n}^\top, \mu_i)^\top$, where $\mathbf{n}^\top$ is the common plane normal. According to Eq.(2.36):

$$\mathbf{m}_i = \mathbf{P}'\mathbf{P}^+ - \mathbf{e}'\frac{(\mathbf{n}^\top, \mu_i)^\top\mathbf{P}^+}{(\mathbf{n}^\top, 0)^\top\mathbf{C}} \qquad (\mathbf{C} = (c_1, c_2, c_3, 0)^\top \text{ is at infinity.})$$

We choose the average affine transformation as the reference transformation:

$$\mathbf{m}_r = \mathbf{P}'\mathbf{P}^+ - \mathbf{e}'\frac{(\mathbf{n}^\top, \bar{\mu})^\top\mathbf{P}^+}{(\mathbf{n}^\top, 0)^\top\mathbf{C}}$$

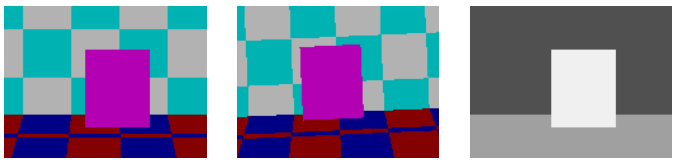
where $\bar{\mu} = \frac{1}{K}\sum_{i=1}^{K}\mu_i$, K is the number of collected affine transformations.

$$\Delta\mathbf{m}_i = \alpha_i\frac{\mathbf{e}'\mathbf{p}_3^+}{(\mathbf{n}^\top, 0)^\top\mathbf{C}}$$

where $\alpha_i = 1/(\bar{\mu} - \mu_i)$, $\mathbf{p}_3^+$ is the third row of $\mathbf{P}^+$.

Therefore,

$$\frac{\Delta\mathbf{m}_i}{\Delta\mathbf{m}_j} = \frac{\alpha_i}{\alpha_j}$$



(a) (b) (c)

449.0	261.2	0.006	0.0007	0.0002	0.0001

(d)

Figure 2.3: 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 $\mathbf{W}_{6\times 31}$.

□

Note that the proof of Result 2.4 uses the fact that the optical center of an affine camera is at infinity. For a finite camera, when the camera motion is in plane motion (not translation in depth direction; not out-of-plane rotation), the 2D image motion is also affine. In this case, if we set the optical center of the first camera as the origin of the world coordinate frame, then $\pi_i^\top \mathbf{C} = 1$ (independent of plane π_i), therefore Result 2.4 still holds.

2.6 Subspace dimensionality: dynamic scene

The assumption of static scenes for deriving Eq.(2.23) 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.(2.3) shows the two rendered frames. Notice each plane is made of many color patches (each color patch is a planar patch). With two views ($F = 1$), and $k = 31$ patches (1 on foreground plane, 15 on background plane, 15 on table plane), the eigenvalues of $\mathbf{W}_{6\times 31}$ are computed and shown in Fig.(2.3d). They clearly show that the dimension of subspace is *two*. Please see

Appendix 2.7.2 for an analytic explanation of why the subspace dimension is two.

This section studies the subspace dimensionality for dynamic scenes.

2.6.1 General motion

Suppose there are L planes in the scene, each moving independently. We are given two views of the scene, with one view being the reference view. Between the reference view and the other view, there is *one* fundamental matrix associated with each independent moving plane.

The homography induced by the i-th plane in the scene is:

$$\mathbf{H}_i \cong \mathbf{A}_i - \mathbf{e}'_i \mathbf{v}^\top \tag{2.24}$$

where $\mathbf{F} = [\mathbf{e}'_i]_\times \mathbf{A}$ is any decomposition of the fundamental matrix associated with the i-th plane. The epipole $\mathbf{e}'_i$ is determined by the *relative* motion between the camera and the i-th plane. The reference homography $\mathbf{A}_i$ is induced by some unknown plane that is static w.r.t. the i-th plane.

We first study the rank condition in general case where the planes in the scene move arbitrarily.

Result 2.5. *Suppose there are B "bodies" in the scene. Each body B_i contains L_i different planes. While each body moves independently, planes inside each body are static w.r.t. each other. The single camera is moving by itself too. We are given two views of the scene, and K affine transformations of local planar patches. These K affine transformations reside in a linear subspace of dimension $d = \min(4B, L-1)$, where $L = \sum_{i=1}^{B} L_i$.*

Proof: Reshape each affine transformation into a column vector. Eq.(2.24) can be

rewritten as:

$$\mathbf{h}_i = \begin{bmatrix} & e_1 & 0 & 0 \\ & 0 & e_1 & 0 \\ & 0 & 0 & e_1 \\ \mathbf{a}_i & e_2 & 0 & 0 \\ & 0 & e_2 & 0 \\ & 0 & 0 & e_2 \end{bmatrix}_{6\times 4} * \begin{bmatrix} 1 \\ v_1 \\ v_2 \\ v_3 \end{bmatrix}$$
$$= \mathbf{E}_i * \mathbf{v}_i \tag{2.25}$$

Planes in the same "body" share the same matrix $\mathbf{E}$. Stack all of the reshaped affine transformation into a single measurement matrix $\mathbf{W}$. For the i-th "body", we have $\mathbf{W}_i = \mathbf{E}_i\mathbf{V}_i$, where $\mathbf{V}_i$ is a $4 \times K_i$ matrix with K_i the number of local patches in Body B_i and $\sum_{i=1}^{B} K_i = K$. We have:

$$\mathbf{W}_{6\times K} = [\mathbf{W}_1, ..., \mathbf{W}_B] = [\mathbf{E}_1, ..., \mathbf{E}_B]_{6\times 4B} \begin{bmatrix} \mathbf{V}_1 & & & \\ & \mathbf{V}_2 & & \\ & & \ddots & \\ & & & \mathbf{V}_B \end{bmatrix}_{4B\times K} = \mathbf{EV}$$

where K is the total number of local measurements (affine transformation). From the above matrix factorization, we have:

$$\text{rank}(\mathbf{W}) \leq \min(6, \text{rank}(\mathbf{E}), \text{rank}(\mathbf{V}))$$

Since $\text{rank}(\mathbf{V}) = \sum_{i=1}^{B} \text{rank}(\mathbf{V}_i) \leq L$, we have:

$$\text{rank}(\mathbf{W}) \leq \min(6, 4B, L)$$

If we subtract the column average of $\mathbf{W}$ from every column in $\mathbf{W}$, we further reduce the rank of $\mathbf{W}$ by one if its rank is bound by L. Therefore we have: $\text{rank}(\mathbf{W}) \leq \min(6, 4B, L-1)$. □

We have similar result for multiple frame case:

Result 2.6. *Given multiple frames of the scene described in Result 2.5, we have* $\text{rank}(\mathbf{W}) \leq \min(4B, L-1)$.

Proof: Denote $\mathbf{E}_i^f$ the epipole matrix of the i-th "body" at Frame f. We have:

$$\mathbf{W}_{6F\times K} = \begin{bmatrix} \mathbf{E}_1^1, & ..., & \mathbf{E}_B^1 \\ & ... & \\ \mathbf{E}_1^F, & ..., & \mathbf{E}_B^F \end{bmatrix}_{6F\times 4B} \begin{bmatrix} \mathbf{V}_1 & & & \\ & \mathbf{V}_2 & & \\ & & \ddots & \\ & & & \mathbf{V}_B \end{bmatrix}_{4B\times K} = \mathbf{EV}$$

It is obvious that

$$\text{rank}(\mathbf{W}) \leq \min(4B, L-1)$$

□

2.6.2 Special motions

In real life, objects do not move pure randomly. This subsection examines a practical special motion, the *instantaneous* motion.

Suppose the relative motion between the i-th plane and the camera is *instantaneous* motion. In other words, the object has small rotation relative to the camera, and its relative forward motion is small compared to its depth. The image motion of the plane between two views can be modelled by an 8-vector $\mathbf{h}_i = (p_1, ..., p_8)^\top$(see [41]), where:

$$\begin{aligned} p_1 &= f'(\gamma e_1 + \omega_2) & p_2 &= \tfrac{f'}{f}(1+\alpha e_1) - \gamma e_3 - 1 \\ p_3 &= -\tfrac{f'}{f}(\omega_3 - \beta e_1) & p_4 &= f'(\gamma e_2 - \omega_1) \\ p_5 &= \tfrac{f'}{f}(\omega_3 + \alpha e_2) & p_6 &= \tfrac{f'}{f}(1+\beta e_2) - \gamma e_3 - 1 \\ p_7 &= \tfrac{1}{f}(\omega_2 - \alpha e_3) & p_8 &= -\tfrac{1}{f}(\omega_1 + \beta e_3), \end{aligned} \tag{2.26}$$

where $\pi = (\alpha, \beta, \gamma)^\top$ is the parameters of plane π ($\pi\mathbf{X} = 1$, for all $\mathbf{X}$ on the plane), $\omega = (\omega_1, \omega_2, \omega_3)^\top$ and $\mathbf{e} = (e_1, e_2, e_3)^\top$ are the rotation and translation of the plane w.r.t. the camera, f and f' are camera focus length at the two views.

The image motion vector (between view j and the reference view) of the i-th plane

can be represented as:

$$\mathbf{h}_i = \begin{bmatrix} fe_1^j & 0 & 0 & f\omega_2^j \\ -e_3^j & e_1^j & 0 & 0 \\ 0 & 0 & e_1^j & -\omega_3^j \\ fe_2^j & 0 & 0 & -f\omega_1^j \\ 0 & e_2^j & 0 & \omega_3^j \\ -e_3^j & 0 & e_2^j & 0 \\ 0 & -\frac{1}{f}e_3^j & 0 & \frac{1}{f}\omega_2^j \\ 0 & 0 & -\frac{1}{f}e_3^j & -\frac{1}{f}\omega_1^j \end{bmatrix} \begin{bmatrix} \gamma \\ \alpha \\ \beta \\ 1 \end{bmatrix} = \mathbf{E}_i^j \mathbf{v}_i \tag{2.27}$$

where f is the camera focus length which is assumed to be fixed, $(\omega^j, \mathbf{e}^j)$ is the instantaneous motion of the i-th plane w.r.t. camera at time j.

If we stack the image motions all local planar patches across all frames into a measurement matrix, we have:

Result 2.7. *Suppose there are B "bodies" in the scene. Each body B_i contains L_i different planes. While each body moves independently, planes inside each body are static w.r.t. each other. The single camera is moving by itself too. Suppose the relative motion between camera and bodies are instantaneous. We are given $F+1$ views of the scene (with one of them being the reference view), and K instantaneous image transformations of local planar patches across all frames. These K transformations reside in a linear subspace of dimension $d = \min(4B, L-1)$, where $L = \sum_{i=1}^{B} L_i$.*

Proof: According to equation (2.27), we have:

$$\mathbf{W}_{8F\times K} = \begin{bmatrix} \mathbf{E}_1^1 & ... & \mathbf{E}_B^1 \\ & ... & \\ \mathbf{E}_1^F & ... & \mathbf{E}_B^F \end{bmatrix}_{8F\times 4B} \begin{bmatrix} \mathbf{V}_1 & & & \\ & \mathbf{V}_2 & & \\ & & \ddots & \\ & & & \mathbf{V}_B \end{bmatrix}_{4B\times K} = \mathbf{E}\mathbf{V} \tag{2.28}$$

From the above factorization, it is obvious that:

$$\text{rank}(W) \leq \min(4B, L-1)$$

□

If we can model the object motion, we can further discover a lower dimensional subspace. Suppose the instantaneous motion of the i-th plane can be modelled as:

$$\mathbf{e}_i = \sum_{j=1}^{N} \alpha_j t^j = \begin{bmatrix} \alpha_{1,1} & ... & \alpha_{1,N} \\ \alpha_{2,1} & ... & \alpha_{2,N} \\ \alpha_{3,1} & ... & \alpha_{3,N} \end{bmatrix}_{3\times N} \begin{bmatrix} t \\ \vdots \\ t^N \end{bmatrix} = \mathbf{A}\mathbf{t} \tag{2.29a}$$

$$\omega_i = \sum_{j=1}^{N} \beta_j t^j = \begin{bmatrix} \beta_{1,1} & ... & \beta_{1,N} \\ \beta_{2,1} & ... & \beta_{2,N} \\ \beta_{3,1} & ... & \beta_{3,N} \end{bmatrix}_{3\times N} \begin{bmatrix} t \\ \vdots \\ t^N \end{bmatrix} = \mathbf{B}\mathbf{t} \tag{2.29b}$$

where $\alpha_j = (\alpha_{j,1}, \alpha_{j,2}, \alpha_{j,3})^\top$ and $\beta_j = (\beta_{j,1}, \beta_{j,2}, \beta_{j,3})^\top$, and t is time parameter.

Equation (2.29) can model most of the rigid motion in the real world (many non-rigid motion too), where N is the number of bases. A lot of object motion in a short time period can be well modelled with $N = 2$. Two special cases are useful in our case, including $N = 1$ (constant speed), and $N = 2$ (acceleration/deceleration).

Result 2.8. *If the motions (w.r.t. camera) of the planes in the scene can be modelled by equation (2.29), then the image motions (8-parameter model) of the planar patches across all frames reside in a linear subspace with dimension* $d = \min(8N, 4B, L-1)$. *For affine cameras,* $d = \min(6N, 4B, L-1)$.

Proof: We just need to show $\text{rank}(\mathbf{W}) \leq 8N$. Substitute equation (2.29) into (2.27), we have:

$$\begin{aligned} \mathbf{h}_i &= diag(f,1,1,f,1,1,1/f,1/f) \begin{bmatrix} \mathbf{t}^\top & & \\ & \ddots & \\ & & \mathbf{t}^\top \end{bmatrix}_{8\times 8N} \begin{bmatrix} \mathbf{a}_1^\top & \mathbf{0} & \mathbf{0} & \mathbf{b}_2^\top \\ -\mathbf{a}_3^\top & \mathbf{a}_1^\top & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{a}_1^\top & -\mathbf{b}_3^\top \\ \mathbf{a}_2^\top & \mathbf{0} & \mathbf{0} & -\mathbf{b}_1^\top \\ \mathbf{0} & \mathbf{a}_2^\top & \mathbf{0} & \mathbf{b}_3^\top \\ -\mathbf{a}_3^\top & \mathbf{0} & \mathbf{a}_2^\top & \mathbf{0} \\ \mathbf{0} & -\mathbf{a}_3^\top & \mathbf{0} & \mathbf{b}_2^\top \\ \mathbf{0} & \mathbf{0} & -\mathbf{a}_3^\top & -\mathbf{b}_1^\top \end{bmatrix}_{8N\times 4} \mathbf{v}_i \\ &= \mathbf{F}_{8\times 8}\mathbf{T}_{8\times 8N}\mathbf{E}_{8N\times 4}\mathbf{v}_i \end{aligned} \tag{2.30}$$

where $\mathbf{a}_r$ is the r-th row of matrix $\mathbf{A}$, $\mathbf{b}_r$ is the r-th row of matrix $\mathbf{B}$ (see (2.29) for

the definition of $\mathbf{A}$ and $\mathbf{B}$). The measurement matrix $\mathbf{W}$ can be factorized as:

$$\mathbf{W}_{8F\times K} = \begin{bmatrix} \mathbf{T}_1 \\ \vdots \\ \mathbf{T}_F \end{bmatrix}_{8F\times 8N} [\mathbf{E}_1 \cdots \mathbf{E}_B]_{8N\times 4B} \begin{bmatrix} \mathbf{V}_1 & & \\ & \ddots & \\ & & \mathbf{V}_B \end{bmatrix}_{4B\times K} \tag{2.31}$$

Therefore, $\text{rank}(\mathbf{W}) \leq 8N$. For affine cameras, the image motion has only six parameters and $\text{rank}(\mathbf{W}) \leq 6N$. Proof done by combining with Result 2.7. □

Given a short image sequence, constant speed assumption is valid in many cases, and the following result shows that under such assumption, the subspace dimension is low (6 for affine cameras) no matter how complex the scene is.

Result 2.9. *If every plane in the scene moves independently but with a constant speed (different planes can have different speed), then the image motions (8-parameter model) of the planar patches across all frames reside in a linear subspace with dimension* $d = \min(8, 4B, L)$. *For affine cameras,* $d = \min(6, 4B, L-1)$.

2.7 Appendix

2.7.1 Appendix 1: Algebraic proof of Theorem 2.3

Algebraic proof: Assuming the projection matrix of the two cameras are, respectively, $\mathbf{P}$ and $\mathbf{P}'$, with $\mathbf{P}$ the reference camera. Notice that we do not assume any canonical form of the cameras, i.e., $\mathbf{P}$ and $\mathbf{P}'$ are 3×4 matrix with rank 3. For any point $\mathbf{X}$ on the plane π, it projects onto the two cameras as (using homogeneous coordinate):

$$\mathbf{x} \cong \mathbf{P}\mathbf{X} \tag{2.32}$$

$$\mathbf{x}' \cong \mathbf{P}'\mathbf{X} \tag{2.33}$$

From (2.32), the ray corresponding to the line joined by $\mathbf{C}$ (the optical center of the first camera) and the point $\mathbf{X}$ is:

$$\mathbf{X}(\lambda) = \mathbf{P}^{+}\mathbf{x} + \lambda\mathbf{C} \tag{2.34}$$

where λ parameterizes this 1D line, and $\mathbf{P}^+ = \mathbf{P}^\top(\mathbf{P}\mathbf{P}^\top)^{-1}$ is the pseudo-inverse of $\mathbf{P}$. Note that $(\mathbf{P}\mathbf{P}^\top)^{-1}$ exists since $\text{rank}(\mathbf{P}\mathbf{P}^\top) = 3$ [3]. Since $\mathbf{X}$ is on both the ray and the plane $\pi\mathbf{X} = 0$, we can determine λ:

$$\lambda = -\frac{\pi^\top \mathbf{P}^+}{\pi^\top \mathbf{C}}\mathbf{x} \tag{2.35}$$

where $\pi^\top \mathbf{C} \neq 0$ since we ignore the degenerate case where the plane π contains $\mathbf{C}$.

From Eq.(2.33), (2.34) and (2.35), we have:

$$\begin{aligned}
\mathbf{x}' &\cong \mathbf{P}'\mathbf{X} \\
&\cong \mathbf{P}'\mathbf{P}^+\mathbf{x} - \mathbf{P}'\mathbf{C}\frac{(\pi^\top \mathbf{P}^+)}{\pi^\top \mathbf{C}}\mathbf{x} \\
&\cong (\mathbf{P}'\mathbf{P}^+ - \mathbf{e}'\mathbf{v}^\top)\mathbf{x} \\
&\cong \mathbf{H}\mathbf{x}
\end{aligned} \tag{2.36}$$

where $\mathbf{e}' = \mathbf{P}'\mathbf{C}$ is the epipole in the second camera (the image point of the first camera center); $\mathbf{v}^\top = \frac{\pi^\top \mathbf{P}^+}{\pi^\top \mathbf{C}}$ parameterizes the plane and is independent of the second camera; $\mathbf{P}'\mathbf{P}^+$ is the homography corresponding to some plane [4]. □

2.7.2 Appendix 2: Notes on the subspace dimension of the synthetic case of Figure (2.3)

Let's go back to see why the rank is two in the synthetic example of Figure (2.3), where the input are two frames of a scene with two planes (Body 1 and 2) moving up and down, and the ground plane (Body 3) fixed w.r.t. the camera. The camera is translating, rotating (around optical axis), and zooming.

From the point of view of scene regularity, we have $\text{rank}(W) \leq (L-1) = 2$, where $L = 3$ is the number of layers in the scene. In the following, we show it via the geometry relationship between the scene and camera.

The motion of a plane is exact affine (since there is not forward/backward trans-

[3] Since $\text{rank}(\mathbf{P}) = 3, \mathbf{P} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, the first three singular values in $\mathbf{\Sigma}$ are larger than 0. $\mathbf{P}\mathbf{P}^\top = \mathbf{U}\mathbf{\Sigma}^\mathbf{2}\mathbf{U}^\top \Rightarrow \text{rank}(\mathbf{P}\mathbf{P}^\top) = 3$.

[4] The fundamental matrix is $\mathbf{F} = [\mathbf{e}']_\times \mathbf{P}'\mathbf{P}^+$ ([37], pp.224). A transformation H is the homography between two images induced by some world plane if and only if the fundamental matrix $\mathbf{F}$ for the two images has a decomposition of $\mathbf{F} = [\mathbf{e}']_\times \mathbf{H}$ ([37], pp.316).

lation w.r.t. camera, and not out of plane rotation). The rotational component is the same for all planes in the scene since only the camera is rotating. Such rotation is cancelled when it is chosen as the reference motion. Therefore we have:

$$\mathbf{W}_{6\times K} = [\mathbf{E}_1, \mathbf{E}_2, \mathbf{E}_3] \begin{bmatrix} \mathbf{V}_1 & & \\ & \mathbf{V}_2 & \\ & & \mathbf{V}_3 \end{bmatrix} \tag{2.37}$$

where $\mathbf{V}_i$ contains all the planar patches in the i-th body. Since Body 1 and 2 moves in opposite direction (up and down), their corresponding epipolar matrix satisfies $\mathbf{E}_1 = -\mathbf{E}_2$. Therefore, we can rewrite equation (2.37) as:

$$\mathbf{W}_{6\times K} = [\mathbf{E}_1, \mathbf{E}_3] \begin{bmatrix} \mathbf{V}_1 & -\mathbf{V}_2 & \\ & & \mathbf{V}_3 \end{bmatrix} \tag{2.38}$$

Since Body 1 and 2 has the same plane normal (only differs in depth), we know that $\mathbf{V}_1 = \lambda \mathbf{V}_2$, with λ some constant. And since the bodies are all planes, we have $\text{rank}(\mathbf{V}_i) = 1$. Therefore:

$$\text{rank}(\mathbf{W}) \leq \text{rank}([\mathbf{V}_1, -\mathbf{V}_2]) + \text{rank}(\mathbf{V}_3) = 2$$

Chapter 3

Subspace Clustering for Extracting Layers from Image Sequences

3.1 Introduction

The major task for layer extraction is to segment the images into *some* number of sub-images, in such a way that pixels within each sub-image share some common 2D parametric transformation (or nonparametric model defined by dense smooth flow field [104]). The three major issues of layer extraction are:

- Segmentation: which portion of the image corresponds to one layer?
- Motion: what motion does a layer undergo?
- Number of layers: how many layers exist in the scene?

These three subproblems are coupled together, making layer segmentation a nontrivial task. On one hand, the model based motion estimation requires knowing the portion of image pixels that can be used to estimate the motion parameters, i.e., the segmentation result. On the other hand, segmentation is to assign pixels to appropriate layer models, which requires knowing the layer model parameters.

EM approach [51, 8, 105, 104, 96, 59] formulates the layer segmentation as a maximum likelihood or maximum a posterior (MAP) estimation. It solves the layer segmentation problem by iteratively solving the above three subproblems, with each

iteration dealing with only one subproblem by fixing the other two. Specifically, EM approach consists of:

- E-step solves the segmentation subproblem, assuming motions are known (including number of motion models/layers).
- M-step solves the motion estimation problem, assuming the layer segmentation is known.
- MDL-step determines the number of layers, assuming segmentation and motion are known.

Initialization (the number of models and the motion for each model) is an important but difficult step for EM approach [81, 96]. A good initialization is required in order for the iterations in the EM algorithm to converge to a desirable solution.

An alternating approach is to solve the layer extraction problem via grouping [103, 81, 106, 55, 56] in a bottom up way without the difficulty of initialization. The assumption is that the motion parameters of local blocks from the same layer will be similar to each other, and form a cluster in the parameter space. Layers can therefore be extracted by grouping local features (blocks or optical flows) into clusters in the parameter space.

Simply applying grouping algorithm to the local features tends to ignore the global constraints existing in the given image sequence, including spatial and temporal constraints. In this chapter, we show that subspace approach can naturally utilize the global spatial-temporal constraints to improve the cluster discriminability, therefore improve the reliability of subsequent clustering algorithm for layer extraction.

3.2 Clustering for layer extraction

According to Theorem 2.1, a plane $\pi^\top \mathbf{X} = 0$ in the scene induces a 2D homography $\mathbf{H}$ between two camera views. The parameters of the homography $\mathbf{H}$ are determined by both the camera geometry and the plane parameters:

$$\mathbf{H} = \mathbf{A} - \mathbf{a}\mathbf{v}^\top$$

where $\mathbf{A}$ and $\mathbf{a}$ are camera parameters, and $\mathbf{v}$ the plane parameters ($\pi = (\mathbf{v}^\top, 1)^\top$). In the following we consider affine cameras, where the plane π induces a 2D affine homography between two affine cameras (Theorem 2.3).

In the bottom-up frame work, the image is divided into local blocks, and each local block is assumed to be the image of a planar patch in the scene. Denote $\mathbf{m}_i$ the local measurement of the i-th planar patch. Given $F+1$ frames in the input video, $\mathbf{m}_i$ is a $6F$ dimensional column vector, which consists of the affine motion parameters between the reference image and every other image in the given image sequence. Suppose the number of local measurements (planar patches) is K, we have:

$$\mathbf{m}_i = \mathbf{m}_l + \epsilon_i \qquad i = 1, 2, ..., K \tag{3.1}$$

where $\mathbf{m}_l$ is the hidden *true* affine homographies of the l-th plane in the scene that contains patch i, and ϵ_i is the measurement noise. We assume that the random variables $\{\epsilon_i \,|\, i = 1, ..., K\}$ are independently and identically distributed (i.i.d.), with *zero* mean and covariance of $\mathbf{\Sigma}_\epsilon$.

Ideally, local measurements of planar patches in the l-th scene plane are identical, since they share the same camera geometry and plane parameters. In reality, due to the measurement noise ϵ_i, local measurements from one common plane will form a cluster centered around $\mathbf{m}_l$ in the parameter space. The shape of the cluster depends on the noise. The task of layer extraction can therefore be formulated as clustering in the motion parameter space.

3.3 Subspace clustering

According to the results in Chapter 2, the true homographies $\{\mathbf{m}_l, l = 1, ..., L\}$ reside in a low dimensional subspace. Instead of grouping in the original high dimensional space, we formulate the layer extraction as clustering in the low dimensional subspace.

Denote the columns of $\mathbf{U}_{6F\times 6F}$ as a set of orthonormal bases of the original $6F$-dimensional parameter space $\mathbb{R}^{6F}$, which consists of the signal space (homography subspace) $\mathbf{U}_S$ and the noise space $\mathbf{U}_\perp$:

$$\mathbf{U} = [\mathbf{U}_S \,|\, \mathbf{U}_\perp]$$

Here $\mathbf{U}$ is an orthonormal matrix. $\mathbf{U}_S$ and $\mathbf{U}_\perp$ are orthogonal ($\mathbf{U}_S^\top \mathbf{U}_\perp = \mathbf{0}$).

According to Eq. (3.1), the local measurement $\mathbf{m}_i$ can be decomposed into two components: the signal component $\mathbf{m}_l$ and the noise component ϵ_i. The true homographies $\{\mathbf{m}_l \,|\, l = 1, ..., L\}$ of the scene planes (layers) resides strictly in the signal subspace. Therefore the projection of $\mathbf{m}_l$ on the noise space vanishes:

$$\mathbf{U}_\perp^\top \mathbf{m}_l = 0$$

The signal decomposition can be achieved by projecting the local measurements onto the signal space and noise space:

$$\begin{aligned} \mathbf{m}_i &= \mathbf{m}_l + \epsilon_i \\ &= \mathbf{U}\mathbf{U}^\top \mathbf{m}_l + \mathbf{U}\mathbf{U}^\top \epsilon_i \\ &= \mathbf{U}_S \mathbf{v}_l + [\mathbf{U}_S | \mathbf{U}_\perp] \mathbf{n}_i \end{aligned} \tag{3.2}$$

where $\mathbf{v}_l = \mathbf{U}_S^\top \mathbf{m}_l$, $\mathbf{n}_i = \mathbf{U}^\top \epsilon_i$.

Since ϵ_i's are i.i.d. zero mean Gaussian noise, $\{\mathbf{n}_i \,|\, i = 1, 2, ..., K\}$ are K i.i.d. multivariate random variables with zero mean and covariance of (see [52]):

$$\mathbf{\Sigma}_n = \mathbf{U}^\top \mathbf{\Sigma}_\epsilon \mathbf{U} \tag{3.3}$$

Denote $\mathbf{v}_i$ as the projection of $\mathbf{m}_i$ onto the signal subspace defined by $\mathbf{U}_S$. From equation (3.2) we have:

$$\mathbf{v}_i = \mathbf{U}_S^\top \mathbf{m}_i = \mathbf{v}_l + (\mathbf{I}_{d\times d} \,|\mathbf{0}) \mathbf{n}_i \tag{3.4}$$

where d is the dimension of the signal subspace. $\mathbf{v}_i$'s are the subspace projection of local measurements $\mathbf{m}_i$'s. It consists of projected signal component $\mathbf{v}_l$ and noise component $(\mathbf{I}_{d\times d} \,|\mathbf{0}) \mathbf{n}_i$. When projecting the local measurements onto the subspace, the signal component is essentially unchanged since it strictly resides in the subspace, but the noise component will be reduced greatly. Intuitively, by subspace projection each cluster in the parameter space pull its data items towards its cluster center. The clusters will become denser in the subspace, and therefore easier to identify.

3.4 Why subspace clustering?

In this section, we use statistical analysis to show the merits of subspace approach resulting from its ability to naturally exploit the global spatial-temporal constraints existing in videos. Specifically, we prove that 1) layers are more discriminative in the subspace, and 2) increasing the frame number in the input video increases the layer discriminability in the subspace, but not in the original space.

3.4.1 Layer discriminability

A data configuration is easier to cluster if data items in the same cluster are closer to each other, while at the same time data from different clusters are further away. Such criteria can be quantified by the *within-cluster* scatter matrix S_w and *between-cluster* scatter matrix S_b [31]:

$$S_w = \sum_{l=1}^{L} \pi_l E[(\mathbf{m}_i - \bar{\mathbf{m}}_l)(\mathbf{m}_i - \bar{\mathbf{m}}_l)^\top \mid i \in C_l] \tag{3.5}$$

$$S_b = \sum_{l=1}^{L} \pi_l (\bar{\mathbf{m}}_l - \mathbf{m}_0)(\bar{\mathbf{m}}_l - \mathbf{m}_0)^\top \tag{3.6}$$

where L is the total number of clusters, and π_l is the prior probability of the l-th cluster C_l with $\sum_{l=1}^{L} \pi_l = 1$. In general, we use equal prior probability $\pi_l = 1/L$. $\bar{\mathbf{m}}_l$ is the expected cluster center of the r-th cluster C_l:

$$\bar{\mathbf{m}}_l = E[\mathbf{m}_i \mid i \in C_l]$$

$\mathbf{m}_0$ is the overall mean:

$$\begin{aligned} \mathbf{m}_0 &= E\left[\frac{1}{K}\sum_{i=1}^{K} \mathbf{m}_i\right] \\ &= \sum_{l=1}^{L} \frac{K_l}{K} \bar{\mathbf{m}}_l \end{aligned}$$

where K_l is the number of local measurements from plane r that forms the l-th cluster, and $\sum_{l=1}^{L} K_l = K$.

The overall cluster tightness (noise strength) can be characterized by the trace of

S_w, denoted by $\mathrm{Tr}(S_w)$. The overall distance among cluster centers (signal strength) can be characterized by the trace of S_b, denoted by $\mathrm{Tr}(S_b)$. Layer discriminability is defined by the overall cluster discriminability, which is in turn defined by q, the signal to noise ratio [31]:

$$q = \frac{\mathrm{Tr}(S_b)}{\mathrm{Tr}(S_w)} \tag{3.7}$$

3.4.2 Subspace projection increases layer discriminability

Layers are more discriminative in the signal subspace than in the original parameter space:

Result 3.1. *Projecting local measurements onto the signal subspace improves the layer discriminability q, since it reduces $\mathrm{Tr}(S_w)$, while at the same time maintains $\mathrm{Tr}(S_b)$. If we further assume that ϵ_i is uncorrelated isotropic Gaussian noise, then subspace projection maximizes the cluster discriminability. Specifically, subspace projection improves the cluster discriminability by $\frac{q_s}{q_0} = \frac{6F}{d}$, where d is the subspace dimension, and q_0, q_s are the cluster discriminability before and after subspace projection, respectively.*

When we increase the number of frames in the input video, the cluster signal strength $\mathrm{Tr}(S_b)$ increases, i.e., the clusters are further away from each other, which is desirable for the clustering algorithm. However, increasing the frame number also increases the noise level $\mathrm{Tr}(S_w)$ in the original space, which is not the case in the subspace:

Result 3.2. *In the subspace, the cluster discriminability q is proportional to the frame number F in the input video ($q \propto F$), while in the original space, q maintains at the same average level when F increases. In other words, increasing the frame number in the input video increases the layer discriminability in the subspace, but not in the original space.*

Proof of Result 3.1

The between-cluster scatter matrix S_{b2} in the subspace is given by:

$$S_{b2} = \sum_{l=1}^{L} \pi_l (\bar{\mathbf{v}}_l - \mathbf{v}_0)(\bar{\mathbf{v}}_l - \mathbf{v}_0)^\top$$

Here $\mathbf{v}_0$ is the overall mean in the subspace:

$$\mathbf{v}_0 = E\left[\frac{1}{K}\sum_{i=1}^{K}\mathbf{v}_i\right] = \sum_{l=1}^{L}\frac{K_l}{K}\mathbf{v}_l$$

where K_l is the size of cluster C_l.

From equation (3.2) and (3.4), we have the expectations of $\mathbf{m}_l$ and $\mathbf{v}_l$:

$$\bar{\mathbf{m}}_l = E[\mathbf{m}_i \,|\, i \in C_l] = \mathbf{U}_s\mathbf{v}_l \tag{3.8}$$

$$\bar{\mathbf{v}}_l = E[\mathbf{v}_i \,|\, i \in C_l] = \mathbf{v}_l \tag{3.9}$$

From equations (3.6,3.2, 3.8,3.9), the between-cluster scatter matrix S_{b1} in the original space is given by:

$$\begin{aligned} S_{b1} &= \sum_{l=1}^{L}\pi_l(\mathbf{U}_S\ \mathbf{U}_\perp)\begin{pmatrix}\bar{\mathbf{v}}_l - \mathbf{v}_0\\ \mathbf{0}\end{pmatrix}\begin{pmatrix}\bar{\mathbf{v}}_l - \mathbf{v}_0\\ \mathbf{0}\end{pmatrix}^\top\begin{pmatrix}\mathbf{U}_S^\top\\ \mathbf{U}_\perp^\top\end{pmatrix} \\ &= \mathbf{U}\begin{pmatrix}S_{b2} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix}\mathbf{U}^\top \end{aligned} \tag{3.10}$$

The derivation of equation (3.10) uses the fact:

$$\mathbf{m}_0 = E\left[\frac{1}{K}\sum_{i=1}^{K}\mathbf{m}_i\right] = \mathbf{U}_s\mathbf{v}_0$$

Since U is orthonormal, its multiplication does not change matrix trace. Therefore:

$$\mathrm{Tr}(S_{b1}) = \mathrm{Tr}\begin{pmatrix}S_{b2} & \mathbf{0}\\ \mathbf{0} & \mathbf{0}\end{pmatrix} = \mathrm{Tr}(S_{b2}) \tag{3.11}$$

In other words, the subspace projection doest not change the overall distance between cluster centers.

By substituting equation (3.1) into (3.5), we have the within-cluster scatter matrix S_{w1} in the original space $\mathbb{R}^{6F}$:

$$S_{w1} = \sum_{l=1}^{L}\pi_l E[\epsilon_i\epsilon_i^\top \,|\, C(i) = r] = \mathbf{\Sigma}_\epsilon \tag{3.12}$$

Since The trace of the matrix $\mathbf{A}$ is equal to the sum of all eigenvalues of the matrix $\mathbf{A}$, we have:

$$\mathrm{Tr}(S_{w1}) = \mathrm{Tr}(\mathbf{\Sigma}_\epsilon) = \sum_{i=1}^{6F} \lambda_i$$

where $\{\lambda_i \,|\, i = 1, ..., 6F\}$ are the non-negative eigenvalues of $\mathbf{\Sigma}_\epsilon$ in non-increasing order.

The within-cluster scatter matrix for the projected data in the subspace is denoted as S_{w2}. From equation (3.5) and (3.4), we have:

$$\begin{aligned} S_{w2} &= \sum_{l=1}^{L} \pi_l E[(\mathbf{v}_i - \bar{\mathbf{v}}_l)(\mathbf{v}_i - \bar{\mathbf{v}}_l)^\top \,|\, i \in C_l] \\ &= (\mathbf{I}_{d\times d} \,|\, \mathbf{0})\mathbf{\Sigma}_n(\mathbf{I}_{d\times d} \,|\, \mathbf{0})^\top \end{aligned} \tag{3.13}$$

Since the orthonormal matrix U does not change the eigenvalues of $\mathbf{\Sigma}_\epsilon$, from equation (3.3) we have:

$$\mathrm{Tr}(\mathbf{\Sigma}_n) = \mathrm{Tr}(\mathbf{\Sigma}_\epsilon) = \sum_{i=1}^{6F} \lambda_i$$

Since $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_{6F}$, from equation (3.13) we have:

$$\mathrm{Tr}(S_{w2}) \leq \sum_{i=1}^{d} \lambda_i < \sum_{i=1}^{6F} \lambda_i = \mathrm{Tr}(S_{w1})$$

If the noise ϵ is assumed to be uncorrelated isotropic Gaussian noise, then $\mathbf{\Sigma}_\epsilon = \lambda\mathbf{I}$. We have:

$$\mathbf{\Sigma}_n = \mathbf{U}^\top \lambda \mathbf{I} \mathbf{U} = \lambda \mathbf{I}$$

From equation (3.12) and (3.13), We have:

$$\mathrm{Tr}(S_{w1}) = 6F\lambda \tag{3.14a}$$

$$\mathrm{Tr}(S_{w2}) = d\lambda \tag{3.14b}$$

From equation (3.14), the cluster discriminability q_0 in the original space and q_s in

the subspace are:

$$q_0 = \frac{\mathrm{Tr}(S_b)}{6F\lambda}$$
$$q_s = \frac{\mathrm{Tr}(S_b)}{d\lambda}$$

Therefore the improve of layer discriminability by subspace projection is:

$$\frac{q_s}{q_0} = \frac{6F}{d}$$

Proof of Result 3.2

Denote the $6F$-dimensional vector $\mathbf{a}_l = \bar{\mathbf{m}}_l - \mathbf{m}_0 = (a_{l,1}, ..., a_{l,6F})^\top$, then:

$$\mathrm{Tr}(\mathbf{a}_l \mathbf{a}_l^\top) = \mathbf{a}_l^\top \mathbf{a}_l = \sum_{f=1}^{6F} a_{l,f}^2$$

Therefore:

$$\begin{aligned} \mathrm{Tr}(S_b) &= \sum_{l=1}^{L} \mathrm{Tr}(\mathbf{a}_l \mathbf{a}_l^\top) \\ &= \sum_{l=1}^{L} \sum_{f=1}^{6F} a_{l,f}^2 \end{aligned} \tag{3.15}$$

Adding one additional frame into the input video will append additional six numbers to the vector $\mathbf{a}_l$, therefore increases $\mathrm{Tr}(S_b)$ according to equation (3.15). If we assume that, on average, each frame in the video contributes the same "energy" λ_s to $\mathrm{Tr}(S_b)$. From equation (3.11), the overall signal level for both the subspace and the original space is $F\lambda_s$. Together with equation (3.14), we have the cluster discriminability for the subspace and the original space:

$$q_s = \frac{F\lambda_s}{d\lambda} \propto F$$
$$q_0 = \frac{F\lambda_s}{6F\lambda} = const$$

Therefore, increasing the frame number F of the input image sequence increases

the layer (cluster) discriminability in the subspace, but not in the original parameter space.

Chapter 4

Robust Subspace Computation

4.1 Introduction

The local measurements reside in a low dimensional subspace (Chapter 2), where the clusters of the local measurements become denser and therefore easier to identify(Chapter 3). To compute the subspace, we need to factorize the measurement matrix $\mathbf{W}$:

$$\mathbf{W}_{P\times K} = \mathbf{U}_{P\times d}\mathbf{V}^{\top}_{d\times K} \tag{4.1}$$

where P is the length of each local measurement (column vector); K is the number of local measurements; d is the dimension of the linear subspace. The columns of matrix $\mathbf{U}$ are the bases of the linear subspace that we want to compute.

SVD algorithm is the most popular algorithm to the task of Eq.(4.1):

$$\mathbf{W}_{P\times K} = \mathbf{U}_{P\times d}\Sigma_{d\times d}\mathbf{V}^{\top} \tag{4.2}$$

where the columns of $\mathbf{U}$ are orthonormal bases of the subspace. It can be shown that the SVD algorithm finds the global minimum (the least squared solution) of the following matrix approximation problem:

$$\min \|\mathbf{W} - \mathbf{U}\mathbf{V}^{\top}\|_2 \tag{4.3}$$

where $\|\cdot\|_2$ is the matrix Frobenious norm ($L2$ norm).

It is well known that such least squared solution to the problem of Eq. (4.3) is

sensitive to outliers. Therefore the SVD algorithm, while effective in dealing with Gaussian noise, is sensitive to outliers in the local measurements. This chapter explores several methods to make the subspace computation robust to outliers. We will show that the subspace not only provides a space where clusters are denser and easier to identify, but also a global constraint that is useful for detecting outliers.

4.2 Probabilistic view of subspace computation

In [93, 94], it was shown that principal subspace can be computed by maximum likelihood estimation, which in turn can be computed by EM algorith [29]. In a similar way, in this section we formulate the subspace computation from a probabilistic view of point, where the subspace computation is viewed as a maximum likelihood estimation problem under different noise model. We will show that maximizing the likelihood is equivalent to minimization some cost function. The format of the cost function is determined by the distribution of the noise in the data.

In general, the observed data (local measurement) $\mathbf{m}_i$ is is a P-dimensional column vector contaminated by additive noise:

$$\mathbf{m}_i = \theta_i + \varepsilon_i \qquad i = 1, \cdots, K \tag{4.4}$$

where θ_i is the unobservable (fixed but unknown) true value of the observed (measured) $\mathbf{m}_i$, and ε_i is the additive noise. We know that θ_i resides in a d dimensional linear subspace ($d < P$) such that:

$$\theta_i = \mathbf{U}\mathbf{v}_i \tag{4.5}$$

where $\mathbf{v}_i$ is the projection of $\mathbf{m}_i$ on the subspace defined by the columns of $\mathbf{U}$.

Assuming that local measurements are independent, the log likelihood of the total K measurements is:

$$l(\theta; \mathbf{m}) = \log p(\mathbf{m}_1, ..., \mathbf{m}_K \,|\, \theta_1, ..., \theta_K) = \sum_{i=1}^{K} \log p(\mathbf{m}_i \,|\, \theta_i) \tag{4.6}$$

Therefore, the goal of subspace computation from measurement data is to find the true values θ_i's that maximize the likelihood of the measurements $l(\theta; \mathbf{m})$, subject to

the condition that these θ_i's reside in a low dimensional subspace (defined by $\mathbf{U}$ in Eq. (4.5)).

Gaussian noise

If the noise ε_i follows zero-mean normal distribution with common standard deviation of σ, then $\mathbf{m}_i \backsim N(\theta_i, \Sigma)$. By further assuming that the elements of each vector ($\mathbf{m}_i$ or θ_i) are independent, the probabilistic distribution of $\mathbf{m}_i$ conditioned on θ_i is:

$$p(\mathbf{m}_i \,|\, \theta_i) \backsim \exp\{-\frac{\|\mathbf{m}_i - \theta_i\|_2^2}{2\sigma^2}\} \tag{4.7}$$

where $\|\mathbf{x}\|_2$ is the $L2$ norm of vector $\mathbf{x}$:

$$\|\mathbf{x}\|_2 = \left(\sum_i x_i^2\right)^{1/2} \tag{4.8}$$

The data log likelihood can be written as:

$$l(\theta; \mathbf{m}) = -c \sum_{i=1}^{K} \|\mathbf{m}_i - \theta_i\|_2^2 \tag{4.9}$$

where c is some positive constant. Maximizing the data log likelihood is therefore equivalent to minimizing the term in the r.h.s. of Eq. (4.9), which is called the cost function or energy function:

$$E(\theta) = \sum_{i=1}^{K} \|\mathbf{m}_i - \theta_i\|_2^2 \tag{4.10}$$

Substitute Eq. (4.5) into Eq. (4.10), we can rewrite Eq. (4.10) in matrix format as:

$$E(\mathbf{U}, \mathbf{V}) = \|\mathbf{W} - \mathbf{U}\mathbf{V}^\top\|_2^2 \tag{4.11}$$

where $\mathbf{W}$ is the measurement matrix with $\mathbf{m}_i$ its i-th column, $\mathbf{v}_i$ is the i-th column of $\mathbf{V}^\top$, and $\|\cdot\|_2$ is the matrix Frobenius norm ($L2$ norm). The assumption of i.i.d. Gaussian noise model transfers the maximum likelihood problem of $\max_\theta l(\theta; \mathbf{m})$ into a $L2$-norm cost function which is convex in $\mathbf{U}$ and $\mathbf{V}$, and has a closed formed solution (SVD algorithm) to compute its global minimum.

Laplacian noise

If we assume the noise ε follows Laplacian distribution instead of normal distribution, we have:

$$p(\mathbf{m}_1, \cdots, \mathbf{m}_K \,|\, \theta_1, ..., \theta_K) \sim \exp\{-\frac{\sum_{i=1}^{K} \|\mathbf{m}_i - \theta_i\|_1}{s}\} \tag{4.12}$$

Therefore the maximum likelihood of the observed data is given by minimizing the following $L1$ norm cost function:

$$E(\theta) = \sum_{i=1}^{K} \|m_i - \theta_i\|_1 \tag{4.13}$$

Written in matrix form, we have:

$$E(\mathbf{U}, \mathbf{V}) = \|\mathbf{W} - \mathbf{U}\mathbf{V}^\top\|_1 \tag{4.14}$$

where $\mathbf{W}$ is the measurement matrix with m_i its i-th column. Unlike the $L2$ norm cost function, the $L1$ norm cost function is in general non-convex in $\mathbf{U}$ and $\mathbf{V}$.

General case

In general, when the noise distributions follow same model but with different model parameters for different data points, the data likelihood is:

$$p(\mathbf{m}_1, \cdots, \mathbf{m}_K \,|\, \theta_1, ..., \theta_K) \sim \exp\{-\sum_{i=1}^{K}\sum_{j=1}^{P} \frac{d(m_{ij} - \theta_{ij})}{s_{ij}}\} \tag{4.15}$$

where $d(\cdot)$ is some distance function, and s_{ij} is related to the parameter of the noise distribution.

The maximum likelihood of the observed data is given by minimizing the following weighted cost function:

$$E(\theta) = \sum_i \sum_j s_{ij} d(m_{ij} - \theta_{ij}) \tag{4.16}$$

Notice that each data item is weighed by different component s_{ij}. If we use the Euclidean distance $d(x) = cx^2$, the above cost function can be simplified as a weighted

sum:

$$E(\theta) = \sum_i \sum_j s_{ij}(m_{ij} - \theta_{ij})^2 \tag{4.17}$$

Written in matrix format, we have:

$$E(\mathbf{U}, \mathbf{V}) = \|\mathbf{S} \otimes (\mathbf{W} - \mathbf{UV})\|_2 \tag{4.18}$$

where $\otimes$ denotes the component wise multiplication. In the low rank approximation context, the above cost function has been studied in robust PCA in [32], and recently in [84]. Unlike the $L2$ norm cost function, the above weighted cost function is in general non-convex in $\mathbf{U}$ and $\mathbf{V}$, due to the weight matrix $\mathbf{S}$.

Summary

The maximum likelihood (ML) solution to the matrix factorization (subspace computation) depends on the assumed noise distribution. When the noise follows independent and identical Gaussian distribution, the ML solution is achieved by minimizing a $L2$ norm cost function. When the noise follows independent and identical Laplacian distribution, the ML solution is achieved by minimizing a $L1$ norm cost function. In general when the noise distributions are no longer identical, the ML solution is achieved by minimizing a non-convex weighted cost function [32, 84], with the weights set according to some problem dependent distance function. Both the cases of $L1$ norm and weighted cost function can be used to deal with outliers, as will be shown in the following sections.

For other noise distributions, such as the generalized exponential family, corresponding cost functions can also be derived [19].

4.3 $L2$-norm based subspace computation

Gaussian distribution is the most often assumed noise model. Under such noise model assumption, the problem of estimating the subspace is equivalent to minimize the following $L2$-norm cost function:

$$E(\mathbf{U}, \mathbf{V}) = \|\mathbf{W}_{P\times K} - \mathbf{U}_{P\times d}\mathbf{V}_{d\times K}^\top\|_2^2 \tag{4.19}$$

where d is the dimension of the subspace defined by $\mathbf{U}$, and $d < P$.

Singular Value Decomposition (SVD) is a popular approach to minimize $E(\mathbf{U}, \mathbf{V})$. The following theory of SVD explains how SVD can be used to minimize $E(\mathbf{U}, \mathbf{V})$ [34]:

Theorem 4.1. *Let the SVD of matrix* $\mathbf{W}$ *be*

$$\mathbf{W}_{P\times K} = \mathbf{A}_{P\times P}\Sigma_{P\times P}\mathbf{B}^{\top}_{P\times K} \tag{4.20}$$

where $\Sigma = diag(\lambda_1, \cdots, \lambda_P)$ *,* $\lambda_1 \geq \cdots \geq \lambda_P \geq 0$*, and* $\mathbf{A}$ *and* $\mathbf{B}$ *orthonormal matrix. Then for* $1 \leq d \leq P$*, we have:*

$$\min E(\mathbf{U}, \mathbf{V}) = \sum_{i=d+1}^{P} \lambda_i^2 \tag{4.21}$$

The above theorem states that the first d columns of $\mathbf{A}$ in Eq. (4.20) defines the subspace that minimizes the $L2$-norm cost function defined in Eq. (4.19), i.e., $\mathbf{U} = \mathbf{A}(:, 1:d)$. Similarly $\mathbf{V} = \mathbf{B}(:, 1:d)$.

The SVD algorithm essentially finds a least-squared solution to the $L2$ norm cost function in Eq. (4.19). The problem with $L2$ norm cost function is that it is sensitive to outliers. With even a single influential outliers, the resulted subspace could be completely different from the desired solution.

4.4 Outlier detection

The local measurements are the 2D affine transformations of predefined local image regions estimated using the algorithm in [10]. For the planar patches belonging to a common plane in the scene, their corresponding affine transformations will ideally share the same 2D image motion parameters, and therefore form a distinct data cluster in the motion parameter space. In real data, some affine motion estimations may become unpredictable due to lighting change, motion or depth discontinuity, and/or non-rigid motions. Instead of forming distinctive clusters, such unpredictable affine motions are often isolated or sparse in the motion parameters. In other words, they are outliers to the major clusters formed by planes in the scene. It is necessary to apply robust methods to detect the outliers for subspace computation.

Outliers are data items that do not follow the global behaviors of the majority of

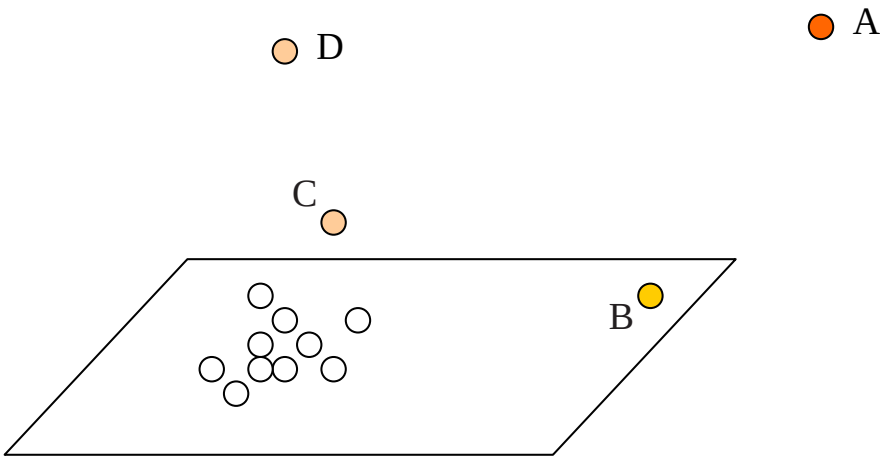


Figure 4.1: Structure outliers and extreme outliers. The 3-dimensional data points reside in a 2-dimensional subspace. Point A,B, and D are extreme outliers, but B is in the subspace. Point C is a structure outlier.

the data (inliers). In our case of layer extraction, the behaviors of inliers are:

- form dense clusters;
- reside in a low dimensional subspace.

According to these two rules, in layer extraction, the outliers can be classified into two types, namely the *extreme outliers* and the *structure outliers*. Extreme outliers are those data points that are far away from the data clusters, and will inflate the covariance matrix of local measurements. *Structure outliers* are data points that do not inflate the variance and covariance, but violate the correlation structure imposed by subspace. Note that an outlier could belong to both extreme outliers and structure outliers. Fig 4.1 shows a 2D subspace in a 3D space, where Point A, B, and D are extreme outliers, but B is in the subspace. Point C is a structure outlier.

Approaches to outlier detection includes parametric approaches and non-parametric approaches. Parametric approaches are suitable for structure outlier detection, and non-parametric approaches are suitable for extreme outlier detection.

4.4.1 Parametric approach

The parametric approaches define a global parametric model that inliers should follow. Outliers are those items that do not follow such parametric model. Specifically, in

parametric approaches, a parametric model is first fit to the data, and then outliers are identified as the data that violate the fit model. A more general scheme is to give each data item a weight in the range of $[0, 1]$ according to the degree that such data item violates the global parametric model. A zero weight indicates an outlier. Robust estimator is often used to weight each data item, where the objective function in Eq. (4.3) is rewritten as:

$$\min \sum_{i,j} \rho(m_{ij} - \mathbf{u}_{i\cdot}\mathbf{v}_{\cdot j}) \tag{4.22}$$

where m_{ij} is the ij-th element of $\mathbf{W}$, $\mathbf{U}$ and $\mathbf{V}$ are the global parametric model (subspace model), $\mathbf{u}_{i\cdot}$ is the i-th row of $\mathbf{U}$, and $\mathbf{v}_{\cdot j}$ is the j-th column of $\mathbf{V}^\top$. The contribution to the cost function of each data element is controlled by the robust M-estimator $\rho(\cdot)$ based on the distance between the data element and the current subspace model, i.e., the residual $(m_{ij} - \mathbf{u}_{i\cdot}\mathbf{v}_{\cdot j})$. For example the Geman-McClure robust function $\rho(x, \sigma) = \frac{x^2}{x^2+\sigma^2}$ is used in [97], where σ is the parameter that controls convexity of the robust estimator.

The use of robust M-estimator in Eq. (4.22) changes the convexity of the cost function. In general, there are many minimums in Eq. (4.22), and iterative procedures are often used to derive a good local minimum. In each iteration, each data item is first weighted based on its distance to the current parametric model, and then a new model is recomputed using the weighted data. When the dimension of the data is too high to afford computing the subspace model multiple times, gradient decent can be used to compute a local minima [97].

The convergence of the above iterative process depends on the model initialization. When a reasonably good initialization is available, the parametric method is highly effective since it takes the global data into account in detecting the outliers. Parametric approaches are effective for detecting structure outliers, since such outliers are not influential and a good initial model is possible if there are not extreme outliers. On the other hand, in the presence of influential extreme outliers, it would be hard, before the removal of the extreme outliers, to obtain a good initial model as the starting point for the iterative procedure.

4.4.2 Non-parametric approach

In some cases, it may be hard to define or compute a global model to detect the outliers. For example, it may be hard to compute a reasonably good subspace model if there are too many extreme outliers in the data. In such cases, we need to explore some local non-parametric criteria. In layer extraction, inliers form clusters, which can be used to define some non-parametric metric that could implicitly use the intrinsic cluster structure of the data, but without knowing the clustering information.

In non-parametric approaches, an outlier is detected based on some distance metric without any model fitting. Such approaches implicitly use the intrinsic cluster structure of the data to detect the outliers. In other words, we want to ignore the isolated data points in the parameter space, and retain only the data points that form clusters to compute the subspace.

A natural non-parameter metric is the kernel-based point density metric [7]. The density of a given data point is computed using the points inside the kernel window centered at the given data point. A data point with low density is declared as an outlier.

Another metric is to use the kNN distance metric, i.e., the distance between the data point under consideration and its k-th nearest neighbor. The intuition of using kNN distance is that for a data point inside a cluster with size larger than k, its kNN distance is small. On the other hand, an extreme outlier will have large kNN distance since they do not form distinctive clusters, and are usually isolated or sparsely distributed.

We prefer kNN distance metric since it has the following advantages over kernel-based density metric:

- The parameter k is much more intuitive than the parameter of kernel window size for users to set. k is the smallest data cluster that users want to declare as inliers. It can be set according to user's requirement without examining the input data. It can be thought of as a kernel-based method, but with kernel size adapted locally to the input data. On the contrary, in kernel-based method the kernel window size (or scale) is data dependent, and is not intuitive for users to set.

- kNN distance metric is more suitable than density-based method in low statis-

tic when the number of data points is small, which is often the case in layer extraction application.

- kNN distance metric is adaptive to the shape of data clusters, which is not the case for kernel-based based.

The kNN distance metric is *one* dimensional, which can be easily characterized by its histogram. The inliers have smaller kNN distance, and form the first peak in the histogram. Detecting the first significant peak is trivial in the one dimensional histogram.

Finally, we note that non-parametric approaches are effective for detecting extreme outliers, but not as effective for detecting structure outliers. The reason is that structure outliers often reside at the boundaries of the clusters.

4.5 Robust subspace computation with outlier detection

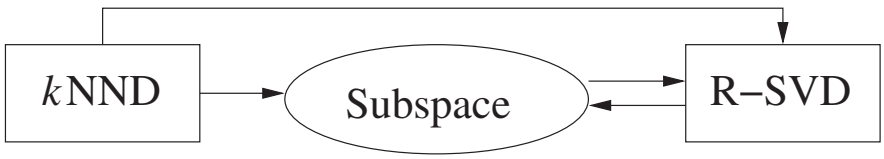


Figure 4.2: Three integrated components in our layer extraction algorithm. Subspace is the core component.

Two important issues arise in order to fully exploit the subspace constraint for a reliable layer extraction algorithm: 1) to achieve low dimensional subspace, especially for dynamic scenes; 2) to make subspace computation robust to outliers that often exist in real data. We achieve the above goals by naturally integrating the following three components in our layer extraction algorithm: subspace, kNND, and robust-SVD, with subspace the centering component, as shown in Figure 4.2. The kNND component, a simple and non-parametric approach based on the k-th nearest neighbor distance metric, implicitly utilizes the intrinsic cluster structure to detect extreme outliers and small layers (cluster size less than k), *without* knowing the clustering information. The remaining large layers are guaranteed to reside in a low dimensional subspace, which in turn provides constraints for the robust-SVD (RSVD) component

to detect the *structure outliers* that violate the correlation structure imposed by subspace. The RSVD is a parametric approach that iterates between subspace model computation and data item weighting. The RSVD component computes an accurate subspace model, due to 1) the removal of influential extreme outliers by kNND and 2) its ability to detect the remaining structure outliers.

4.5.1 kNND: detecting extreme outliers and assuring low-dimensional subspace

The kNND procedure implicitly utilizes the intrinsic cluster structure to detect extreme outliers and small clusters (size less than k), *without* knowing the cluster structure. We use a simple example to illustrate the kNND procedure. In Figure 4.3, there are 35 data points reside in a 2D parameter space, with two large clusters, five sparse points, and two small clusters. For each data point i, its distance to its k-th nearest neighbor is denoted as $d_k(i)$. If point i resides in a cluster with size larger than k, then its $d_k(i)$ will be small. Otherwise, $d_k(i)$ will be large. The points in *all* clusters with size larger than k will form the first dominant peak in the *one* dimensional histogram of $\{d_k(i) \,|\, i = 1, \cdots, K\}$. If points not in this peak are removed as outliers, then the size of any remaining cluster must be larger than k. In the example shown in Figure 4.3, if we set $k = 5$, the two clusters with sizes larger than 5 will form the first peak in the 1D histogram of $d_k(i)$. The sparse points and the two small clusters will be removed as outliers. From this example, we see that kNND a data-driven procedure for detecting and removing outliers without parametric modelling.

We now show how to choose k in kNND to assure a desired low dimensional subspace using Result 2.6, which is especially useful for dynamic scenes. Suppose the image size (pixel number) is w, and each local patch contains b pixels. After the kNND procedure, the size of a survived cluster is expected to be larger than k, and its corresponding layer in the image will have more than kb pixels. Therefore, the number of survived clusters (layers), denoted as L, must be bound by: $L \leq \frac{w}{kb}$. According to Result 2.6, the dimension of the subspace d is no more than $(L-1)$. Therefore, via kNND, we are guaranteed a linear subspace whose dimension d is bound by:

$$d \leq L - 1 \leq \frac{w}{kb} - 1$$

We can assure the subspace dimension to be lower than a pre-defined value d_0 by

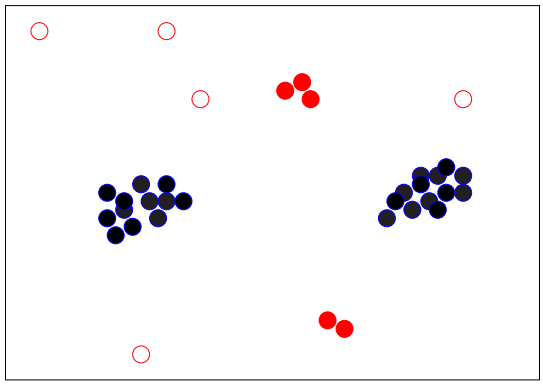


Figure 4.3: kNND example.

setting the parameter k as:

$$k \geq \frac{w}{(d_0+1)b} \tag{4.23}$$

In real scenes, it is often the case that the large layers are from static backgrounds, in which case the subspace dimension is no more than *three*.

kNND metric was used to model point process in clutter backgrounds [17]. It was also used to determined the window size in kernel density estimation [39], which can also be used to detect sparse points. We prefer directly using kNND metric because of its following advantages:

- kNND transforms high dimensional data into one dimensional data, where extreme outliers are clearly distinguished from inliers.
- the parameter k is directly related the expected size of large layers, and therefore can be used to guarantee a low dimensional subspace.
- kNND is non-parametric and has high break-down point, suitable for detecting influential extreme outliers.
- kNND is adaptive to cluster shape.

4.5.2 Robust SVD with structure outliers detection

Structure outliers may survive the kNND procedure since they may reside nearby to cluster boundaries. They are less influential than extreme outliers on subspace computation. However, structure outliers affect the accuracy of subspace computation

and obscures the boundaries among clusters. It is therefore desirable to detect such structure outliers before the subspace clustering.

After the influential extreme outliers are removed, the remaining data are used to construct the measurement matrix $\mathbf{W}$, and the SVD algorithm [1] can be safely applied to obtain an initial subspace model:

$$\mathbf{W}_{6F\times K} = \mathbf{U}_{6F\times 6F}\mathbf{\Sigma}_{6F\times 6F}\mathbf{V}^{\top}_{6F\times K} \tag{4.24}$$

The diagonal elements of $\mathbf{\Sigma}$ are the singular values λ_i of $\mathbf{W}$ in non-increasing order.

The actual rank of $\mathbf{W}$ depends on the camera and the planes in the scene, and is detected by [45]:

$$d = \min(d_u, d_t) \tag{4.25}$$

where d_u is the user-defined upper-bound of the subspace dimension that is guaranteed by choosing k according to equation (4.23), and d_t is the rank bound by the expected noise in the input data. d_t is computed using the singular values λ_i's. There are several methods to determine d_t [53]. The one that used in [45] is to use the ratio of the signal energy to total energy:

$$d_t = \arg\min_m \left(\frac{\sum_{i=0}^{m} \lambda_i^2}{\sum_{i=0}^{6F} \lambda_i^2} > t \right) \tag{4.26}$$

Here $(1-t)$ determines the noise level we want to tolerate. The other often used method is to compute the difference of adjacent singular values [53]:

$$\delta_i = \lambda_i - \lambda_{i+1}$$

The rank of $\mathbf{W}$ is determined as the place where δ_i is big, i.e., the singular values have a sharp jump.

If we assume that the measurement noise ϵ_i in equation 4.4 follows a zero-mean multivariate Gaussian distribution, then the d dimensional homography subspace is optimally approximated by the signal subspace defined by the first d columns of U in equation (4.24).

Once the rank is determined, the data can then be decomposed into the signal

[1] SVD is the most often used algorithm to compute subspace. Classic SVD is least-squared based and is sensitive to outliers.

component and the noise component:

$$\begin{aligned}\mathbf{W} &= (\mathbf{U}_S\mathbf{U}_\perp)\begin{pmatrix}\Sigma_S & \\ & \Sigma_\perp\end{pmatrix}(\mathbf{V}_S\mathbf{V}_\perp)^\top \\ &= \mathbf{U}_S\Sigma_S\mathbf{V}_S^\top + \mathbf{U}_\perp\Sigma_\perp\mathbf{V}_\perp^\top \\ &= \mathbf{S} + \mathbf{N}\end{aligned}$$

Since $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$, we have:

$$\begin{pmatrix}\mathbf{U}_S^\top \\ \mathbf{U}_\perp^\top\end{pmatrix}(\mathbf{U}_S\mathbf{U}_\perp) = \begin{pmatrix}\mathbf{U}_S^\top\mathbf{U}_S & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_bot^\top\mathbf{U}_\perp\end{pmatrix} = \mathbf{I}$$

Therefore, we have:

$$\mathbf{U}_\perp^\top\mathbf{U}_\perp = \mathbf{I} \tag{4.27}$$

The noise component $\mathbf{N}$ also follows zero-mean Gaussian distribution [52]. The sampled covariance matrix $\mathbf{C}$ of $\mathbf{N}$ is:

$$\begin{aligned}\mathbf{C} &= \frac{1}{K-1}\mathbf{N}\mathbf{N}^\top \\ &= \frac{1}{K-1}\mathbf{U}_\perp\Sigma_\perp^2\mathbf{U}_\perp^\top\end{aligned} \tag{4.28}$$

The noise component in the i-th data point is $\mathbf{n}_i = \mathbf{U}_\perp\Sigma_\perp\mathbf{v}_i$, where $\mathbf{v}_i^\top$ is the i-th row of $\mathbf{V}_\perp$, a $(6F-d)$ dimensional vector. The Mahalanobis distance z_i^2 of $\mathbf{n}_i$ is:

$$\begin{aligned}z_i^2 &= \mathbf{n}_i^\top\mathbf{C}^{-1}\mathbf{n}_i \\ &= (K-1)\mathbf{v}_i^\top\mathbf{v}_i \qquad \text{(Eq. 4.27)} \\ &= (K-1)\sum_{p=1}^{6F-d} v_{i,p}^2\end{aligned} \tag{4.29}$$

Since the noise components follow Gaussian distribution, z_i^2 follows χ_N^2 distribution [53], with $N = (6F-d)$ degrees of freedom [2]. A data point with z_i^2 lies outside the p-th confidence interval of the corresponding χ_N^2 distribution is marked as a structure outlier.

[2] In reality, the elements in the noise vector $\mathbf{n}_i$ may not be independent, which means that the true degree of freedom of the χ^2 distribution is less than $(6F-d)$. We perform another rank detection based on equation (4.26) to detect the effective degree of freedom.

Our algorithm for robust subspace computation with structure outliers detection consists of the following steps:

1. Use SVD to compute an initial subspace using all remaining data points.
2. Compute Mahalanobis distance z_i^2 for each data point. Data whose z_i^2 are outside the p-th confidence interval of χ^2 distribution are marked as structure outliers.
3. Apply SVD to the set of inliers to recompute the subspace, including the dimension of the subspace.
4. Repeat Step 2 and 3 until the set of inlier stabilizes.

Since kNND procedure removes the influential extreme outliers, the initial subspace model in Step 1 is reasonably good, and the above iterative procedure will converge quickly to a good subspace model that is defined by the first d columns of the matrix $\mathbf{U}$.

The above algorithm is a special case of the weighted cost function (see Section 4.2). Here the weight is binary ($\{0,1\}$), and every item in the same column of the measurement matrix receives the same weight. In such special case, the weighted cost function has a global minimum that can still be computed by the singular value decomposition of the weighted measurement matrix [34, 49, 84].

4.5.3 Experimental results

Fig. 4.4 and 4.5 show the experimental results of robust subspace computation on the *mobile & calendar* sequence. The input image sequence has 11 frames, with the middle frame the reference frame. Fig. 4.4(a) & (b) show the first and last frames of the sequence. The reference frame is divided into 48×48 blocks, with adjacent blocks overlap by half. The local measurements are the affine transformation estimated for each block from the reference frame to other frames.

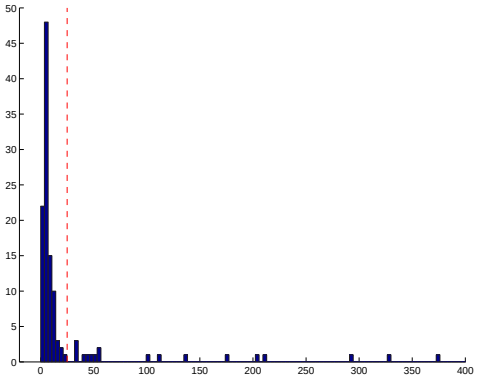
For the i-th local measurement $\mathbf{m}_i$, its distance to its k-th nearest neighbor $\mathbf{m}_k$ is computed as:

$$d_i(k) = \|\mathbf{m}_i - \mathbf{m}_k\|_2 \tag{4.30}$$

where $\|\cdot\|_2$ is the $L2$ norm. Fig. 4.4(c) shows the histogram of $d_i(k)$. As we can see, the inliers form the first peak. The extreme outliers are identified as those data

(a) input: first frame

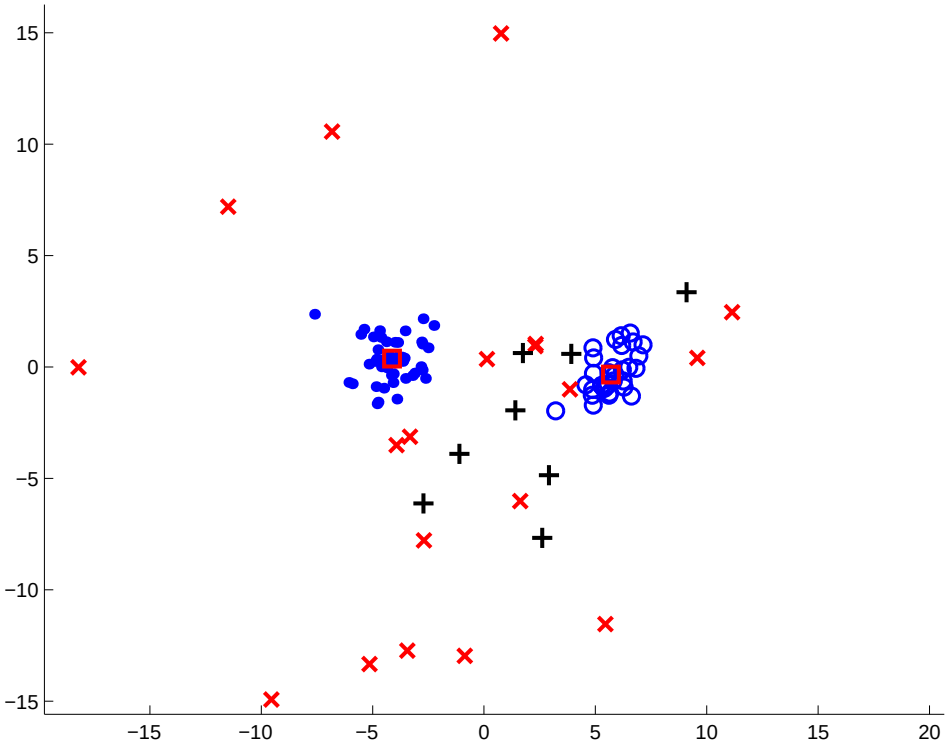
(b) last frame



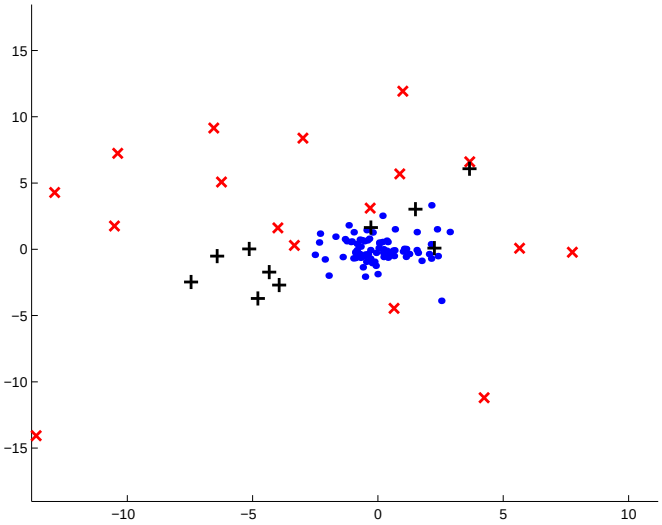
(c) histogram of kNND

(d) excluded outliers by kNND

Figure 4.4: Robust subspace computation on *Mobile* sequence. (a) & (b): two frames of the 11-frame input image sequence; (c): the 1-D histogram of the kNND; (d): the outliers excluded as not on the first peak in the histogram in (c);



(a) signal subspace



(b) noise subspace

Figure 4.5: Subspace. (a): projecting data onto signal subspace; (b) projecting data onto noise subspace. Blue • and ○ are inliers. Red ×'s are extreme outliers. Black +'s are structure outliers.

items not on the first peak. Fig. 4.4(d) shows the detected extreme outliers. They correspond to the blocks strapping over different layers, or small moving objects in the scene.

After the extreme outliers are detected and excluded by the kNND procedure, the remaining data are processed by robust-SVD to compute the subspace, as well as detect the structure outliers. In this example, a 2D subspace is derived. Fig. 4.5(a) shows the projected data on this 2D subspace. As we can see, the inliers (blue "$\bullet$" and "$\circ$") form two distinct dense clusters in this subspace. Extreme outliers are shown in red "$\times$", and structure outliers are shown in black "+". Fig. 4.5(b) shows the projected data on the first two dimension of the noise subspace. The projected inliers on the noise subspace form Gaussian cloud.

4.6 Discussion: $L1$-norm based subspace computation

We have shown a robust method to compute the subspace using kNND and robust SVD. The cost function of the robust SVD is essentially $L2$ norm with special weighting. In this section, we discuss the potential advantages of using $L1$ norm metric for subspace computation [57]. Minimizing the $L1$ norm metric corresponds to the maximum likelihood estimation under Laplacian noise model. We first show that $L1$ norm metric is more robust than $L2$-norm through a simple illustrative line-fitting example. We then propose two algorithms to compute the subspace using $L1$ norm metric: alternating weighted-median algorithm and alternating convex quadratic programming. These two algorithms are efficient: weighted median has fast algorithm [90], and convex quadratic programming (see [65]) is well studied and has very efficient software package available. More extensive experiments of these above two approaches to subspace computation is part of the future work.

4.6.1 Robust $L1$ norm metric

One important advantage of using $L1$ norm is that it is more robust to outliers than $L2$ norm in statistical estimation, as can be seen from the following simple line fitting example. We are given 10 two-dimensional points $\{(x_i, y_i)|i = 1, ..., 10\}$ where the response variable y is corrupted by Gaussian noise. We want to fit a line $y = kx$ to

these 10 points, where k is the parameter (slope) that we need to estimate. More specifically, we use the following linear model:

$$y = kx + \varepsilon \tag{4.31}$$

where k is the parameter to estimate and ε is the noise that corrupts the response variable y.

If ε is assumed to be Gaussian noise, then the ML estimation of the parameter k is given by minimizing the follow $L2$-norm cost function (sum of squared difference):

$$E(k) = \sum_{i=1}^{10} (y_i - kx_i)^2 \tag{4.32}$$

The least squared solution to minimize the above cost function is:

$$k = \frac{\sum_{i=1}^{10} x_i y_i}{\sum_{i=1}^{10} x_i^2}$$

If ε is assumed to have Laplacian distribution, then the ML estimation of the parameter k is given by minimizing the follow $L1$-norm cost function:

$$\sum_{i=1}^{10} |y_i - kx_i| = \sum_{i=1}^{10} |x_i| |\frac{y_i}{x_i} - k| \tag{4.33}$$

The global minimum is the weighted median of $\{\frac{y_i}{x_i} | i = 1, ..., 10\}$, with $|x_i|$ the corresponding weights. If $x_i = 0$, then its corresponding data point is removed from the accumulation in Eq. (4.33), since the weight is equal to zero too.

Fig.4.6 shows the results when there is NOT any outlier in the given data. As we can see, the $L1$ norm and $L2$ norm cost function give similar estimation of k.

When there are outliers in the data, the results are different. In Fig.4.7 there are two outliers. The $L1$ norm cost function still gives good results, while the $L2$ norm cost function gives erroneous estimation.

In summary, we have the following well-known result [90]:

Result 4.1. *The global minimum of the $L1$-norm cost function $E(u) = \sum_{i=1}^{K} \|y_i - ux_i\|_1$ is given by the weighted median of $\{\frac{y_i}{x_i} | \ i = 1, ..., K\}$, where $|x_i|$ is the weight of the i-th cost item.*

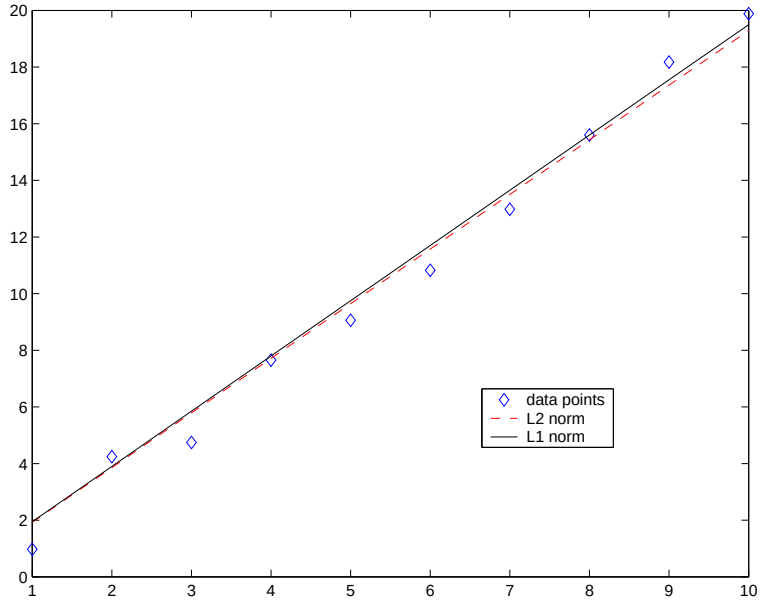


Figure 4.6: Fit a line to 10 given data points. All data points are inliers. The result of using $L2$ norm cost function is similar to that of using $L1$ norm.

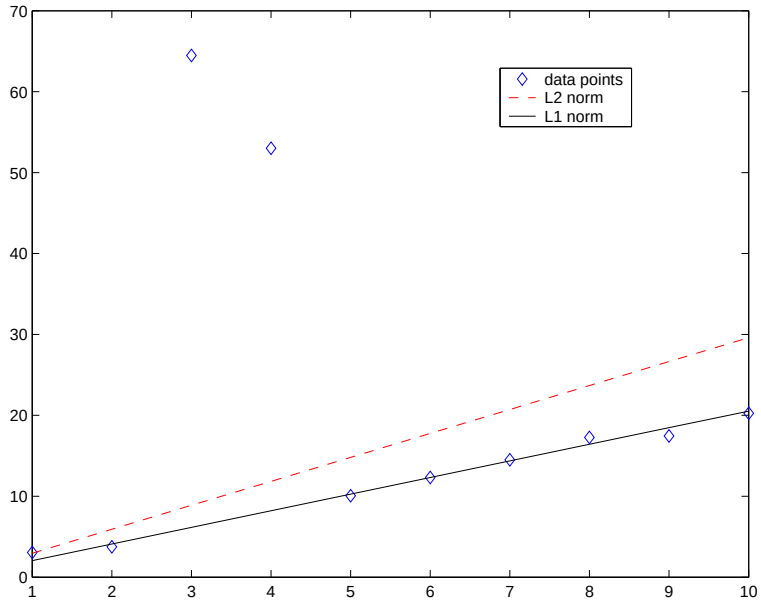


Figure 4.7: Fit a line to 10 given data points. Two data points are outliers. Using $L2$ norm cost function gives erroneous result, while using $L1$ norm cost function still gives correct result.

4.6.2 $L1$-norm based robust subspace computation

We have shown that by assuming Laplacian noise distribution, the maximum likelihood estimation of matrix factorization or linear model fitting corresponds to $L1$-norm based statistical estimation, which is more robust to outliers than $L2$-norm based. In this section, we present algorithms on how to maximize the likelihood, i.e., minimize the $L1$-norm based cost function:

$$E(\mathbf{U}, \mathbf{V}) = \|\mathbf{W} - \mathbf{U}\mathbf{V}^\top\|_1 \tag{4.34}$$

$\mathbf{W}_{P\times K}$ is the measurement matrix with Column i the observed (measured) data x_i. The columns of $\mathbf{U}_{P\times d}$ are the d bases of the subspace to be estimated, with $d < \min(P, K)$.

Alternative minimization by weighted-median

While Eq.(4.33) has a global minimum that can be computed via the weighted median, the cost function for matrix factorization in Eq.(4.34) is in general non-convex, since both $\mathbf{U}$ and $\mathbf{V}$ are unknown. It requires some iterative scheme to achieve a good local minimum.

If $\mathbf{U}$ or $\mathbf{V}$ is known, then we can use weighted median to compute the global minimum of Eq.(4.34). Such fact suggests an scheme that minimizes the cost function alternatively over $\mathbf{U}$ or $\mathbf{V}$, each time optimizing one argument while keeping the other one fixed.

We use the alternating minimization procedure [26]. To simplify the presentation, we first consider the case where the dimension of the subspace is one. The alternating minimization problems are:

$$\text{For } i = 1 \cdots P, \quad u_i^{(t)} = \arg\min_u \|\mathbf{m}_{i\cdot} - u\mathbf{V}^{(t-1)^\top}\|_1 \tag{4.35a}$$

$$\text{For } j = 1 \cdots K, \quad v_j^{(t)} = \arg\min_v \|\mathbf{m}_{\cdot j} - v\mathbf{U}^{(t)}\|_1 \tag{4.35b}$$

where $u_i^{(t)}$ is the i-th element of the column vector $\mathbf{U}$ (similar definition of $v_i^{(t)}$) , t is the index of the iteration steps. The solutions (global minimums) to Eq. (4.35) can be obtained through weighted median according to Result 4.1.

When the subspace dimension d is more than one, $\mathbf{U}$ and $\mathbf{V}$ contain more than one

//Initialization
Set $\mathbf{U} = 0$ and $\mathbf{V} = 0$
//Cycle through d columns of $\mathbf{U}$ *for N times*
For $n = 1, \cdots, N, c = 1, \cdots, d$:
//Optimize $\mathbf{u}_{\cdot c}$*, the c-th column of* $\mathbf{U}$ *with other columns fixed*
If $n = 1$, initialize $\mathbf{v}_{\cdot c}^{(0)}$ randomly
Set $\mathbf{W} = \mathbf{W} - \sum_{k \neq c} u_{ik} v_{kj}$
For $t = 1, \cdots,$ convergence

$$\text{For } i = 1 \cdots P, \quad u_i^{(t)} = \arg\min_u \|\mathbf{m}_{i\cdot} - u\mathbf{v}_{\cdot c}^{(t-1)^\top}\|_1$$
$$\text{For } j = 1 \cdots K, \quad v_j^{(t)} = \arg\min_v \|\mathbf{m}_{\cdot j} - v\mathbf{u}_{\cdot c}^{(t)}\|_1$$

Figure 4.8: Algorithm of using iterative weighted median to minimize the $L1$ norm cost function, and therefore to compute the subspace.

column. Our algorithm cycles through the d columns of $\mathbf{U}$ and $\mathbf{V}$, optimizing each column while fixing the others. The problem is therefore broken into d subproblems of Eq. (4.35). The overall algorithm is shown in Fig 4.8.

Alternative minimization by convex quadratic programming

We have presented the approach that cycle through the principal vectors (subspace bases) by optimizing over one principal vector while fixing the others. Such approach is simple to implemented by weighted median. In this subsection, we convert the subspace computation problem to alternative convex optimization problem, which updates all principal vectors instead of only one in each iteration. Potentially alternative convex optimization might be faster and achieve better local minimum than the alternative weighted median approach presented above.

The alternative optimization can be written as:

$$\mathbf{U}^{(t)} = \arg\min_{\mathbf{U}} \|\mathbf{W} - \mathbf{U}\mathbf{V}^{(t-1)^\top}\|_1 \tag{4.36a}$$
$$\mathbf{V}^{(t)} = \arg\min_{\mathbf{V}} \|\mathbf{W} - \mathbf{U}^{(t)}\mathbf{V}^\top\|_1 \tag{4.36b}$$

In the following we show how to solve Eq. (4.36b). Eq. (4.36a) can be solved similarly.

The cost function of Eq. (4.36b) can be written as:

$$\begin{aligned} E(\mathbf{V}) &= \|\mathbf{W} - \mathbf{U}^{(t)}\mathbf{V}^\top\|_1 \\ &= \sum_{j=1}^{K} \|\mathbf{m}_{\cdot j} - \mathbf{U}^{(t)}\mathbf{v}_{\cdot j}\|_1 \end{aligned}$$

where $\mathbf{m}_{\cdot j}$ is the j-th column of $\mathbf{W}$, $\mathbf{v}_{\cdot j}$ is the j-th column of $\mathbf{V}^\top$. The problem of Eq. (4.36b) is therefore decomposed into K sub-problems, each one optimizing $\mathbf{v}_{\cdot j}$. Each sub-problem has the following general formulation:

$$\mathbf{x} = \arg\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_1 \tag{4.37}$$

This problem can be reduced to a simple convex quadratic programming problem whose global minimum can be computed efficiently [65]:

$$\begin{aligned} &\min_{\mathbf{x},\mathbf{z},\mathbf{t}} \frac{1}{2}\|\mathbf{z}\|_2^2 + \gamma \mathbf{e}^\top \mathbf{t} \\ &s.t. -\mathbf{t} \leq \mathbf{A}\mathbf{x} - \mathbf{b} - \mathbf{z} \leq \mathbf{t} \end{aligned} \tag{4.38}$$

where $\mathbf{e}$ is a column vector of ones. γ is a small positive constant.

In summary, the $L1$ norm formulation of subspace computation $\min_{\mathbf{U},\mathbf{V}} \|\mathbf{W} - \mathbf{U}\mathbf{V}^\top\|_1$ can be decomposed into two alternative minimization problem. Each alternating problem is further divided into $P + K$ independent sub-problems. Each sub-problem can be in turn reduced to a simple convex quadratic problem whose global minimum can be computed efficiently. Notice that while the global minimum of each sub-problem can be derived by convex quadratic programming, the original problem $\min_{\mathbf{U},\mathbf{V}} \|\mathbf{W} - \mathbf{U}\mathbf{V}^\top\|_1$ is in general non-convex.

Chapter 5

Implementation and Experimental Results

5.1 Introduction

In this chapter, we present the detailed algorithm of layer extraction using subspace. We formulate the layer extraction as clustering in the subspace, where layers (clusters) become denser and easier to classify. To assure a low dimensional subspace and to extract small layers that do not have enough local measurements to form clusters in the feature space, we propose a semi-progressive layer extraction algorithm, where large layers are simultaneously extracted in the subspace. The large layers then provides guidelines for the extraction of small layers. We present experimental results of layer extraction on a variety of image sequences to demonstrate the effectiveness of our layer extraction algorithm.

5.2 Layer extraction algorithm

We are given an image sequence consisting of $F + 1$ images. Denote I_0 the reference image that are being segmented, and $\{ I_f \mid f = 1, 2, ..., F\}$ the other F frames. The core task of layer extraction algorithm is to segment the reference image I_0 into some number of 2D sub-images in such a way that pixels within each sub-image share common 2D parametric motions. The segmentation result of the reference image I_0 can then be used to initialize the layer segmentation of other images in the same

sequence.

Fig. 5.1 shows the overall procedure of the layer extraction algorithm. Its major steps are:

1. Divide the reference image into local regions, such as the homogeneous color regions or pre-defined $n \times n$ blocks. Estimate the motion of each local region between the reference frame and any other frame in the image sequence. Reshape the motion parameters of each local region into a column vector, and stack all of such column vectors into a matrix, which is the measurement matrix.

2. Robustly compute the low dimensional subspace by factorizing the measurement matrix. Outliers and small layers are excluded in this step.

3. Project the local measurements onto the subspace and cluster them the subspace into initial layers, which are large layers;

4. Progressively extract the previously excluded small layers;

5. Refine final layers using layer competition.

In the following we present each major step in more details.

5.2.1 Measurement matrix construction

We divide the reference image (the image being segmented) into K small $n \times n$ overlapped blocks, where adjacent blocks are overlapped with each other by half of the block size [1]. The first step in our layer extraction algorithm is 2D image motion estimation for each local block. We use hierarchical motion estimation algorithm [10] to estimate an affine motion $\mathbf{m}_f$ between the reference frame I_0 (the middle frame of the input images) and any other frame $\{I_f \mid f = 1, 2, ..., F\}$. We make use of the temporal coherence by using $\mathbf{m}_f$ as initialization when estimating $\mathbf{m}_{f+1}$. The reference affine motion is computed using the trimmed mean of all local measurements. According to Result 2.1, such trimmed mean corresponds to some (virtual) world plane. The local measurement of each block is then set to the relative affine transformation:

$$\Delta\mathbf{m}_i = \mathbf{m}_i - \mathbf{m}_f \tag{5.1}$$

[1] Overlapped blocks can effectively deal with occlusions or other noises.

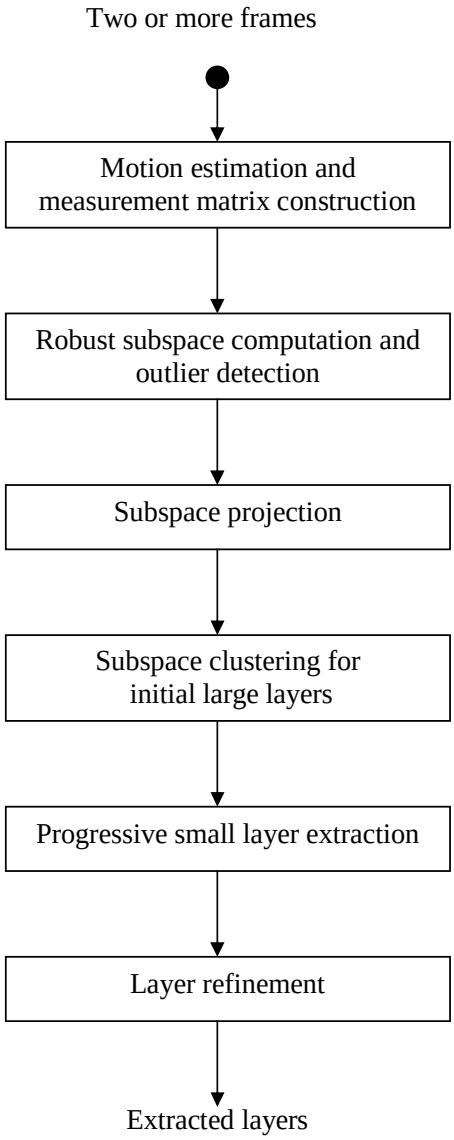


Figure 5.1: The procedure of layer extraction algorithm.

Notice that outliers may appears in the local measurements due to various reasons such as motion discontinuity, lighting change, non-rigid motion violating the assumed model, etc. Such outliers will be detected in the step of robust subspace computation.

For each block, its total $6F$ motion parameters are reshaped into a $6F$-dimensional column vector. The total K vectors are stacked together to form the measurement matrix $\mathbf{W}_{6F\times K}$. We scale the different components in the relative affine transformation $\Delta\mathbf{m}_f$ such that a unit distance along any component in the parameter space corresponds to approximately a unit displace at the image boundaries [103]. Such scaling makes the parameter space approximately *isotropic*. We use the image width w as the scale factor. Specifically, the matrix $\mathbf{W}_{6F\times K}$ is left-multiplied by the following scale matrix $\mathbf{S}$:

$$\mathbf{S}_{6F\times 6F} = \begin{pmatrix} \mathbf{s}_{6\times 6} & & \\ & \ddots & \\ & & \mathbf{s}_{6\times 6} \end{pmatrix}$$

where $\mathbf{s}_{6\times 6} = diag(w, w, 1, w, w, 1)$. Such linear transformation does not change the rank of $\mathbf{W}$, or the dimension of the subspace. In practice, we found that $\mathbf{S}$ is not a sensitive parameter. The final results do not change for a wide range of the scale amount w in matrix $\mathbf{S}$.

Frames nearer to the reference frame usually have better motion estimation. Therefore, we add a time-dependent decay parameter $\alpha(f)$ to weight the motions according to their frame distance to the reference frame: $\tilde{\mathbf{m}}_f = \alpha(f)\mathbf{m}_f$. Again, such weighting is equivalent to left multiplying $\mathbf{W}$ a non-singular diagonal matrix, a basic matrix transformation, and therefore does not change the rank of $\mathbf{W}$.

A local EM-like algorithm to improve local measurements

The affine motion directly estimated from a small block may over-fits the data inside that block, and could differs greatly from the global optimal motion, i.e., the corresponding layer model. Simply increasing the block size reduces over-fitting effect, but at the same time increases the chance that such block contains multiple motions, and therefore no longer corresponds to a planar patch in the scene.

To deal with the above problem, we design a local process to gradually expanding each small block into a larger *k-connected component (KCC)*. Each *KCC* should cover the original starting block, while at the same time the residuals inside it should be less

than some small pre-defined value ε. Being *k-connected* discards the thinly connected pixels, which are usually textureless and are from other layers other than the layer containing the starting block. The shape of a KCC could be irregular. Given a $n \times n$ block r in the reference frame, its KCC is derived by the following local process:

- Step 1 (motion estimation): Compute the homography of current KCC w.r.t. every frame in the sequence, and compute its motion compensated residuals using temporal selection [54].
- Step 2 (expanding/shrinking): Remove pixels with residuals larger than ε from current KCC. For each pixel p on the boundary of current KCC, a $n \times n$ ($n = 5$ in our experiments) test window w is centered at p. w is added to (removed from) KCC if the majority of the pixels inside w have residuals less (larger) than ε. In our experiments, Step 2 is repeat four times in one iteration to save computation time.
- Step 3 (evaluation): Enforce the *k-connected* requirement. If the resulted KCC becomes stable (the change of area or motion parameters is small enough) or the maximum number of iterations has been reached, then the block r is declared as an inlier. If the resulted KCC does not cover the original block r, then r is marked as an outlier. Otherwise, goto Step 1.

The parameter ε specifies the noise level to be tolerated. It depends on the geometry (the planarity) and the texture of the underlying layer, and is set to a conservative small value (15 in our experiments, with the pixel intensity range of $[0, 255]$). The above procedure is similar to EM algorithm, but the assignment of pixels (expanding/shrinking) happens only along the current boundaries of each KCC.

Fig.(5.2) shows an example of the above local process, given a ten-frame sequence. The small rectangle in Fig.(5.2a) shows the initial block r. Fig.(5.2b) shows the residuals [2] after compensating the motion estimated based on pixels inside r. Although pixels inside r are well compensated, those pixels outside r but inside the same layer of r (flower bed) are not well compensated. Fig.(5.2c) shows the *k-connected component* after five iterations of the above process. Fig.(5.2d) is the final converged result. We can see that the pixels in the same layer are much better compensated in (c) and (d) than that in (b), which indicates a better global motion estimation result under

[2]The residuals are scaled up for visualization.

the tolerable noise level ε. In our experiments, we find that three to five iterations are enough to obtain stabilized motion parameters. For our purpose, we do not require the local process to converge. Three iterations are enough to produce the desired local measurements. In Fig.(5.2e), the initial region r contains two motions (the tree and the flower garden). The resulted KCC does not cover r, therefore r is marked as an outlier.

5.2.2 Robust subspace computation

The subspace is computed by factorizing the measurement matrix $\mathbf{W}$:

$$\mathbf{W} = \mathbf{U}\mathbf{V}^\top \tag{5.2}$$

where the columns of $\mathbf{U}$ are the orthonormal bases of the linear subspace. Due to the outliers in the local measurements, robust method is necessary here. Small layers are also excluded as outliers by the kNND procedure at this stage. The dimension of the subspace can be bound by choosing appropriate value of k according to Eq. (4.23).

Please see Chapter 4 for details on how the subspace is computed.

5.2.3 Subspace clustering for initial layer models

Once we compute the subspace, we project the local measurements onto the subspace by:

$$\mathbf{V} = \mathbf{W}^\top\mathbf{U} \tag{5.3}$$

where the i-th column of $\mathbf{V}^\top$ is the projected result of the i-th local measurement (the i-th column in measurement matrix $\mathbf{W}$).

We now apply a simple clustering algorithm to cluster the inliers in the d-dimensional subspace for the initial layers. The clustering task becomes much easier in the low dimensional space with outliers that have been detected. There are many existing clustering algorithm (see [50]) available. We use the simple mean-shift [18] based clustering algorithm. Mean-shift algorithm has also been successfully applied to color segmentation, motion estimation, and non-rigid object tracking [21, 22, 23, 20, 24]. We adopt this algorithm in our clustering step because: (1) it is non-parametric and robust; and (2) it can automatically derive the number of clusters and the cluster

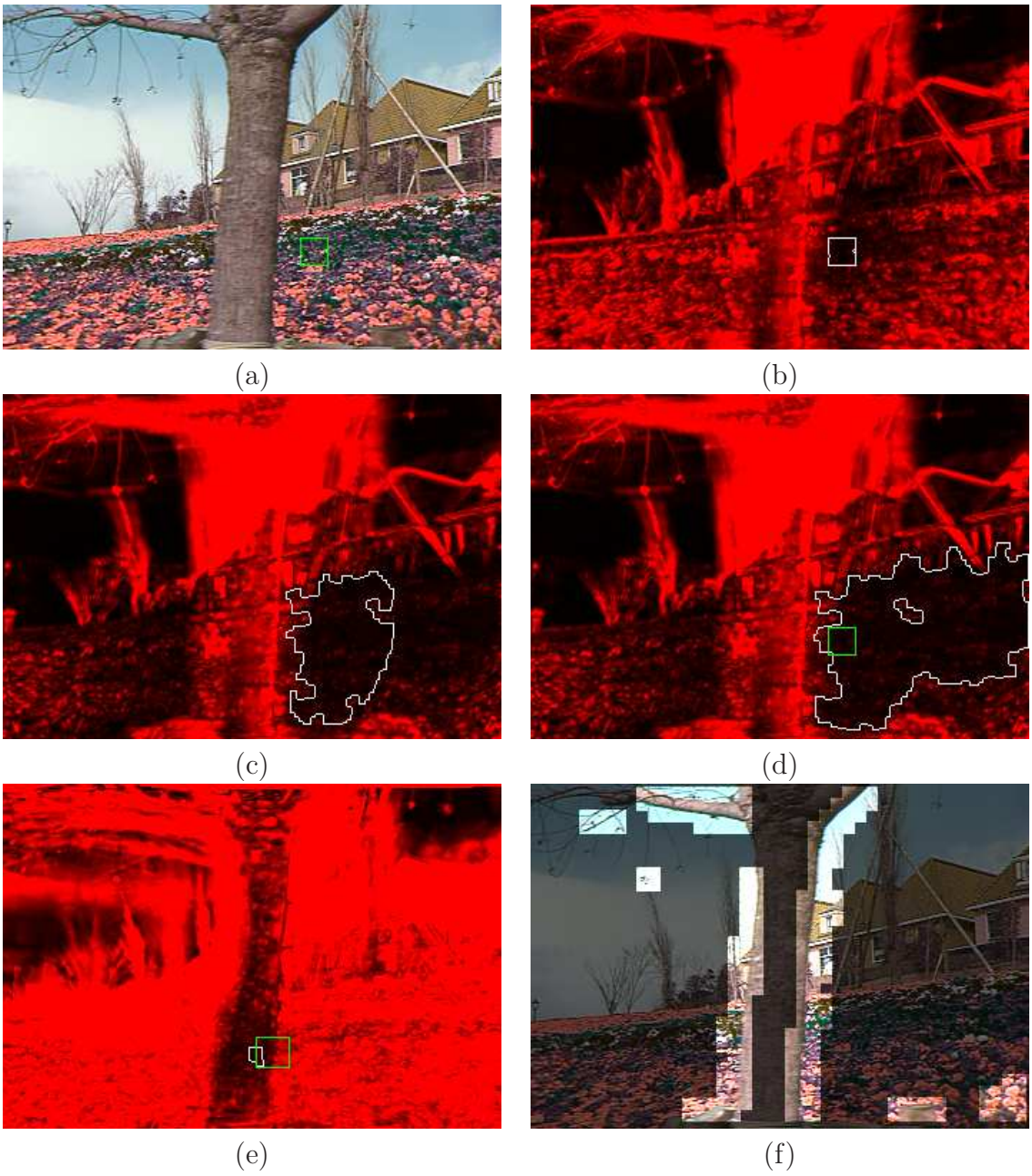


Figure 5.2: Deriving the *k-connected component (k=5)*: (a) initial region r given by the white rectangle; (b) residuals after compensation based on the motion estimated using the initial region r; (c) residuals and *k-connected component* after five iterations of the local process; (d) converged residuals and *k-connected component*; (e) outlier region (final KCC does not cover the original block); (f) detected outlier blocks in local measurements using local EM procedure.

centers. Refer to [21, 22] for a clear description and details on this algorithm. Here we just point out how we set the window radius r, a parameter that determines the resolution of clustering, therefore determines the number of layers.

r can be set according to the covariance matrix of $\mathbf{W}$ [21]. In all of our experiments it is set adaptively by:

$$r = 0.2\sqrt{\frac{\lambda_d^2 + \lambda_{d+1}^2}{2}} \tag{5.4}$$

where d is the dimension of the subspace, and λ_i are the eigenvalues of $\mathbf{W}$. In other words, r is set to cover the noise variance, while be less than the smallest of signal variances. We have found by experiments that a wide range of r produces the desired results, due to the use of subspace and multiple frames, which verifies the theoretical analysis of Result 3.1 and 3.2. In general, the parameter r is much more intuitive to set the the "coding length" in the MDL approach that used to determine the number of layers.

5.2.4 Layer competition and refinement

Once we are given the initial layers and their model, we can refine the layers by re-assigning pixels to layers and re-compute the layer model.

To utilize the spatial coherence existing in single image, we assign color regions, instead of individual pixels, to layers. The assumption we use here is that each homogeneous color region is a planar patch. Such assumption is generally valid for images of natural scenes, and has been used in motion analysis and stereo [12, 105, 33, 88]. Very fine-grained color over-segmentation [21] is used here to assure that each homogeneous color segment is contained inside only one layer.

We use layer competition to assign each color segment s in the reference frame to the layer L that best describes the motion of the pixels inside s. To deal with occlusions, temporal selection [54] is used here to determine the best layer model for a color region. The idea of temporal selection is based on the assumption that a pixel occluded in one temporal direction is usually not occluded in the other temporal direction. We use only the non-occluded frames to determine the ownership of each segment s. The following steps are used to assign segment s to one of the current layers using temporal selection:

- Compute the motion-compensated image residuals $e(f, L)$ for s:

$$e(f, L) = \sum_{(x,y) \in s} [I_0(x, y) - I'_{f,L}(x, y)]^2 \tag{5.5}$$

  where I_0 is the reference image, and $I'_{f,L}$ is the resulted image of warping Frame f towards the reference image using the motion of layer L from the reference frame to Frame f.

- Sort $\{e(f, L) \,|\, f = 1, ..., F\}$ in non-decreasing order.

- Use temporal selection to compute the cost $c(L)$ of assigning s to Layer L as the sum of the first half of the sorted sequence $\{e'(i, L) \,|\, i = 1, ..., F\}$:

$$c(L) = \sum_{i=1}^{F/2} e(i, L)$$

  where $e'(1, L) \leq ... \leq e'(F, L)$.

- s is assigned to Layer L, where $L = \arg\min_L c(L)$.

we may still observe some spurious regions, largely due to noises in the images, or the disparities between color segment boundaries and motion boundaries in some places. Such spurious regions appear at different random positions in different frames, and can be removed by using the fact that each layer corresponds to a *rigid and consistent* plane in the scene, such that its shape in the image domain is coherent or changes *gradually* across time [89].

5.3 Layer depth ordering from visibility reasoning

For any two layers that share common boundaries, we can determine their depth ordering using visibility reasoning. Suppose we are given the layer mask of the reference image, the following steps are used to determine the depth ordering between two layers L_a and L_b:

1. Warp L_a and L_b from the reference image to other frames in the given image sequence, using the layer motions.

2. If L_a and L_b overlaps in the warped mask at the f-th frame I_f, then accumulate the color difference at the overlapped pixels:

$$e_a = \sum_i \left[I_f(i) - I'_a(i)\right]^2$$
$$e_b = \sum_i \left[I_f(i) - I'_b(i)\right]^2$$

where i is the index of overlapped pixels, I'_a and I'_b are the warped image of L_a and L_b respectively.

3. The layer with less color difference has smaller depther. For example, if $e_a < e_b$ then $Depth(L_a) < Depth(L_b)$.

We derive the depth ordering for all layer pairs as long as they share common boundaries. The overall depth ordering among layers can be deduced from all of the depth relationship of layer pairs.

Note that if two layers do not overlap with each other, then we can not use the above simple visibility reasoning to determine their depth ordering relationship. In many applications we do not need the depth ordering between two un-overlapped layers. Or we can apply structure from motion (SFM) using layer representation to recover the projective depth of the layers. Projective depth is enough for the purpose of determining depth ordering.

5.4 Progressive small layer extraction

Small layers are harder to identify, since (1) they do not have enough data points to form distinctive clusters, and (2) they often have unreliable motion estimations. Large layers, on the other hand, are easier to extract since (1) they reside in a low dimensional subspace, and (2) they contain enough local measurements to form distinctive clusters in the subspace. Therefore, in the low dimensional subspace, the large layers form very discriminative clusters that can be reliably identified. Instead of trying to extract the difficult small layers at the very beginning, our semi-progressive algorithm first simultaneously extracts the large layers in the low-dimensional subspace. The excluded small layers, previously hard to identify, are then progressively extracted against the initial large layers. The final layer description is derived in a competition

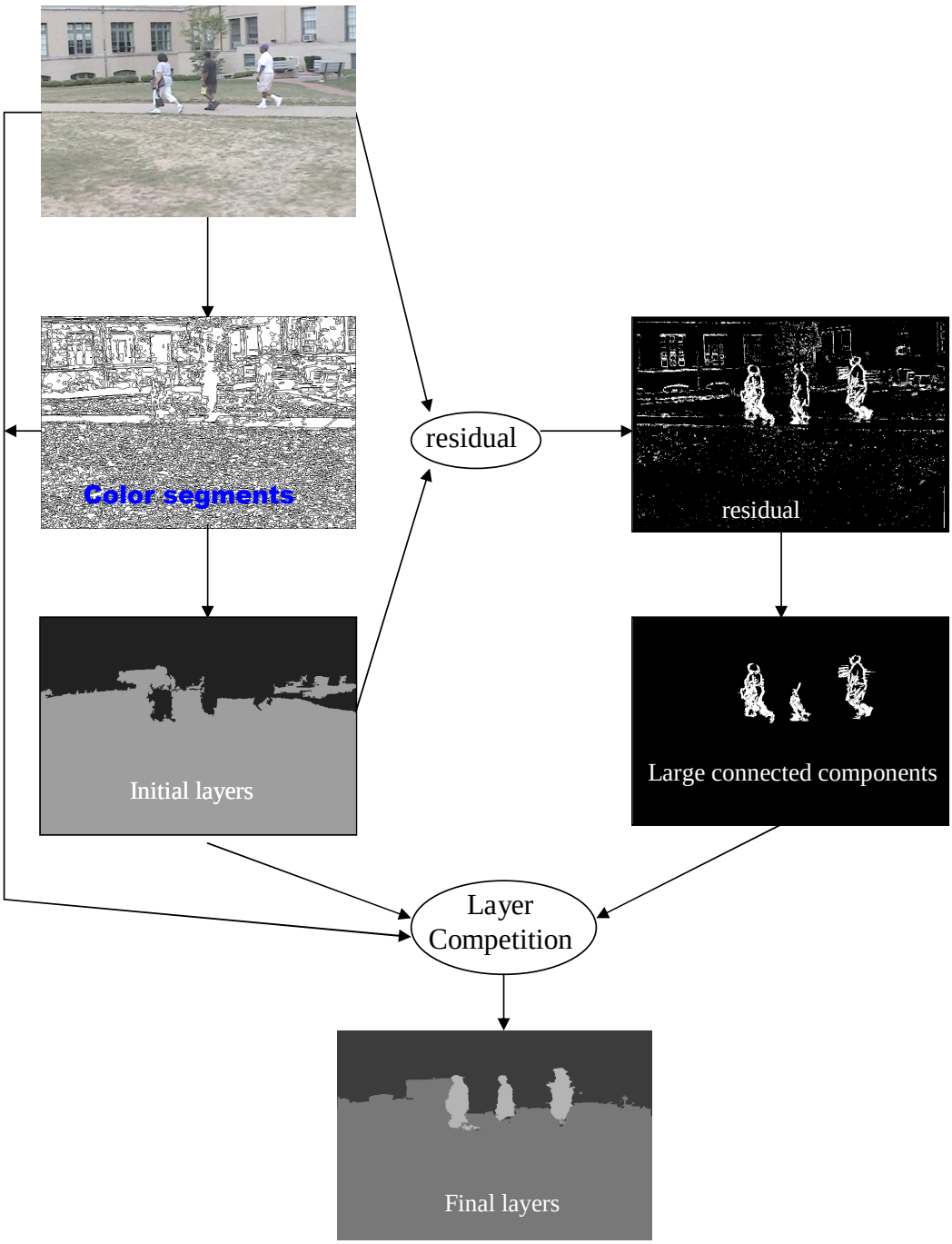


Figure 5.3: The overall procedure of semi-progressive layer extraction algorithm. The initial layers are first derived by subspace clustering, then the integrated squared residual image (ISRI) is computed according to Eq. 5.6. The large connected components in the ISRI are detected and initialized as new layers, which compete with the initial layers for final layer support to produce the final layer map.

frame work where all existing layers compete for over-segmented color regions.

Fig. 5.3 shows the overall procedure of semi-progressive layer extraction algorithm. The overall procedure is very similar to background modelling [98] for foreground object detection, except that we uses multiple layers to model the background. In our approach, the large layers are first extracted using the algorithm presented in Section 5.2. The models (motions and supports) of large layers are used to compensated the images in the input video, which produce one residual image R_f for each frame I_f. Denote $r(p, f)$ the pixel p in R_f, then the *integrated squared residual image* $s(p)$ is defined as as the sum of squared residuals across F frames at every pixel p:

$$s(p) = \sum_{f=1}^{F} r^2(p, f) \tag{5.6}$$

We assume the residual $r(p, f)$ is drawn from a common zero-mean Gaussian distribution (i.i.d.) [3], therefore $s(p)$ follows χ^2_F distribution. Pixel p is marked as an outlier if its $s(p)$ is outside the 95 percentile confidence interval of the χ^2_F distribution.

The small layers excluded by the kNND procedure will form connected components of outliers due to their large values in the integrated residual image. The largest connected component is initialized as a new layer, which then competes with the existing layers for layer support. The new layer survives only if the resulted layer segmentation increases the overall likelihood according to some application dependent criteria, or model selection criteria such as MDL (see [8]). The progressive procedure stops when a newly initialized layer can not survive the layer competition.

5.5 Experimental Results

We show the experimental results of applying our subspace approach on several real video sequences of dynamic and static scenes.

5.5.1 *flower garden* sequence

We use *flower garden* sequence to illustrate the steps in layer extraction algorithm. The input image sequence contains the five images. Fig. 5.4 shows the first, middle

[3] Variance is estimated using median: $\hat{\sigma} = 1.4826\,\text{median}_i\,|r_i|$.

(a) first frame

(b) middle frame

(c)last frame

Figure 5.4: *flower garden* sequence. The scene is static and the camera is translating to the right approximately horizontally

and last frames. In this sequence, the scene is static and the camera is translating to the right approximately horizontally.

Local measurements and measurement matrix

The middle frame is chosen as the reference image to be segmented into layers. We divide this image into 48×48 (pixels) small blocks, with adjacent blocks overlap with each other by 24 pixels. We estimate the affine transformation for each block.

For each block, there are four affine transformations, one for each frame in the image sequence (except the reference image). These four affine transformations are reshaped into a 24-dimensional column vector. We derive the measurement matrix $\mathbf{W}$ by collecting all of such column vectors from all blocks.

Robust subspace computation

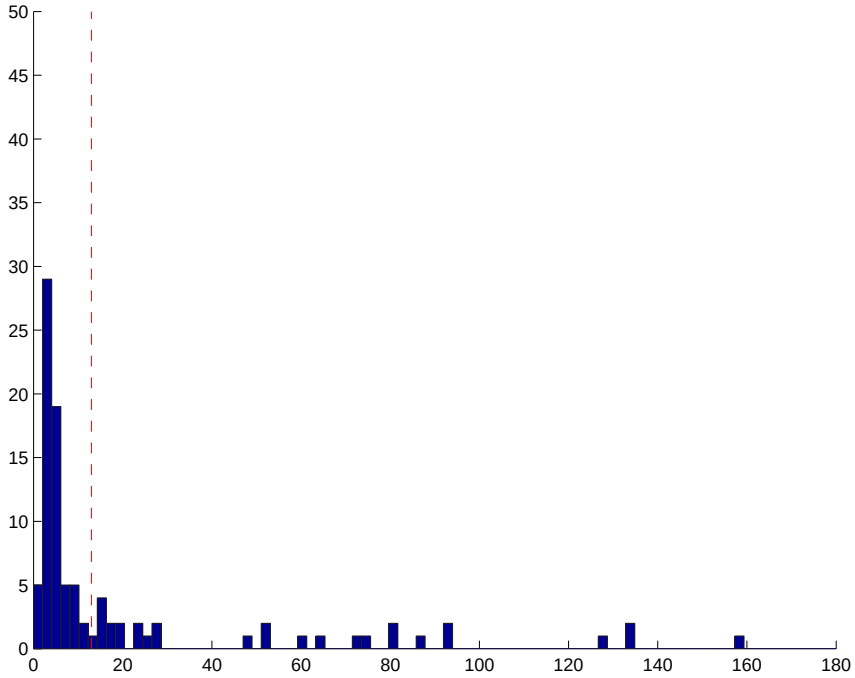
We use the kNND metric to detect the extreme outliers. For each column in the measurement matrix $\mathbf{W}$, we compute $d_i(k)$, the distance to its k-th nearest neighbor

in the 24-dimensional parameter space. According to Eq. (4.23), we choose $k = 6$ to assure that the subspace dimension will be less than 4. It also specifies that we expect the layer size in the initial layer map will contain at least 6 local blocks (overlapped). Fig. 5.5(a) shows the 1D histogram of all $d_i(k)$'s. As we can see, there is one obvious peak which consists of $d_i(k)$'s of inliers. We simply detect the first peak by using the first valley following the first peak, or the 90 percentile, whichever is smaller. The red vertical line shows the boundary (threshold) to determine the extreme outliers, i.e., blocks whose corresponding $d_i(k)$ is on the right of the line are declared as extreme outliers. Notice that in this step, we are conservative in the sense that we want to remove as many as possible the extreme outliers.

Fig. 5.5(b) shows the corresponding extreme outliers in the image domain. The sources of extreme outliers are:

- The blocks containing multiple motions are excluded, such as the one that contain tree trunk and background (garden or house).
- Blocks with few texture suffers aperture problem and also result in unreliable motion estimation [82]. The textureless blocks, mostly from the sky, are excluded as extreme outliers.
- The tree branch has several places that self occludes, and result in bad motion estimation.
- Some remaining blocks in the tree branch has good motion estimation. But the cluster in the parameter space is too small, and is excluded as small layers. This can also be seen from Fig. 5.7(b), where in the bottom a small cluster consisting five data items is marked as extreme outliers.

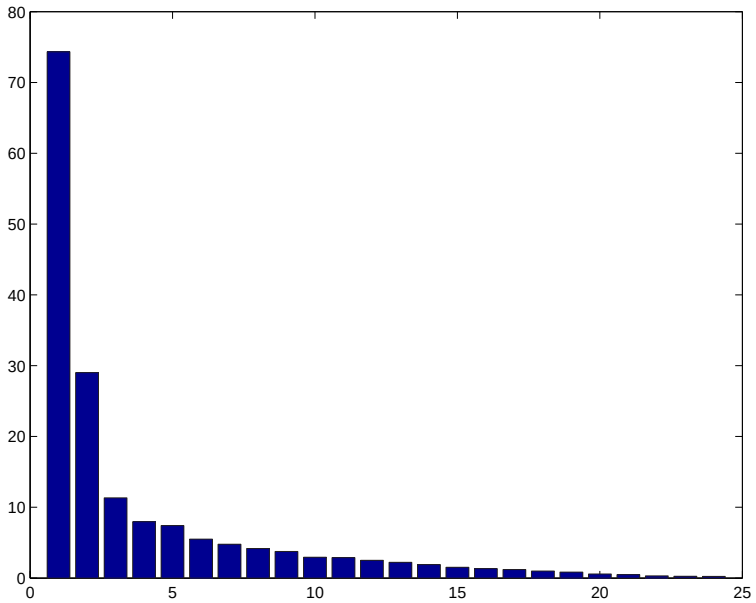
Once the extreme outliers are identify, we can safely apply the robust SVD algorithm to compute the subspace and detect the structure outliers. Fig. 5.6 shows the singular values of the robust SVD result. It clearly indicates that there is a 2D subspace of the original 24 dimensional space! Fig. 5.7 shows the subspace projection in the 2D signal subspace. As we can see, the inliers form three distinct clusters in the signal subspace. We also plot the components of projecting the data points onto the first two dimensions of the noise subspace (22 dimensional). The projection of the inliers in the noise subspace roughly forms a Gaussian cloud.



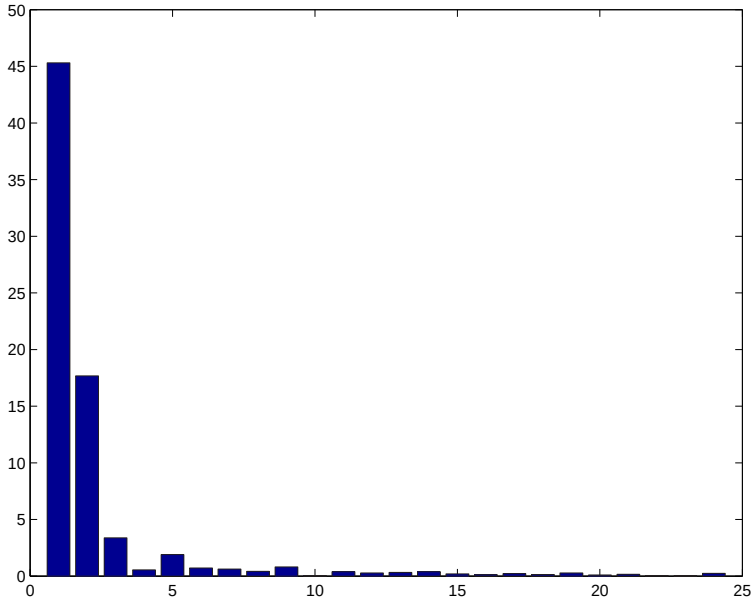
(a)

(b)

Figure 5.5: Detecting outliers using kNND. (a) Histogram of the 1D kNND; (b) Extreme outliers in the image domain.

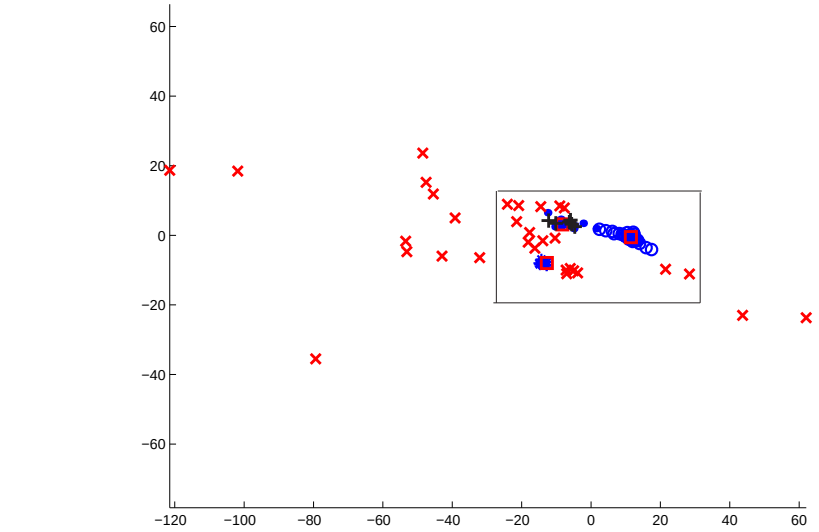


(a) singular values $\lambda_i, i = 1, ..., 24$

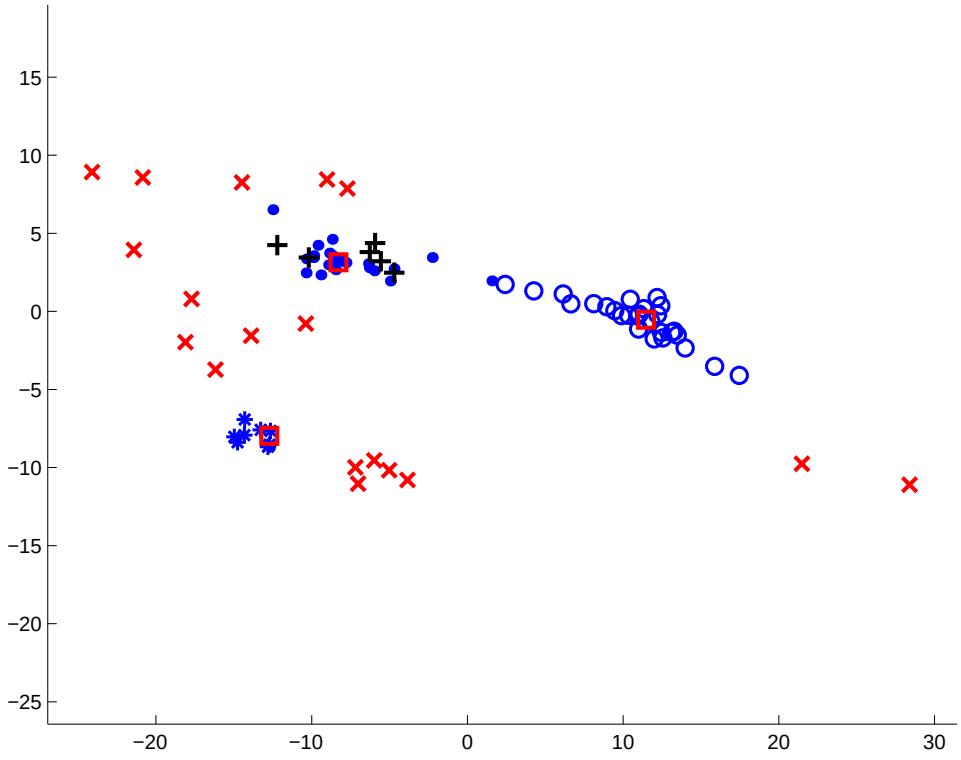


(b) difference between adjacent singular values.

Figure 5.6: Singular values of SVD algorithm applied to the measurement matrix after removing outliers.



(a)



(b) zoom-in of the rectangle part in (a)

Figure 5.7: Projecting local measurements on to the 2D signal space. (b) shows the zoom-in of the data points in the rectangle in (a). The blue data points are inliers. The red × indicates the extreme outliers. The black + shows the detected structure outliers. The red □ shows the cluster centers of the mean shift clustering algorithm.

Subspace clustering for initial layers

We apply the mean shift algorithm to the inliers in the 2D signal subspace. The window size r of the mean shift algorithm is determined according to Eq. (5.4). Fig. 5.7 shows the clustering result, where the red □ indicates the cluster center. Three clusters are detected. Fig. 5.9(a) shows the three clusters in the image domain (the masks of the initial layers).

In experiments, we find a wide range of r produces same results. Specifically, $r \in [1.5, 10]$ results in same clusters when using subspace clustering. The reason of having a wide range of r is because the use of subspace and multi-frames, which result in very discriminative clusters.

For comparison, we show the clustering result ($r = 5.0$) without using subspace (i.e., clustering in the original subspace) in Fig. 5.9(b), where 16 layers are resulted. Many layers are over-segmented, while some blocks from different layers are clustered together as one same layer.

Layer competition

Once we have the initial layers, we can recompute the layer motion, and re-assign pixels to layers, under the frame work of layer competition. The motion computation and pixel assignment can be iterated multiple times. Fig. 5.10(b) shows the result of just one iteration of layer competition, where layers compete for homogeneous color regions shown in Fig. 5.10(a). The color segmentation is very fine grained. As we can see, even visually homogeneous regions (sky, tree) are still segmented into many small regions. Therefore it is generally assured that a homogeneous color region belongs to one single layer.

Progressive small layer extraction

Fig. 5.11(a) shows the integrated residual image of the tree layer. It indicates that the tree branch has different motion from the tree trunk. The reason is due to different depth. Such motion difference is accumulated across frames. If desirable, the progressive small layer extraction algorithm can be applied here. Fig. 5.11(b) shows the progressive layer extraction result. The tree layer is further divided into two layers: the tree trunk layer and the tree branch layer.

Fig. 5.12 shows more final results of layer segmentation of the flower garden sequence. The segmentation result is consistent across the whole sequence.

5.5.2 Mobile & calendar Sequence

This sequence contains eleven frames, where there are three moving objects and one static background plane. The calendar is moving upward, the train is pushing the ball, and the camera is zooming and tracking the train. Figure 5.13(a-c) show three frames.

The local measurement is 48×48 pixels in size, where adjacent blocks are overlapped by half. The histogram of kNND metric $d_k(i)$ is shown in Figure 5.13(d), with $k = 10$ set according to equation (4.23). Measurements from *all* large layers form the first dominant peak in this one-dimensional histogram. Outliers and small layers have large $d_k(i)$, and therefore most of them are far away from the first peak. The vertical dash line show the boundary of the detected peak. The excluded outliers and small layers are highlighted in Figure 5.13(e). As we can see, these are measurements of patches containing either discontinuous motion or small layers (ball and train).

We detect a two-dimensional signal subspace of $\mathbb{R}^{60}$ based on equation (4.25) with $t = 95\%$. In this subspace, the inliers from two distinctive clusters that can be reliably identified by the mean shift algorithm, as shown by the blue "$\bullet$" and "$\circ$" in Figure 5.13(f) [4]. For illustration, we also project outliers onto this subspace, with extreme outliers shown by red "$\times$", and structure outliers by black "+". Figure 5.13(g) shows the noise subspace, where blue "$\bullet$" represents the noise components of data from the two different clusters. Note that the dimension of the noise subspace is usually very high, and we only plot the first two dimensions. We can see that the noise components of inliers follow Gaussian distribution quite well.

Figure 5.13(h) shows the result of color over-segmentation. Figure 5.13(i) shows the two initial layers by assigning each color segment to its best initial layer model. Figure 5.13(j) shows the integrated squared-residual image, which is also used to indicate the uncertainty of layer ownership in (i). The small layers (ball and train) are clearly outliers in this residual image. They are progressively extracted by initializing new layers from the largest connected components of the residual outliers, and then

[4] The red "$\square$" in (f), visible if zoomed-in or printed in color, indicates the cluster centers derived by mean shift algorithm.

compete with the existing layers for layer support. Two such small layers are detected. Figure 5.13(k) shows the final layer map. Note that in Figure 5.13(k), the ball casts a shadow to its right on the white part of the calendar, which results in the "tail" of the ball layer. Figure 5.13(l) shows another final layer result on the last image in this sequence. As we can see, four layers (each has rigid motion) are extracted, including the background, calendar, train, and ball.

Fig. 5.14 shows more final results of layer segmentation of the mobile and calendar sequence. The segmentation result is consistent across the whole sequence.

5.5.3 Walking person sequence (I)

In this sequence, a hand-held camera is tracking three people walking in a static background. Figure 5.15(a-c) show three frames in this eleven-frame sequence. The human motions are non-rigid or articulated, and their corresponding local measurements are often unreliable. Figure 5.15(d) shows the excluded outliers and small layers by the kNND procedure with $k = 10$. A two-dimensional subspace is derived based on the remaining data. The ground and building walls are simultaneously extracted as large layers, as shown in Figure 5.15(f). The integrated squared residual image is shown in Figure 5.15(g), where the three walking people clearly form connect components with large values, based on which we progressively extract three layers of the walking people. Figure 5.15(g) also indicates the layer ownership uncertainty in (f). Figure 5.15(h) shows the final layer extraction result. Some parts of the human (e.g.,feet) with articulated motions not modelled by the major human body are mistakenly assigned to the ground layer or the building wall layer. Still, it clearly indicates five layers in the scene: the three walking people, the ground, and the building. Such results are useful for subsequent image analysis for scene understanding (such as human detection).

Fig. 5.16 shows more final layer segmentation results of the walking-people sequence. The segmentation result is consistent across the whole sequence.

5.5.4 Walking person sequence (II)

In this sequence (Fig. 5.17), two persons are walking together on a road in campus. The kNND procedure effectively detect and remove the extreme outliers, including

small layers (two walking person), blocks containing motion boundaries, textureless blocks, as shown in Fig. 5.17(c). Three initial layers are extracted by clustering small blocks in the subspace as shown in Fig. 5.17(e). The initial layers after layer competition is shown in Fig. 5.17(f). The walking persons are assigned to tree layer, which has big residual values in the integrated residual image shown in Fig. 5.17(g). The progressive layer extraction algorithm identifies these large components, and detect them as new layers, as shown in the final layer segmentation in Fig. 5.17(h). Fig. 5.18 shows more results of this sequence.

5.6 Discussion

5.6.1 Planar approximation

Layer representation assumes that the 3D scene can be approximated by some number of planes. In real world, perfect planes do not exist in general. In this section we study when a portion of the scene can be approximated by a plane.

The factors we need to consider are:

1. the noises in 2D local measurements
2. the position of the cameras, including the distance from the camera to the scene, and the distance between cameras
3. the 3D parallax of the scene
4. the size of the window ($n \times n$ block) used in local measurement
5. the size of the planes in the scene.

For example, consider the scene shown in Fig 5.19. There are two parallel planes in the scene taken by two cameras. The 2D parallax is measured by the difference between two planes Δd (i.e., 3D parallax), and the translational distance between the two cameras $\mathbf{t}$. When the parallax is not *measurable* in the images, then the scene can be well approximated by one plane. By *measurable* we mean the two clusters corresponding to the two planes in the parameter space are distinctive.

To simplify the analysis, let us assume that the internal camera matrix is $diag(1, 1, 1)$. The homography induced by a plane $\boldsymbol{\pi} = (\mathbf{v}^\top, 1)^\top$ is:

$$\mathbf{H} = \mathbf{R} - \mathbf{t}\mathbf{v}^\top$$

The difference between these two homographies induced by two planes is:

$$\Delta\mathbf{H} = \mathbf{t}(\mathbf{v}_1 - \mathbf{v}_2)^\top$$

The distance between these two homographies in the parameter space is:

$$e = \|\mathbf{S}\mathbf{t}(\mathbf{v}_1 - \mathbf{v}_2)^\top\|_2$$

where $\mathbf{S} = diag(s, s, 1)$ is the scaling matrix with s a constant, and $\|\cdot\|_2$ is the Frobenius norm ($L2$ norm). As we can see, e is determined by the 2D parallax, which in turn is determined by the 3D scene parallax $\Delta\mathbf{v}^\top = (\mathbf{v}_1 - \mathbf{v}_2)^\top$ and the translational distance between cameras $\mathbf{t}$.

Now assume the noises in the local measurements can be modelled by Gaussian distribution with standard deviation σ. The cluster (layer) discriminability is then:

$$q = \frac{e}{\sigma} = \frac{\|\mathbf{S}\mathbf{t}(\mathbf{v}_1 - \mathbf{v}_2)^\top\|_2}{\sigma} \tag{5.7}$$

In the example shown in Fig. 5.19, the cluster discrininability is:

$$q = \frac{1}{\sigma}\frac{t}{d}\left[\frac{1}{1 - \frac{\Delta d}{d}} - 1\right] \tag{5.8}$$

As we can see from Eq. (5.8), there are several coupled factors that determine whether the two layers is separable in the image domain: the measurement noise σ, the distance between two cameras t, the 3D parallax Δd, and the distance between the scene and the camera. Fig. 5.20 shows the effect of changing the above parameters. As we can see, in case (a) it is easy to separate the two layers shown in Fig. 5.19. When the cameras are too far away from the scene, or the measurement noise is too big, the two planes will be mixed together in the parameter space and will be extracted out as one layer in the image domain. Similar effects have been seen in the experimental results of the *flower garden* sequence (see Fig. 5.4) and the *walking person* sequence (see Fig. 5.15), where the planes in the background are far away from the camera

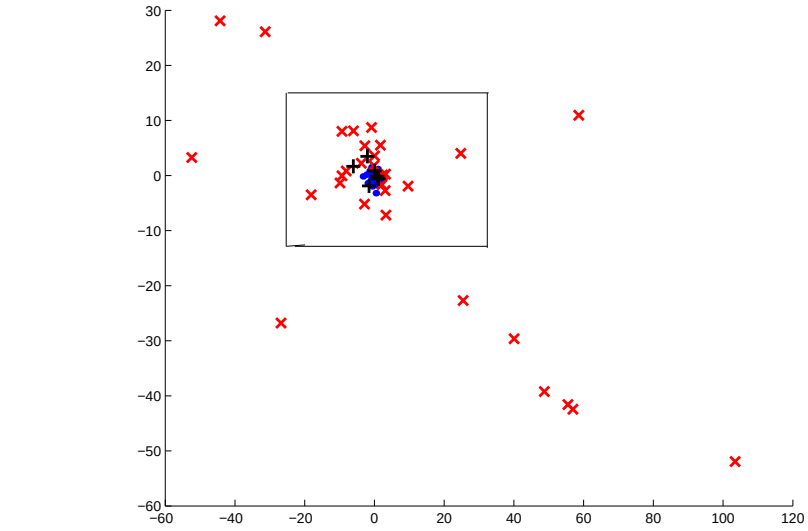
(i.e., large value of d in (5.8)), and can not be separated into different layers due to measurement noise.

We also need to take into account the size of the planes. If the 3D planes are very small (in the image domain, much smaller than the window size used to get the 2D local measurements), then they will not be able to be separated into different planes. The reason is that in the step of local measurement, one local block (region) in the image will contain several planes in the scene, and the difference among planes in the scene is not measurable. Fig. 5.21 shows such effect.

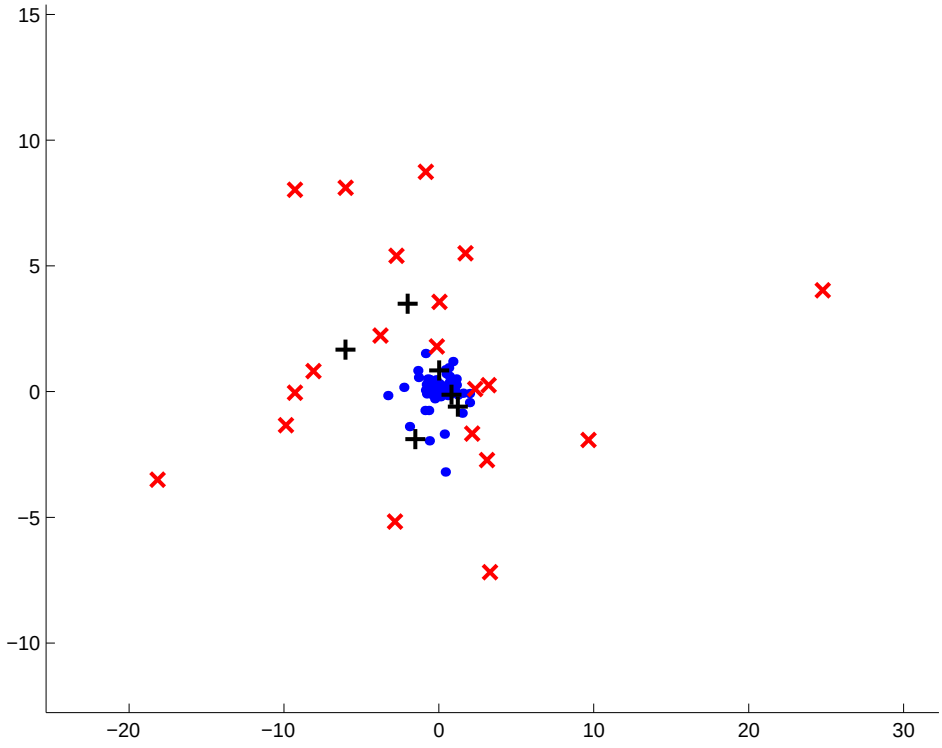
5.6.2 Evaluation

We have demonstrated the effectiveness of our layer extraction approach using real data in a qualitative way. We also use some standard video sequences (e.g., *flower garden* sequence and *mobile & calendar* sequence) to compare our approach against other approaches. Our approach achieve much better results in terms of the quality of layer boundaries, as well as the extraction of small layers that tends to be missed in other approaches.

In general, how should one evaluate the quality of the result? Some quantitative metric is desirable for comparing layered representations obtained by our approach against other methods. Ideally, a quantitative measure would allow us to close the loop of layer extraction. In other words, we could show the sensitivity of the result to some of the parameters as the number of layers, the number of frames used, the dynamic range of the imagery, sensor noise, etc. One possible quantitative measure is the difference between the original video frames and the synthesized video frames using the estimated layer model. This is part of our future work.

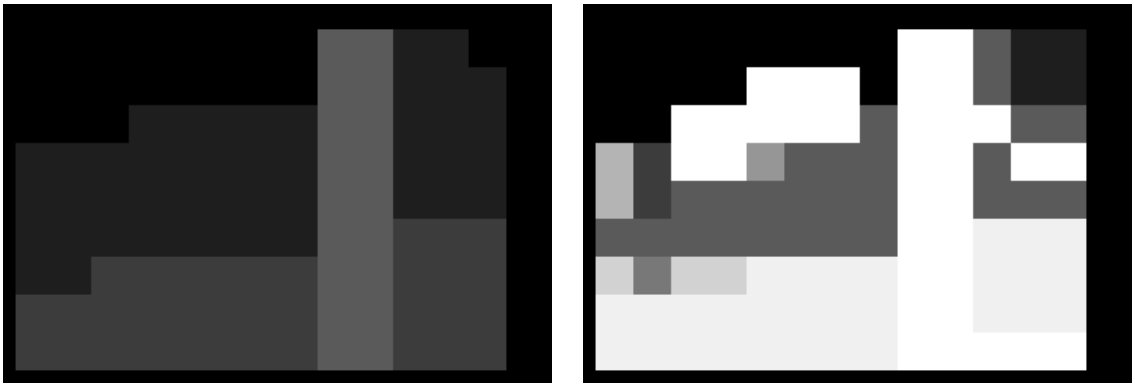


(a)



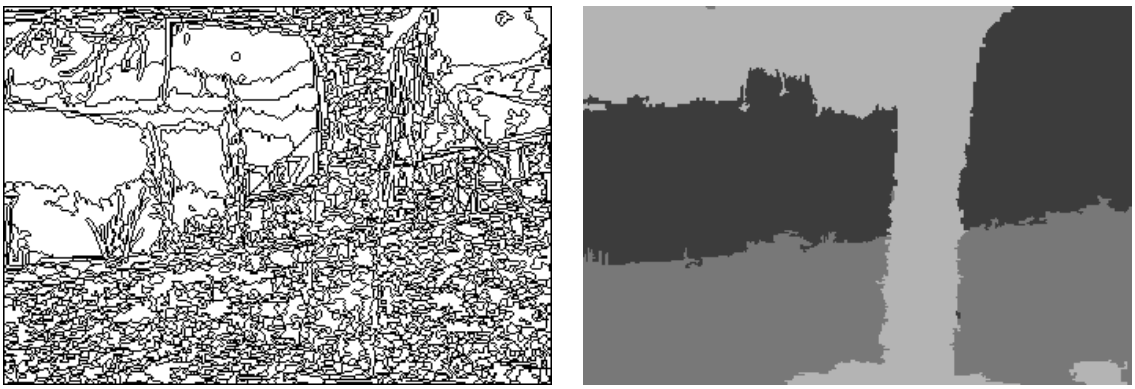
(b) zoom-in of the rectangle part in (a)

Figure 5.8: Projecting local measurements on to the noise subspace. (b) shows the zoom-in of the data points in the rectangle in (a). For the purpose of illustration, we only plot the components of the first two dimensions in the subspace. The blue data points are inliers. The red × indicates the extreme outliers. The black + shows the detected structure outliers.



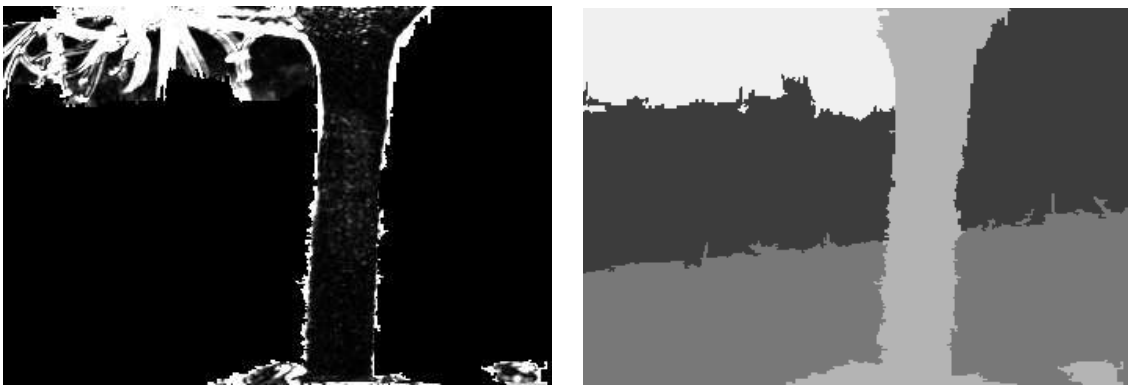
(a) (b)

Figure 5.9: Initial layer segmentation from clustering. (a) initial layers from clustering blocks in the subspace, where black pixels are those excluded as outliers; (b) initial layers from clustering blocks in the original space.



(a) edges of color segments (b) result of layer competition

Figure 5.10: Layer segmentation after layer competition. (a) edge map of over color segmentation; (b) result of initial layers competing for color segments.



(a) (b)

Figure 5.11: Progressive layer extraction. (a) Integrated residual image of the tree layer; (b) final layer segmentation.

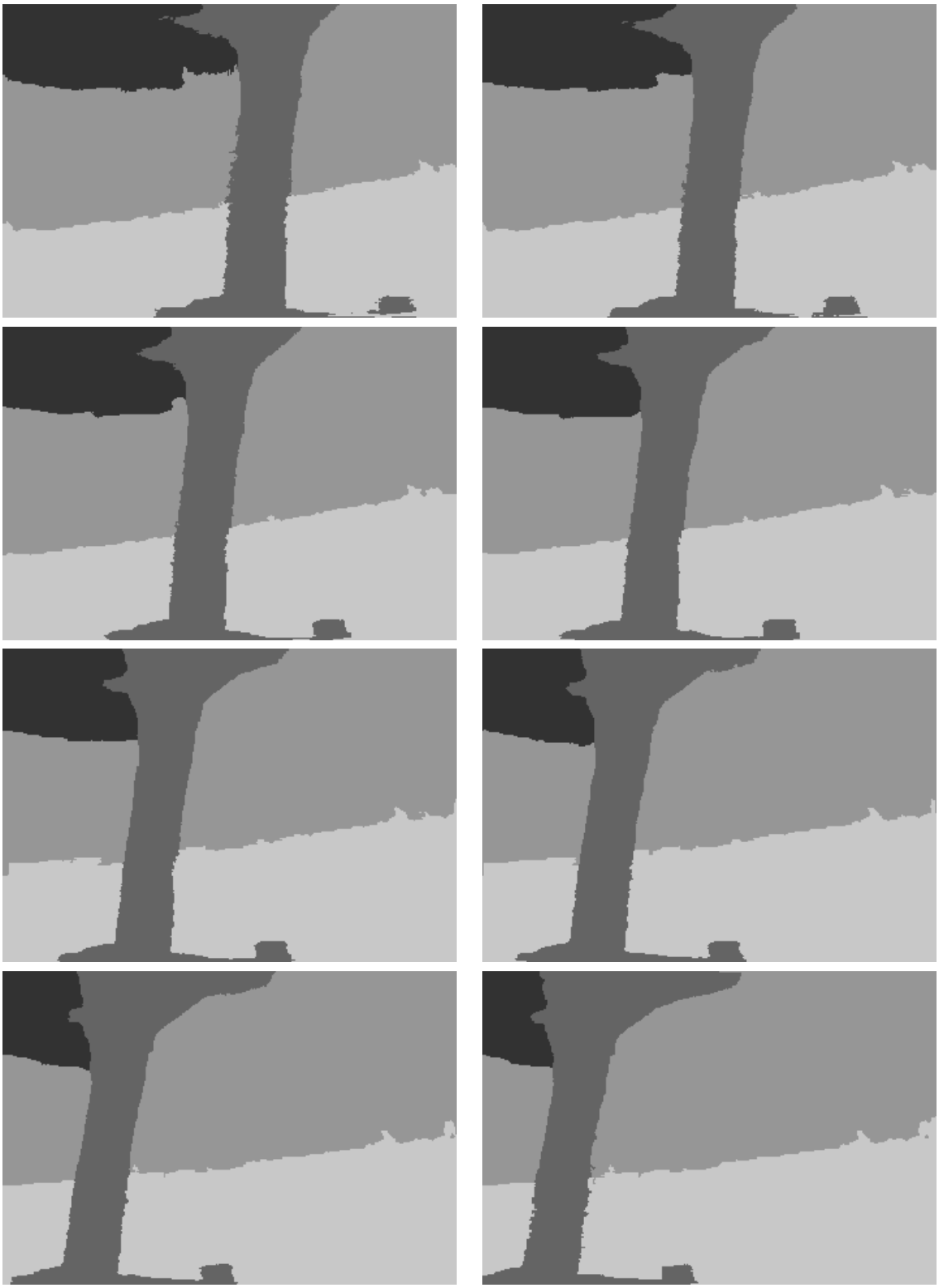
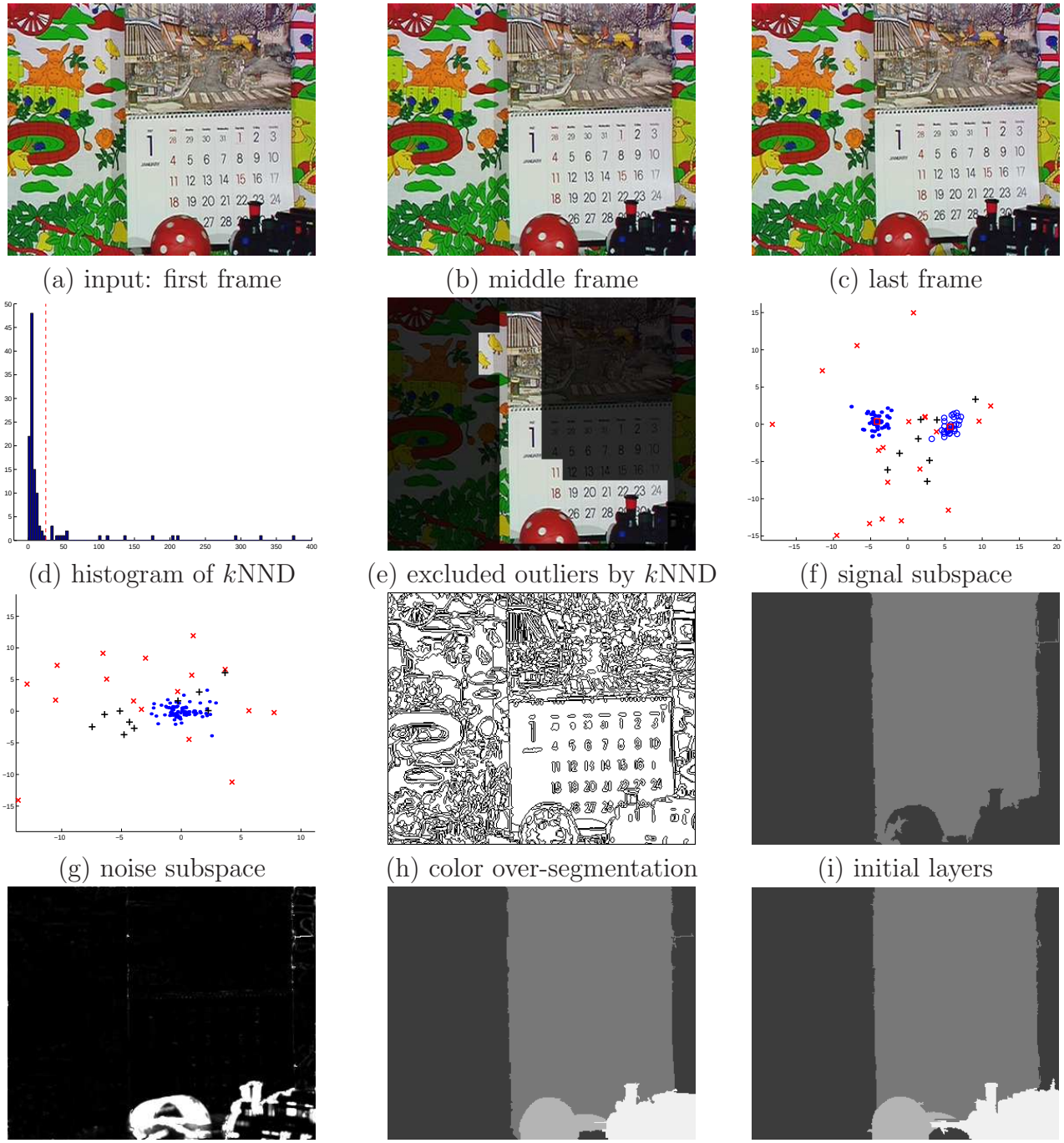


Figure 5.12: Final layer segmentation of flower garden sequence.



(a) input: first frame (b) middle frame (c) last frame

(d) histogram of kNND (e) excluded outliers by kNND (f) signal subspace

(g) noise subspace (h) color over-segmentation (i) initial layers

(j) integrated squared residuals (k) four layers extracted (l) another res

Figure 5.13: *Mobile* sequence. (f)&(g): blue "•" and "○" are inliers, red "×" are extreme outliers, black "+" are structure outliers.

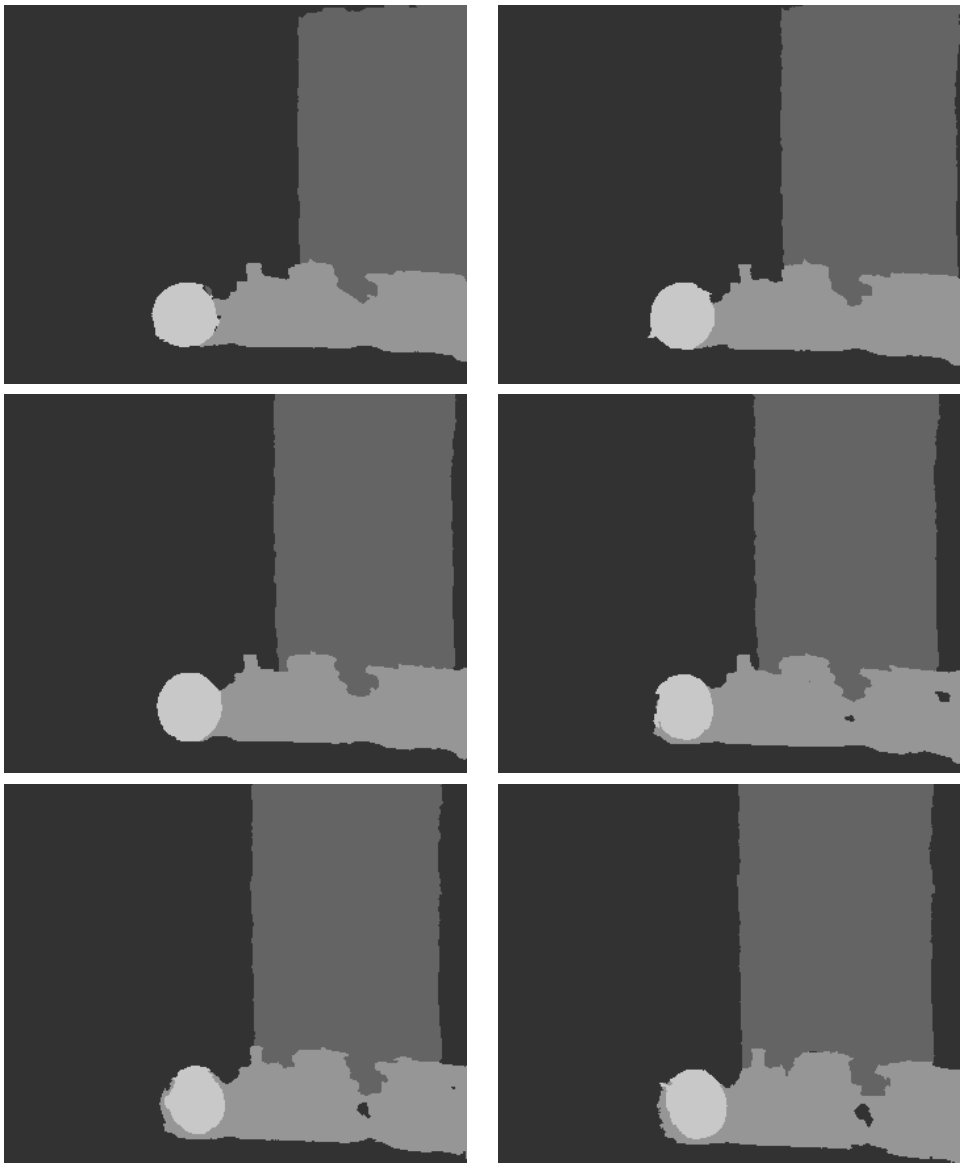
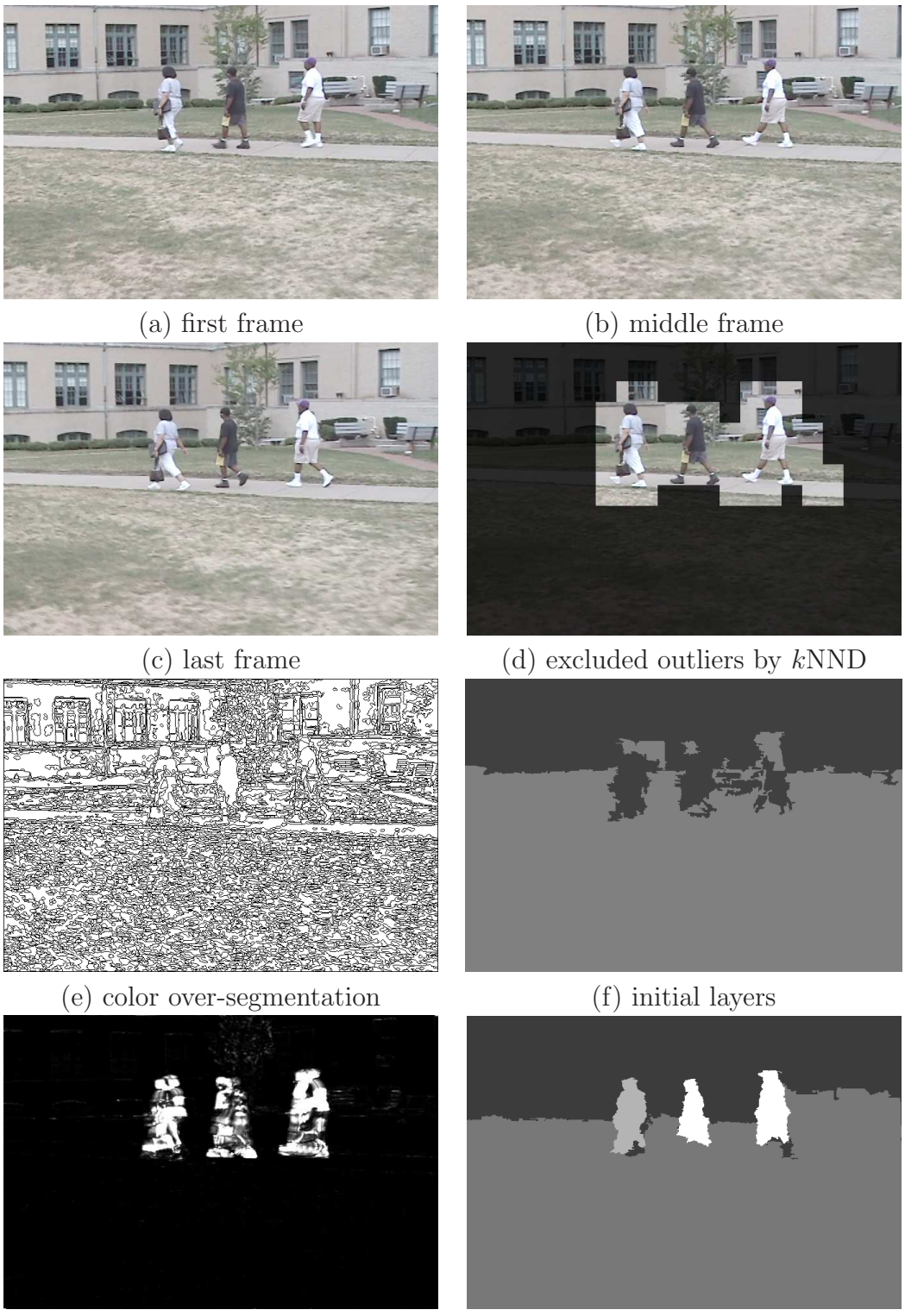


Figure 5.14: Final layer segmentation of mobile and calendar sequence.



(a) first frame (b) middle frame

(c) last frame (d) excluded outliers by kNND

(e) color over-segmentation (f) initial layers

(g) integrated squared residuals (h) five layers extracted

Figure 5.15: Walking-people sequence. Five layers are extracted, including three walking people, ground, and building walls.

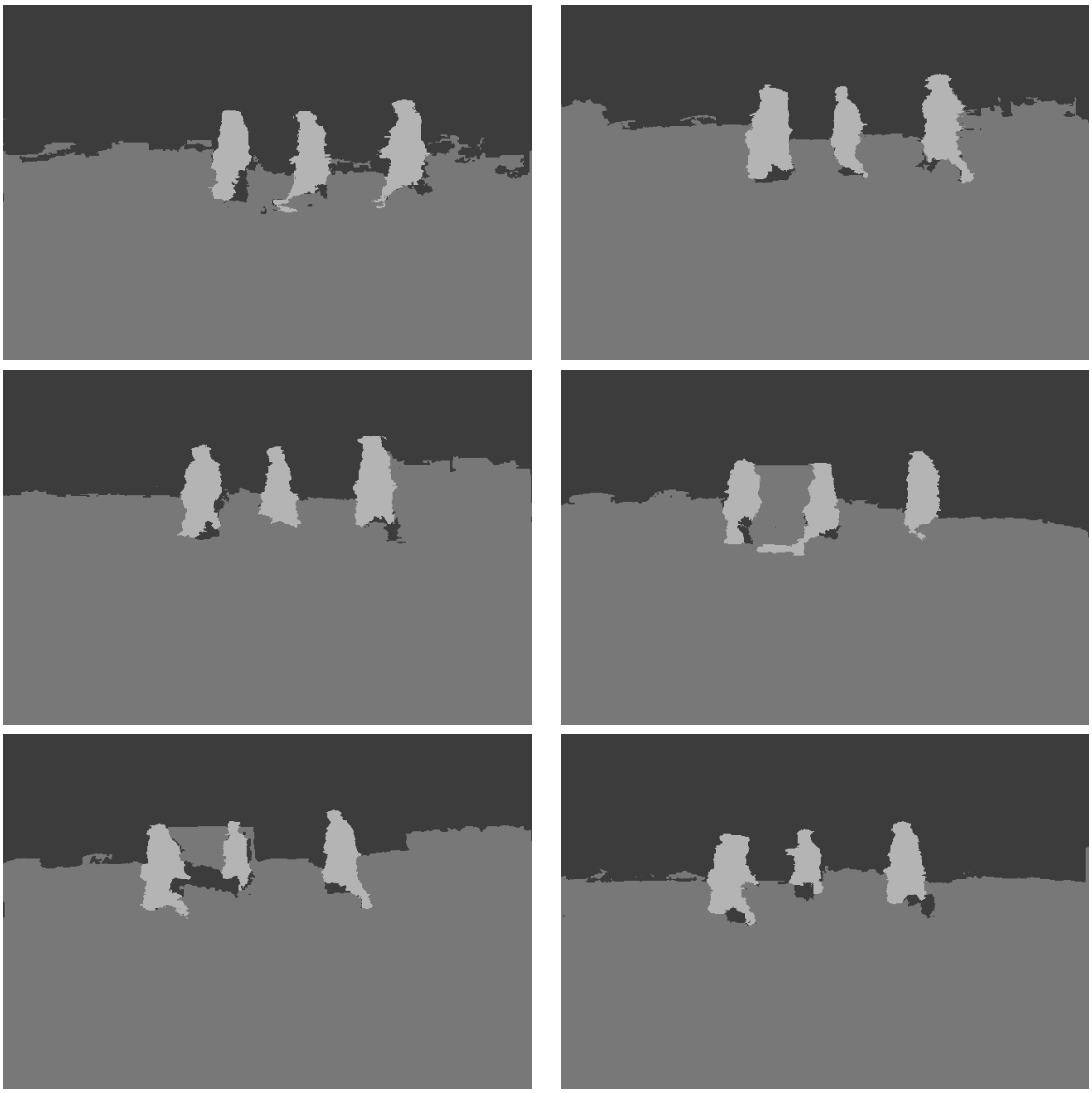
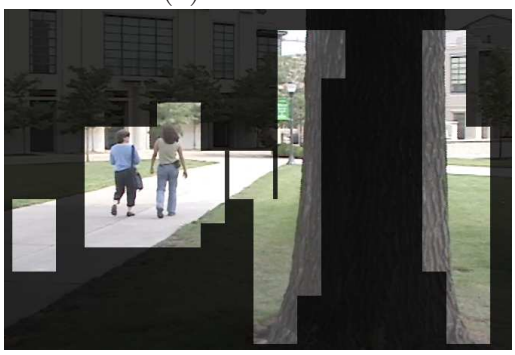


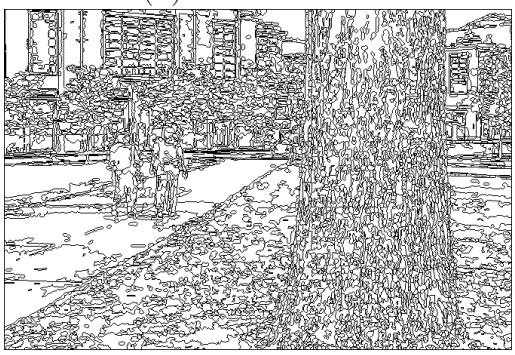
Figure 5.16: Final layer segmentation of walking people sequence (I).

(a) first frame

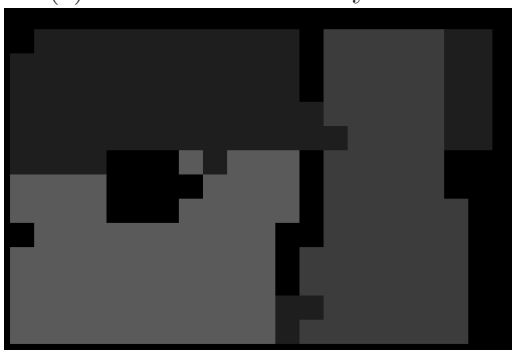
(b) middle frame



(c) excluded outliers by kNND

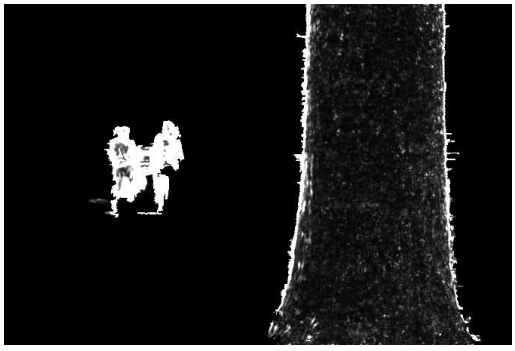


(d) color over-segmentation



(e) initial layers by clustering local blocks in the subspace

(f) initial layers after layer competition



(g) integrated residual of the tree layer

(h) final four layers

Figure 5.17: Walking-people sequence II. Four layers are extracted, including two walking people, ground, and buildings.

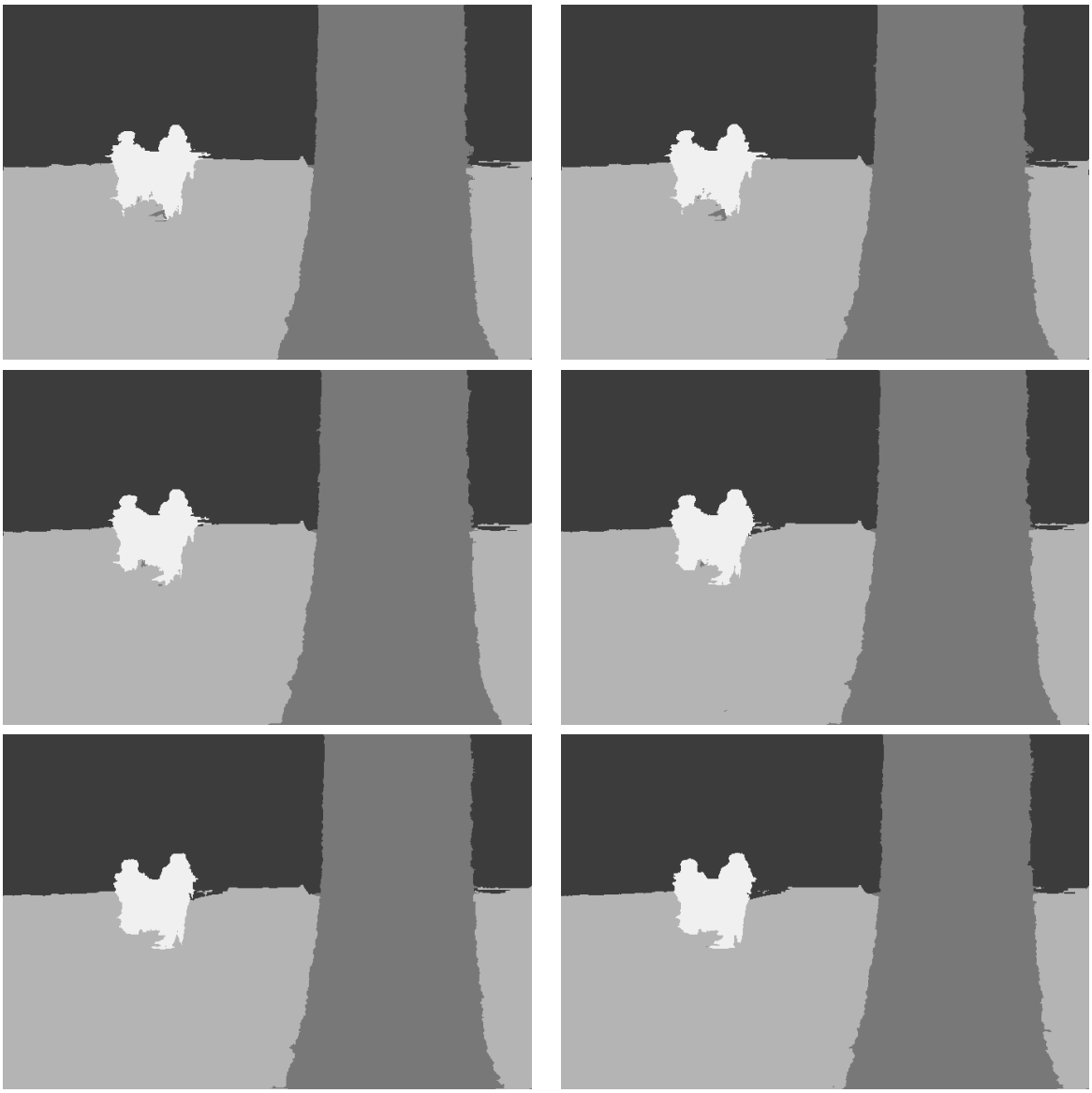


Figure 5.18: Final layer segmentation of walking people sequence (II).

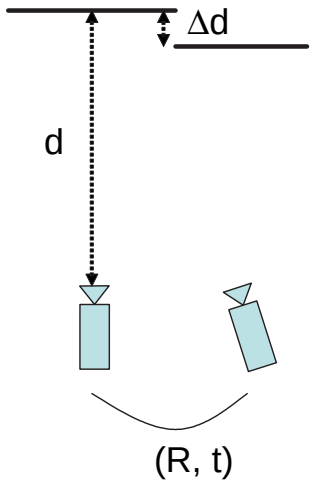
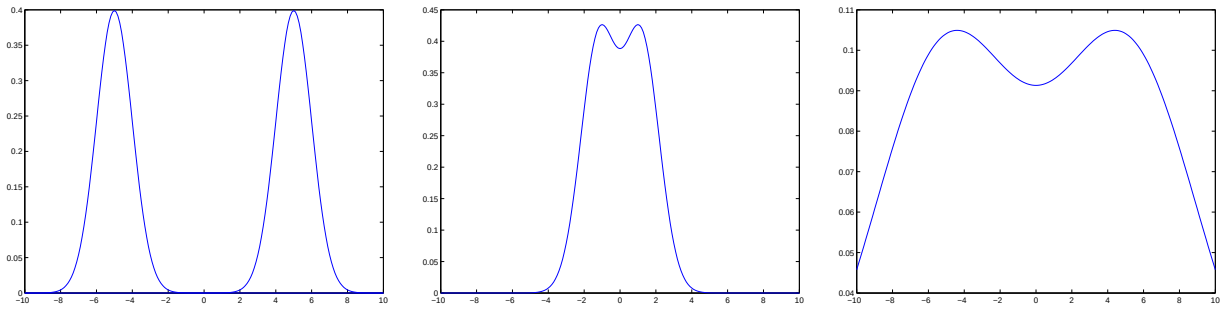


Figure 5.19: A simple scene contains two planes.



(a) (b) (c)

Figure 5.20: Analysis of layer discriminability. (a) the original data distribution in the parameter space; (b) effect of increasing distance d; (c) effect of increasing noise σ.

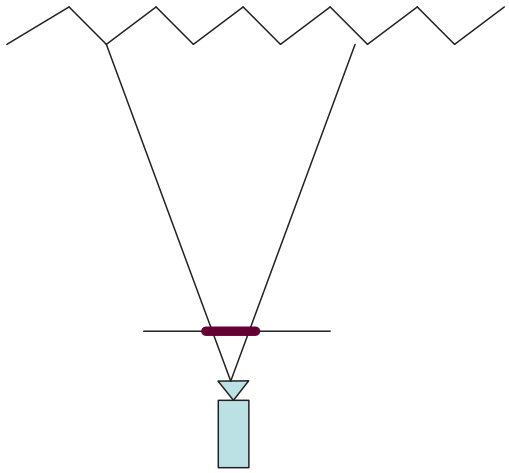


Figure 5.21: The effect of small 3D planes. One local region (shown in bold line segment in the image plane) contains many small planar patches in the scene. The estimated 2D motion of such local region is the result of averaging multiple planar patches. As a result, even though the 3D parallax among these planar patches is large, they are not measurable in the image domain and are likely grouped together as one layer.

Chapter 6

Applications

Layer representation has been successfully used in many areas in computer vision, including motion analysis [102, 51, 43], 3D scene analysis [9], video compression [62], etc. In this chapter we exploit the potential application of our layer extraction approach to object detection and recognition in image understanding, video compression, and super-resolution.

6.1 Breaking the chicken-egg cycle in image understanding

Traditional analysis approach to image understanding faces a chicken-egg cycle. Image analysis is usually formulated as a bottom-up process, where low level features (edge, texture, motion, etc) are extracted and aggregated to higher level understanding [66]. However, it is in general difficult to reliably extract and estimate low level features for such bottom-up analysis. On the other hand, if some initial information or understanding about the image is available, then the whole bottom-up process becomes much easier. For example, the boundaries between meaningful objects, if available, can greatly simplify the video analysis tasks including shape, motion, or texture analysis. The subsequent higher level analysis for understanding therefore becomes easier. Unfortunately, the boundaries (segmentation) are usually not available at the very beginning.

In the following, we will present how layer segmentation can help break such

analysis-understanding cycle, using the following example sequences: the walking person sequence, traffic sign sequence, desktop sequence, and Humanoid robot sequence.

6.1.1 Walking person sequence

As an example, given one single image shown in Fig. 6.1(a), humans can easily figure out that the scene consists of a "horizontal ground" plane in the front, one or two "vertical building-wall" planes in the back, and three "people". Once such understanding is obtained, further analysis on the shape and texture will give more information, such as that the three persons are walking instead of standing still, etc. For machines, the prior knowledge at the very beginning is generally not available, and therefore the above "easy" tasks for humans are currently tough image analysis tasks for machines.

Given two or more frames of the same scene in Fig. 6.1(a), humans do not seem to gain much more information because the content is already understood. However, the additional frames provide critical temporal information that machines can utilize to acquire some initial but useful information about the scene. One such information is the layer segmentation. Fig. 6.1(b) is the layer segmentation result using our subspace approach. Such initial segmentation breaks the vicious cycle of understanding and analysis. It facilitates the subsequent video analysis task for understanding. For example, by analyzing the shape and texture of the three foreground blobs, the machine can eventually understand that they are humans.

6.1.2 Traffic sign sequence

Detecting traffic signs is important for vehicle navigation. Given five frames of a scene shown in Fig 6.2(a), our algorithm extracts four layers as shown in Fig 6.2(b). Further analysis on the foreground layer, such as video text detection, will lead to the understanding that it is a stop-sign. Other layers in this example are two buildings and the ground, which are also useful for vehicle navigation.

(a) (b)

Figure 6.1: Layer segmentation of a sample frame in a short video sequence where there persons are walking on a road in front of a building. (a): One frame in the walking-people sequence. (b) Layer segmentation.

Figure 6.2: Layer extraction result on the traffic sign sequence. (a) The middle frame of the five-frame sequence; (b) Layer extraction result.

6.1.3 Desktop sequence

In some applications we want to detect and recognize the characters in video scene. Characters in the scene can be distorted in different ways in the image, depending on the camera view point. Such distortion makes character detection and recognition even more difficult. Being able to bring the characters in the image to the frontal view will greatly simplify the subsequent task of detecting and recognizing the characters in the scene. Fig. 6.3 shows an example. The input video contains five frames of a desktop scene. The layer extraction algorithm extracts two layers, one for the desktop and one for the wall. We assume a rough calibration matrix (intrinsic) of $diag(f, f, 1)$ with f the focus length. Given the layers and their homographies, we can rectify the layer images to the frontal view. Since characters are usually written on the planes, they are also brought to the frontal view, where they are much easier to be detected and recognized.

6.1.4 Humanoid image sequence: ground detection

Detecting obstacles on the ground is useful for robot navigation. Fig. 6.4(a) shows one of the images taken by a Honda humanoid robot. Given five frames, the layer extraction result is shown in Fig. 6.4(b). Again, further analysis on the layers will lead to the detection of the ground, and the two obstacles (boxes) on the ground. Once the ground plane is available, we can estimate robot's ego-motion with respect to the ground, which is useful in robot control.

6.2 Video compression

Layer representation extends the mosaic representation to compress videos that contain multiple moving objects or parallax information due to the translational movement of the cameras. Once the layer segmentation, we can construct a mosaic for the corresponding layer across frames in the given video sequence. Such layer mosaic removes the temporal redundant information, just as the ordinary mosaic does, and can therefore significantly compress the video.

We show the preliminary results of applying layer representation to compress two short video sequences. Each input video segment is compactly represented by layer

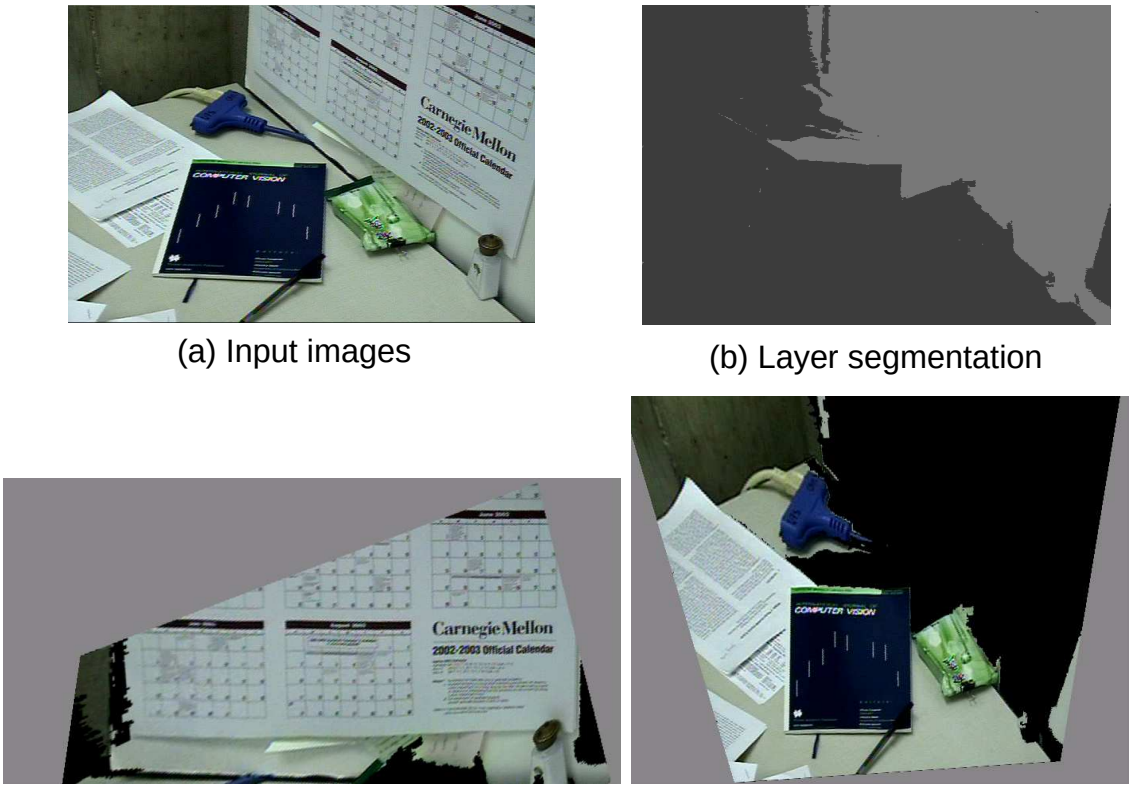


Figure 6.3: Bringing the layers to the frontal view to facilitate the task of detecting and recognizing the characters in the scene.

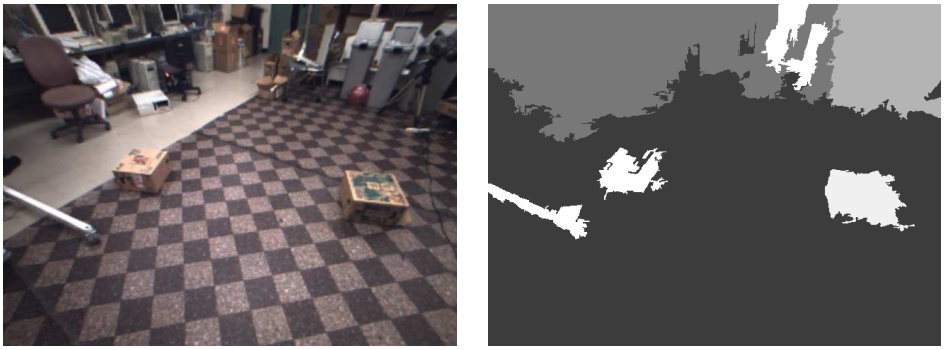


Figure 6.4: Layer extraction results on the humanoid image sequence. (a) The middle frame of the five-frame humanoid image sequence; (b) Layer extraction result. Note the objects on the ground.

mosaics [1]. Fig.(6.5a & b) shows the four layer mosaics of the flower garden sequence (30 frames). We are able to compress the original video sequence from about $7MB$ to $40KB$. Fig.(6.5c) shows the recovered frame based on the layer representation, whose original frame is shown in Fig.(6.5d). The remaining images in Fig.(6.5) show the results for *mobile & calendar* sequence. We are able to compress it from about $9MB$ (30 frames) to about $45KB$. Note that higher compression ratio can be achieved with longer sequence.

6.3 Super-resolution

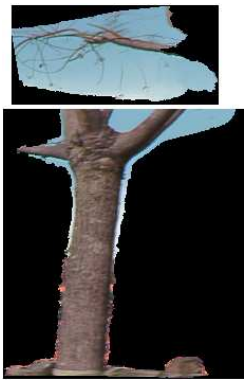
Video super-resolution, such as the reconstruction-based algorithm [46], requires accurate optical flow estimation that is in general hard to compute. But once we have the layer segmentation, it is much easier to compute the accurate optical flows. The reason is that the smoothness constraint can be safely applied inside each layer, but not across the boundaries among layers.

To show the potential of layer representation in super-resolution application, we simply fuse together the "stop-sign" layer extracted from five frames in the traffic-sign sequence shown in Fig.6.2. The fusion contains the following two simple steps:

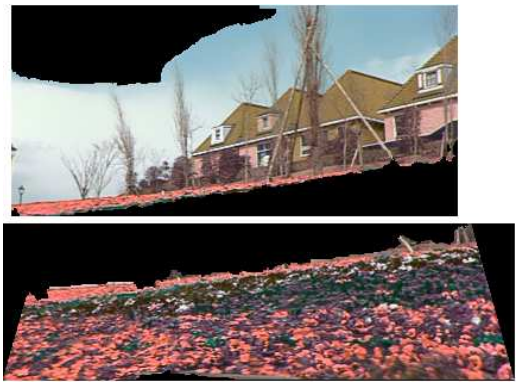
- Warp the "stop-sign" layer from each frame to the reference frame.
- Blend the overlapped pixels using median operator.

Fig. 6.6 shows the results. Such a simple fusion algorithm itself has already enhanced the resolution. It is doing better than the bi-cubic interpolation. This result indicates that the layer motion model has provided a good optical flow estimation. We anticipate better result if the reconstruction-based algorithm, instead of the above simple fusion process, is applied.

[1] Sprite VOP in MPEG-4



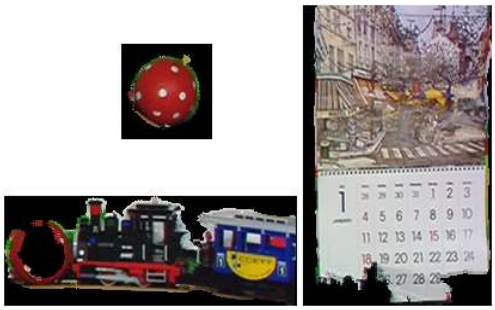
(a) Layer mosaic 1 & 2



(b)Layer mosaic 3 & 4

(c) rended frame

(d) original frame



(e) Layer mosaic 1, 2, and 3

(f)Layer mosaic 4

(g) rended frame

(h) original frame

Figure 6.5: Layer mosaics and synthesized frame. (a) & (b) layer mosaics of the flower garden sequence; (c): a recovered frame based on the layer representation; (d): original frame corresponding to (c); (e)–(h) same results on mobile & calendar sequence.

Figure 6.6: Supper-resolution result on the "stop-sign" layer (image shown by zooming-in a factor of four). (a) The original image; (b) Fusion result; (c) Bi-cubic interpolation.

Chapter 7

Explicit Subspace for Ego-motion Estimation and Ground Layer Segmentation

In previous chapters, the subspace is estimated from local measurements. In some special platform, the subspace becomes explicit due to the setup of the camera and its special motion. For example, in visual car navigation, the planar motion constraint provides an explicit subspace. We use such explicit subspace constraint for robust ego-motion estimation and ground layer segmentation.

This chapter presents a robust method to solve the two coupled problems: ground layer detection and vehicle ego-motion estimation, which appear in visual navigation. We virtually rotate the camera to the downward-looking pose in order to exploit the fact that the vehicle motion is roughly constrained to be planar motion on the ground. This camera geometry transformation, together with planar motion constraint, will: 1) eliminate the ambiguity between rotational and translational ego-motion parameters, and 2) improve the Hessian matrix condition in the direct motion estimation process. The virtual downward-looking camera enables us to estimate the planar ego-motions even for small image patches. Such local measurements are then combined together, by a robust weighting scheme based on both ground plane geometry and motion compensated intensity residuals, for a global ego-motion estimation and ground plane detection. We demonstrate the effectiveness of our method by experiments on both synthetic and real data.

7.1 Introduction

Ego-motion estimation and ground plane detection have many applications, such as visual navigation, computer vision based driving assistance, and 3D environment map reconstruction. In this paper we address the case of a single camera rigidly mounted on a car moving in traffic scenes that includes cluttered background including other static or moving objects on the ground plane. It is difficult to apply traditional Structure from Motion algorithms here since they usually require estimating the depth for such cluttered background. To overcome such difficulty, planes in the scene have been used for ego-motion estimation [70, 48].

Ground plane is of special interest. Methods to obtain ground plane include 2D dominant motion estimation [47] and layer extraction [51, 103, 8, 105, 104, 81, 96, 55, 59, 56]. These approaches can be classified into two categories: top-down approaches and bottom-up approaches. Top-down approaches either assume that the ground plane is a dominant plane, or assume that the scene can be approximated with a few planar layers who simultaneously compete for layer support. In our traffic scenarios, the ground plane is not necessary a dominant plane, and the cluttered background is hard to be modelled with a small number of planar layers. It is therefore hard to apply the top-down approaches here. In the bottom-up approaches, images are first divided into small patches, and local measurement (such as 2D image transformation) for each patch is then computed. These local measurements are then grouped into layers. Due to the typical forward motion in vehicle moving, it is necessary to use projective homography for local 2D measurements. Given *small* local support area and low texture on the road (ground), the recovery of projective homography is not reliable due to large number of unknown parameters, small field of view, and ambiguities among its parameters.

In this chapter, we assume the camera is calibrated such that its focus length and its relative pose with respect to the vehicle is known. In such a particular setup, the ground plane normal is explicitly constrained too [1]. We can therefore use ego-motion, instead of projective homography, as the local measurement. Given an image patch that is *assumed* to be on the ground, the estimated ego-motion is the local measurement of such image patch. Using ego-motion as the local measurement is an improvement over using projective homography, since it exploits the ground plane

[1] We do not need to know the distance from the camera to the ground due to the scale ambiguity between the camera translation and scene depth.

geometry. However, estimating ego-motion based on small image patch still suffers from ambiguities among its parameters due to small field of view [3, 27].

To overcome the above difficulty, we exploit the fact that the vehicle motion can be approximated by planar motion on the ground. Such planar motion is of great practice importance and has been used in structure from motion and camera calibration [67, 6], and vehicle ego-motion estimation [85]. In this chapter, we use a virtual downward-looking camera to exploit the planar motion constraint. Thinking of a virtual downward-looking camera on planar motion has the following advantages: 1) It eliminates the ambiguity between rotational and translational ego-motion parameters; 2) It improves the Hessian matrix condition in the direct motion estimation process; 3) It induces image motions that are linear in terms of image coordinates, and therefore can be reliably estimated.

The virtual camera is used to collect the local measurements, i.e., to estimate the planar ego-motions based on small image patches. Such local measurements are then combined together, by a robust weighting scheme for the global ego-motion estimation and ground plane detection. Regularization is then applied for the recovery of the remaining small non-planar motions.

7.2 Ego-motion estimation

In this section, we describe the direct method to estimate the vehicle ego-motion with respect to a small image patch that is assumed to be on the ground.

7.2.1 Ego-motion model

Given a sequence of images $I_0, I_1, ..., I_N$ under a perspective camera with internal matrix of $diag(f, f, 1)$, we want to compute the camera ego-motion between the reference image I_0 and another image $I_i, i = 1, 2, ..., N$. The *incremental* image motion at an image point $\mathbf{p} = (x, y)^\top$ in I_i is given by (see [42]):

$$\mathbf{v}_i(\mathbf{p}) = \mathbf{B}_p \mathbf{\Omega}_i + \frac{1}{Z(\mathbf{p})} \mathbf{A}_p \mathbf{T}_i \tag{7.1}$$

where $\mathbf{\Omega}_i = (\omega_X, \omega_Y, \omega_Z)_i^\top$ and $\mathbf{T}_i = (T_X, T_Y, T_Z)_i^\top$ are the camera rotational and translational velocity, $Z(\mathbf{p})$ is the 3D scene depth at Point $\mathbf{p}$,

$$\mathbf{B}_p = \begin{bmatrix} -\frac{xy}{f} & (f + \frac{x^2}{f}) & -y \\ -(f + \frac{y^2}{f}) & \frac{xy}{f} & x \end{bmatrix} \tag{7.2}$$

$$\mathbf{A}_p = \begin{bmatrix} f & 0 & -x \\ 0 & f & -y \end{bmatrix} \tag{7.3}$$

If we are given a 3D plane $\mathbf{n}^\top \mathbf{P} + d = 0$ with $\mathbf{n} = (n_1, n_2, n_3)^\top$ the plane normal and $\mathbf{P} = (X, Y, Z)^\top$ the 3D coordinate of points on the plane, then we can rewrite Eq.(7.1) as:

$$\mathbf{v}_i(\mathbf{p}) = \mathbf{B}_p \mathbf{\Omega}_i + \frac{\mathbf{n}^\top \mathbf{F}}{d} \mathbf{A}_p \mathbf{T}_i \tag{7.4}$$

where $\mathbf{F} = (-\frac{x}{f}, -\frac{y}{f}, -1)^\top$.

Eq.(7.4) shows that there is a scale ambiguity between d and the camera translation $\mathbf{T}_i$, which means that we can only recover the direction of the camera translation. Without loss of generality, we set $d = -1$ in our experiments.

7.2.2 Direct estimation of ego-motion

As has been pointed out in [85], in typical traffic scenarios, direct method [42, 35, 10, 64] is more preferable than optical-flow based approach [2, 40, 91, 83] for ego-motion estimation. The reason is that the road usually has weak texture or linear image structure, while the cluttered background including moving objects often contains many feature points.

Given calibrated camera and ground plane normal, we use direct method to estimate the incremental ego-motion based on the brightness constancy assumption, by minimizing the sum square difference (SSD) with respect to the incremental camera motion parameters $\Theta = (\mathbf{\Omega}_i, \mathbf{T}_i)$:

$$\begin{aligned} E(\Theta) &= \sum_p \left[I_i(\mathbf{p} + \mathbf{v}_i(\mathbf{p}, \Theta)) - I_0(\mathbf{p}) \right]^2 \\ &\approx \sum_p \left[\mathbf{g}_p^\top \mathbf{J}_p^\top \Theta + e_p \right]^2 \end{aligned} \tag{7.5}$$

where $e_p = I_i(p) - I_0(p)$ is the temporal difference at Pixel p, $\mathbf{g}_p^\top = \nabla I_i(\mathbf{p})$ is the image gradient at Pixel $\mathbf{p}$ in image I_i, and $\mathbf{J}_p$ is the Jacobian at p:

$$\mathbf{J}_p = \frac{\partial \mathbf{v}_i(\mathbf{p})}{\partial \Theta} = \begin{bmatrix} \mathbf{B}_p^\top \\ \frac{1}{Z(p)}\mathbf{A}_p^\top \end{bmatrix} = \begin{bmatrix} \mathbf{B}_p^\top \\ \frac{\mathbf{n}^\top \mathbf{F}}{d}\mathbf{A}_p^\top \end{bmatrix} \tag{7.6}$$

From Eq.(7.5), we can see that every pixel inside the image patch with non-zero intensity derivative makes a contribution to the final solution of Θ. To achieve robustness to outliers, the contribution of each pixel should be weighted according to some robust criteria. For example, robust estimator uses the residual e_p to determine the weight $w_p = w(e_p) = \frac{\dot{\rho}(e_p)}{e_p}$, where $\rho(\cdot)$ is some robust M-estimator.

The weighted least square solution of Eq.(7.5) is given by:

$$\Theta = \mathbf{L}^{-1}\mathbf{b} \tag{7.7}$$

where

$$\mathbf{L} = \sum_p w_p \mathbf{J}_p \mathbf{g}_p \mathbf{g}_p^\top \mathbf{J}_p^\top \tag{7.8}$$

is the *Hessian*,

$$\mathbf{b} = \sum_p (-w_p e_p \mathbf{J}_p \mathbf{g}_p) \tag{7.9}$$

is the *accumulated residual*.

Once we recover the incremental camera motion parameters $\mathbf{\Theta} = (\mathbf{\Omega}_i, \mathbf{T}_i)$, we perform an incremental update to the ego-motion $\mathbf{M}_i$:

$$\mathbf{M}_i \leftarrow \mathbf{M}_i \begin{bmatrix} \mathbf{R}(\mathbf{\Omega_i}) & \mathbf{T}_i \\ 0 & 1 \end{bmatrix} \tag{7.10}$$

where $\mathbf{R}(\mathbf{\Omega_i})$ is the incremental rotation matrix given by the Rodriguez's formula:

$$\mathbf{R}(\mathbf{\Omega}) = \mathbf{I} + [\tilde{\mathbf{n}}]_\times \sin\theta + [\tilde{\mathbf{n}}]_\times^2 (1 - \cos\theta) \tag{7.11}$$

where $\theta = \|\Omega\|$, and

$$[\tilde{\mathbf{n}}]_\times = \frac{1}{\theta}\begin{bmatrix} 0 & -w_z & w_y \\ w_z & 0 & -w_x \\ -w_y & w_x & 0 \end{bmatrix}. \tag{7.12}$$

The overall direct ego-motion computation is an iterative Gauss-Newton gradient decent process. Each iteration consists of the following three steps:

1. Compute the incremental motion parameters (Eq.(7.7)).
2. Perform the incremental update to the ego-motion $\mathbf{M}_i$ (Eq.(7.10)).
3. Warp the image I_i towards the reference image I_0, using the homography induced by the ground plane $(\mathbf{n}, d)$ under current ego-motion: $\mathbf{H} = \mathbf{K}(\mathbf{R}(\mathbf{\Omega}) - \frac{\mathbf{T}}{d}\mathbf{n}^\top)\mathbf{K}^{-1}$, where $\mathbf{K} = \text{diag}(f, f, 1)$ is the camera internal matrix.

7.3 Camera models for planar ego-motion estimation

There are several difficulties in estimating the full vehicle ego-motion based on a *small* image patch:

- During a short period of time, the vehicle undergoes approximately planar motion. For a camera rigidly mounted on such vehicle [2], its ego-motion consists of a rotation around an axis vertical to the ground plane, and two translations parallel to the ground plane. Therefore, full ego-motion model contains more parameters than necessary. Estimating such diminishing parameters are inherently ill-conditioned.
- There are inherent ambiguities between rotation and translation. Given a small image patch, therefore small field of view (FOV), it is hard to differentiate the w_X-induced flow from the T_Y-induced flow, and the w_Y induced flow from the T_X-induced flow, respectively [3, 27]. These inherent ambiguities introduce elongated valley in the SSD error function [4], resulting in slow convergence and bad local minima.

It is therefore necessary to exploit the planar motion constraint. To do so, we divide the six ego-motion parameters into two triples. The first triple consists of the planar motion parameters, and the second triple consists of the diminishing non-planar motion parameters that can be ignored at the stage of local measurement.

[2]The camera can have any orientation, but is otherwise fixed w.r.t. the vehicle body.

In the following, we introduce two virtual cameras and analyze how the selection of camera models affects the effectiveness in exploiting planar motion constraint.

Virtual cameras can be achieved by rectifying the images using the homography induced by the ground plane and the relative pose between the original camera and the virtual camera. Doing so requires camera calibration for the camera rotational pose with respect to the vehicle. We assume the camera is fixed with respect to the vehicle, which means that the calibration can be done before hand (see the Appendix for a simple calibration method). It is important to keep the virtual cameras always on the same plane so that the camera motions among the rectified images are still planar motions.

7.3.1 Virtual forward-looking camera

In a typical setting, the camera is mounted on the vehicle looking at the ground at some angle, as shown in Fig.(7.8). A simple way to make use of the planar motion constraint is to virtually rotate the camera such that its optical axis (Z axis) points forward horizontally and its XZ plane parallel to the ground plane, as has been done in [85]. We will call it the *forward-looking* camera.

The planar ego-motion parameters are then reduced to $\Theta_f = (w_Y, T_X, T_Z)$. There still exists ambiguity between w_Y and T_X. In [85], the dominant camera motion set is chosen to be (w_X, w_Y, T_Z). But in real experiments we have observed non-negligible T_X, especially when the vehicle is changing lanes or turning. Moreover, the camera motion is not longer planar due to w_X.

In the coordinate frame of the forward-looking camera, the normal of the ground plane is $(0, 1, 0)$, and the Jacobian w.r.t. Θ_f is:

$$\mathbf{J}_p = \frac{\partial \mathbf{v}_i(\mathbf{p})}{\partial \Theta_f} = \begin{bmatrix} f + \frac{x^2}{f} & y & -\frac{xy}{f} \\ \frac{xy}{f} & 0 & -\frac{y^2}{f} \end{bmatrix}^\top \tag{7.13}$$

In addition to the ambiguity between w_Y and T_X, the above Jacobian also indicates the following problems:

- It is usually hard to estimate w_Y and T_Z within a small FOV since they introduce image motions that are second order polynomial terms of the image coordinate (x, y).

- The Hessian matrix is determined by both the image texture and the Jacobian. When the texture is low,the second order terms in the Jacobian will contribute to a badly-conditioned Hessian matrix.

Coordinate normalization and translation are useful technique to improve the matrix condition number [38]. In our case, coordinate normalization does not change the condition of the Hessian matrix, since every element in the Jacobian is multiplied by a same constant [3]. Translating the coordinates to center around $(0, 0)$ will improve the matrix condition. Doing so effectively translates the camera such that its optical axis passes through the center of the input image patch. In the forward-looking camera, it is impossible to do so given an image patch on the ground that is parallel to the camera optical axis.

7.3.2 Virtual downward-looking camera

The above analysis on forward-looking camera geometry motivates us to rotate and translate (parallel to the ground plane) the camera geometry such that we think of a virtual camera whose optical axis is vertical to the ground plane and passing through the center of input image patch. We call it the *downward-looking* camera.

In the coordinate frame of downward-looking camera, the normal of the ground plane is $(0, 0, 1)$. The dominant motion becomes $\Theta_d = (w_Z, T_X, T_Y)$, and the Jacobian w.r.t. Θ_d is:

$$\mathbf{J}_p = \frac{\partial \mathbf{v}_i(\mathbf{p})}{\partial \Theta_d} = \begin{bmatrix} -y & f & 0 \\ x & 0 & f \end{bmatrix}^\top \tag{7.14}$$

The advantages of using the downward-looking camera are:

- The above Jacobian consists of only zero and first order polynomial terms, which, together with the virtual camera translation so that its image coordinates are center around $(0, 0)$, will result in a well-conditioned Hessian matrix even when the road has low texture.
- There is not inherent ambiguities among the parameters in Θ_d. It is easy to differentiate the flow induce by w_Z from the flow induced by (T_X, T_Z). Indeed,

[3]Notice that f also needs to be scaled according to the normalization to preserve the correctness of Eq.(7.1).

Θ_d can be reliably estimated since they induce image motions that are linear in terms of image coordinate (x, y) (no perspective distortion).

Notice that equally treating the pixels in the rectified image is equivalent to give larger weights to pixels (in the original un-rectified image) that correspond to points further-away on the ground plane, due to the perspective distortion (front-shorten) in the un-rectified image. We can adjust such scene-dependent weighting by non-uniform image sampling. Also notice that translating the camera to look at the patch center effectively enlarges the camera field of view (FOV). We avoid the degenerate case of infinite rectified image area by using only the image pixels below the horizon line (vanishing line of the ground plane), since pixels above the horizon line in the image are obvious non-ground pixels. Given the camera pose relative to the vehicle, it is straightforward to calculate the horizon line (see Appendix for details).

7.4 Ground plane detection and global ego-motion estimation by virtual downward-looking camera

This section describes a robust technique to combine locally estimated ego-motions for ground plane detection and global ego-motion estimation. The general framework of the algorithm is:

1. Bootstrap from local estimations: Divide the image into small $n \times n$ patches [4]. For each patch, estimate an ego-motion (Section 7.3.2) and compute its robust weights based on both geometry and intensity residuals (Section 7.4.1).

2. Combine the local estimations according to their weights for global ego-motion estimation, including the non-planar motions.

3. Recompute the robust weight based on current ego-motion.

Step 1 is an important bootstrap step to provide a good initialization for further global estimation. Step 2 and 3 are the two iterative steps. In our experiments, we have found one or two iterations are enough, due to the accurate local ego-motion

[4] We use overlap image patches.

estimation by the downward-looking camera. The ground plane is detected based on the final weights.

7.4.1 Geometry based robust weighting

Traditional robust weighting uses motion compensated pixel intensity residuals e_p, i.e., $w_p = w(e_p)$ in Eq.(7.7). The residual e_p depends on both geometry and texture. Pixels not on the ground plane but with low texture will also have low residuals when compensated by the motion corresponding to the ground plane, and will be given large weights if weighting is purely based on intensity residuals. When the ground layer has low texture, the inclusion of those false pixels will affect the final ego-motion estimation. We should exclude such false pixels by exploiting the ground plane geometry in the robust weighting.

For each patch in the image, we initialize its plane normal as the ground plane normal, then refine its plane normal (Section 7.4.1) under the currently estimated ego-motion. If the patch is in fact on the ground, the refined plane normal will be close to the ground normal due to accurate initialization. Otherwise, we will end up with a plane normal that is distinct from the plane normal of the ground [5].

Our final weighting scheme use both the intensity residuals, and the angle between the re-estimated plane normal $\mathbf{n}$ and the ground normal $\mathbf{n}_g$:

$$w_p = w(e_p, \theta) \tag{7.15}$$

where e_p is the intensity residual, $\theta = arccos\left(\frac{\mathbf{n}^\top \mathbf{n}_g}{\|\mathbf{n}\| \cdot \|\mathbf{n}_g\|}\right)$, and $w(\cdot, \sigma)$ is the robust weighting with scale σ that is set to the robust standard deviation (see [8]) by $\sigma = 1.4826 \cdot median_p|e_p|$.

Compute plane normal

This section describes the direct method to estimating the plane normal based on current ego-motion. We re-use Eq.(7.7) and the corresponding algorithm in Section 7.2.2, except that the unknowns are the plane normal $\mathbf{n} = (n_1, n_2, n_3)$ instead of ego-motion Θ. We therefore need to derive the new Jacobian $\mathbf{J}_p = \frac{\partial \mathbf{v}_i(\mathbf{p})}{\partial \mathbf{n}}$. Given the ego-motion

[5] We do not care if such non-ground plane normal is actually correct, as long as it is distinct from the normal of the ground plane.

and the ground plane equation, we prefer using the exact homography to represent $\mathbf{v}_i(\mathbf{p})$, instead of using the instantaneous representation in Eq.(7.4). The reason is the following. In each step of incremental ego-motion estimation, the instantaneous representation is a good approximation since the incremental ego-motion is very small. But once the final ego-motion Θ is recovered, instantaneous representation is no longer a good approximation, especially when multiple frames are used.

Suppose the initial plane normal is $\mathbf{n}$, we want to compute the incremental plane normal update $\mathbf{m}$ to $\mathbf{n}$. The homography $\mathbf{H}$ induced by the updated plane $(\mathbf{n}+\mathbf{m})\mathbf{P}+d=0$ is:

$$\begin{aligned}\mathbf{H} &= \mathbf{K}(\mathbf{R}(\mathbf{\Omega})-\frac{\mathbf{T}}{d}\mathbf{n}^\top)\mathbf{K}^{-1}-\mathbf{K}\frac{\mathbf{T}}{d}\mathbf{m}^\top\mathbf{K}^{-1} \\ &= \tilde{\mathbf{R}}-\mathbf{K}\frac{\mathbf{T}}{d}\mathbf{m}^\top\mathbf{K}^{-1} \end{aligned} \tag{7.16}$$

where $\mathbf{K}=\mathrm{diag}(f,f,1)$ is the camera internal matrix.

Denote $\mathbf{r}_i^\top$ the i-th row of $\tilde{\mathbf{R}}$, and $[\tilde{x},\tilde{y}]=[\frac{\mathbf{r}_1^\top\mathbf{p}}{\mathbf{r}_3^\top\mathbf{p}},\frac{\mathbf{r}_2^\top\mathbf{p}}{\mathbf{r}_3^\top\mathbf{p}}]$. The incremental image motion at point $\mathbf{p}=(x,y,1)^\top$ is:

$$\mathbf{v}_i(\mathbf{p})=\begin{bmatrix} u \\ v \end{bmatrix}=\begin{bmatrix} \frac{\mathbf{h}_1^\top\mathbf{p}}{\mathbf{h}_3^\top\mathbf{p}}-\tilde{x} \\ \frac{\mathbf{h}_2^\top\mathbf{p}}{\mathbf{h}_3^\top\mathbf{p}}-\tilde{y} \end{bmatrix} \tag{7.17}$$

where $\mathbf{h}_i^\top$ is the i-th row vector of $\mathbf{H}$.

The Jacobian $\mathbf{J}_p$ with respect to $\mathbf{m}$ is: $\mathbf{J}_p=\frac{\partial\mathbf{v}_i(\mathbf{p})}{\partial\mathbf{m}}=$

$$\frac{\mathrm{diag}(x,y,1)}{d\mathbf{r_3}^\top\mathbf{p}}\begin{bmatrix} \frac{\tilde{x}}{f}T_Z-T_X & \frac{\tilde{x}}{f}T_Z-T_X & \tilde{x}T_Z-fT_X \\ \frac{\tilde{y}}{f}T_Z-T_Y & \frac{\tilde{y}}{f}T_Z-T_Y & \tilde{y}T_Z-fT_Y \end{bmatrix}^\top$$

At each iteration, the plane normal is updated by:

$$\mathbf{n}\leftarrow\mathbf{n}+\mathbf{m}$$

The new plane normal is then plugged into Eq.(7.16) to compute the new homography for the next iteration.

7.4.2 Recovering remaining non-planar motion parameters

After the planar ego-motions have been recovered, we can estimate other small non-planar motions, which might exhibit due to vehicle bouncing or non-planar road condition. Under the coordinate frame of the downward-looking camera, the non-planar motion set is $\Theta_2 = (w_X, w_Y, T_Z)$. The Jacobian $\mathbf{J}_p$ w.r.t. the non-planar motion parameters Θ_2 is:

$$\mathbf{J}_p = \frac{\partial \mathbf{v}_i(\mathbf{p})}{\partial \Theta_2} = \begin{bmatrix} -\frac{xy}{f} & f + \frac{x^2}{f} & -x\frac{\mathbf{n}^\top \mathbf{F}}{d} \\ -(f + \frac{y^2}{f}) & \frac{xy}{f} & -y\frac{\mathbf{n}^\top \mathbf{F}}{d} \end{bmatrix}^\top$$

Estimating Θ_2 is inherently ill-conditioned since it induces very small or diminished image motions that are second order polynomial terms of image coordinates. Nevertheless, small or diminishing motions mean that it is safe to apply strong regularization to improve the condition. The regularized cost function is: $E(\Theta_2) =$

$$\sum_{\mathbf{p}} \left[\tilde{I}_i(\mathbf{p} + \mathbf{v}_i(\mathbf{p}, \Theta_2)) - I_0(\mathbf{p}) \right]^2 + \lambda \sum_{\mathbf{p}} \mathbf{v}_i(\mathbf{p}, \Theta_2)^2$$

where $\tilde{I}_i$ is the image I_i warped by the homograpy induced by the ground plane under current ego-motion Θ_1. The second summation term is the regularization term, which states that the image motion induced by parameter set Θ_2 must be small. $\lambda \geq 0$ is a constant parameter. A larger λ enforces stronger regularization.

By setting $\frac{\partial E(\Theta_2)}{\partial \Theta_2} = 0$, the weighted least square solution is:

$$\Theta_2 = \left[\sum_{\mathbf{p}} w_p \mathbf{J}_p (\mathbf{g}_p \mathbf{g}_p^\top + \lambda \mathbf{I}) \mathbf{J}_p^\top \right]^{-1} \sum_{\mathbf{p}} (-w_p e_p \mathbf{J}_p \mathbf{g}_p) \tag{7.18}$$

Enforcing the regularization is equivalent to "virtually improve" the texture, as shown by the diagonal matrix $\lambda \mathbf{I}$ in Eq.(7.18), and will therefore improve the condition of Hessian matrix.

	synthetic case (Fig.(7.1))		real case (Fig.(7.3))	
	condition num.	ego-motion	condition num.	ego-motion
8-parameter	1.6312e+006	N/A	1.7618e+006	N/A
full ego-motion	3.0674e+003	$\begin{bmatrix} 0.2162^\circ & -0.1360^\circ & -1.5926^\circ \\ -0.0301 & 0.0226 & -0.1264 \end{bmatrix}$	8.5083e+004	$\begin{bmatrix} -0.7710^\circ & -0.1182^\circ & 0.2130^\circ \\ -0.2001 & -0.2367 & 0.0636 \end{bmatrix}$
forward-looking	2.1349e+002	$[-0.4725^\circ, 0.0058, 0.0903]$	4.5254e+003	$[0.0108^\circ, -0.0064, 0.0595]$
downward-looking	8.4357e+000	$[-0.9991^\circ, -0.0181, 0.1066]$	5.5469e+001	$[-0.0840^\circ, -0.1222, 0.2497]$

Table 7.1: Ego-motion estimation and condition of Hessian (larger condition number means worse condition). For synthetic case, the ground truth of ego-motion is: $(w_Y, T_X, T_Z) = (-1.0°, -0.0175, 0.1)$, in the coordinate frame of forward-looking camera. Translations are measured by the unit of image height. The motion parameters of the 8-parameter model do not directly indicate the ego-motion parameters, and are not shown here.

7.5 Experimental results

7.5.1 Local planar ego-motion estimation

This section presents the experimental results on estimating the planar ego-motion based on small image patches, which is an important bootstrap step for further global ego-motion estimation and ground plane detection. To deal with large motion, in the experiments we use a multi-resolution Gaussian pyramid of the input images. In all experiments, we only use the image pixels below the horizon line, since pixels above the horizon line in the image are obvious non-ground pixels. We also compare the results of our downward-looking model against the classical 8-parameter model and full ego-motion model [10], and the recently proposed forward-looking model [85].

Synthetic case

To compare different motion models, we use a synthetic image sequence with ground truth. Fig.(7.1) shows the two synthesized images, where the camera simulates a moving vehicle on the ground plane by simultaneously moving forward and turning left (around an axis at some distance to the vehicle and vertical to the ground). The normal of the ground plane and the camera focus length are known.

The synthetic case in Table (7.1) quantitatively compares the condition number of the Hessian matrix and the recovered ego-motion parameters using four different motion models. The image patch used to compute the ego-motion is indicated by the

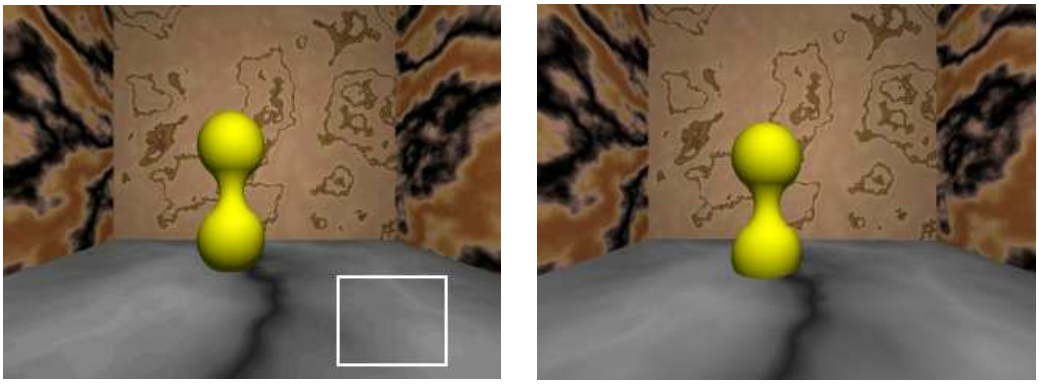


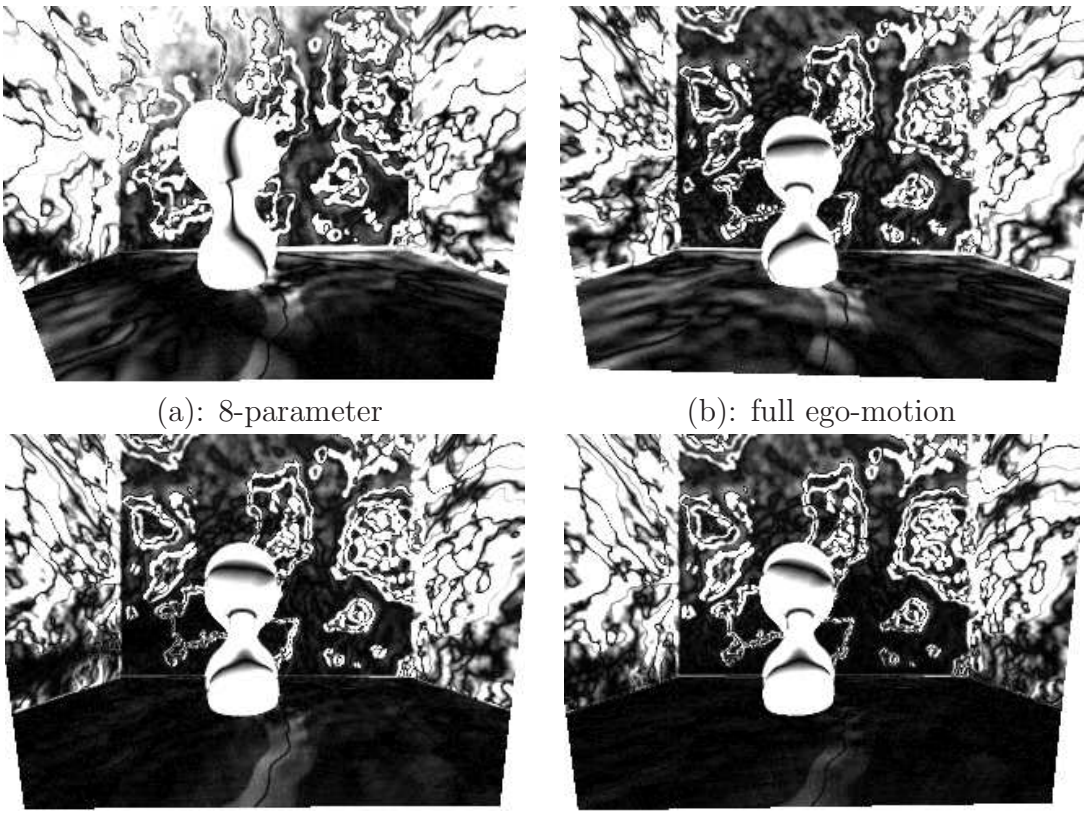
Figure 7.1: Synthesized images where the ground plane has low textures. The rectangle shows one of the patch used to compute the camera ego-motion.

rectangle in Fig.(7.1a).

From Table (7.1), we can see that removing the diminishing parameters greatly improves the condition of Hessian matrix, since diminishing parameters are inherently ill-conditioned. As a result, the 8-parameter model has the worst condition since its number of unknown parameters is far more than necessary. The downward-looking camera improves the condition number by orders of magnitudes, and recovers the most accurate ego-motion, which supports our observations in Section 7.3.2. The forward-looking model performs better than the full ego-motion model. But it appears that part of the left-turn has been confused by left-translation in the forward-looking model.

Fig.(7.2) shows the motion-compensated residual images for qualitative comparison. Given the global vehicle ego-motion with respective to the ground, we can directly compute the 2D homography $\mathbf{H}$ of the ground plane between the given two video frames ($\mathbf{H} = \mathbf{K}(\mathbf{R} - \frac{\mathbf{t}}{d}\mathbf{n}^\top)\mathbf{K}$). We warp the second image towards the first image using $\mathbf{H}$. The residuals are defined as the difference between the original first image and the warped result of the second image. The more accurate the estimated ego-motion, the smaller the resulted residuals.

As we can see from Fig.(7.2), pixels inside the used rectangle are well-compensated in all models, but only the downward-looking camera fully compensates all pixels in the ground plane, which means that it actually recovers a better global motion model based on a small image patch.



(a): 8-parameter (b): full ego-motion

(c): forward-looking (d): downward-looking

Figure 7.2: Motion compensated residual images by motions from Table (7.1). The residuals are scaled up by a factor of 4 for visibility.

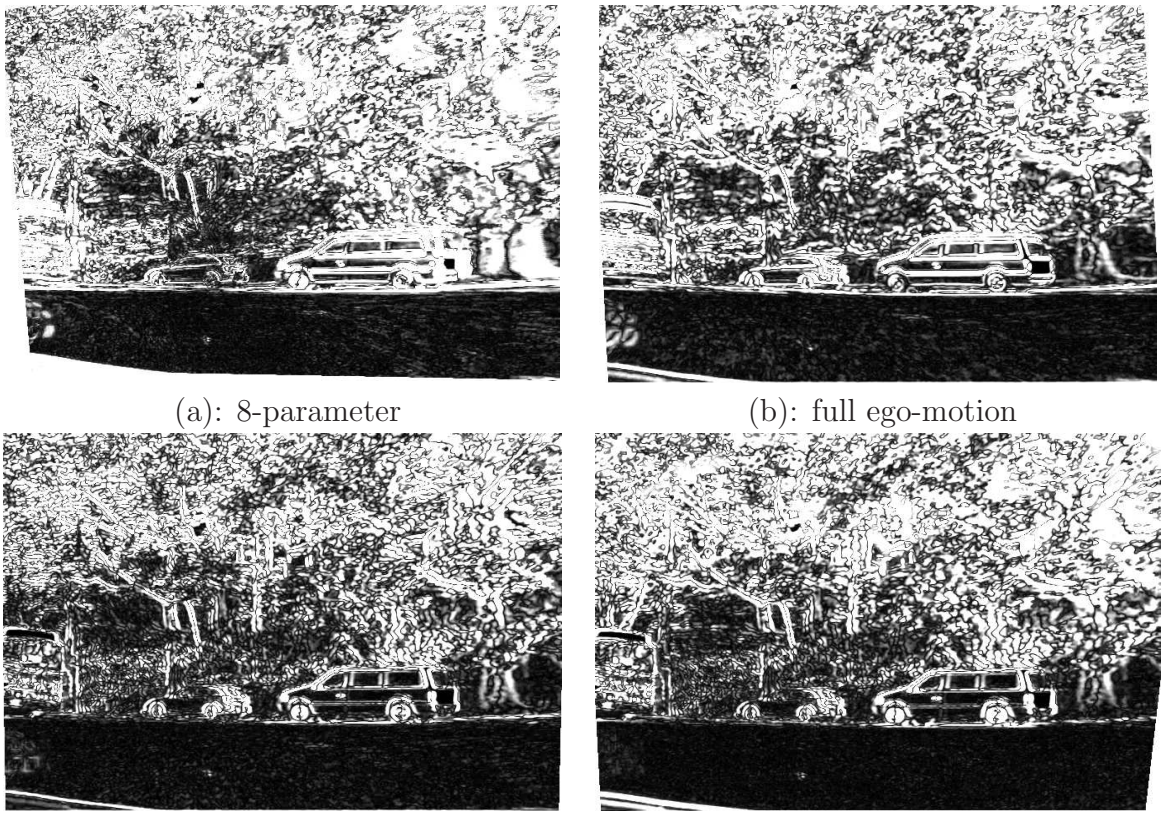
Figure 7.3: Real images with low textures on the ground plane, and moving cars/bus in the background.

Real case

In this subsection, we use real images to compare the performance of ego-motion estimation based on small image patches. Fig.(7.3) shows the images we use, where the camera is put on a car that is simultaneously moving forward and turning left (around an axis at some distance to the vehicle and vertical to the ground). The rectangle shows the image patch we randomly select to compute the ego-motion. It is quite a challenging task due to the very low texture of the road and the small image patch.

The last two columns in Table (7.1) show the condition number of the Hessian matrix and the recovered ego-motion. As we can see, the downward-looking camera has the best condition number, and its recovered ego-motion correctly indicates that the car is moving forward and turning left. The forward-looking camera model does not recover the correct left-turn motion, which appears to be caused by the confusion between w_Y and T_X. The full ego-motion model has large non-planar motions, which is obviously incorrect.

The motion compensated residual images in Fig.(7.4) qualitatively show the performance. As we can see, all motion models well compensate the pixels inside the rectangle that are used for estimation, but only the downward-looking camera compensates all the pixels on the ground, as can be indicated by the yellow lane marks at the bottom-left of the images. The darker pixels at the very bottom-left of the images (right below the yellow lane marks) in Fig.(7.3) are part of the car dash-board, and their corresponding residuals in the downward-looking camera model show correct



(a): 8-parameter (b): full ego-motion

(c): forward-looking (d): downward-looking

Figure 7.4: Motion compensated residual images. The residuals are scaled up by a factor of 4 for visibility. Notice the residuals of lane-marks at the bottom left, and the residuals of car dash-board right below the lane-marks. The downward-looking camera model compensates the lane marks best, and shows correct parallax on the car dash-board.

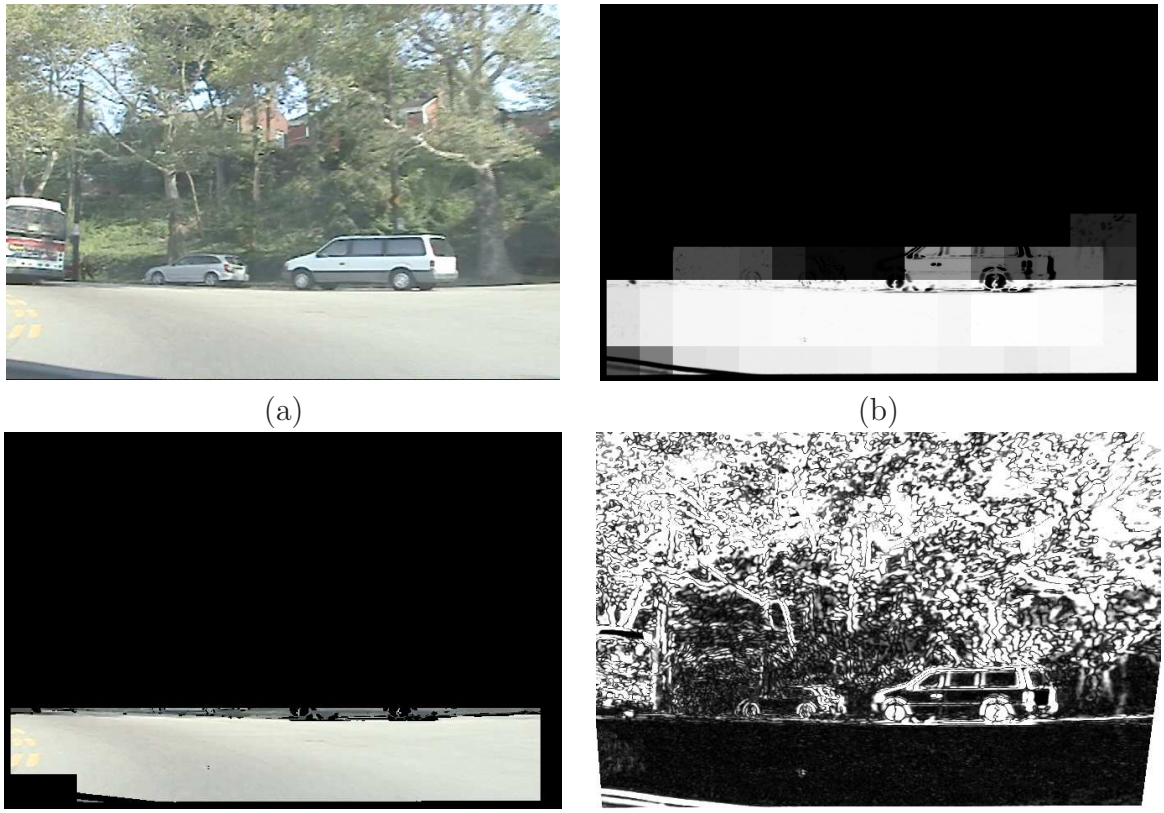
Figure 7.5: Traffic scene in city with cluttered background containing moving cars. The road has weak or linear textures.

parallax. We also use the shape of the image boundaries to indicate the ego-motion. As we can see, only the downward-looking camera has correct shape corresponding to forward and left-turn motions.

7.5.2 Ground layer detection and global ego-motion estimation

In this section, we show the results of ground layer detection and the global ego-motion estimation. Fig.(7.3) and Fig.(7.5) show the two image sequences we used in this experiment. The roads have either very weak texture, or linear image structure. The backgrounds are cluttered and contain moving objects.

Fig.(7.6) and (7.7) shows the experimental results. The first row is the result on Fig.(7.3), and the second row is the result on Fig.(7.5). Fig.(7.6b) shows the weights (see Eq.(7.15)) indicating the ownership (ground layer or non-ground layer) of the pixels. Outliers, such as moving cars (and their shadows), buildings, and trees on the side, are clearly indicated by low weights. Fig.(7.6c) shows the detected ground layer using a simple histogram-based threshold scheme. The lane marks on the road are included in the ground layer although their colors are quite different from the majority pixels of the road plane. The car dash-board at the bottom-left in the first image sequence, and the trees and moving cars on the ground in both image sequences, are excluded due to significantly lower weights. Notice that some of the outliers, such as the trees at the right side of the road in the second sequence, have very low texture and therefore low intensity residuals, but still be excluded due to the



(a) (b)

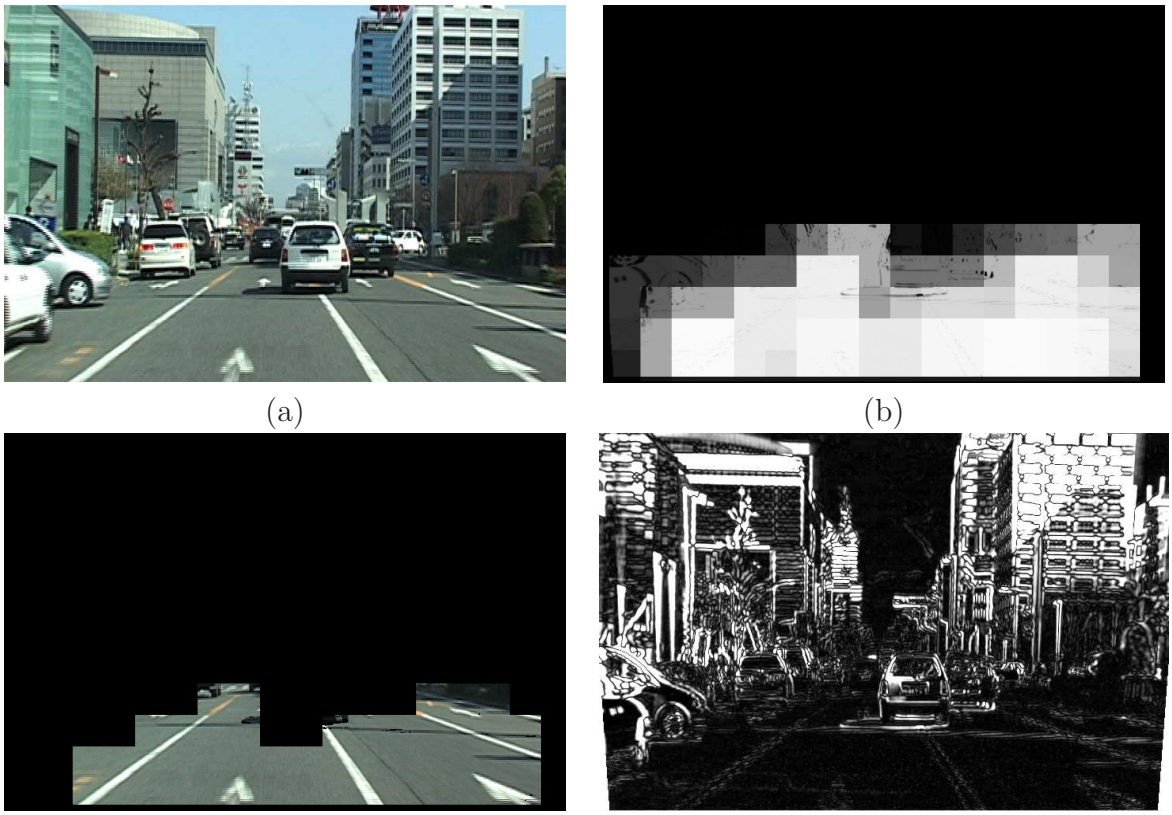
(c) (d)

Figure 7.6: Ground layer detection and global ego-motion estimation. (a): reference frame of the input images; (b): weights indicating the ownership of pixels (brighter means larger weight); (c): detected ground layer using weights in (a); (d): motion compensated residuals by the global ego-motion. The residuals are scaled up by a factor of 4 for visibility.

fact that their geometries (plane normals) are significantly different from the ground plane normal. Fig.(7.6d) shows the motion compensated residuals by the global ego-motion estimated based on the weights in (b). As we can see, the pixels on the road are well compensated, while pixels from other objects, such as the buildings, trees, and the moving cars with their shadows, show correct parallax.

7.6 Conclusion

Vehicle ego-motion estimation and ground layer detection are challenging tasks due to low texture on the road and the non-linear perspective distortion. By ways of virtual



(a) (b)

(c) (d)

Figure 7.7: Ground layer detection and global ego-motion estimation. (a): reference frame of the input images; (b): weights indicating the ownership of pixels (brighter means larger weight); (c): detected ground layer using weights in (a); (d): motion compensated residuals by the global ego-motion. The residuals are scaled up by a factor of 4 for visibility.

camera, we have made use of the constraint that the vehicle is undergoing planar motion on the ground. Enforcing such constraint is necessary to avoid the estimation of diminishing parameters that are ill-conditioned. By using virtual downward-looking camera, we further improve the condition of the Hessian matrix, and eliminate the ambiguities among the unknown parameters, which are linear in terms of image coordinates and can be reliably estimated. Together with a geometry-based robust weighting scheme, we have shown promising results on vehicle ego-motion estimation and ground layer detection.

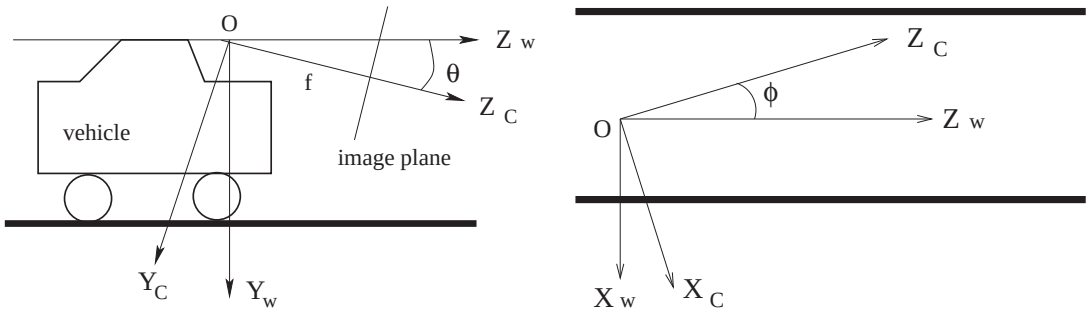
We have assumed that the camera focus length is known. In practice, we only require a rough initialization of the focus length, since the error in the focus length only introduces systematic bias on the estimated ego-motion, but does not affect the ground layer detection. We can therefore use the algorithm presented in this paper to derive the ground plane. Then use the detected ground plane to calibrate the camera [99, 60] to correct the bias in ego-motion.

Appendix: calibration of camera look-at point

Fig.(7.8) shows the coordinate frames of the vehicle (X_W, Y_W, Z_W) and the camera (X_C, Y_C, Z_C). We must know the camera look-at point [6] *w.r.t. the vehicle* in order to virtually rotate the camera desired pose. The look-at point is defined, in the frame of (X_W, Y_W, Z_W), as the intersection point of axis Z_C and the plane $Z_W = 1$. Since the camera is fixed w.r.t. the vehicle, the look-at point is fixed too. To derive the look-at point, we drive the car along a straight road with parallel lane marks, and take a few images. If the optical axis Z_C is identical to the axis Z_W, the vanishing point in each image, $v = (x_v, y_v)$, of the parallel lane marks will be coincident with the camera principal point $c = (x_c, y_c)$ [7]. Therefore the look-at point is $\left(\frac{x_v - x_c}{f}, \frac{y_v - y_c}{f}\right)$, where f is focus length. Each image gives an estimation of the look-at point. If using multiple images, we use the mean of them. The horizon line is then defined as $y = y_v$. Automatic techniques have been developed by researchers to compute vanishing points. In our experiments, the lane marks are semi-automatically identified.

[6] We assume x_C is parallel to the ground plane. Violation of such assumption will only introduce a constant bias in the rotation around the camera axis, which cancels out when computing the relative ego-motion among the views.

[7] Assumed to be at the image center.



(a): side view (b): top view

Figure 7.8: Coordinate frames.

Chapter 8

Conclusions and Future Work

8.1 Conclusion

A layer consists of one or more 2D regions in the image domain, inside which the pixels share the same 2D motion model. Layer(ed) representation can be considered as the first cut to the image/video analysis task. Layers are therefore useful mid-level video representation for many applications in computer vision, including motion analysis, object tracking, scene analysis, video coding, etc. In recent years, various approaches have been proposed to extract layers from image sequences, including EM algorithm, grouping (clustering), tensor voting, etc.

This thesis presented a novel subspace approach to layer extraction. It can naturally utilize the spatial-temporal constraints existing in the video to improve the reliability of layer extraction. Specifically, using statistical analysis we prove that 1) layers are more discriminative in the subspace, and 2) increasing the frame number in the input video increases the layer discriminability in the subspace, but not in the original space.

Computing the subspace from the local measurements is a critical step. Such step is usually performed via low rank matrix approximation, which in turn is achieved by factorizing the measurement matrix. The L2-norm based matrix approximation leads to the SVD algorithm, which is sensitive to outliers in local measurements. Based on the fact that inliers form clusters and lie in the subspace, we present the kNND metric to detect extreme outliers that are far away from clusters, and the robust SVD algorithm to detect the structure outliers that violate the subspace constraint.

We also discuss the use of L1-norm based metric for low rank matrix approximation, which is intrinsically robust to outliers, and can be done via quadratic programming.

To deal with small layers that are harder to extract than the large layers, we propose kNND and semi-progressive layer extraction algorithm. The idea behind is that large (easier) layers should be extracted first and used to guide the extraction of subsequent small (harder) layers.

We applied our approach to a variety of real image sequences, and achieve good results of layer segmentation.

8.2 Future work

There are several lines of research that can be further conducted:

1. One important area is to exploit the applications of layer(ed) representation, including navigation, layer-based structure from motion, camera self-calibration, image-based modelling and rendering, video super-resolution.

2. The subspace computation (factorization of measurement matrix) can be formulated as minimizing the following cost function:

$$E(\mathbf{U}, \mathbf{V}) = \|\mathbf{S} \otimes (\mathbf{W} - \mathbf{U}\mathbf{V})\|_p$$

When the weighting matrix $\mathbf{S}$ has rank higher than 1, the cost function is usually no longer convex in $\mathbf{U}$ and $\mathbf{V}$. The local measurement for each patch is the homographies between the reference frame against every other frame in the video. For each patch, if one homography between the reference frame and one single other frame is missed or has large error (outlier), we must discard such patch even if the homographies related to all the other frames are good measure. The reason is to guarantee the rank 1 condition of $\mathbf{S}$, therefore to guarantee a convex cost function.

If we want to use such patch, we will face the problem of optimizing a non-convex function $E(\mathbf{U}, \mathbf{V})$. Iterative procedure such as alternating minimization over $\mathbf{U}$ and $\mathbf{V}$ is usually required to obtain a good local minimum of $E(\mathbf{U}, \mathbf{V})$. The scale space theory in computer vision can be applied here to obtain an approximated

global minimum. The idea is to smooth the cost function $E(\mathbf{U}, \mathbf{V})$ through smart formation of the weighting matrix $\mathbf{S}$.

3. A closely related field is motion segmentation, where the local measurements are the translation of feature points. Structure from motion using line segments is also well studied in computer vision. Feature points, lines, and planar patches (affine homographies) have their own pros and cons. A framework to unify these different features to bring the pros together for layer extraction is desirable.

4. Although constraints from all frames in the given image sequence can be fused together in the subspace approach to layer extraction, as far as we know, currently all of the layer extraction algorithms segment only one image at a time. New framework that can simultaneously segment all frames in the input image sequence is desirable, because existing (may be unknown) physical constraints (such as the rigidity of the object in the scene) can be better utilized.

5. In the experiments we have used 2D affine motion model. Affine motion model is generally used in practice. As we have presented in Chapter 2, the subspace theory applies to projective motion model too. Our approaches can be used as long as reasonable local measurements (the 2D projective motions) for local patches are available. In the future, we will experiment with scenes with high parallax where projective model may be more desirable at the very beginning. In such case, for local measurements, we can still start from affine model. We can then aggregate locally 2D affine motions into more global projective motions in the loop of layer estimation. In other words, once the support of each 2D region for local measurement is large enough, we can reliably estimative its 2D projective motion.

6. In general, layer representation is well-suited for scenes that can be approximated by some smooth surfaces. Some scenes may contain dense parallax where part of the image may not be described using layers. In such case, we would extract the possible layers, and then use the concept of plane + parallax to describe the dense parallax. Layers will still be advantageous since existing scene regularity can be exploited at the representation level.

Bibliography

[1] E.H. Adelson. Layered representations for image coding. Technical Report 181, MIT Media Lab, 1991.

[2] G. Adiv. Determining 3-d motion and structure from optical flow generated by several moving objects. *PAMI*, 7(4):384–401, July 1985.

[3] G. Adiv. Inherent ambiguities in recovering 3-d motion and structure from a noisy flow field. *PAMI*, 11(5), May 1989.

[4] Y. Aloimonos. Harmonic computational geometry: A new tool for visual correspondence. In *BMVC 2002*, 2002.

[5] Yucel Altunbasak, P. Erhan Eren, and A. Murat Tekalp. Region-based parametric motion segmentation using color information. *Graphical Models and Image Processing*, 60(1), January 1998.

[6] Martin Armstrong, Andrew Zisserman, and Richard I. Hartley. Self-calibration from image triplets. In *ECCV96*, pages 3–16, 1996.

[7] Christopher G. Atkeson, Andrew W. Moore, and Stefan Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1-5):11–73, 1997.

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

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

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

[11] M. J. Black and P. Anandan. The robust estimation of multiple motions: Affine and piecewise-smooth flow fields. Tr, Xerox PARC, CA, Dec. 1993.

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

[13] Michael J. Black and Allan D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. In *ECCV (1)*, pages 329–342, 1996.

[14] M.J. Black and P. Anandan. A framework for the robust estimation of optical flow. In *ICCV93*, pages 231–236, 1993.

[15] J.F. Blinn. Jim blinns corner: Compositing, part 1: Theory. *IEEE Computer Graphics and Applications*, 14(5):83–87, September 1994.

[16] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3d shape from image streams. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

[17] Simon Byers and Adrian E. Raftery. Nearest neighbor clutter removal for estimating features in spatial point processes. *Journal of the American Statistical Association*, 93(442):577–584, 1998.

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

[19] M. Collins, S. Dasgupta, and R. E. Schapire. A generalization of principal components analysis to the exponential family. In T. G. Dietterich, S. Becker, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[20] Robert Collins. Mean-shift blob tracking through scale space. In *Computer Vision and Pattern Recognition (CVPR'03)*, June 2003.

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

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

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

[24] Dorin Comaniciu. Nonparametric information fusion for motion estimation. In *CVPR*, Madison, June 2003.

[25] Joao Costeira and Takeo Kanade. A multi-body factorization method for motion analysis. In *ICCV*, pages 1071–1076, 1995.

[26] I. Csiszár and G. TTusnády. Information geometry and alternating minimization procedures. *Statistics and Decisions, Supplement Issue*, 1:205–237, 1984.

[27] K. Daniilidis and H.H. Nagel. The coupling of rotation and translation in motion estimation of planar surfaces. In *CVPR93*, pages 188–193, 1993.

[28] T. Darrell and A. Pentland. Robust estimation of a multilayered motion representation. In *IEEE Workshop on Visual Motion*, pages 173–178, Princeton, 1991.

[29] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximal likelihood form incomplete data via the em algorithm. *RoyalStat*, B 39:1–38, 1977.

[30] 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.

[31] K. Fukunaga. *Statistical Pattern Recognition*. New York: Academic Press, 1989.

[32] Ruben Gabriel and S. Zamir. Lower rank approximation of matrices by least squares with choise of weights. *Technometrics*, 21(4):489–498, November 1979.

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

[34] G.H. Golub and C.F. Van Loan. *Mattrix Computation*. Johns Hopkins University Press, Baltimore, maryland, 2nd edition, 1989.

[35] K.J. Hanna. Direct multi-resolution estimation of ego-motion and structure from motion. In *MOTION91*, pages 156–162, 1991.

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

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

[38] R.I. Hartley. In defence of the 8-point algorithm. In *ICCV95*, pages 1064–1070, 1995.

[39] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning.* Springer Series in Statistics. Springer-Verlag, 2001.

[40] D.J. Heeger and A.D. Jepson. Subspace methods for recovering rigid motion i: Algorithms and implementation. *IJCV*, 7(2):95–117, January 1992.

[41] B. K. P. Horn. *Robot Vision.* MIT Press, Cambridge, MA, 1986.

[42] B.K.P. Horn and E.J. Weldon, Jr. Direct methods for recovering motion. *IJCV*, 2(1):51–76, June 1988.

[43] S. Hsu, P. Anandan, and S. Peleg. Accurate computation of optical flow by using layered motion representations. In *ICPR94*.

[44] Hsu S. Irani, M. and P. Anandan. Video compression using mosaic representations. *Signal Processing: Image Communication, special issue on Coding Techniques for Low Bit-rate Video*, 7(4-6):529–552, 1995.

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

[46] M. Irani and S. Peleg. Improving resolution by image registration. *CVGIP:Graph. Models Image Process*, 53:231–239, 1991.

[47] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *ECCV92*.

[48] M. Irani, B. Rousso, and S. Peleg. Recovery of ego-motion using image stabilization. In *CVPR94*, pages 454–460, 1994.

[49] Michal Irani and P. Anandan. Factorization with uncertainty. In *ECCV (1)*, pages 539–553, 2000.

[50] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

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

[52] R.A. Johnson and D.W Wichern. *Applied Multivariate Statistical Analysis*. Prentice Hall, 2nd edition, 1988.

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

[54] S.B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *CVPR 2001*.

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

[56] Qifa Ke and Takeo Kanade. A robust subspace approach to layer extraction. In *IEEE Workshop on Motion and Video Computing (Motion 2002)*, 2002.

[57] Qifa Ke and Takeo Kanade. Robust subspace computation using L1 norm. Technique report CMU-CS-03-172, Computer Science Department, Carnegie Mellon University, 2003.

[58] Qifa Ke and Takeo Kanade. Transforming camera geometry to a virtual downward-looking camera: Robust ego-motion estimation and ground-layer detection. In *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR 2003)*, Madison, June 2003.

[59] S. Khan and M. Shah. Object based segmentation of video using color, motion and spatial information. In *CVPR01*.

[60] Joss Knight, Andrew Zisserman, and Ian Reid. Linear auto-calibration for ground plane motion. In *CVPR 2003*, June 2003.

[61] R Kumar, P Anandan, and K Hanna. Shape recovery from multiple views: a parallax based approach. In *ARPA Image Understanding Workshop, Monterey, CA, Nov. 1994*. Morgan Kauffmann Publishers, November 1994.

[62] 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.

[63] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[64] R. Mandelbaum, G. Salgian, and H. Sawhney. Correlation-based estimation of ego-motion and structure from motion and stereo. In *ICCV99*, pages 544–550, 1999.

[65] O. L. Mangasarian and David R. Musicant. Robust linear and support vector regression. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(9):950–955, 2000.

[66] D. Marr. *Vision: a Computational Investigation into the Human Representation and Processing of Visual Information*. W.H. Freeman, San Francisco, 1982.

[67] S. Maybank. *Theory Of Reconstruction From Image Motion*. Springer-Verlag, 1992.

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

[69] H. Murase and S. Nayar. Visual learning and recognition of 3d objects from appearance. *IJCV*, 1(14):5–24, 1995.

[70] S. Negahdaripour and B.K.P. Horn. Direct passive navigation. *PAMI*, 9(1):168–176, January 1987.

[71] T. Poggio, V. Torre, and C. Koch. Computational vision and regularization theory. *Nature*, 317:314–319.

[72] Tomaso Poggio and Federico Girosi. A theory of networks for approximation and learning. Technical Report AIM-1140, MIT AI Lab, 1989.

[73] J. Rissanen. A universal prior for integers and b¿estimation by minimum description length. *The Annals of Statistics*, 11(2):416–431, 1983.

[74] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*. John Wiley and Sons, New York, 1987.

[75] Henry A. Rowley, Shumeet Baluja, and Takeo Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998.

[76] Yoichi Sato and Katsushi Ikeuchi. Temporal-color space analysis of reflection. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 570 – 576, June 1993.

[77] H.S. Sawhney and S. Ayer. Compact representations of videos through dominantand mulitple motion estimation. *PAMI*, 18:814–831, 1996.

[78] Henry Schneiderman and Takeo Kanade. A statistical model for 3d object detection applied to faces and cars. In *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, June 2000.

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

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

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

[82] Jianbo Shi and Carlo Tomasi. Good features to track. In *CVPR'94*.

[83] S. Soatto and P. Perona. Recursive 3-d visual-motion estimation using subspace constraints. *IJCV*, 22(3):235–259, March 1997.

[84] Nathan Srebro and Tommi Jaakkola. Weighted low-rank approximations. In *ICML 2003 (to appear)*.

[85] G. Stein, O. Mano, and A. Shashua. A robust method for computing vehicle ego-motion. In *IEEE Intelligent Vehicles Symposium (IV2000)*, 2000.

[86] R. Szeliski. Image mosaicing for tele-reality applications. In *WACV94*, pages 44–53, 1994.

[87] Richard Szeliski and Heung-Yeung Shum. Creating full view panoramic image mosaics and environment maps. *Computer Graphics*, 31(Annual Conference Series):251–258, 1997.

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

[89] H. Tao, H.S. Sawhney, and R. Kumar. Object tracking with bayesian estimation of dynamic layer representations. *PAMI*, 24(1):75–89, January 2002.

[90] T.H.Cormen and C.E.Leiserson andR.L.Rivest. *Introduction to Algorithms*. MIT Press, McGraw-Hill, New York, NY, 1990.

[91] T.Y. Tian, C. Tomasi, and D.J. Heeger. Comparison of approaches to egomotion computation. In *CVPR'96*, pages 315–320, 1996.

[92] A. N. Tikhonov and V. A. Arsenin. *Solutions of Ill-posed Problems*. Winston & Sons, 1977.

[93] M. Tipping and C. Bishop. Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Aston University, 1997.

[94] M. Tipping and C. Bishop. Mixtures of probabilistic principal component analysers. *Neural Computation*, 11(2):443–482, 1999.

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

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

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

[98] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *International Conference on Computer Vision*, pages 255–261, 1999.

[99] W. Triggs. Autocalibration from planar scenes. In *ECCV'98*.

[100] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuro Science*, 3(1):71–86, 1991.

[101] Richard Voyles, James Morrow, and Pradeep Khosla. The shape from motion approach to rapid and precise force/torque sensor calibration. *Journal of Dynamic Systems, Measurement and Control*, 119(2):229 – 235, June 1997.

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

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

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

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

[106] Yair Weiss. Segmentation using eigenvectors: a unifying view. In *IEEE International Conference on Computer Vision*, pages 975–982, 1999.

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

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