VISUAL ANALYSIS AND SYNTHESIS WITH PHYSICALLY GROUNDED CONSTRAINTS

BY

JIA-BIN HUANG

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Electrical and Computer Engineering
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2016

Urbana, Illinois

Doctoral Committee:

Professor Narendra Ahuja, Chair
Professor Thomas S. Huang
Professor Minh N. Do
Professor Mark Hasegawa-Johnson
Associate Professor Derek Hoiem

ABSTRACT

The past decade has witnessed remarkable progress in image-based, data-driven vision and graphics. However, existing approaches often treat the images as pure 2D signals and not as a 2D projection of the physical 3D world. As a result, a lot of training examples are required to cover sufficiently diverse appearances and inevitably suffer from limited generalization capability. In this thesis, I propose inference-by-composition approaches to overcome these limitations by modeling and interpreting visual signals in terms of physical surface, object, and scene. I show how we can incorporate physically grounded constraints such as scene-specific geometry in a non-parametric optimization framework for (1) revealing the missing parts of an image due to removal of a foreground or background element, (2) recovering high spatial frequency details that are not resolvable in low-resolution observations. I then extend the framework from 2D images to handle spatio-temporal visual data (videos). I demonstrate that we can convincingly fill spatio-temporal holes in a temporally coherent fashion by jointly reconstructing the appearance and motion. Compared to existing approaches, our technique can synthesize physically plausible contents even in challenging videos. For visual analysis, I apply stereo camera constraints for discovering multiple approximately linear structures in extremely noisy videos with an ecological application to bird migration monitoring at night. The resulting algorithms are simple and intuitive while achieving state-of-the-art performance without the need of training on an exhaustive set of visual examples.

To my parents, for their love and support.

ACKNOWLEDGMENTS

This thesis would not have been possible without the support and guidance from many people around me during my PhD.

I want to first thank my thesis advisor, Prof. Narendra Ahuja, for his patience and wisdom. I have learned a lot from him about how to ask important questions and conduct high-quality research.

My gratitude goes to four other members of my thesis commitee, Prof. Derek Hoiem, Prof. Minh Do, Prof. Mark Hasegawa-Johnson, and Prof. Thomas S. Huang. All of them have provided insightful and valuable comments that guide me to improve the thesis.

During my PhD study, I have been fortunate to have the opporunity to work with several excellent researchers, including Dr. Sing Bing Kang, Dr. Johannes Kopf, Dr. Zhengyou Zhang, Dr. Zicheng Liu, Dr. Qin Cai, Dr. Leonid Sigal, Dr. Sung Ju Hwang, and Dr. Rich Caruana.

Also, I would like to thank my labmates over the years, including John, Bernard, Sanketh, Esther, Emre, Xianbiao, Qingxiong, Hsien-Ting, Abhishek, Avinash, Huiguang, and Shengnan. I enjoyed having you around in the computer vision and robotics laboratory.

Last but not least, I express my sincere gratitude to my family: my dearest dad, mom, brother, and wife for their unconditioned love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Inverse problems in image and video processing such as image denoising, super-resolution, inpainting, and motion deblurring have been fundamental and long-standing problems in computer vision and image processing. Developing an effective approach for these inverse problems will have great impact in both engineering and scientific fields as these problems naturally emerge from various imaging scenarios.

Learning and leveraging good natural image priors arguably is the most critical step for addressing these ill-conditioned or under-determined linear inverse problems. During the past several decades, a variety of natural image priors have been proposed, including smoothness, total variation, sparse modeling, and self-similarity. In particular, data-driven priors learned from large external image datasets using advanced machine learning algorithms have demonstrated great success in many challenging tasks.

On the other hand, the internal fractal structures of natural images have been exploited as a strong image-specific prior. A classical example of such prior is the non-local means approach to image denoising. The key idea behind a non-local image processing algorithm is that natural image patches tend to re-occur many times within the same image. Therefore, grouping similar patches together allows one to accurately reconstruct the underlying clean signal by simply averaging them. A more sophisticated collaborating filtering algorithm based on non-local means has achieved state-of-the-art performance in a wide range of problems.

However, all these priors treat the observed image as a pure 2D signal. They are agnostic to the fact that an image is a perspective projection of the physical 3D world. This substantially limits the ability to use internal natural priors for solving challenging cases.

In this thesis, we investigate the use of mid-level representation of the scene for tackling these ill-posed problems induced from the imaging process. The core idea is to model and interpret image and visual signals in terms of a physical

surface, scene, and motion. In particular, we extract mid-level representation of the scene, such as plane localization, translational symmetry detection, and per-pixel dense motion representation. We then make use of the inferred information to guide the low-level patch-based optimization algorithm in a unified optimization framework. We have shown that such mid-level representation greatly helps improve the performance of several important problems, including image completion, image super-resolution, and video completion.

This thesis addresses the following four major topics.

Image Completion [1] We propose a method for automatically guiding patch-based image completion using mid-level structural cues. Our method first estimates planar projection parameters, softly segments the known region into planes, and discovers translational regularity *within* these planes. This information is then converted into soft constraints for the low-level completion algorithm by defining prior probabilities for patch offsets and transformations. Our method simultaneously handles multiple planes, and in the absence of any detected planes falls back to a baseline fronto-parallel image completion algorithm. We validate our technique through extensive comparisons with state-of-the-art algorithms on a variety of scenes.

Single Image Super-Resolution [2] Self-similarity based super-resolution (SR) algorithms are able to produce visually pleasing results without extensive training on external databases. Such algorithms exploit the statistical prior that patches in a natural image tend to recur within and across scales of the same image. However, the internal dictionary obtained from the given image may not always be sufficiently expressive to cover the textural appearance variations in the scene. We extend self-similarity based SR to overcome this drawback. We expand the internal patch search space by allowing geometric variations. We do so by explicitly localizing planes in the scene and using the detected perspective geometry to guide the patch search process. We also incorporate additional affine transformations to accommodate local shape variations. We propose a compositional model to simultaneously handle both types of transformations. We extensively evaluate the performance in both urban and natural scenes. Even without using any external training databases, we achieve significantly superior results on urban scenes, while maintaining comparable performance on natural scenes as other state-of-the-art SR algorithms.

Temporally Coherence Video Completion We present an automatic video completion algorithm that synthesizes missing regions in videos in a temporally coherent fashion. Our algorithm can handle dynamic scenes captured using a moving camera. State-of-the-art approaches have difficulties handling such videos because viewpoint changes cause image-space motion vectors in the missing and known regions to be inconsistent. We address this problem by jointly estimating optical flow and color in the missing regions. Using pixel-wise forward/backward flow fields enables us to synthesize temporally coherent colors. We formulate the problem as that of non-parametric patch-based optimization. We demonstrate our technique on numerous challenging videos.

Video-based Bird Detection [3] Bird migration is a critical indicator of environmental health, biodiversity, and climate change. Existing techniques for monitoring bird migration are either expensive (e.g., satellite tracking), labor-intensive (e.g., moon watching), indirect and thus less accurate (e.g., weather radar), or intrusive (e.g., attaching geolocators on captured birds). We present a vision-based system for detecting migrating birds in flight at night. Our system takes stereo videos of the night sky as inputs, detects multiple flying birds and estimates their orientations, speeds, and altitudes. The main challenge lies in detecting flying birds of unknown trajectories under high noise level due to the low-light environment. We address this problem by incorporating stereo constraints for rejecting physically implausible configurations and gathering evidence from two (or more) views. Specifically, we develop a robust stereo-based 3D line fitting algorithm for geometric verification and a deformable part response accumulation strategy for trajectory verification. We demonstrate the effectiveness of the proposed approach through quantitative evaluation of real videos of birds migrating at night collected with near-infrared cameras.

Other Work not in the Thesis During my doctorial study, I also visited problems other than exploiting physically grounded constraints [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. Interested readers can refer to these publications for more details.

CHAPTER 2

IMAGE COMPLETION USING PLANAR STRUCTURE GUIDANCE

2.1 Introduction

Replacing or filling regions in images with plausibly synthesized content is a common image editing operation. This task, known as image completion, is used in applications ranging from the removal of unwanted objects in personal photos to movie post-production. It is also an important step in many graphics algorithms, e.g., for generating a clean background plate or reshuffling image contents.

While much progress has been made, image completion remains a challenging problem. This is because some higher level understanding of the scene is often required. The state-of-the-art automatic algorithms typically rely on low-level cues; they synthesize the missing region as a field of overlapping patches copied from the known region [19]. Here, they attempt to synthesize an image that *locally* appears like the known input everywhere, and such that overlapping patches agree as much as possible. Barnes et al. [20] showed how this algorithm can be sped up using a random search and propagation scheme.

Most of these algorithms have two important limitations. First, since they only directly copy translated patches from the input, the performance degrades with scenes that are not fronto-parallel. They would not be able to effectively handle the perspective foreshortening as shown in Figure 2.1.[1] The other limitation is in the tendency of converging to local minima, due to the strong non-convexity of the objective. This second problem is somewhat alleviated by applying the algorithm in a coarse-to-fine manner.

Recent approaches handle the fronto-parallel limitation by considering patch transformations such as rotation, scale, and gain/bias color adjustments [21, 17]. While this improves the algorithm's ability to complete general scenes, it results in an exponential increase of the search space from 2 degrees of freedom per output

[1] All the images in the thesis are used under Creative Commons license.

Figure 2.1: Our image completion algorithm automatically extracts mid-level constraints (perspective and regularity) and uses them to guide the filling of missing regions in a semantically meaningful way. Our method is capable of completing challenging scenes such as multiple building facades (left), strong perspective distortion (middle) and large regular repetitive structures (right). We significantly outperform three representative state-of-the-art image completion techniques for these images (see Figure 2.2). Image credits (left to right): Flickr users micromegas, Theen Moy, Nicu Buculei.

pixel up to 8 (or more). This adds many local minima to the solution space and hence worsens the tendency of giving rise to bad configurations. This effect can be observed in Figures 2.2, 2.8, 2.9, and 2.10.

In this chapter, we show how image completion can be substantially improved by automatically *guiding* the low-level synthesis algorithm using *mid-level structural analysis* of the known region. Specifically, we estimate planar projection parameters (i.e., the local perspective slope of the scene) as well as translational regularity in the affine rectified domain (explained later in Section 2.4.1); this information is used to constrain the search space in the missing region. These constraints are cast as a prior probability of the patch transformation parameters. As a result, we can use an even richer patch transformation model than previous work (i.e., full homographies) since our constraints effectively reduce this high dimensional model to a lower degree subspace.

We handle multiple detected planes (that may be perspectively distorted) by using a soft proximity-based weighting scheme and relying on the power of the low-level algorithm for finding good transitions. Note that while we model the world as piecewise planar, we are *not* just restricted to such scenes: just as the

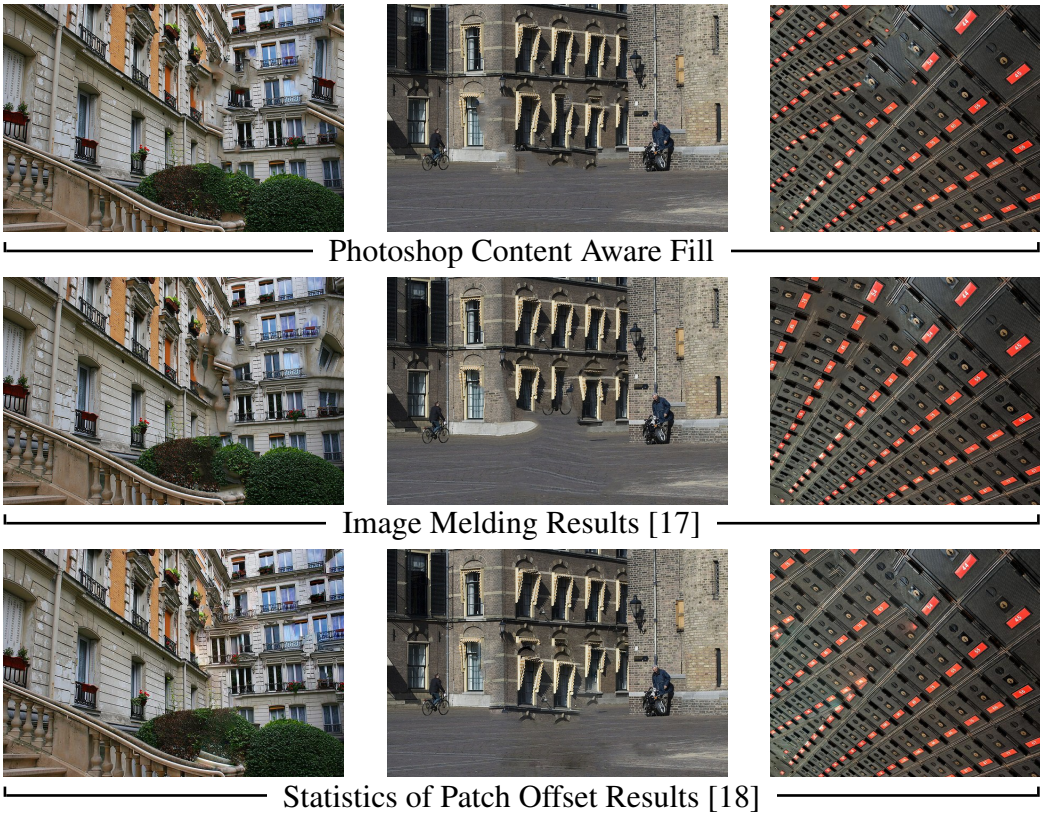


Figure 2.2: Limitations of current state-of-the-art methods. Compare these results with ours in Figure 2.1.

original completion algorithm was not limited to fronto-parallel scenes, ours allows significant deviation from the piecewise planar model, as evidenced by our results.

Our algorithm significantly improves performance for challenging man-made scenes such as those of architecture and indoors. In the absence of any detected structural cues, e.g., for most natural landscape images (Figure 2.9), our algorithm falls back to standard unconstrained completion, i.e., our implementation of Wexler et al.'s algorithm [19]. We validate our method by comparing against state-of-the-art algorithms. We show numerous representative results in Section 2.6.

2.2 Previous Work

In this section, we review representative techniques for image completion. Image completion techniques can be roughly categorized as diffision-based or example-based.

Diffusion-based techniques fill in small or narrow holes by propagating adjacent image structures. This class of techniques is pioneered by Bertalmio et al. [22] and later extended by Ballester et al. [23] and Bertalmio et al. [24]. These techniques are less effective in handling large missing regions due to their inability to synthesize textures.

By comparison, example-based methods exploit redundancy in natural images for filling missing pixels. They are based on example-based texture synthesis methods [25, 26]. Variants of these methods include using structure-based priority [27], deterministic EM-like schemes [28, 19, 20], or MRF models with patches as labels, which can be solved efficiently using belief propagation [29] or graph cut [30]. These techniques still fundamentally rely on low-level cues, which are less effective for images with larger structures.

In many real scenes, the appearance can change significantly due to local scene shape variation such as perspective distortion. As a result, it may be difficult to synthesize plausible regions if only translated patches are considered. Recent approaches address this problem by increasing the motion parameter search space to similarity transform with reflection and accommodate slight photometric variations [21, 17]. While the additional motion parameters do help when needed, the increased dimensionality and complexity render the nearest neighbor searching algorithm even harder to find a good solution. We handle this issue by constraining the transformation based on mid-level structural analysis of the image.

Missing regions in images could also be completed with the help of external image datasets. Hays and Efros [31] retrieve semantically similar images from a large dataset and copy a single large region to fill the missing pixels. A similar scene matching strategy was adopted by Zhang et al. [32]; the main difference is that they transfer the self-similarity field to guide the completion instead of the actual contents of the matched image. Another example of using external database is through instance-level matching methods to fill in missing regions via appropriate geometric and photometric transformation of the retrieved image [33]. In this work, we consider only the known region in the input image.

The notion of automatic guidance maps for image completion has been used in a number of approaches. For example, Jia et al. [34] inferred the line and contour continuation in the missing regions and used them for completion. A similar type of salient line matching was used in problems of tele-registration [35]. Kopf et al. [36] used tile-based search space constraints to constrain the source of texture examples in synthesis. He and Sun [18] identify a number of

representative translation offsets from the known region of the input image and use these offsets to complete the image. However, their method detects regularity only in the image plane and is thus not very effective on larger structures with significant perspective effects.

While it is desirable to have a fully automatic approach, the image completion technique may still fail on occasion because computer vision techniques are typically far from perfect. Interactive methods allow users to explicitly provide high-level expertise to guide the completion. User-specified constraints include label map [37], line structure continuation [38], perspective [39], lattice [40], and symmetry [9].

2.3 Overview

We implement as baseline algorithm the non-parametric optimization algorithm of Wexler et al. [19], and use random search and propagation as in PatchMatch [20]. We use two types of mid-level constraints of the scene to guide the low-level completion process: planar perspective and translational regularity. Given an image and a user-specified mask that specifies the region (or hole) to fill, we first detect multiple planes, estimate their perspective parameters, and determine their spatial supports within the scene (Section 2.4). To determine translational regularity within each plane, we perform feature matching using SIFT features [41]. The positions of all matched feature pairs are then affine rectified[2] using the corresponding plane parameters. This allows the dominant translational shifts to be easily detected through clusters of displacement vectors in the rectified domain.

We use the detected perspective planes and the translational regularity within each plane as soft constraints to guide the low-level image completion (Section 2.5). We achieve this by integrating these derived constraints as prior probabilities of the search space. The regularity detection step provides "positional" guidance, i.e., where the source patch should be copied from. In contrast, the plane orientation constraints provide "non-positional" guidance of source patches, i.e., how the source patch should be deformed. By incorporating both positional and non-positional constraints on searching source patches from the known region, we show that these two types of mid-level image analysis can significantly improve

[2]Affine rectification means mapping vanishing points to infinity so that parallel lines in 3D space project to parallel 2D lines in the rectified image.

the quality of the completed region in a semantically meaningful way.

2.4 Detecting Planar Surfaces and Regularity

In this section, we describe our analysis of the known image region to detect planar surfaces (Section 2.4.1) and translational regularity within these planes (Section 2.4.2). The results of this analysis will be used to constrain the low-level completion algorithm, as described in Section 2.5.

2.4.1 Planes

Many techniques have been proposed for identifying and rectifying planes [42, 43, 44], i.e., converting a perspectively distorted plane to a fronto-parallel version. We use a technique that involves line segment extraction, vanishing point estimation [45], and grouping based on vanishing points. Since this part of our algorithm is relatively standard, we provide only a brief description here. We first detect edges and fit line segments in the known region of the image. We then detect up to three vanishing points (VPs) using a RANSAC-based voting approach. This means we assume there are only up to three different plane orientations in the scene. This is reasonable for typical man-made structures. We show a sample result in Figure 2.3.

Given the three VPs, we can recover up to three plane orientations, one from each pair of the detected VPs. We compactly represent the parameters of plane m using the vanishing line $\mathbf{l}_\infty^m$ (the image of the line at infinity on the world plane connecting the two distinct VPs):

$$\mathbf{l}_\infty^m = [l_1^m, l_2^m, l_3^m]^\top. \tag{2.1}$$

Note that $\mathbf{l}_\infty^m$ is homogeneous and has two degrees of freedom. The perspective image of a plane can then be affine rectified (so that parallel lines in 3D appear parallel in the image) using a pure perspective transformation matrix

$$\mathbf{H}_m = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1^m & l_2^m & l_3^m \end{bmatrix}. \tag{2.2}$$

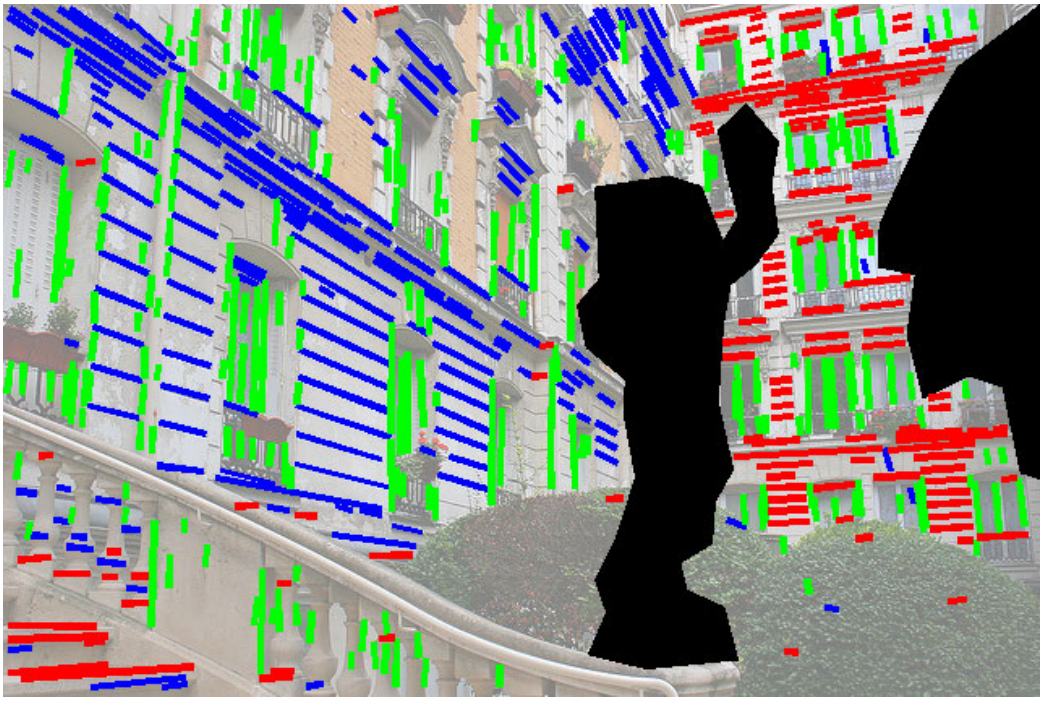


Figure 2.3: Vanishing point detection from a man-made environment. The red, green, and blue line segments correspond to three detected vanishing points, respectively.

However, the plane parameters provide no information on the spatial support of the plane in the image domain. While there are many computer vision algorithms available for automatic single-view reconstruction of man-made environment, they are usually quite sophisticated, e.g., see Barinova et al.'s work [46]. Instead, we address this problem via a rather simple and straightforward approach.

Our key insight is that a plane typically consists of two sets of parallel 3D lines. In other words, there are usually two sets of the line segments with two distinct VPs that should reside within the same image region. We identify the support of each plane by locating positions where the two sets of line segments corresponding to the two VPs overlap with each other.

We first estimate the spatial support of each VP by diffusing its corresponding line segments using a wide Gaussian kernel. Then, we estimate the spatial support for the planes by performing element-wise multiplication of its VP's support line density maps. These product maps have a high response where the two sets of the line segments overlapped with each other. Note that we always add the fronto-parallel plane with parameters $l_\infty^0 = [0, 0, 1]^\top$ and assign a fixed density value

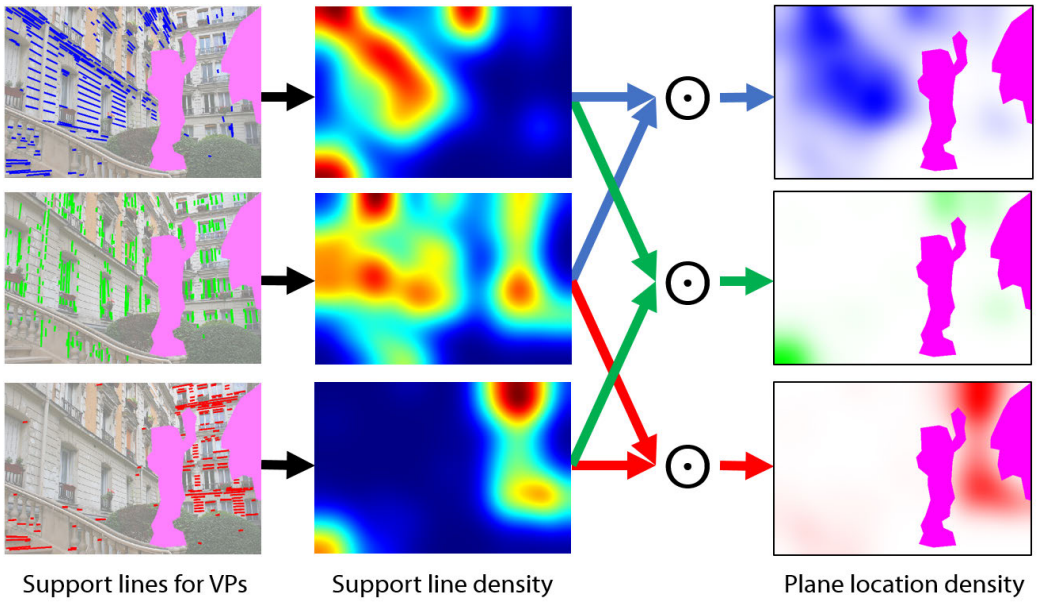


Figure 2.4: Plane localization in the known region using support line segments from pairs of vanishing points. In the hole region we assign plane probabilities in a different manner, as shown in Figure 2.5.

10^{-5} uniformly across the image. We then perform *per-pixel* normalization of this density product map so that the sum over the plane membership probability is 1; we call this the "posterior probability" $\Pr[m|\boldsymbol{x}]$ for assigning plane membership m at pixel $\boldsymbol{x}$. This process is illustrated in Figure 2.4. Here, the posterior probability distributions are shown as color-coded density maps on the right column (note that the density map for the fronto-parallel plane is not shown).

As lines can only be detected in the known region of the image, the posterior probabilities within the unknown region are highly unreliable. To address this problem, we assign to every missing pixel the probabilities of the closest boundary pixel. The posterior probability map of an example image is shown in Figure 2.5.

2.4.2 Regularity extraction

Regular and near-regular structures are ubiquitous in man-made environments as well as in many natural objects. Detection of such regularity has been shown to be a compact means for understanding scene structure. Liu et al. [47] provide a good survey of work in this area.

Similar to He and Sun [18], we also detect translational regularity using offsets of matched image features. However, we detect regularity in a localized manner

Figure 2.5: Visualization of plane posterior probability in the known region, and propagation into the hole region. The posterior probabilities of the three recovered planes are shown in blue, green and red. (Note that the fronto-parallel plane is not shown here.) The boundary pixels between the known and unknown regions are shown in white. In the hole region every pixel is assigned the plane probabilities of the nearest boundary pixel.

and in affine rectified space in order to account for possibly multiple foreshortened planes. We begin with detecting standard difference of Gaussians feature points in the known image region and compute the SIFT descriptors for each feature point [41]. We choose to extract features in the original image rather than rectified space because the rectification would severely distort the image for slanted planes (e.g., rectifying the ground plane in the middle image in Figure 2.1 would lead to extreme distortions near the horizon). We compute the two nearest neighbors for each feature using a kd-tree. We only retain matches whose ℓ_2 feature distances are below a threshold of 0.1.

Next, for each plane m, we extract all feature matches, where both feature positions have a high posterior probability $\Pr[m|\boldsymbol{x}]$ (defined in Section 2.4.1). Specifically, we check if the product of two posterior probabilities (from two detected feature positions) exceeds 0.5.

Repetitive structures in man-made environments are usually equidistant in 3D.

However, the equal spacing is not preserved in image space (and, hence, in our feature matches), due to perspective distortion. We undo this distortion by affinely rectifying the positions of the matched feature points. The displacement of two rectified feature points is now spatially invariant, and consequently, we can detect translational repetition: if certain regular structures exist, these displacement vectors form a dense cluster in the 2D affine rectified space. We use the mean-shift algorithm [48] to detect these modes (setting the bandwidth parameter to 10 pixels, and rejecting spurious modes with fewer than 10 members). We denote the set of the modes as $\mathscr{D}_m = \{\mathbf{d}_i\}$, where $\mathbf{d}_i \in R^2$ is the displacement vector in the rectified space.

Figure 2.6 (c-d) shows detected modes in both rectified (left) and axis-aligned space (right). In (e-f) we also show their positions relative to a target patch in the image (white square). This figure highlights the importance of having perspective correction in computing the displacement vectors. In addition to the accurate position suggestion, the plane parameters explicitly provide how the source patches should deform spatially. The recovered candidates, outlined in blue, have to be significantly deformed to match with the image axis aligned target patch, in white. It is difficult to recover such geometric transformations using low-level algorithms alone.

He and Sun [18] have shown that regularity using statistics of matched patch offsets can be helpful in the context of image completion. However, they assume global translational regularity in the image plane, i.e., they assume a single fronto-parallel surface. Detection and optimization are both done in image space. For our technique, while the detection is in rectified affine space, the objective function is optimized in image space using constrained homographies (as described in the next section).

Our regularity detection handles more general scenes because we deal with each plane independently (but with soft membership). As a result, we are able to detect different repetitive structures on multiple building facades with different orientations. Given the major differences, He and Sun's technique will have to be substantially modified to work on such scenes.

2.5 Guided Image Completion

In this section, we describe how the detected planes and extracted regularity from the previous section are used to guide the low-level image completion algorithm. We build on Wexler et al.'s algorithm [19] using random search and propagation as in PatchMatch [20]. Please refer to these papers for details on the base algorithm.

We incorporate sampling from planes by modifying the patch distance function (Sections 3.4.1-2.5.3), and the regularity by modifying the random sample generation (Section 2.5.4).

2.5.1 Objective function

We augment the image completion objective function in two ways: First, we augment the patch distance (called "coherence measure" in the original paper [19]) by including a guidance term. Second, we augment the search space by the plane index, which determines the patch transformation.

The objective function takes the form

$$\min_{\{\mathbf{t}_i,\mathbf{s}_i,m_i\}} \sum_{i\in\overline{\Omega}} E_{\text{color}}(\mathbf{s}_i,\mathbf{t}_i,m_i) + E_{\text{guide}}(\mathbf{s}_i,\mathbf{t}_i,m_i), \tag{2.3}$$

where Ω and $\overline{\Omega}$ are the sets of known and unknown pixel indices, $\mathbf{t}_i = (t_i^x, t_i^y)^\top$ is the center position of a target patch in $\overline{\Omega}$, $\mathbf{s}_i = (s_i^x, s_i^y)^\top$ is the center position of the corresponding source patch in Ω, and m_i the plane index of an unknown target patch $\mathbf{t}_i$. The two terms E_{color} and E_{guide} are the appearance and guidance terms, respectively, which together make up the patch distance.

Note that the target patches are image-aligned with no geometric transformation such as scaling or rotation, while the source patches have a geometric transform that is implicitly derived from geometry of the plane they are sampled from. The geometric transform is described in the next section.

2.5.2 Appearance cost

Our appearance cost is the sum of the absolute values of two sampled patches in the RGB space:

$$E_{\text{color}}(\mathbf{s}_i,\mathbf{t}_i,m_i) = \left\| q(\mathbf{s}_i,\mathbf{t}_i,m_i) - p(\mathbf{t}_i) \right\|_1, \tag{2.4}$$

where $p(\mathbf{t}_i)$ denotes the 7×7 patch sampled around the center position $\mathbf{t}_i$ and $q(\mathbf{s}_i, \mathbf{t}_i, m_i)$ denotes the sampled patch centered at $\mathbf{s}_i$ with geometric transformation subject to the plane orientation defined by target patch position $\mathbf{t}_i$ and the plane parameter of plane m_i.

Most prior approaches use pure translational patches [19, 29] or explicitly search geometric transformations, e.g., rotation, scale, and flip [21, 17]. Instead, we sample patches using homographies. Rather than searching for all parameters of the homography, we derive it implicitly from the combination of the coordinates $\mathbf{s}_i, \mathbf{t}_i$ with their corresponding plane index m_i.

We first compute the transformation that maps a 7×7 patch at $\mathbf{t}_i$ to transformed patch sampled at $\mathbf{s}_i$. Let $\tilde{\mathbf{t}}_i = [t_i^x, t_i^y, 1]^\top$ and $\tilde{\mathbf{s}}_i = [s_i^x, s_i^y, 1]^\top$ as homogenous representations of $\mathbf{t}_i$ and $\mathbf{s}_i$, respectively. Let $\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3$ be the row vectors of $\boldsymbol{H}_{m_i}$. The source and target patch positions in the affine rectified space are computed as:

$$\tilde{\mathbf{t}}_i' = \left[\boldsymbol{h}_1\tilde{\mathbf{t}}_i,\ \boldsymbol{h}_2\tilde{\mathbf{t}}_i,\ \boldsymbol{h}_3\tilde{\mathbf{t}}_i\right]^\top, \tag{2.5}$$

$$\tilde{\mathbf{s}}_i' = \left[\boldsymbol{h}_1\tilde{\mathbf{s}}_i,\ \boldsymbol{h}_2\tilde{\mathbf{s}}_i,\ \boldsymbol{h}_3\tilde{\mathbf{s}}_i\right]^\top. \tag{2.6}$$

We define (d^x, d^y) as the displacement vector from target to source patch positions in the rectified space. The term $\tilde{\mathbf{s}}_i$ is represented as

$$\tilde{\mathbf{s}}_i' = \begin{bmatrix} \boldsymbol{h}_1 + \boldsymbol{h}_3 d^x \\ \boldsymbol{h}_2 + \boldsymbol{h}_3 d^y \\ \boldsymbol{h}_3 \end{bmatrix} \tilde{\mathbf{t}}_i. \tag{2.7}$$

By applying the inverse of the rectifying matrix $\mathbf{H}_{m_i}^{-1}$, we have

$$\tilde{\mathbf{s}}_i = \mathbf{H}_{m_i}^{-1}\tilde{\mathbf{s}}_i' = \mathbf{H}_{m_i}^{-1} \begin{bmatrix} \boldsymbol{h}_1 + \boldsymbol{h}_3 d^x \\ \boldsymbol{h}_2 + \boldsymbol{h}_3 d^y \\ \boldsymbol{h}_3 \end{bmatrix} \tilde{\mathbf{t}}_i. \tag{2.8}$$

To get the motion parameters of the patch around $\mathbf{s}_i$ (i.e., factoring out the dependency of $\mathbf{t}_i$), we apply a translation matrix with offset $\mathbf{t}_i$:

$$\tilde{\mathbf{s}}_i = \boldsymbol{H}_{m_i}^{-1} \begin{bmatrix} \boldsymbol{h}_1 + \boldsymbol{h}_3 d^x \\ \boldsymbol{h}_2 + \boldsymbol{h}_3 d^y \\ \boldsymbol{h}_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & t_i^x \\ 0 & 1 & t_i^y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \boldsymbol{T}_{\mathbf{s}_i} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \tag{2.9}$$

where $\boldsymbol{T}_{\mathbf{s}_i}$ compactly represents the domain transformation of the sampled source patch. Note that in the special case of fronto-parallel plane ($\boldsymbol{H}_0 = \boldsymbol{I}_3$), $\boldsymbol{T}_{\mathbf{s}_i}$ reduces to a translation matrix with offset (s_i^x, s_i^y).

2.5.3 Guidance cost

Our guidance cost includes three constraints derived from the analysis stage:

$$\begin{aligned} E_{\text{guide}}(\mathbf{s}_i, \mathbf{t}_i, m_i) \; &= \lambda_1 E_{\text{plane}}(\mathbf{s}_i, \mathbf{t}_i, m_i) + \lambda_2 E_{\text{direction}}(\mathbf{s}_i, \mathbf{t}_i, m_i) + \\ &\quad \lambda_3 E_{\text{proximity}}(\mathbf{s}_i, \mathbf{t}_i), \end{aligned} \tag{2.10}$$

where $\lambda_1 = 10$, $\lambda_2 = 10^3$, and $\lambda_3 = 1$ are the weighting parameters for plane compatibility, orthogonal direction, and proximity cost, respectively. Next, we describe each of these constraints in detail.

Plane compatibility. In the analysis stage, we computed the posterior probability map $\Pr[m_i|\boldsymbol{x}]$ for assigning plane membership m_i for position located at $\boldsymbol{x}$. We directly convert this into a penalty term using the negative log-likelihood. Specifically,

$$E_{\text{plane}}(\mathbf{s}_i, \mathbf{t}_i, m_i) = -\log \Pr[m_i|\mathbf{s}_i] - \log \Pr[m_i|\mathbf{t}_i], \tag{2.11}$$

i.e., the term encourages sampling from a plane that has a high probability both in the source and target location.

Orthogonal direction cost. Urban scenes often consist of repetitive structures along horizontal and vertical directions, e.g., windows on a building facade. This term encourages using source patches located on either one of the orthogonal directions. Note that affine rectification makes the support lines for each VP parallel; however, the lines from the two VPs are not necessarily orthogonal to each other. We estimate the rotation angle that maps the set of line segments for each VP to align with the horizontal axis. This mapping is denoted as $\mathbf{H}^1_{m_i}$ and $\mathbf{H}^2_{m_i}$ for the two VPs defining the plane m_i:

$$\mathbf{H}^j_{m_i} = \begin{bmatrix} cos(\theta_j) & -sin(\theta_j) & 0 \\ sin(\theta_j) & cos(\theta_j) & 0 \\ l_1^{m_i} & l_2^{m_i} & l_3^{m_i} \end{bmatrix}. \tag{2.12}$$

We define the orthogonal direction cost as a truncated L1-norm:

$$E_{\text{direction}}(\mathbf{s}_i, \mathbf{t}_i, m_i) = \psi\big(\min(|\boldsymbol{H}^1_{m_i}(\tilde{\mathbf{s}}_i)^y - \boldsymbol{H}^1_{m_i}(\tilde{\mathbf{t}}_i)^y|, \\ |\boldsymbol{H}^2_{m_i}(\tilde{\mathbf{s}}_i)^y - \boldsymbol{H}^2_{m_i}(\tilde{\mathbf{t}}_i)^y|)\big), \tag{2.13}$$

where $\psi(z) = min(|z|, c)$ is the function that caps the cost to a constant $c = 0.02$. To ensure that the cost is invariant to the scale of the image, we divide the distances in y-axis in the rectified space by the largest image dimension. For cases of target patches with no available source samples on the both directions, this constraint has no effect on searching of the source patch because it is constant. We visualize the directional cost given a target patch in Figure 2.7.

Proximity cost. It has been shown by Kopf et al. [36] that constraining the search space to nearby regions can improve the synthesis result. In addition to the above mid-level constraints, we also introduce a low-level search space constraint which favors nearby source patches for completion. This constraint implicitly avoids copying patches from extremely different scales. We define the proximity cost as

$$E_{\text{proximity}}(\mathbf{s}_i, \mathbf{t}_i) = \frac{||\mathbf{s}_i - \mathbf{t}_i||_2^2}{\sigma_d(\mathbf{t}_i)^2 + \sigma_c^2}, \tag{2.14}$$

where $\sigma_d(\mathbf{t}_i)^2$ is the squared distance of target position to the nearest border to the known region and $\sigma_c^2 = (W/8)^2$ is the parameter for adjusting the strength of the proximity constraint (W is the largest image dimension).

2.5.4 Structure guided sampling and propagation

We extend the random location sampling in the PatchMatch algorithm [20] to incorporate our computed plane probabilities and translational regularity. In addition to the regular random location sampling, we also sample from the clustered regularity modes computed in Section 2.4.2. We did not include the regularity as a prior term in the previous sections because the detection is sometimes not reliable. Using regularity instead for random location sample generation provides a more robust way of incorporating this constraint. In our implementation of PatchMatch, we use 5 iterations of *plane probability guided* sampling and *regularity guided* sampling (as described below) in the search and propagation stage.

Plane probability guided sampling. For a given target patch $\mathbf{t}_i$, we first sample the plane index m_i according to the posterior probability $\Pr[m_i|\mathbf{t}_i]$. Then, we sample $\mathbf{s}_i$ by drawing random samples from $\Pr[\mathbf{s}_i|m_i]$ using rejection sampling. This effectively biases the search space toward finding the correct patches from the same plane.

Regularity guided sampling. While the plane probability guided random sampling scheme samples from the right plane, it does not impose constraints on where in the plane it should sample from. This usually leads to visible artifacts when regular structures are present. Sampling from our detected regularity modes alleviates this problem.

For each target patch $\mathbf{t}_i$, we first draw a plane index m_i as above, then we randomly draw one displacement in rectified space from $\mathscr{D}_m$. Using the target patch position $\mathbf{t}_i$, the plane index m_i, and the displacement in the rectified space, we can then directly compute the candidate positions and their geometric transformations. Examples of candidate source patches are shown in Figure 2.6. This regularity guided sampling scheme greatly improves the completion quality when repetitive structures exist.

2.6 Results

We compare our results against several state-of-the-art image completion algorithms. Specifically, we choose Photoshop Content Aware Fill [20, 19], He and Sun's method [18], and Image Melding [17]. All of these methods use frontoparallel translational patches, except Image Melding, which allows similarity transformation and flip of patches.

2.6.1 Comparison with the state-of-the-art methods

In Figure 2.8, we show a series of comparisons on challenging scenes. In the first row, the building consists of near regular structures. We can see that the competing algorithms fail to synthesize such large structure because they only minimize localized texture energy without considering a global consistency. Our method, on the other hand, fills in the hole with repetitive pattern similar to the

known region. In addition, with the recovered plane orientation, our synthesized result is physically plausible.

In the second row, we show a single planar building facade with regular patterns. Even with only mild perspective distortion, translational patches are insufficient to synthesize the foreshortening effect and thus result in broken line structures. Image Melding, while theoretically equipped with the ability to apply appropriate scaling of the patches, fails to find such solution in high-dimensional space. Our algorithm effectively uses the plane constraint to extend the facade with minimally visible artifacts. The results on the 3rd to 6th rows show that our algorithm is *not* limited to ideal piecewise plane scenes with homogeneous textures. With the plane support detection and the weighting scheme, we leverage the low-level algorithm to find a good transition boundary between one structure and another. Examples illustrating good transition boundaries are the stairs regions and the pure texture region around the tree in the fourth row, and multiple unknown surface discontinuities in the fifth row. In the last row, we demonstrate the effectiveness of combining plane constraints and regularity-guided sampling.

From these examples of realistic scenes, we can see that our image completion algorithm is robust to deviations from perfectly textured planar surfaces. In other words, our completion algorithm does not require perfect plane orientation recovery, support estimation, segmentation, and symmetry detection. In fact, the analysis in many regions contains errors because vision algorithms are far from perfect. However, as exemplified here, by combining a powerful low-level algorithm with mid-level constraints, we are able to extend the state-of-the-art in image completion.

2.6.2 Comparisons on Natural Scenes

For images of natural scenes our analysis usually does not detect any planes because there are no reliable features to detect planes and translational regularity. In such cases our algorithm automatically reverts to the baseline image completion algorithm, i.e., our implementation of Wexler et al.'s algorithm [19]. Four such examples are shown in Figure 2.9. We compare to the unguided version of our completion algorithm (fourth column) to validate that our result looks visually similar to the baseline.

2.6.3 Failure modes

We used relatively simple algorithms in our image analysis stages, which can fail to detect vanishing points or plane regularities, or more severely, return false positives. In the former case our algorithm just reverts to fronto-parallel completion, while the latter case might lead to some artifacts. The performance of the analysis stage could likely be improved using more sophisticated computer vision methods, which we leave to future work.

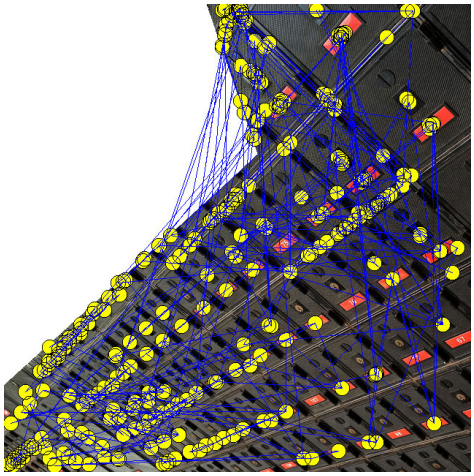
The first two rows in Figure 2.10 demonstrate the difficulty of finding demarcation lines between different perspective planes when the unknown region is large. The results in the third row shows that the falsely detected plane may over-constrain the patch synthesis and lead to poor results near the bushes. Notice, though, that the competing techniques also fail to generate satisfactory results.

2.7 Concluding Remarks

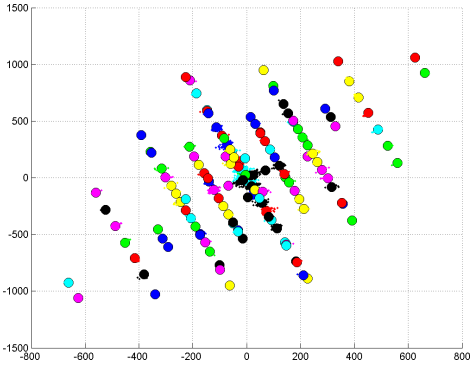
We have presented an automatic image completion algorithm that exploits extracted mid-level scene structures for guiding the low-level completion. Our algorithm detects multiple planes and their corresponding translational regularity. These constraints are incorporated into the augmented patch distance and the sampling scheme. In the absence of reliable plane detection, our algorithm automatically reverts to a baseline completion algorithm. We demonstrated that our method consistently outperforms state-of-the-art image completion algorithms for a wide range of challenging scenes.

Historically, conventional statistical texture synthesis methods formulate texture synthesis as an "analysis then synthesis" framework. However, this type of framework has been mostly set aside due to the simplicity and the effectiveness of example-based methods. Our method demonstrates the benefit of and the need for image analysis. We show that the quality of image completion can be significantly improved by striking a balance between analysis and synthesis.

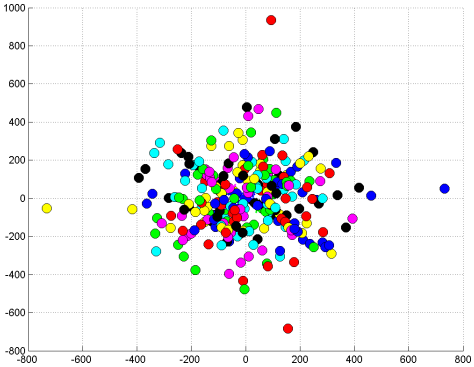
(a) Input image



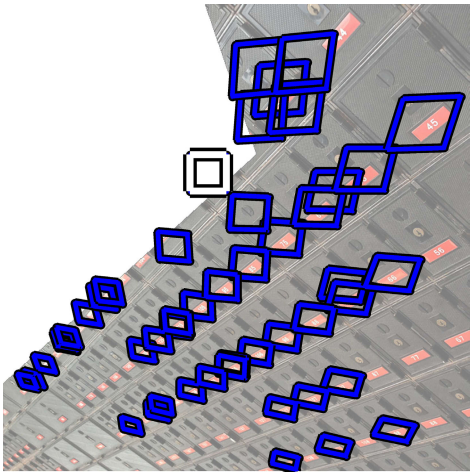
(b) Matched features



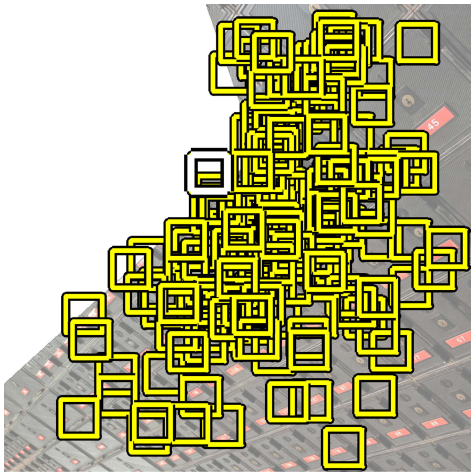
(c) Rectified space modes



(d) Fronto-parallel modes



(e) Rectified displacements



(f) Fronto-parallel displacements

Figure 2.6: Detecting regularity from modes of displacement vectors between matched features. (a) and (b) show the input image and matched features. (c) and (e) (on the left) show the detected modes and a visualization of candidate source patch locations and shapes for a single target patch (in white) in the *affine rectified space*. The repetitive structure is clearly revealed. (d) and (f) show the corresponding illustrations for the fronto-parallel space. Here, the displacements are scattered and do not reveal any structure.

Figure 2.7: Visualization of the directional cost given a target patch (shown in white). The dark regions indicate lower costs. The directional cost encourages finding matches along the two dominant, orthogonal directions, leading to semantically more plausible completion results.

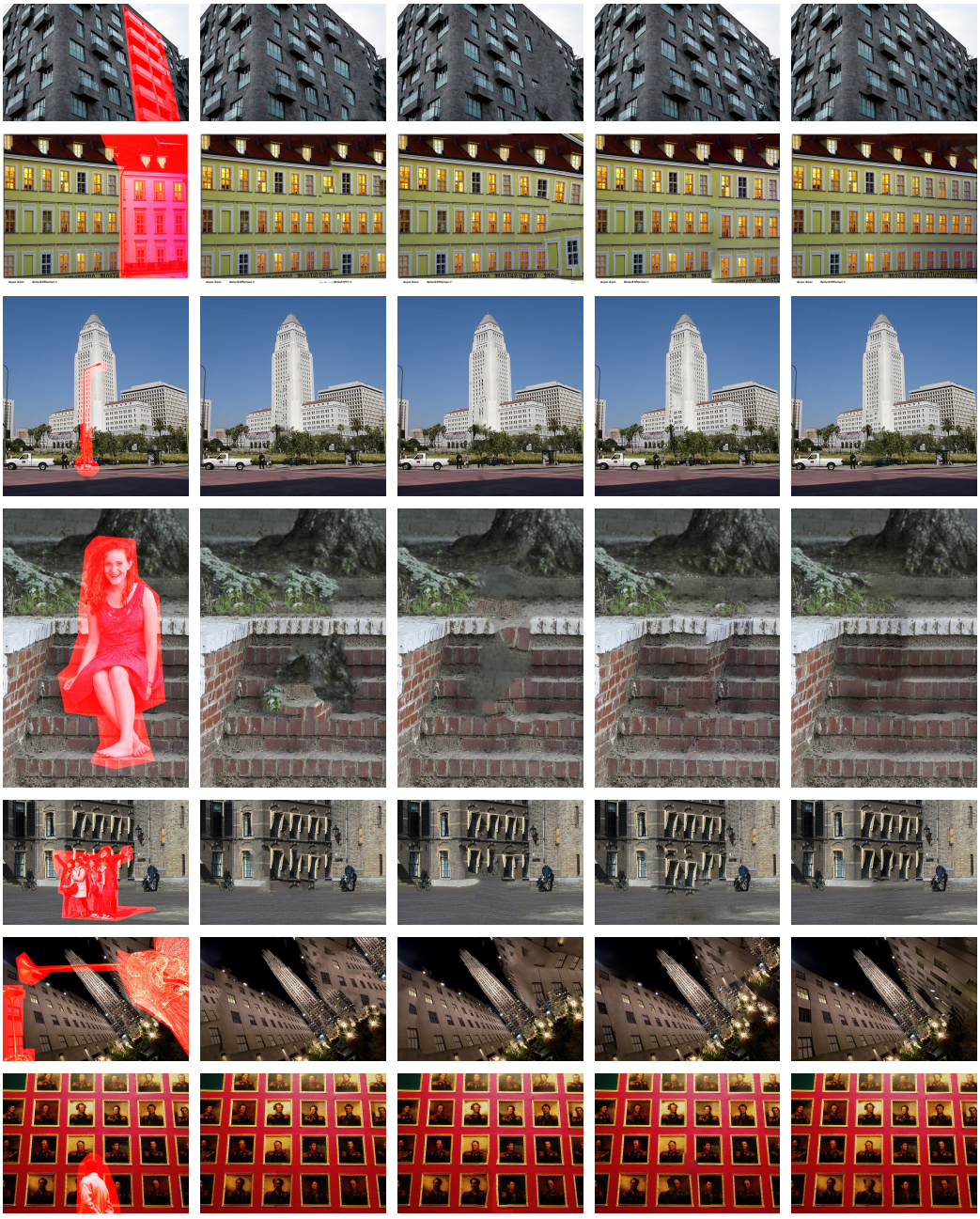
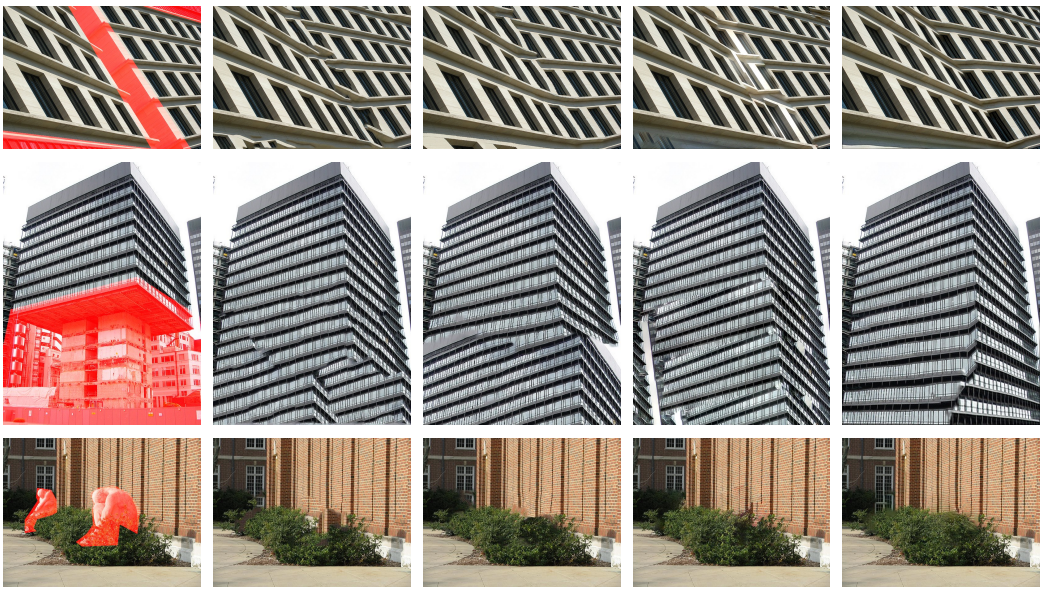


Figure 2.8: Comparisons with representative state-of-the art algorithms. Image credits: Flickr users Daniel Foster, Moyan Brenn, Savannah Roberts, Remon Rijper, Chris Ford, and marie-ll.

Input & hole | Photoshop | Image Melding [17] | Our result (unguided) | Our result (guided)

Figure 2.9: Image completion on images of natural scenes.



Input & hole | Photoshop | Image Melding [17] | [18] | Our result

Figure 2.10: Failure examples. First two rows: our algorithm has difficulty in finding the good demarcation lines when the missing regions are too large. The last row: our method may overconstrain the patch synthesis with falsely detected planes, which leads to artifacts near the bushes. Image credits: Flickr users Reto Fetz, David Barrie, and brokenthoughts.

CHAPTER 3

SINGLE IMAGE SUPER-RESOLUTION WITH TRANSFORMED SELF-EXEMPLARS

3.1 Introduction

The single image super-resolution (SR) problem is fundamentally under-constrained because a large number of high-resolution (HR) pixels need to be estimated from many fewer ones available in a given low-resolution (LR) image. Many SR algorithms address this problem by exploiting various image priors for regularization, including the simple smoothness assumption or sophisticated statistical priors learned from large collections of natural images.

Most modern single image super-resolution (SR) methods rely on machine learning techniques. These methods focus on learning the relationship between low-resolution (LR) and high-resolution (HR) image patches. A popular class of such algorithms uses an external database of natural images as a source of LR-HR training patch pairs. Existing methods have employed various learning algorithms for learning this LR to HR mapping, including nearest neighbor approaches [49], manifold learning [50], dictionary learning [51], locally linear regression [52, 53, 54], and convolutional networks [55].

However, methods that learn LR-HR mapping from external databases have certain shortcomings. The number and type of training images required for satisfactory levels of performance are not clear. Large-scale training sets are often required to learn a sufficiently expressive LR-HR dictionary. For every new scale factor by which the resolution has to be increased, or SR factor, these methods need to re-train the model using sophisticated learning algorithms on large external datasets.

To avoid using external databases and their associated problems, several approaches exploit internal patch redundancy for SR [56, 57, 58, 59]. These methods are based on the fractal nature of images [60], which suggests that patches of a natural image recur within and across scales of the same image. Internal LR-HR patch database can be built using the scale-space pyramid of the given image itself. Internal dictionaries have been shown to contain more *relevant* training patches, as compared to external dictionaries [61].

While internal statistics have been successfully exploited for SR, in most algorithms the LR-HR patch pairs are found by searching only for "translated" versions of patches in

Figure 3.1: Examples of self-similar patterns deformed due to local shape variation, orientation change, or perspective distortion.

the scaled down images. This effectively assumes that an HR version of a patch appears in the same image at the desired scale, orientation and illumination. This amounts to assuming that the patch is planar and the images of the different assumed occurrences of the patch are taken by a camera translating parallel to the plane of the patch. This fronto-parallel imaging assumption is often violated due to the non-planar shape of the patch surface, common in both natural and man-made scenes, as well as perspective distortion. Figure 3.1 shows three examples of such violations, where self-similarity across scales will hold better if suitable geometric transformation of patches is allowed

In this chapter, we propose a self-similarity driven SR algorithm that expands the internal patch search space. First, we explicitly incorporate the 3D scene geometry by localizing planes, and use the plane parameters to estimate the perspective deformation of recurring patches. Second, we expand the patch search space to include affine transformation to accommodate potential patch deformation due to local shape variations. We propose a compositional transformation model to simultaneously handle these two types of transformations. We modify the PatchMatch algorithm [20] to efficiently solve the nearest neighbor field estimation problem. We validate our algorithm through a large number of qualitative and quantitative comparisons against state-of-the-art SR algorithms on a variety of scenes. We achieve significantly better results for man-made scenes containing regular structures. For natural scenes, our results are comparable with current state-of-the-art algorithms.

Our Contributions:

1. Our method effectively increases the size of the limited internal dictionary by allowing geometric transformation of patches. We achieve state-of-the-art results without using any external training images.

2. We propose a decomposition of the geometric patch transformation model into (i) perspective distortion for handling structured scenes and (ii) additional affine transformation for modeling local shape deformation. This allows us to adaptively

and efficiently take advantage of scene geometry when it is present.

3. We use and make available a new dataset of urban images containing structured scenes as a benchmark for SR evaluation.

3.2 Related Work

The core of image SR algorithms has shifted from interpolation and reconstruction [62] to learning and searching for best matching existing image(s) as the HR map of the given LR image. We limit our discussion here to these more current learning-based approaches and classify the corresponding algorithms into two main categories: external and internal, depending on the source of training patches.

External database driven SR: These methods use a variety of learning algorithms to learn the LR-HR mapping from a large database of LR-HR image pairs. These include nearest neighbor [49], kernel ridge regression [63], sparse coding [51, 64, 65, 7], manifold learning [50] and convolutional neural networks [55]. The main challenges lie in how to effectively model the patch space. As opposed to learning a global mapping over the entire dataset, several methods alleviate the complexity of data modeling by partitioning or pre-clustering the external training database, so that relatively simpler prediction functions could be used for performing the LR-HR mapping in each training cluster [52, 53, 54]. Instead of learning in the 2D patch domain, some methods learn how 1D edge profiles transform across resolutions [66, 67]. Higher-level features have also been used in [68, 69, 70] for learning the LR-HR mapping. In contrast, our algorithm has the advantage of neither requiring external training databases, nor using sophisticated learning algorithms.

Internal database driven SR: Among internal database driven SR methods, Ebrahimi and Vrscay [56] combined ideas from fractal coding [60] with example-based algorithms such as non-local means filtering [71], to propose a self-similarity based SR algorithm. Glasner et al. [57] unified the classical and example-based SR by exploiting the patch recurrence within and across image scales. Freedman and Fattal [58] showed that self-similar patches can often be found in limited spatial neighborhoods, thereby gaining computational speed-up. Yang et al. [72] refined this notion further to seek self-similar patches in extremely localized neighborhoods (in-place examples), and performed first-order regression on them. Michaeli and Irani [73] used self-similarity to jointly recover the blur kernel and the HR image. Singh et al. [74] used the self-similarity principle for super-resolving noisy images.

Expanding patch search space: Since internal dictionaries are constructed using only the given LR image, they tend to contain a much smaller number of LR-HR patch pairs compared to external dictionaries which can be as large as desired. Singh and Ahuja used

orientation selective sub-band energies for better matching textural patterns [75] and later reduced the self-similarity based SR into a set of problems of matching simpler sub-bands of the image, amounting to an exponential increase in the effective size of the internal dictionary [59]. Zhu et al. [76] proposed to enhance the expressiveness of the dictionary by optical flow based patch deformation during searching, to match the deformed patch with images in external databases. We use projective transformation to model the deformation common in urban scenes to better exploit internal self-similarity. Fernandez-Granda and Candes [77] super-resolved planar regions by factoring out perspective distortion and imposing group-sparse regularization over image gradients. Our method also incorporates 3D scene geometry for SR, but we can handle multiple planes and recover regular textural patterns beyond orthogonal edges through self-similarity matching. In addition, our method is a generic SR algorithm that handles both man-made and natural scenes in one framework. In the absence of any detected planar structures, our algorithm automatically falls back to searching only affine transformed self-exemplars for SR.

Our work is also related to several recent approaches that solve other low-level vision problems using over-parameterized (expanded) patch search spaces. Although more difficult to optimize than 2D translation, such over-parametrization often better utilizes the available patch samples by allowing transformations. Examples include stereo [78], depth upsampling [79], optical flow [80], image completion [1], and patch-based synthesis [17]. Such expansion of the search space is particularly suited for the SR problem due to the limited size of internal dictionaries.

3.3 Overview

Super-resolution scheme: Given a LR image I, we first blur and subsample it to obtain its downsampled version I_{D}. Using I and I_{D}, our algorithm to obtain an HR image I_{H} consists of the following steps:

1) For each patch P (target patch) in the LR image I, we compute a transformation matrix $\mathbf{T}$ (homography) that warps P to its best matching patch Q (source patch) in the downsampled image I_{D}, as illustrated in Figure 3.2 (c). To obtain the parameters of such a transformation, we estimate a nearest neighbor field between I and I_{D} using a modified PatchMatch algorithm [20] (details given in Section 3.4).

2) We then extract Q_{H} from the image I, which is the HR version of the source patch Q.

3) We use the inverse of the computed transformation matrix $\mathbf{T}$ to 'unwarp' the HR patch Q_{H}, to obtain the self-exemplar P_{H}, which is our estimated HR version of the target patch P. We paste P_{H} in the HR image I_{H} at the location corresponding to the LR patch P.

4) We repeat the above steps for all target patches to obtain an estimate of the HR image

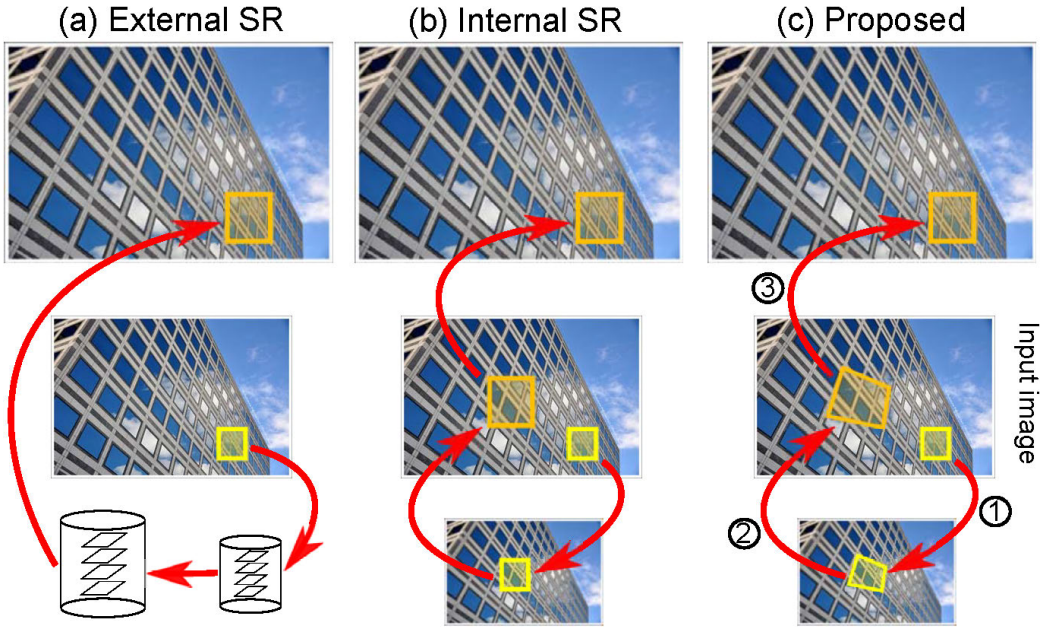


Figure 3.2: Comparison with external dictionary and internal dictionary (self-similarity) approaches. Middle row: Given LR image I. Our method allows for geometrically transforming the target patch from the input image, while searching for its nearest neighbor in the downsampled image. The HR version of the best match found is then pasted on to the HR image. This is repeated for all patches in the input image I.

I_{H}.

5) We run the iterative backprojection algorithm [62] to ensure that the estimated I_{H} satisfies the reconstruction constraint with the given LR observation I.

Figure 3.2 schematically illustrates the important steps in our algorithm, and compares it with other frameworks.

Motivation for using transformed self-exemplars: The key step in our algorithm is the use of the transformation matrix $\mathbf{T}$ that allows for geometric deformation of patches, instead of simply searching for the best patches under translation. We justify the use of transformed self-exemplars with two illustrative examples in Figure 3.3. Matching using the affine transformation and planar perspective transformation achieves both lower matching errors and more accurate prediction of the HR content than matching patches under translation.

3.4 Nearest Neighbor Field Estimation

3.4.1 Objective function

Let Ω be the set of pixel indices of the input LR image I. For each target patch $P(\mathbf{t}_i)$ centered at position $\mathbf{t}_i = (t_i^x, t_i^y)^\top$ in I, our goal is to estimate a transformation matrix $\mathbf{T}_i$ that maps the target patch $P(\mathbf{t}_i)$ to its nearest neighbor in the downsampled image I_{D}. A dense nearest neighbor patch search forms a nearest-neighbor field (NNF) estimation problem. In contrast to the conventional 2D translation (or offsets) field, here we have a field of *transformations* parametrized by θ_i for i^{th} pixel in the input LR image. Our objective function for this NNF estimation problem takes the form

$$\min_{\{\theta_i\}} \sum_{i \in \Omega} \mathbf{E}_{\text{app}}(\mathbf{t}_i, \theta_i) + \mathbf{E}_{\text{plane}}(\mathbf{t}_i, \theta_i) + \mathbf{E}_{\text{scale}}(\mathbf{t}_i, \theta_i), \tag{3.1}$$

where θ_i is the unknown set of parameters for constructing the transformation matrix $\mathbf{T}_i$ that we need to estimate (in a way explained later). Our objective function includes three costs: (1) appearance cost, (2) plane cost, and (3) scale cost. Below we first describe each of these costs.

Appearance cost $\mathbf{E}_{\text{app}}$: This cost measures similarity between the sampled target and source patches. We use Gaussian-weighted sum-of-squared distance in the RGB space as our metric:

$$\mathbf{E}_{\text{app}}(\mathbf{t}_i, \theta_i) = ||\boldsymbol{W}_i \left(P(\mathbf{t}_i) - Q(\mathbf{t}_i, \theta_i)\right)||_2^2, \tag{3.2}$$

where the matrix $\boldsymbol{W}_i$ is the Gaussian weights with $\sigma^2 = 3$, and $Q(\mathbf{t}_i, \theta_i)$ denotes the sampled patch from I_{D} using the transformation $\mathbf{T}_i$ with parameter θ_i.

We now present how we design and construct the transformation matrix $\mathbf{T}_i$ from estimated parameter θ_i for sampling the source patch $Q(\mathbf{t}_i, \theta_i)$. The geometric transformation of a patch in general can have up to 8 degrees of freedom (i.e., a projective transformation). One way to estimate the patch geometric transformation is to explicitly search in the additional patch space (e.g., scale, rotation) [81, 82, 17] beyond translation. However, perspective distortion can only be approximated by scaling, rotation and shearing of affine transformations. Therefore, affine transformations by themselves are less effective in modeling the appearance variations in man-made, structured scenes. Huang et al. [1] addressed this problem by detecting planes (and their parameters) and using them to determine the perspective transformation between the target and source patch. In Figure 3.4, we show a visualization of vanishing point detection and posterior probability map for detection of planes, as yielded by [1].

In this chapter, we combine the explicit search strategy of [81, 82, 17], along with the perspective deformation estimation approach of [1]. Using the algorithm of [1],[1] we detect and localize planes and compute the planar parameters, as shown by the example in Figure 3.4. We propose to parameterize $\mathbf{T}_i$ by $\theta_i = (\mathbf{s}_i, m_i)$, where $\mathbf{s}_i = (\mathbf{s}_i^x, \mathbf{s}_i^y, \mathbf{s}_i^s, \mathbf{s}_i^\theta, \mathbf{s}_i^\alpha, \mathbf{s}_i^\beta)$ is the 6-D affine motion parameter of the source patch and m_i is the index of detected plane (using [1]). We propose a factored geometric transformation model $\mathbf{T}_i(\theta_i)$ of the form:

$$\mathbf{T}_i(\theta_i) = \mathbf{H}\left(\mathbf{t}_i, \mathbf{s}_i^x, \mathbf{s}_i^y, m_i\right) \mathbf{S}\left(\mathbf{s}_i^s, \mathbf{s}_i^\theta\right) \mathbf{A}\left(\mathbf{s}_i^\alpha, \mathbf{s}_i^\beta\right), \tag{3.3}$$

where the matrix $\mathbf{H}$ captures the perspective deformation given the target and source patch positions and the planar parameters (as described in [1]). The matrix

$$\mathbf{S}\left(\mathbf{s}_i^s, \mathbf{s}_i^\theta\right) = \begin{bmatrix} \mathbf{s}_i^s \mathbf{R}(\mathbf{s}_i^\theta) & \mathbf{0} \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{3.4}$$

captures the similarity transformation through a scaling parameter $\mathbf{s}_i^s$ and a 2×2 rotation matrix $\mathbf{R}(\mathbf{s}_i^\theta)$, and the matrix

$$\mathbf{A}\left(\mathbf{s}_i^\alpha, \mathbf{s}_i^\beta\right) = \begin{bmatrix} 1 & \mathbf{s}_i^\alpha & 0 \\ \mathbf{s}_i^\beta & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

captures the shearing mapping in the affine transformation.

The proposed compositional transformation model resembles the classical decomposition of a projective transformation matrix into a concatenation of three unique matrices: similarity, affine, and pure perspective transformation [83]. Yet, our goal here is to "synthesize" rather than "analyze" the transformation $\mathbf{T}_i$ for sampling source patches. The proposed formulation allows us to effectively factor out the dependency of the positions of the target $\mathbf{t}_i$ and source patch $(\mathbf{s}_i^x, \mathbf{s}_i^y)$ for estimating the perspective deformation in $\mathbf{H}\left(\mathbf{t}_i, \mathbf{s}_i^x, \mathbf{s}_i^y, m_i\right)$ from estimating affine shape deformation parameters using $(\mathbf{s}_i^s, \mathbf{s}_i^\theta, \mathbf{s}_i^\alpha, \mathbf{s}_i^\beta)$ for matrices $\mathbf{S}$ and $\mathbf{A}$. This is crucial because we can then exploit piecewise smoothness characteristics in natural images for efficient nearest neighbor field estimation.

Photometric compensation: Local photometric compensation is usually carried out in prior work via normalizing the patch to be zero-mean and unit variance before comparing patches. While this certainly reduces the complexity of patch appearance variations when learning the LR-HR mapping using external dataset, we argue that the normalization might hallucinate erroneous HR details when matching the target patch with self-exemplars. Inspired by [82], we use the difference of target and source patch means to compensate the bias in the RGB space reasonable predefined ranges. Specifically, we

[1] Available at `https://github.com/jbhuang0604/StructCompletion`

compensate $\pm 25\%$ of the intensity variations for each channel.

Plane compatibility cost $\mathbf{E}_{\text{plane}}$: For man-made images, we can often reliably localize planes in the scene using standard vanishing point detection techniques. The detected 3D scene geometry can be used to guide the patch search space. We modify the plane localization code in [1] and add a plane compatibility cost to encourage the search over the more probable plane labels for source and target patches.

$$\mathbf{E}_{\text{plane}} = -\lambda_{\text{plane}} \log \left(\Pr[m_i|(\mathbf{s}_i^x, \mathbf{s}_i^y)] \times \Pr[m_i|(\mathbf{t}_i^x, \mathbf{t}_i^y)] \right), \tag{3.6}$$

where the $\Pr[m_i|(x,y)]$ is the posterior probability of assigning label m_i at pixel position (x,y) (see Figure 3.4 (b) for an example).

Scale cost $\mathbf{E}_{\text{scale}}$: Since we allow continuous geometric transformations, we observed that the nearest neighbor field often converged to the trivial solution, i.e., matching target patches to itself in the downsampled image I_{D}. Such a match has small appearance cost. This trivial solution leads to the conventional bicubic interpolation for SR. We avoid such trivial solutions by introducing the scale cost $\mathbf{E}_{\text{scale}}$:

$$\mathbf{E}_{\text{scale}} = \lambda_{\text{scale}} \min(0, \text{SRF} - \text{Scale}(\mathbf{T}_i)), \tag{3.7}$$

where SRF indicates the desired SR factor, e.g., 2x, 3x, or 4x, and the function $\text{Scale}(\cdot)$ indicates the scale estimation of a projective transformation matrix. We approximately estimate the scale of the source patch sampled using $\mathbf{T}_i$ with the first-order Taylor expansion [84]:

$$\text{Scale}(\mathbf{T}_i) = \sqrt{\det \left(\begin{bmatrix} \mathbf{T}_{1,1} - \mathbf{T}_{1,3}\mathbf{T}_{3,1} & \mathbf{T}_{1,2} - \mathbf{T}_{1,3}\mathbf{T}_{3,2} \\ \mathbf{T}_{2,1} - \mathbf{T}_{2,3}\mathbf{T}_{3,1} & \mathbf{T}_{3,1} - \mathbf{T}_{2,3}\mathbf{T}_{3,2} \end{bmatrix} \right)},$$

where $\mathbf{T}_{u,v}$ indicates the value of u^{th} row and v^{th} column in the transformation matrix $\mathbf{T}_i$ with $\mathbf{T}_{3,3}$ normalized to one. Intuitively, we penalize if the scale of the source patches is too small. Therefore, we encourage the algorithm to search for source patches that are similar to the target patch and at the same time to have larger scale in the input LR image space, and thereby to provide more high-frequency details for SR. We soft-threshold the penalty to zero when the scale of the source patch is sufficiently large.

3.4.2 Inference

We need to estimate 7-dimensional ($\theta_i \in R^7$) nearest neighbor field solutions over all overlapping target patches. Unlike the conventional self-exemplar based methods [57, 58], where only a 2D translation field needs to be estimated, the solution space in our formulation is much more difficult to search. We modify the PatchMatch [20] algorithm for this task with the following detailed steps.

Initialization: Instead of using the random initialization done in PatchMatch [20], we initialize the nearest neighbor field with zero displacements and scales equal to the desired SR factor. This is inspired by [58, 72], suggesting that good self-exemplars can often be found in a localized neighborhood. We found that this initialization strategy provides a good start for faster convergence.

Propagation: This step efficiently propagates good matches to neighbors. In contrast to propagating the transformation matrix $\mathbf{T}_i$ directly, we propagate the parameter $\theta_i = (\mathbf{s}_i, m_i)$ instead so that the affine shape transformation is invariant to the source patch position.

Randomization: After propagation in each iteration, we perform randomized search to refine the current solution. We simultaneously draw random samples of the plane index based on the posterior probability distribution, randomly perturb the affine transformation and randomly sample position (in a coarse-to-fine manner) to search for the optimal geometric transformation of source patches and reduce the matching errors.

3.5 Experiments

Datasets: Yang et al. [85] recently proposed a benchmark for evaluating single image SR methods. Most images therein consist of natural scenes such as landscapes, animals, and faces. Images that contain indoor, urban, architectural scenes, etc., rarely appear in this benchmark. However, such images feature prominently in consumer photographs. We therefore have created a new dataset *Urban 100* containing 100 HR images with a variety of real-world structures. We constructed this dataset using images from Flickr (under CC license) using keywords such as urban, city, architecture, and structure.

In addition, we also evaluate our algorithm on the *BSD 100* dataset, which consists of 100 test images of natural scenes taken from the Berkeley segmentation dataset [86]. For this dataset, we evaluate for SR factors of 2x, 3x, and 4x.

Methods evaluated: We compare our results against several state-of-the-art SR algorithms. Specifically, we choose four SR algorithms trained using a large number of external LR-HR patches for training. The algorithms we use are: Kernel rigid regression (Kim) [63], sparse coding (ScSR) [51], adjusted anchored neighborhood regression (A+)

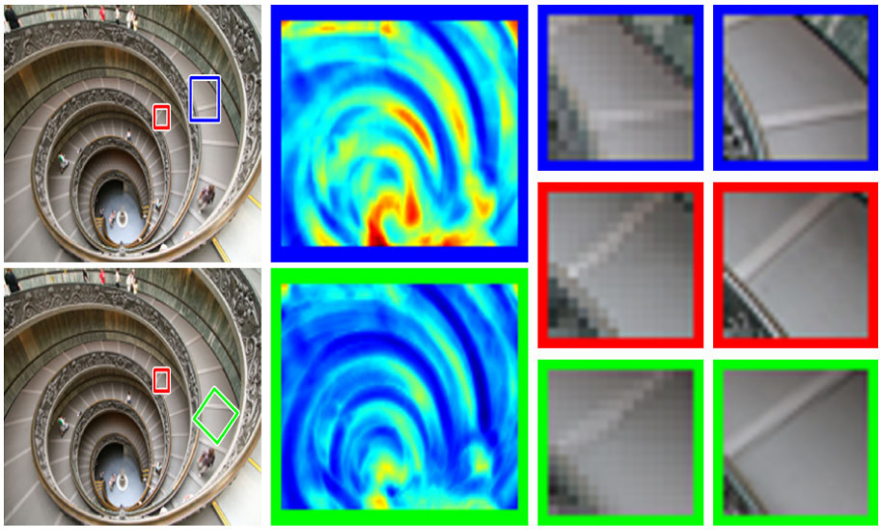
[54], and convolutional neural networks (SRCNN) [55].[2] We also compare our results with those of the internal dictionary based approach (Glasner) [57] [3] and the sub-band self-similarity SR algorithm (Sub-Band) [59].[4] All our datasets, results, and the source code are publicly available.[5]

Implementation details: We use 5×5 patches and perform SR in multiple steps. We achieve 2x, 3x, 4x SR factors in three, five and six upscaling steps, respectively. At the end of each step, we run 20 iterations of the backprojection algorithm [62] with a 5×5 Gaussian filter with $\sigma^2 = 1.2$. The NNF solution from a coarse level is upsampled and used as an initialization for the next finer level. We empirically set the parameters $\lambda_{\text{plane}} = 10^{-3}$ and $\lambda_{\text{scale}} = 10^{-3}$. The parameters are kept fixed for all our experiments.

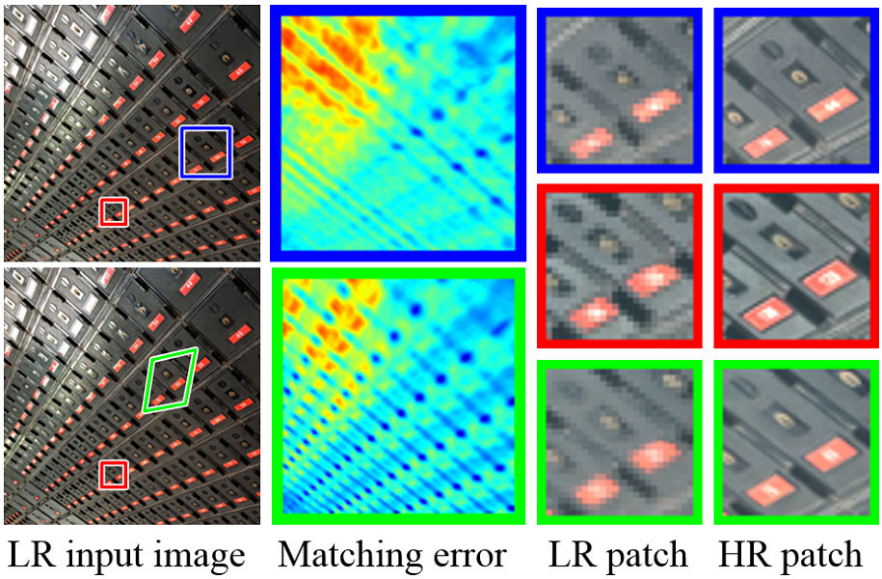
Qualitative evaluation: In Figure 3.5, we show visual results on images from the Urban 100 dataset. We show only the cropped regions here. We find that our method is capable of recovering structured details that were missing in the LR image by properly exploiting the internal similarity in the LR input. Other approaches, using external images for training, often fail to recover these structured details. Our algorithm well exploits the detected 3D scene geometry and the internal natural image statistics to super-resolve the missing high-frequency contents. In Figures 3.6 and 3.7, we demonstrate that our algorithm is not restricted to images of a single plane scene. We are able to automatically search for multiple planes and estimate their perspective and affine transformations to robustly predict the HR image.

In Figures 3.8 and 3.9, we show two results on natural images where no regular structures can be detected. In such cases, our algorithm reduces to searching for affine transformations only in the nearest neighbor field, similar to [81]. On natural images without any particular geometric regularity, our method performs as well as the recent, state-of-the-art methods such as [55, 54], although, as can be seen in both examples, our results contain slightly sharper edges and fewer artifacts such as ringing. We present more results for both Urban 100 and BSD 100 datasets in our project page `https://sites.google.com/site/jbhuang0604/publications/struct_sr`.

[2] Implementations of [63, 51, 54, 55] are available on authors' websites.
[3] We implement this from the paper [57].
[4] Results were provided by the authors.
[5] `https://github.com/jbhuang0604/selfexsr`



(a) Affine transformation



(b) Planar perspective transformation

Figure 3.3: Examples demonstrating the need for using transformed self-exemplars in our self-similarity based SR. Red boxes indicate a selected target patch (to be matched) in the input LR image I. We take the selected target patch, remove its mean, and find its nearest neighbor in the downsampled image I_{D}. We show the error found while matching patches in I_{D} in the second column. Blue boxes indicate the nearest neighbor (best matched) patch found among only translational patches, and green boxes indicate the nearest neighbor found under the proposed (a) affine transformation and (b) planar perspective transformation. In the third and fourth columns we show the matched patches Q in the downsampled images I_{D} and their HR version Q_{H} in the input image I.

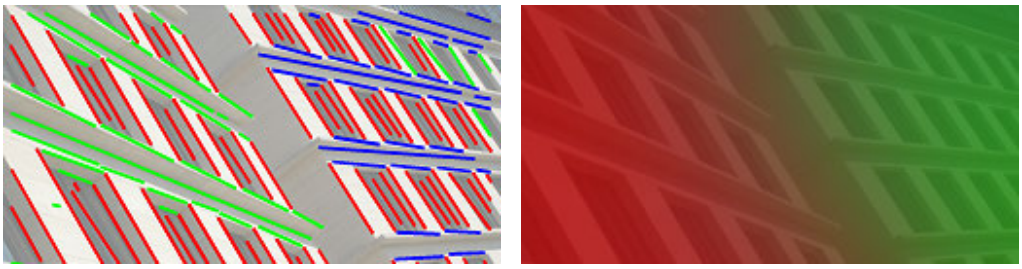


Figure 3.4: (a) Vanishing point detection and (b) Visualization of posterior plane probability.

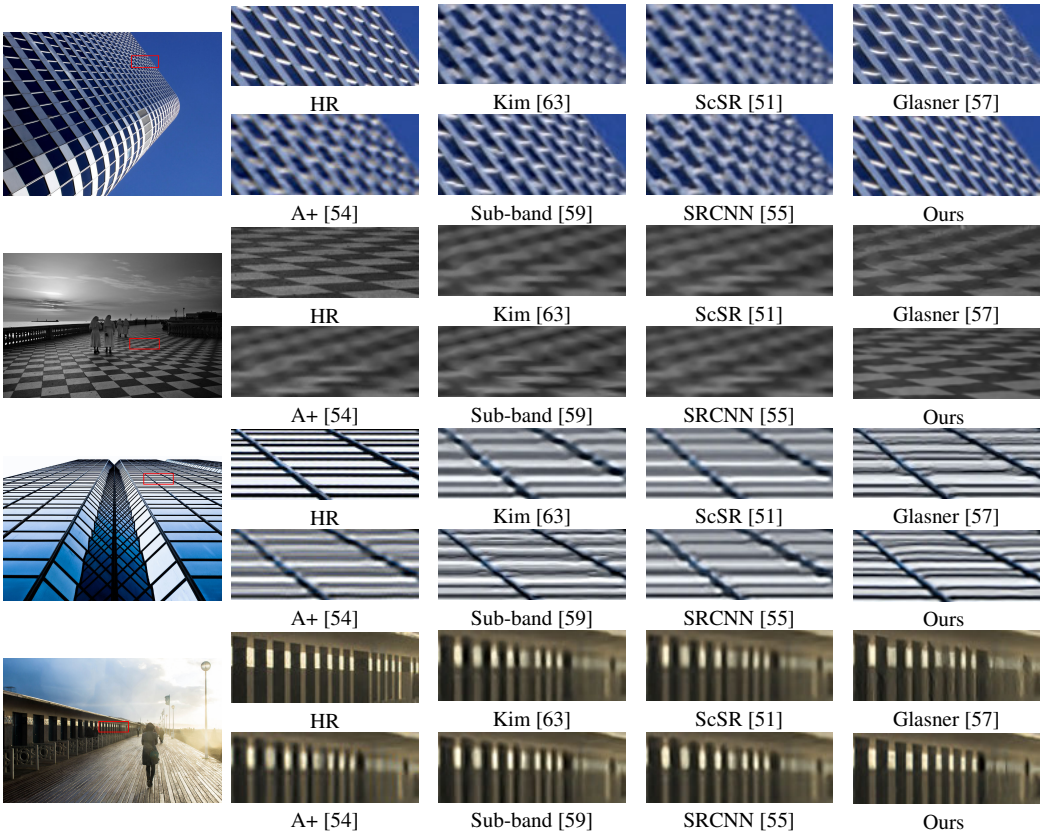
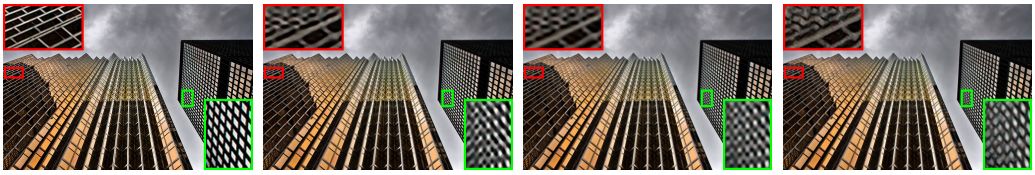


Figure 3.5: Visual comparison for 4x SR. Our method is able to explicitly identify perspective geometry to better super-resolve details of regular structures occurring in various urban scenes.



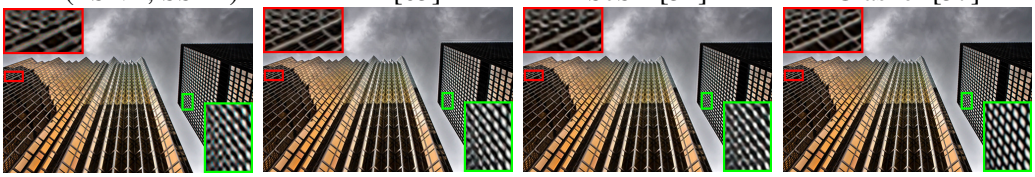


Figure 3.6: Visual comparison for 4x SR. Our algorithm is able to super-resolve images containing multiple planar structures. Image credit: Flickr user thelearningcurvedotca.

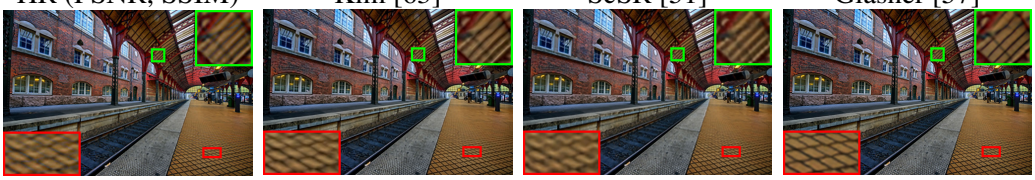


Figure 3.7: Visual comparison for 4x SR. Our algorithm is able to better exploit the regularity present in urban scenes than other methods. Image credit: Flickr user jimnix.

Figure 3.8: Visual comparison for 3x SR. Our result produces sharper edges than other methods. Shapes of fine structures (such as the horse's ears) are reproduced more faithfully in our result.

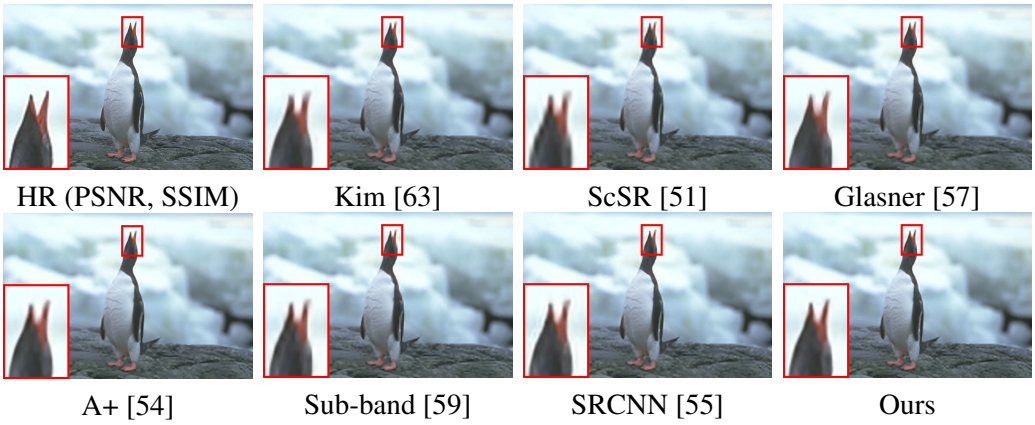


Figure 3.9: Visual comparison for 3x SR. Our result shows slightly sharper reconstruction of the beaks.

Table 3.1: Quantitative evaluation on *Urban 100* and *BSD 100* datasets. Red indicates the best and blue indicates the second best performance.

Metric	Scale	Bicubic	ScSR [51]	Kim [63]	SRCNN [55]	A+ [54]	Sub-band [59]	Glasner [57]	Ours
PSNR (*Urban*)	2x	26.66	28.26	28.74	28.65	28.87	28.34	28.15	29.38
	4x	23.14	24.02	24.20	24.14	24.34	24.21	23.79	24.82
SSIM (*Urban*)	2x	0.8408	0.8828	0.8940	0.8909	0.8957	0.8820	0.8743	0.9032
	4x	0.6573	0.7024	0.7104	0.7047	0.7195	0.7115	0.6838	0.7386
PSNR (*BSD*)	2x	29.55	30.77	31.11	31.11	31.22	30.73	30.56	31.18
	3x	27.20	27.72	28.17	28.20	28.30	27.88	27.36	28.30
	4x	25.96	26.61	26.71	26.70	26.82	26.60	26.38	26.85
SSIM (*BSD*)	2x	0.8425	0.8744	0.8840	0.8835	0.8862	0.8774	0.8675	0.8855
	3x	0.7382	0.7647	0.7788	0.7794	0.7836	0.7714	0.7490	0.7843
	4x	0.6672	0.6983	0.7027	0.7018	0.7089	0.7021	0.6842	0.7108

Quantitative evaluation: We also perform quantitative evaluation of our method in terms of PSNR (dB) and structural similarity (SSIM) index [87] (computed using luminance channel only). Since such quantitative metrics may not correlate well with visual perception, we invite the reader to examine the visual quality of our results for better evaluation of our method.

Table 3.1 shows the quantitative results on *Urban 100* and *BSD 100* datasets. Numbers in red indicate the best performance and those in blue indicate the second best performance. Our algorithm yields the best quantitative results for this dataset, 0.2-0.3 dB PSNR better than the second best method (A+) [54] and 0.4-0.5 dB better than the recently proposed SRCNN [55]. We are able to achieve these results *without* any training databases, while both [54] and [55] require millions of external training patches. Our method also outperforms the self-similarity approaches of [57] and [59], validating our claim of being able to extract better internal statistics through the expanded internal search space. In the *BSD 100* dataset our results are comparable to those obtained by other approaches on this dataset, with ≈ 0.1 dB lower PSNR than the results of A+ [54]. Our quantitative results are slightly worse than the state-of-the-art in this dataset since it is difficult to find geometric regularity in such natural images, which our algorithm seeks to exploit. Also A+ [54] is trained on patches that contain natural textures quite suitable for super-resolving the BSD100 images. While we achieve slightly worse quantitative performance on BSD100, our results are often visually more pleasing compared to others and do not have artifacts.

Effect of the number of NNF iterations: We investigate the effect of the number of iterations for nearest neighbor field estimation using our algorithm in Figure 3.10, for one step 2x SR. We show the intermediate results after 1, 2, and 5 iterations. The second row shows a visualization of the source patch positions in the nearest neighbor field and the matching cost in each stage. The in-place initialization (zero iterations) already provides good matches for smooth regions. We can see a significant reduction in the matching cost even with one iteration. We use 10 iterations for generating all our results.

Effect of patch size: Patch size is an important parameter for example-based SR algorithms. Larger patches may be difficult to map to HR since they may contain complex structural details. Very small patches may not contain sufficient information to accurately predict their HR versions. In Figure 3.11, we plot PSNR/SSIM for patch sizes ranging from 3×3 to 15×15. We obtain these plots by averaging over 25 images. We observe that there is a wide range of patch sizes for which our algorithm is able to perform consistently.

Limitations: Our method has difficulty dealing with fine details when the planes are not accurately detected. We show one such case in Figure 3.12 where we fail to recover the regular structures. Another limitation of our approach is processing time. While external database driven SR methods require time-consuming training procedures, they run quite fast during test time [54, 53]. While our algorithm does not require an explicit

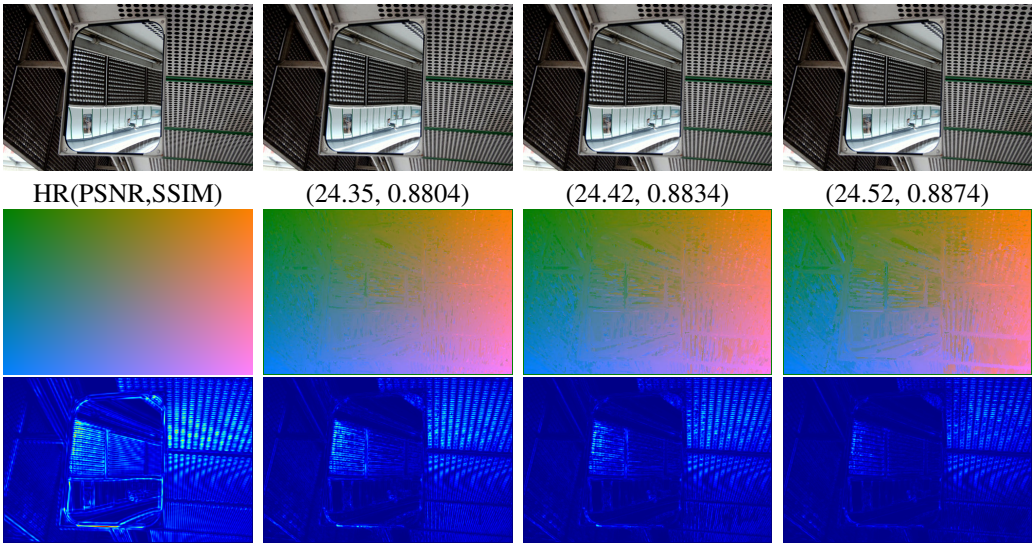


Figure 3.10: Effect of iterations. First row: HR and the SR results on 1, 2, and 5 iterations. Second row: the visualization of the nearest neighbor field. Third row: the patch matching cost.

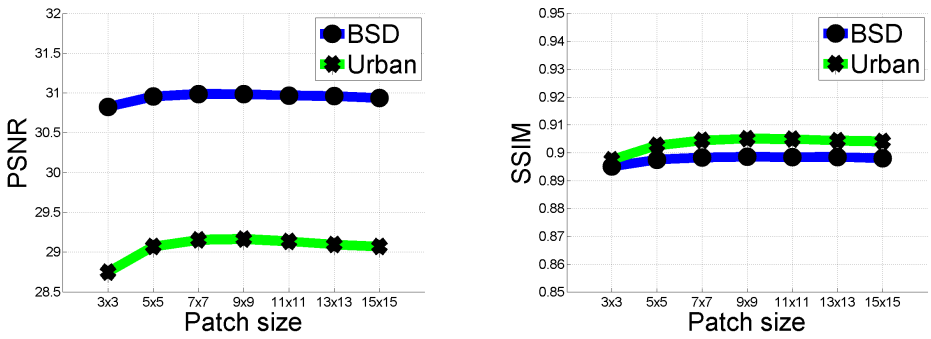


Figure 3.11: Quantitative performance as a function of patch size.

training step, it is slow to super-resolve a test image. This drawback is associated with all self-similarity based approaches [57, 59]. On an average, our Matlab implementation takes around 40 seconds to super-resolve an image in *BSD 100* by 2x with a 2.8 GHz Intel i7 CPU and 12 GB memory.

3.6 Concluding Remarks

We have presented a self-similarity based image SR algorithm that uses transformed self-exemplars. Our algorithm uses a factored patch transformation representation for simultaneously accounting for both planar perspective distortion and affine shape deformation of

Figure 3.12: A failure case with SR factor 4x.

image patches. We exploit the 3D scene geometry and patch search space expansion for improving the self-examplar search. In the absence of regular structures, our algorithm reverts to searching affine transformed patches. We have demonstrated that even without using external training samples, our method outperforms state-of-the-art SR algorithms on a variety of man-made scenes while maintaining comparable performance on natural scenes.

CHAPTER 4

TEMPORALLY COHERENT COMPLETION OF DYNAMIC VIDEO

4.1 Introduction

Video completion methods are designed to fill user-specified spatio-temporal holes with plausible content using remaining parts of the video. An effective video completion method has many practical applications in video post-production, such as unwanted object removal, full-frame video stabilization (as a byproduct), logo or watermark removal in broadcast videos, and restoration of damaged vintage films.

Much progress has been made on automatic single-image completion, to a point of where commercial solutions are now available.[1] However, automatic video completion algorithms have fared less well. This is due to the additional time dimension which introduces major challenges: (1) viewpoint changes cause non-trivial appearance changes in image-space; (2) the synthesized content needs to be temporally coherent; (3) there is exponentially increased computational complexity due to the larger number of missing pixels.

Many state-of-the-art video completion algorithms synthesize the missing (target) regions by sampling spatio-temporal patches from the known (source) regions [19, 88] or by solving spatio-temporal shift-maps using graph cuts [90]. While good results have been shown, these approaches have two major limitations.

One limitation for translation-based sampling is the degradation caused by source and target regions having inconsistent color, texture, or motion, which may be caused by viewpoint changes in hand-held videos or non-periodic object motion. The inconsistency problems can be partially alleviated by warping spatio-temporal object samples [92], stabilizing the input video [88], or compensating geometric distortion through projective transformation [89]. However, these approaches assume either a clean foreground and background layer separation, simple parametric global motion, or static background and therefore have difficulties in handling general situations.

Another limitation is that motion is not explicitly reconstructed even though motion-

[1] See, for example, `http://www.adobe.com/technology/projects/content-aware-fill.html`

Table 4.1: Comparisons with state-of-the-art video completion algorithms. The cells highlighted in red indicate limitations of an algorithm. The optimization techniques used in [88] and our approach that alternate between patch search and patch voting steps can be viewed as a "Hard EM" algorithm for estimating maximum-likelihood solution [19]. The visibility assumption refers to each missing pixel needing to be visible in at least one frame.

Method	[89]	[90]	[88]	[91]	Ours
Synthesis unit	Spatial	Spatiotemporal	Spatiotemporal	Spatial	Spatial
Optimization	Graph-cut	Graph-cut	Hard-EM	Greedy	Hard-EM
Dynamic background	No	Yes	Yes	No	Yes
Visibility assumption	Yes	No	No	No	No
Flow estimation	No	No	No	Yes	Yes
Warping	Homography	N/A	Affine	N/A	Dense flow field
Temporal consistency	No	No	No	Yes	Yes

based features are used as part of the similarity metric. As shown in our experiments, video completion algorithms without explicit motion reconstruction often yield results that are plausible when viewed as separate images, but are not temporally coherent. Several recent approaches address the temporal consistency problem by explicitly synthesizing flow field in the target regions [91, 93]. However, these approaches smoothly interpolate the flow field with diffusion-based techniques, and are less effective in synthesizing dynamic backgrounds.

In this chapter, we present a new video completion algorithm that does not make any simplifying assumptions about the video content; it works on casual hand-held videos with moving camera, dynamic content, lighting and color variations, and missing pixels that are not seen in any of the known regions. Our method can fill missing regions in such videos with plausible content in a temporally coherent manner.

Our method jointly estimates appearance (color) and dense flow field in the missing region. The key idea is that the reconstructed pixel-wise forward and backward flow fields allow us to explicitly promote temporal coherence. Since our flow field is non-parametric, it can handle general motion parallax and dynamic scenes. This is in comparison with existing video completion algorithms that use parametric motion to mostly compensate for camera motion (e.g., affine [88] or homography [89]). We formulate video completion as non-parametric patch-based optimization. The combination of non-parametric spatial patch-based optimization and dense flow field estimation facilitate synthesizing colors that are spatially coherent (i.e., locally appear similar to the known regions everywhere) in each frame while maintaining temporal coherence (i.e., with small flow warp errors) across frames.

We have tested our algorithm on numerous challenging videos that span a wide variety of situations, including hand-held and stationary camera, static and dynamic back-

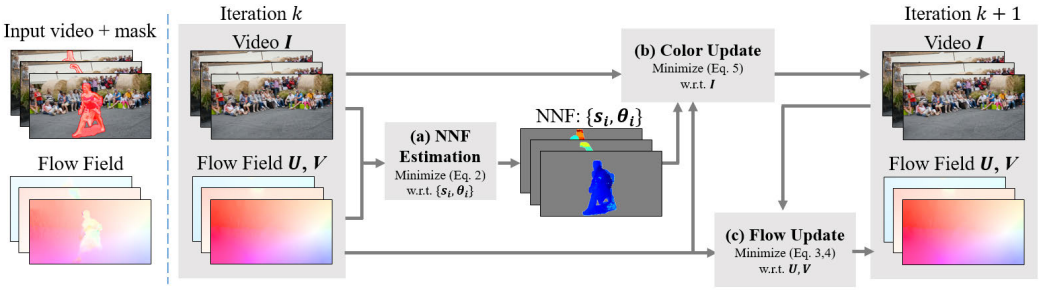


Figure 4.1: Algorithm pipeline. Given the input video and user-selected mask, we start with computing the flow fields. After initialization (Sec 4.4.4) at the coarsest level, in each scale our algorithm iterates through three steps (Sec 4.4.3): (a) nearest neighbor field estimation: minimize the color spatial cost by finding dense approximate nearest neighbor source patches for all target patches; (b) color update: minimize the color spatial and color temporal cost so that the synthesized colors are both spatially and temporally coherent; and (c) flow update: refine the forward and backward flow fields. We then upsample the solution of the nearest neighbor field and flow fields to the next finer level. The color at the finer level is estimated by spatial patch voting (using the upsampled nearest neighbor field).

ground, and multiple moving objects. We show representative still frames from the videos throughout the chapter.

4.2 Related Work

Image and video completion are well-explored topics; surveys by Guillemot and Le Meur [94] and Ilan and Shamir [95] provide a more comprehensive review. Here we limit our discussion to state-of-the-art or representative approaches. Table 4.1 shows a feature-by-feature comparison with representative techniques.

Patch-based synthesis These methods fill missing regions by sampling non-local spatio-temporal patches (e.g., $5 \times 5 \times 5$) from the known input. Efros and Leung [25] first introduced non-parametric sampling techniques for texture synthesis. The approach is later extended and applied to image completion [27, 96] and video completion [97, 98]. However, for video completion, these techniques either assume static cameras [97] or constrained camera motion [98] so that the foreground and background layers can be easily separated and independently filled. Furthermore, the greedy patch filling process inevitably propagates errors at early steps to the subsequent steps, yielding globally inconsistent results.

To address the global inconsistency issue, patch-based completion algorithms have been cast as a global optimization problem which is minimized by alternating between patch search and reconstruction steps [19, 28]. Newson et al. [88] recently extend the patch-based optimization approach by incorporating texture features, compensate dominant camera motion with global affine transformation, and use a spatio-temporal version of PatchMatch [20] for fast approximated nearest neighbor search.

Our algorithm builds upon the non-parametric optimization framework with three major differentiators. First, we fill the hole by sampling *spatial* patches rather than *spatio-temporal* patches. While spatio-temporal patches provide a simple way for encoding local appearance *and* motion, the assumption that spatio-temporal blocks appear repeatedly through time is typically not valid under the hand-held camera condition and non-repetitive scene motion. Spatial patches, on the other hand, have no such restriction and thus are applicable to general scenarios. Second, our approach explicitly estimates dense forward and backward flow fields. The estimated dense flow fields allow us to explicitly enforce temporal coherence of the synthesized contents as well as propagate known content into the unknown regions. Third, similar to Darabi et al. [17], we augment the patch search space to account for texture and structural inconsistency between the source and the target regions. In addition to random search, we use the local flow vectors to predict and propagate transformation parameters from frame to frame.

Segment-based synthesis Such methods pose completion as a labeling problem by solving a correspondence map (or a shift map) between the source and target pixels. They typically apply graph cuts to find optimal seams and thus do not need to specify patch sizes. Kwatra et al. [99] propose a texture synthesis algorithm by iteratively selecting a shifted version of an input texture and using graph cut to find minimally noticeable seams between the original and shifted textures. Moving beyond texture synthesis, Pritch et al. [30] generalize segment-based approaches to several other image editing tasks, including image reshuffling, retargeting and completion using a global multi-label graph cut based optimization. Granados et al. [90] extend this method to video completion and introduce an interactive interface for users to help constrain the search space.

The major limitation of segment-based methods is that the synthesized content has to be copied from the unoccluded regions "as is" in the known input. Therefore, these methods in general cannot handle objects that undergo appearance changes (e.g., scale variations when an object moves toward or away from the camera) as well as video captured with a hand-held camera. Granados et al. [89] show that the appearance variations between the source and target regions can be compensated by homographies. However, this approach assumes manual labeling of foreground regions, piecewise planar and static background, and visibility of the occluded region (i.e., the occluded region must be visible in some

other frames). In contrast, our method offers greater flexibility for handling appearance variations and can handle videos captured by a freely moving camera.

Flow-based synthesis To address the temporal consistency problem, techniques have been developed to fill motion field in missing regions, e.g., through a greedy selection of spatio-temporal patches of local motion [100], per frame diffusion [91], or iterative optimization [93] with propagated colors from the known boundary.

Our method differs in two ways. First, instead of treating flow estimation as an independent step from color estimation [100, 91], our approach iteratively computes and refines the dense flow field from the synthesized colors and vice versa. Second, filling the colors only through propagating from the known boundary (e.g., [100, 93]) inevitably generate blurry results due to successive averaging of colors and thus are applicable only for holes with very narrow temporal span. In contrast, we synthesize color by patch-based optimization and use the flow to enforce the temporal consistency. As a result, our approach can handle holes with arbitrary temporal span.

Flow-based video editing and manipulation Our work is also related to several flow-based video synthesis and editing tasks. Flow representation provides a direct way to enforce temporal consistency for texture synthesis [28], fluid animation [101], video editing [102] and high dynamic range video reconstruction [103]. The work most related to our work is that of Xue et al. [104], which decomposes a set of images into a clean background and occlusion/reflection layers. Unlike [104], our approach does not require every pixel in the occluding region to be visible in at least one frame. That is, we are able to hallucinate plausible contents for the unknown regions that are *not* visible in the entire image sequence.

4.3 Overview

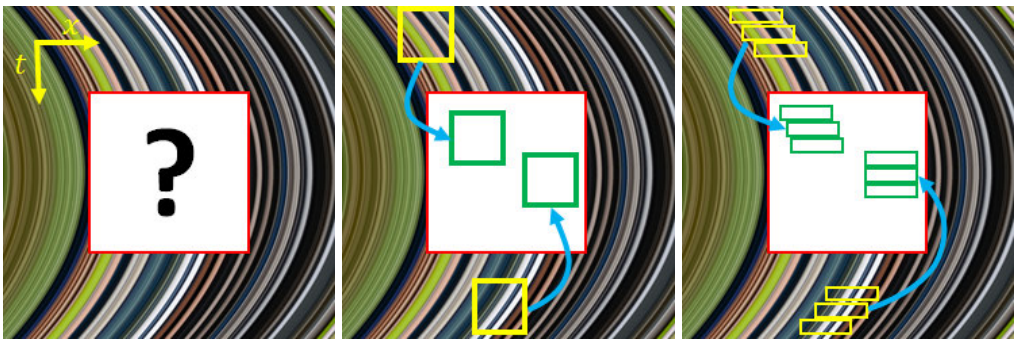
In our video completion algorithm, we jointly estimate the unknown color values and local motion in the target regions. We start with computing forward and backward optical flow in the *known* region for all adjacent frames using a two-frame optical flow algorithm [105]. We implement our color synthesis algorithm similar to the non-parametric patch-based optimization algorithms of [19, 28]. In each iteration, we loop over three main steps: (1) patch search, (2) color update and (3) forward/backward flow update. We summarize our method in Algorithm 1 and illustrate the pipeline in Figure 4.1.

Algorithm 1: Proposed video completion algorithm.

Input : Video $\boldsymbol{I}$, user-specified mask $\overline{\Omega}$
Output: Completed video $\boldsymbol{I}$

Compute forward/backward flow fields $\boldsymbol{U}, \boldsymbol{V}$ in Ω
Initialization: filling hole $\overline{\Omega}$ in $\boldsymbol{I}, \boldsymbol{U}, \boldsymbol{V}$ at coarsest scale (Sec. 4.4.4)
for scale s from 1 to n_s **do**
    for iteration k from 1 to K_s **do**
        (a) **NNF estimation:**
        Minimize Eq. 4.2 w.r.t. $\{\mathbf{s}_i, \theta_i\}$, with $\boldsymbol{I}, \boldsymbol{U}, \boldsymbol{V}$ fixed.
        (b) **Color update:**
        Minimize Eq. 4.6 w.r.t. $\boldsymbol{I}$, with $\boldsymbol{U}, \boldsymbol{V}, \{\mathbf{s}_i, \theta_i\}$ fixed.
        (c) **Flow update:**
        Minimize Eqs. 4.3 and 4.4 w.r.t. $\boldsymbol{U}, \boldsymbol{V}$, with $\boldsymbol{I}, \{\mathbf{s}_i, \theta_i\}$ fixed.
    end
    Upsample $\boldsymbol{U}, \boldsymbol{V}$ using bicubic interpolation.
    Upsample $\{\mathbf{s}_i, \theta\}$ using nearest-neighbor interpolation.
end

Patch search In this step, for all overlapping patches in the target regions, we search their corresponding patches in the source region with most similar local appearances. Our algorithm differs from existing non-parametric patch sampling methods in two major aspects. First, in contrast to methods using spatio-temporal patches $5\times5\times5$ pixels for synthesis, we use *spatial-only* patches (5×5 patch in our experiments) as synthesis units. We enforce temporal consistency by augmenting the patch matching cost with color consistency along the flow vectors. Such representation allows us to handle complex flow fields resulted from hand-held cameras and scenes with depth variations. For example, in a video captured by a hand-held camera, spatial patches with the exact same appearance may be embedded in significantly different spatio-temporal patches because the spatially-varying motion field caused by the moving camera. Therefore, working with spatio-temporal patches would have fundamental limitations in exploiting the non-local repetition of patches for video completion. In contrast, working with spatial patches does not suffer from such cases (Figure 4.2). Second, in contrast to copying exactly the local color patches from the source to the target region, we augment the patch search space to accommodate appearance variations. Specifically, we search additional geometric transformation (scale and rotation) of patches. Similar to the single-image case [17], the additional patch transformation offers greater flexibility in synthesizing visually plausible contents even with limited appearance variations in the source regions.



(a) Input and hole Spatiotemporal Spatial

Figure 4.2: Limitation of using spatio-temporal patches/segments. We use a frame from sequence DANCE-TWIRL and synthetically generate translational motion along the x-axis. (a) A spatio-temporal x-t slice of the sequence with mask overlay. (b) Using spatiotemporal patches (2D patches here) is not able to properly fill the missing region because the motion between the source (yellow) and target (green) regions are not consistent. (c) Using spatial patches (1D slices here), on the other hand, offers greater flexibility by adapting to the local flow.

Color update In the conventional patch-based optimization framework, the reconstruction step involves "patch voting," where the pixel color at an unknown pixel is estimated by averaging the colors of the corresponding nearest neighbors from all overlapping patches [19]. This encourages spatial coherence, i.e., reconstructed patches look similar to some place in the known region. We extend this step to incorporate also diffused colors from the nearest *transitive* temporal neighbors in the known region (i.e., found by walking along the flow vectors until a known pixel is reached). We use forward/backward flow consistency to identify areas of unreliable flow, e.g., in occluded or dis-occluded regions.

Forward and backward flow update After estimating color in the target region, we update and refine the forward and backward flow fields. This step resembles the two-frame optical flow computation in the known region. Here, we fix the flow in the source regions and only update the flow in the target regions, resulting in an optical flow estimation algorithm with spatio-temporal hole boundary constraints.

4.4 Completion as Optimization

In this section, we provide details for our approach. We start by introducing the problem formulation and objective function. Next, we describe our optimization procedure for joint color and flow estimation.

4.4.1 Problem formulation

Let $\boldsymbol{I}$ be the input video of height H, width W and number of frames L, and $\boldsymbol{U}, \boldsymbol{V}$ the forward and backward flow fields, respectively. The forward flow at position (x, y, f) is given by $\boldsymbol{U}(x, y, f) = (\mathrm{d}x, \mathrm{d}y, +1)$, indicating the flow vector $(\mathrm{d}x, \mathrm{d}y)$ from a point located at (x, y, f) to a point $(x+\mathrm{d}x,\ y+\mathrm{d}y,\ f+1)$ in the video $\boldsymbol{I}$. Similarly, the backward flow at (x, y, f) is $\boldsymbol{V}(x, y, f) = (\mathrm{d}x, \mathrm{d}y, -1)$. We denote the set of unknown pixels by $\overline{\Omega}$ (i.e., the user-specified spatio-temporal regions) and the set of known pixels by Ω.

We define $\mathbf{t}_i = (t_i^x, t_i^y, t_i^f) \in \overline{\Omega}$ as the i^{th} target pixel position, where (t_i^x, t_i^y) is the 2D spatial position and t_i^f is the frame index. Our goal is to estimate the unknown color values $\boldsymbol{I}(\mathbf{t}_i)$ for all target pixels, as well as the forward/backward flow vectors $\boldsymbol{U}(\mathbf{t}_i), \boldsymbol{V}(\mathbf{t}_i)$. We estimate the colors through non-parametric patch-based optimization. Specifically, for the i^{th} target (unknown) pixel, we seek source (known) pixel position $\mathbf{s}_i = (s_i^x, s_i^y, s_i^f) \in \Omega$ and 2D patch geometric transformation $\theta_i \in \mathbb{R}^2$ (patch scaling and rotation) to minimize spatial reconstruction errors. Temporal consistency of synthesized colors $\boldsymbol{I}(\mathbf{t}_i)$ is achieved by enforcing color consistency along forward and backward flow fields $\boldsymbol{U}, \boldsymbol{V}$. We describe the objective function and iterative optimization steps in the following subsections.

4.4.2 Objective function

We solve the following problem:

$$\underset{\boldsymbol{I}, \boldsymbol{U}, \boldsymbol{V}, \{\mathbf{s}_i, \theta_i\}}{\operatorname{argmin}} \boldsymbol{E}_{\textit{color-spatial}} + \boldsymbol{E}_{\textit{color-temporal}} + \boldsymbol{E}_{\textit{flow-spatial}}, \tag{4.1}$$

where: (1) $\boldsymbol{E}_{\textit{color-spatial}}$ penalizes *spatial* patch-based reconstruction errors of colors in the target regions, (2) $\boldsymbol{E}_{\textit{color-temporal}}$ penalizes *temporal* inconsistency of the synthesized colors, and (3) $\boldsymbol{E}_{\textit{flow-spatial}}$ is a spatial regularization term for the forward and backward flow fields.

Spatial color cost This cost encourages local neighborhoods P_i in the target region to appear similar to local neighborhoods Q_i in the source region. It also encourages spatial coherence, since we consider overlapping target patches. P_i is an axis-aligned 5×5 spatial color patch, centered around $\mathbf{t}_i$, and Q_i is centered around $\mathbf{s}_i$, with scale and rotation transformation specified by θ_i. The spatial color cost is the sum of square losses for all overlapping patches in $\overline{\Omega}$ and their correspondences:

$$\boldsymbol{E}_{\textit{color-spatial}} = \sum_{i \in \overline{\Omega}} \|P_i - Q_i\|_2^2. \tag{4.2}$$

We use a Gaussian falloff function with $\sigma = 1$ to give higher weights to pixels closer to the center of the neighborhood.

Temporal color cost This cost encourages temporal color consistency between adjacent frames along the forward and backward flow vectors. We use a term that is commonly employed in optical flow algorithms:

$$\begin{aligned} \boldsymbol{E}_{\textit{color-temporal}} = \sum_{i \in \overline{\Omega}} \alpha \phi\Big(\big\| \boldsymbol{I}(\mathbf{t}_i) - \boldsymbol{I}\big(\mathbf{t}_i + \boldsymbol{U}(\mathbf{t}_i)\big) \big\|_2^2 \Big) + \\ \alpha \phi\Big(\big\| \boldsymbol{I}(\mathbf{t}_i) - \boldsymbol{I}\big(\mathbf{t}_i + \boldsymbol{V}(\mathbf{t}_i)\big) \big\|_2^2 \Big), \end{aligned} \tag{4.3}$$

where the Charbonnier penalty $\phi(x^2) = \sqrt{x^2 + \varepsilon}$ with a small constant ε is a robust function (a differentiable variant of ℓ_1-norm), and $\alpha = 0.5$ is the weight coefficient for this cost.

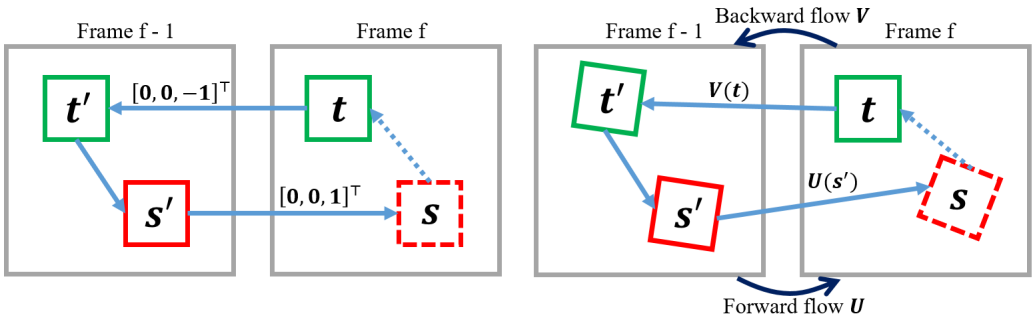
Spatial flow cost To promote piecewise smooth flow fields, we introduce spatial regularization:

$$\begin{aligned} \boldsymbol{E}_{\textit{flow-spatial}} = \beta \iint \phi\Big(\|\bigtriangledown \boldsymbol{U}_x(\mathbf{t}_i)\|_2^2 + \|\bigtriangledown \boldsymbol{U}_y(\mathbf{t}_i)\|_2^2 \Big) \mathrm{d}x\mathrm{d}y + \\ \beta \iint \phi\Big(\|\bigtriangledown \boldsymbol{V}_x(\mathbf{t}_i)\|_2^2 + \|\bigtriangledown \boldsymbol{V}_y(\mathbf{t}_i)\|_2^2 \Big) \mathrm{d}x\mathrm{d}y, \end{aligned} \tag{4.4}$$

where $\beta = 0.1$ is the weight coefficient for this cost, and $\bigtriangledown$ denotes the gradient operator. This cost penalizes large magnitudes in the gradient of the flow field with a robust function $\phi(x)$ and regularizes the flow vectors between the synthesized colors in adjacent frames.

4.4.3 Optimization

Since Equation 4.1 is non-convex, we use an iterative optimization algorithm that alternates between minimizing different subsets of the variables. We apply the optimization in a coarse-to-fine manner to reduce the chance of prematurely locking into a bad local minimum. At each iteration, we first fix the colors $\boldsymbol{I}$ and flow fields $\boldsymbol{U}, \boldsymbol{V}$, and solve for source patch position and transformation $(\mathbf{s}_i, \theta_i)$ for each target patch $\mathbf{t}_i$. This corresponds to the patch search step for estimating the dense approximate nearest neighbor field $\{\mathbf{s}_i, \theta_i\} i \in \overline{\Omega}$. We then fix the estimated nearest neighbor field $(\mathbf{s}_i, \theta_i)$ and flow fields $\boldsymbol{U}, \boldsymbol{V}$ and estimate colors $\boldsymbol{I}$. This color synthesis step minimizes local appearance differences between the source and target patches spatially and the color differences along flow vectors temporally. Finally, we fix the synthesized colors $\boldsymbol{I}$ and update the forward/backward flow fields $\boldsymbol{U}, \boldsymbol{V}$. Algorithm 1 summarizes this procedure in pseudocode. We now describe each step



(a) PatchMatch propagation (b) Flow-guided propagation

Figure 4.3: Flow-guided temporal propagation. (a) The direct extension of the PatchMatch algorithm [20] to 3D [88]. Similar to the spatial propagation case in PatchMatch, candidate patches are propagated along the temporal axis. (b) The proposed flow-guided temporal propagation relaxes the constraints of axis-aligned propagation and uses local forward and backward flow vectors for accurate prediction of the candidate source patch position and transformation.

in detail.

Nearest neighbor field estimation Given the currently estimated color $\boldsymbol{I}$ and forward/backward flow fields $\boldsymbol{U}, \boldsymbol{V}$, we want to search for each target pixel position $\mathbf{t}_i$ (1) the source position $\mathbf{s}_i$, and (2) the geometric patch transformation θ_i that minimizes the overall objective in (Eq. 4.1). This is equivalent to just minimizing the first term of the objective function (Eq. 4.2). To this end we extend the generalized PatchMatch algorithm [81] to the spatio-temporal case. We initialize the source patch position $\mathbf{s}_i$ and patch transformation θ_i through random sampling, and then update the estimation by alternating between the random search and propagation steps.

Random search: At this step, we generate a sequence of random samples $(\mathbf{s}_i, \theta_i)$ (i.e., source position, rotation and scale) from an exponential distribution [81]. We update the nearest neighbor field if any of the randomly selected sample achieves a lower cost.

Spatial and temporal propagation: This step involves propagating good nearest neighbor candidates spatially and temporally. For the spatial propagation, we follow the generalized PatchMatch algorithm to take into account the geometric transformation of the source patch. The temporal propagation step, however, has two important differences regarding candidate patch position and transformation:

(1) Propagating patch position: In a straightforward extension of the PatchMatch algorithm (e.g., as in [88]), for the target patch centered at $(t_i^x, t_i^y, t_i^f \pm 1)$, the algorithm would consider candidate source patches centered at $(s_i^x, s_i^y, s_i^f \pm 1)$. However, this temporal propagation strategy implicitly assumes that the motion at the target pixel $\mathbf{t}_i$ is the

same as the motion at the source pixel $\mathbf{s}_i$. We avoid this by temporally propagating the candidate patches along the estimated forward and backward flow vectors; see Figure 4.3 as illustration of forward temporal propagation. For target patch $\mathbf{t}_i$, we generate the candidate source patch position $\mathbf{s}_i$ by: (1) finding temporal neighbor of $\mathbf{t}_i$ using backward flow: $\mathbf{t}'_i = \mathbf{t}_i + \boldsymbol{V}(\mathbf{t}_i)$, (2) finding the corresponding source patch $\mathbf{s}'_i$ of the target patch $\mathbf{t}'_i$ from the current nearest neighbor field, and (3) propagating the source patch to the next frame using forward flow $\mathbf{s}_i = \mathbf{s}'_i + \boldsymbol{U}(\mathbf{s}_i)$. The backward temporal propagation follows a similar procedure. The flow-guided temporal propagation removes the assumption that the motion must be consistent in the source and target patches.

(2) Propagating patch transformation: We observe that the local patches of flow vectors provide cues of the *patch transformation* from the current frame to the adjacent frames. For example, the apparent size of an object increases under camera zoom. Scaling of patches can be inferred from the relative densities of flow vectors from frame to frame. More specifically, we warp a grid of pixel positions using optic flow and estimate the propagated candidate transformation matrix as $T(\boldsymbol{U}(\mathbf{s}'_i))S(\theta'_i)T(\boldsymbol{V}(\mathbf{t}_i))$, where $S(\theta'_i)$ is the transformation matrix of source patch $\mathbf{s}'_i$ and $T(\boldsymbol{V}(\mathbf{t}_i))$ is the estimated patch transformation matrix using local flow vectors at position $\mathbf{t}_i$. $T(\boldsymbol{U}(\mathbf{s}'_i))$ is similarly defined.

Flow-guided color synthesis Given the currently estimated forward/backward flow fields $\boldsymbol{U}, \boldsymbol{V}$, we estimate the color in the missing regions by minimizing $\boldsymbol{E}_{\textit{color-spatial}} + \boldsymbol{E}_{\textit{color-temporal}}$.

The spatial color cost $\boldsymbol{E}_{\textit{color-spatial}}$ by itself could be minimized using standard patch voting, i.e., computing the weighted average of the overlapping source patches. Incorporating the temporal color cost $\boldsymbol{E}_{\textit{color-temporal}}$ turns the problem into a 3D Poisson equation with temporal connections specified by the forward/backward flow vectors. It can be solved with the Gauss-Seidel method. Let $\boldsymbol{Z}$ denote the color field obtained by spatial voting, i.e., the minimizer of $\boldsymbol{E}_{\textit{color-spatial}}$. Applying the Gauss-Seidel method involves iteratively averaging $\boldsymbol{Z}$ and the warped images from the forward/backward flow. However, we found that this algorithm requires many iterations to convergence, particularly when the hole contains a long temporal span. In addition, the repeated application of bicubic interpolation (for sampling colors in sub-pixel positions) introduces unwanted blur and ringing artifacts.

We propose a heuristic to address this issue. First, we categorize the missing pixels into two classes:

Connected pixels are pixels that are visible somewhere in the video. By following their forward or backward flow connections *transitively* we eventually reach a known pixel. We call these known pixels the transitive temporal neighbors of the connected pixel, designated $\boldsymbol{r}_i^f$ (forward) and $\boldsymbol{r}_i^b$ (backward) for i^{th} pixel. For connected

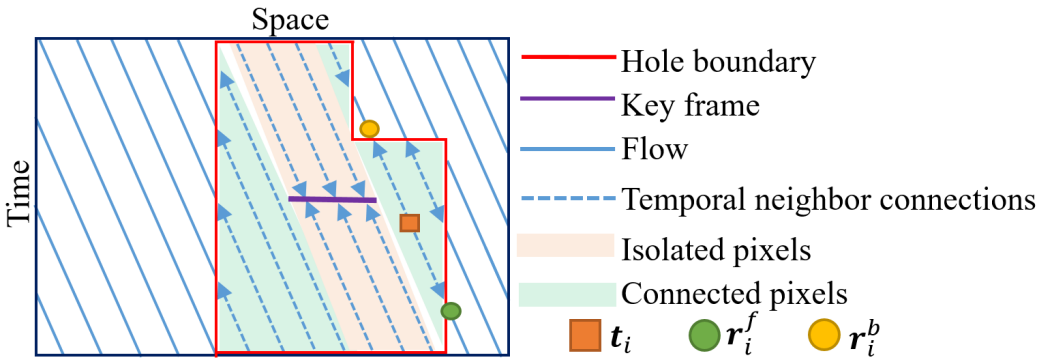


Figure 4.4: Flow-guided color synthesis. For all target pixels, we find their temporal neighbors in the source regions by traversing the estimated flow vectors. We then use these temporal neighbors to enforce temporal coherence.

pixels we directly penalize deviation from the transitive temporal neighbor colors, avoiding error accumulation from repeated resampling.

Isolated pixels are not transitively connected to any known pixels. It is difficult to synthesize isolated pixels consistently using the Gauss-Seidel solver described above. To address this issue we find the frame that has most isolated pixels and designate it a *key frame*. The isolated pixels in the key frame are treated as if they are known pixels, i.e., they become the transitive temporal neighbors for other isolated pixels, turning them into connected pixels. We greedily pick key frames in this manner until all isolated pixels are connected. The key frame pixels themselves do not have temporal neighbors, and use spatial voting only. In a way they are synthesized like in a standard *image* completion problem, albeit jointly solved in a video completion problem.

Figure 4.4 illustrates the situations described above. Formally, the heuristic above is implemented by minimizing the following objective:

$$\underset{\boldsymbol{I},\boldsymbol{Z}}{\operatorname{argmin}} \sum_{i\in\overline{\Omega}} \|\boldsymbol{I}(\mathbf{t}_i)-\boldsymbol{Z}(\mathbf{t}_i)\|_2^2 + \alpha\phi\left(\left\|\boldsymbol{I}(\mathbf{t}_i)-\boldsymbol{I}(\boldsymbol{r}_i^f)\right\|_2^2\right) + \alpha\phi\left(\left\|\boldsymbol{I}(\mathbf{t}_i)-\boldsymbol{I}(\boldsymbol{r}_i^b)\right\|_2^2\right). \tag{4.5}$$

For pixels that are missing either $\boldsymbol{r}_i^f$ or $\boldsymbol{r}_i^b$ we drop the respective terms (key frames do not have temporal neighbors, all other pixels have at least one temporal neighbor).

As this objective function decomposes into $|\overline{\Omega}|$ independent loss functions, we can drop the pixel index i for simplicity. We use the spatial voting results $\boldsymbol{Z}$ as an initialization, which corresponds to only minimizing spatial cost and ignoring temporal coherence.

We then iteratively solve the best increment $\mathrm{d}\boldsymbol{I}$ by setting the derivative of the objective function to zero. Denote the differences of the $\boldsymbol{Z}(\mathbf{t}_i) - \boldsymbol{I}(\mathbf{t}_i)$ as $\mathrm{d}\boldsymbol{Z}$, $\mathbf{I}(\boldsymbol{r}_i^f) - \boldsymbol{I}(\mathbf{t}_i)$ as $\mathrm{d}\boldsymbol{I}^f$, and $\mathbf{I}(\boldsymbol{r}_i^b) - \boldsymbol{I}(\mathbf{t}_i)$ as $\mathrm{d}\boldsymbol{I}^b$. In each iteration, we can compute the optimal increment in close-form:

$$\mathrm{d}\boldsymbol{I}(\mathbf{t}_i) = \frac{1}{C}\left[\mathrm{d}\boldsymbol{Z} + \alpha\left(\phi'\big(\mathrm{d}\boldsymbol{I}^f\big)\,\mathrm{d}\boldsymbol{I}^f + \phi'\Big(\mathrm{d}\boldsymbol{I}^b\Big)\,\mathrm{d}\boldsymbol{I}^b\right)\right], \tag{4.6}$$

$$C = 1 + \alpha\left(\phi'\big(\mathrm{d}\boldsymbol{I}^f\big) + \phi'\Big(\mathrm{d}\boldsymbol{I}^b\Big)\right), \tag{4.7}$$

where $\phi'(x^2) = \frac{1}{\sqrt{x^2+\varepsilon^2}}$ is the first-order derivative of the robust function $\phi(\cdot)$. We find that five iterations are sufficient for convergence. Solving the modified objective function in Eq. 4.6 strikes a good balance between temporal consistency and spatial coherence.

Forward and backward flow field estimation In this step, we fix the color $\boldsymbol{I}$ and refine and update the forward/backward flow fields $\boldsymbol{U}, \boldsymbol{V}$. The minimization problem corresponds to a typical two-frame optical flow estimation problem with a data term and a spatial regularization term. We use the previously estimated flow as initialization and iteratively estimate the flow increments. We use the Iterative Reweighted Least Squares (IRLS) formulation [105] to compute the forward and background flow field in the target regions.

4.4.4 Initialization

To bootstrap the optimization process we need to initialize color, flow, and nearest neighbor fields at the coarsest scale. We first initialize the nearest neighbor field $\{\mathbf{s}_i, \theta_i\}$ with random samples, and the flow fields $\boldsymbol{U}, \boldsymbol{V}$ by smoothly interpolating the known values at the boundary inward (independently for each frame). Next, we initialize the colors $\boldsymbol{I}$ by running the nearest neighbor field estimation and the flow-guided color synthesis step, and finally update the flow fields using the synthesized colors.

4.4.5 Implementation details

We use a relatively small 5×5 patches in our synthesis. While typical image completion algorithms use larger patches, e.g., 10×10 [17], we observe that increasing the patch size does not substantially improve the visual quality of the results while significantly increasing the runtime. We attribute this effect to the additional temporal connections by the forward and backward flow vectors. These connections help efficiently propagate good candidates along the temporal dimension for nearest neighbor search. We set the

color temporal cost weight $\alpha = 0.5$ and the flow spatial weight $\beta = 0.1$. We fix the weights for all our experiments.

We optimize the objective function in a multi-resolution fashion. We set the number of image pyramid levels such that the height at the coarse scale is between 32 and 64 pixels. At the coarsest scale, we run 20 iterations and decrease iteration count in each subsequent scale by 4 (though we never use less than 4 iterations).

Our algorithm relies on the estimated flow to enforce temporal consistency. Optical flow algorithms are far from perfect, however, particularly in occluded/dis-occluded regions. We are thus interested in identifying unreliable flow pixels. While existing learning-based flow confidence measures [106] appear to be robust, they are computationally intensive. Instead, we use forward-backward flow consistency to identify unreliable flow. Specifically, we compute confidence scores c_i^f, c_i^b for forward and backward flow,

$$c_i^f = \exp\left(-\frac{\|\boldsymbol{U}(\mathbf{t}_i) + \boldsymbol{V}(\mathbf{t}_i + \boldsymbol{U}(\mathbf{t}_i))\|_2^2}{2\sigma_F^2}\right),$$
$$c_i^b = \exp\left(-\frac{\|\boldsymbol{V}(\mathbf{t}_i) + \boldsymbol{U}(\mathbf{t}_i + \boldsymbol{V}(\mathbf{t}_i))\|_2^2}{2\sigma_F^2}\right),$$

where $\sigma_F = 1$ controls the sensitivity. We label the forward flow $\boldsymbol{U}(\mathbf{t}_i)$ at $\mathbf{t}_i$ as "unreliable" when the confidence score $c_i^f < 0.5$, and similar for the backward flow. We discard these unreliable flow vectors when searching for the temporal neighbors of the missing pixels. As a result, we drop the respective terms in Eq. 4.6 when performing the flow-guided color update, preventing the influence by unreliable flows.

4.5 Results

We implement the completion algorithm in MATLAB. For optical flow computation, we use the C++ implementation from Liu [105]. Processing a short video with 854×480 pixels and 90 frames and with a moderately sized missing region (e.g., CAMEL: 6.5M = 17.81% of pixels missing) our non-optimized implementation took around 3 hours on a desktop computer with 2.8 GHz Intel i7 CPU (quad-core) and 12GB memory. **All the data, results, and source code will be made publicly available**.

Removing dynamically moving objects in natural scenes We evaluated our algorithm on a variety of challenging sequences from a recent benchmark dataset [107]. While the dataset was intended for evaluating video object segmentation algorithms, the image sequences present multiple instances of challenges for evaluating video completion algorithms "in the wild." The major challenges including dynamic background, mo-

tion blur, camera shake, background clutter, and complicated hole shapes. We dilate the ground truth pixelwise annotation using a 15×15 structuring element. We then use the RotoBrush tool in Adobe AfterEffect to include cast shadows in the mask.

Figure 4.5 shows sample frames from five input image sequences with mask overlay (odd rows) and our completion results (even rows). In the first three sequences CAMEL, BREAKDANCE, and KITE-SURF, we demonstrate that our algorithm can seamlessly fill the missing dynamic background for videos captured with freely moving camera. The HORSEJUMP-LOW and FLAMINGO sequences highlight the advantage of our flow-guided color synthesis for accurately propagating known colors into the hole.

Comparisons with state-of-the-art methods We qualitatively compare our method with a recent state-of-the-art patch-based video completion algorithm [88]. We used the code released by the author and tested it on the image sequences [107] using the default parameters provided by the authors. Figure 4.6 shows the representative frames from four sequences and the completion results. Newson et al. [88] fill the hole by sampling spatio-temporal patches from source regions. We can see that such a technique introduces severe artifacts because of the inconsistent motion between the source and target regions. Our method, on the other hand, fills the missing regions with convincing contents.

In Figure 4.7, we compare with a segmentation-based technique for background [89] and foreground inpainting [90]. As the code is not publicly available, we compare with them using sequences from their papers. Our method achieves comparable quality without the need to provide dense pixel-wise mask of the foreground objects [89] or specify spatio-temporal search regions [90].

In Figure 4.8, we highlight the temporal coherence aspect of the completion results. In Figure 4.8(b), we show a spatio-temporal x-t slice of the video along the marked profile. In Figure 4.8(c)(d), we show the x-t slice of the *completed* video by [88] and our approach, respectively. From the x-t slice visualization we can clearly see that our results are temporally coherent and are adapted to the non-trivial camera motion in the known regions.

Contributions of each component We evaluate the contributions of each of the major components to the final performance in Figure 4.9. We tested the sequence ROLLERBLADE by disabling (1) patch-based optimization (i.e., propagate colors from the known boundary into the hole) and (2) flow estimation (i.e., using interpolated flow only), respectively.

As the input image sequence inevitably has bias, gain, color, or tone differences from one frame to another, the completion results may contain visible boundary along the hole. We compensate this photometric inconsistency by blending the synthesized contents with the original input video through solving a 3D spatio-temporal Poisson problem. Fig-

Figure 4.5: Object removal from video sequences CAMEL, BREAKDANCE, KITE-SURF, HORSEJUMP-LOW, and FLAMINGO. For each input sequence (odd row), we show representative frames with mask overlay. We show the completed results in even rows.

Input + mask [88] Our results

Figure 4.6: Comparison to [88] on sequences BMX-BUMPS, SWING, and TENNIS. These sequences are challenging due to the motion blur from the fast camera motion. Our algorithm seamlessly removes the dynamic object under shaky motion. Newson et al. [88], on the other hand, produces visible artifacts spatially and fails to generate temporally coherent results.

ure 4.10 shows one example of the blending result that hides the visible seams on the boundary.

Comparison to single image completion algorithms Figure 4.11 shows the advantage of video completion (or multi-frame image completion) over conventional image completion from a single image. In the sequence DANCE-JUMP, searching for usable texture from the same source image fails to complete the missing region with plausible contents. We show in Figure 4.11 results from two state-of-the-art image completion algorithms: Photoshop Content Aware Fill and [1]. Our video completion result (top-right) shows a convincing completion by transferring available contents that are visible in other frames in the sequence.

Limitations Our video completion has several limitations. First, the computational complexity of the proposed algorithm is still high and far from interactive rate. The long computational time makes the method impractical for many video editing applications. In our experiments, about 65% of the overall computation is spent on iteratively computing and refining the forward and backward flow fields in the image sequence. Using GPU to

speed up the flow computation may help speed up the completion process.

Second, as our algorithm relies on dense flow fields to guide the completion, we often fail to generate convincing completion of dynamic textures, e.g., waves, due to the unreliable flows.

Third, when a large area is occluded throughout the entire image sequence (i.e., a large number of isolated pixels), the performance of our algorithm is limited by how well an image completion algorithm can fill the hole. We show such a case in Figure 4.12. While our completion is still temporally coherent, spatial artifacts are visible with a closer examination.

4.6 Conclusions

In this chapter, we presented a robust video completion algorithm that is capable of handling a wide variety of challenging scenarios. Our main contribution is to combine the advantages of patch-based optimization framework and pixel-wise flow field representation for temporally coherent synthesis. We formulate the filling process as a global optimization of color and flow and present an alternating optimization approach to minimize the objective function. Experimental results show that our approach significantly extends the capacity of existing video completion algorithms on videos containing multiple dynamic objects, and with scene depth variations.

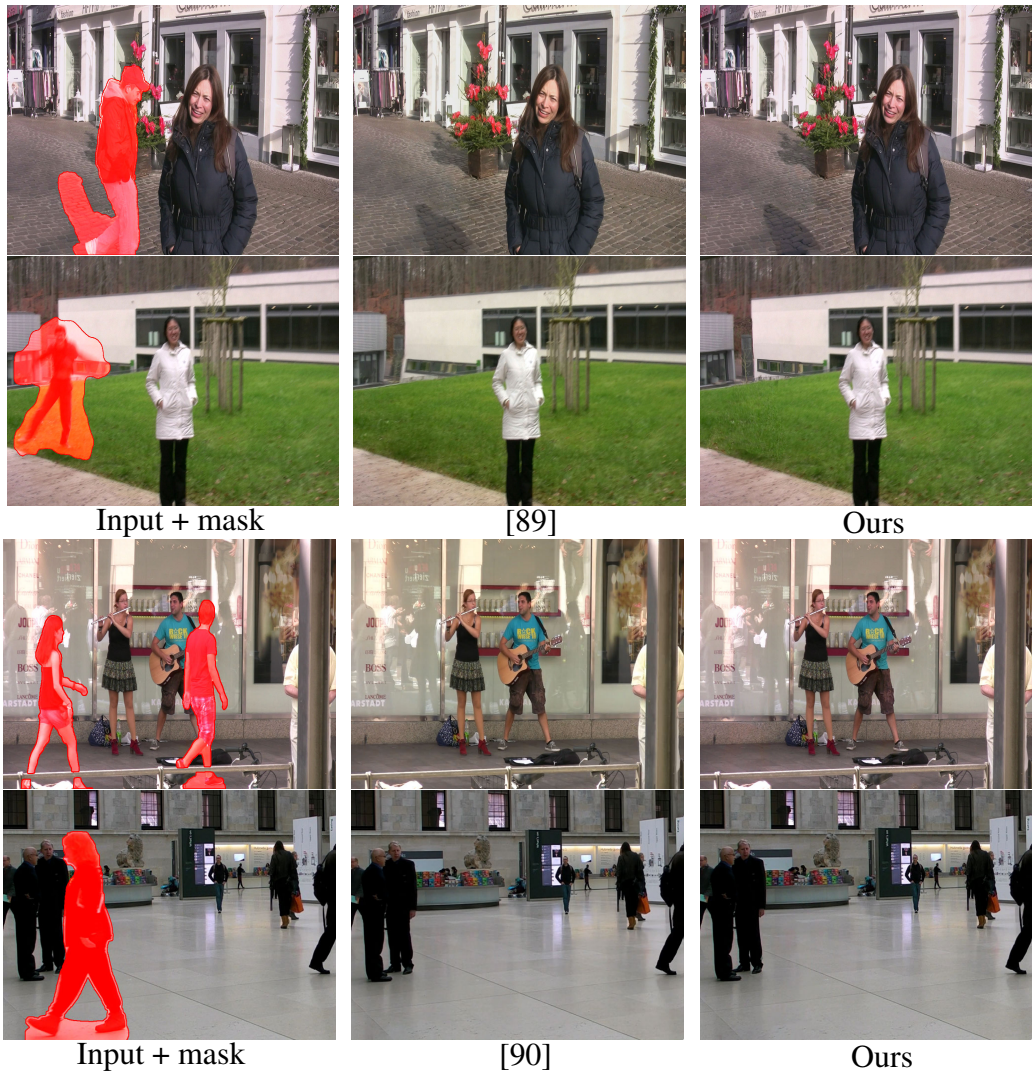
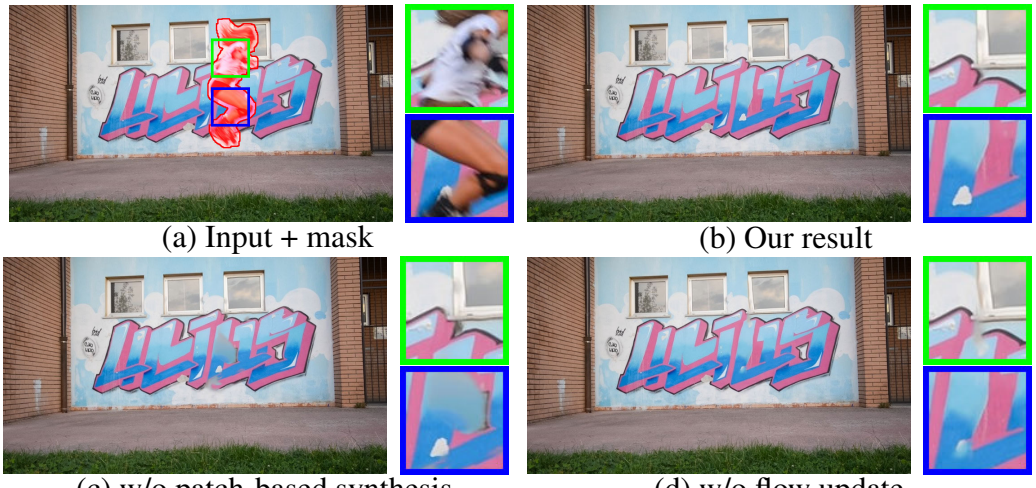


Figure 4.7: Comparison to segmentation-based methods for background [89] and foreground [90] inpainting on sequences from their paper. Our approach achieves similar visual quality without the need of manually segmenting out the dynamic foreground objects in the scene or manually specifying search regions.

(a) Sample frame (b) Input + mask

(c) [88] (d) Ours

Figure 4.8: Temporally coherent completion. We take the sequence CAMEL and visualize the completion results using spatiotemporal x-t slice of the video along the profile (yellow line) in (a). (b) The x-t slice of the video with mask marked as red. (c) Results from [88]. (d) Our results. We can clearly see that the completion results in (c), while seeming locally plausible, fail to maintain long-term temporal consistency. The combination of patch-based optimization and dense flow field allows us to preserve the temporal continuity with high spatial frequency.



(a) Input + mask (b) Our result

(c) w/o patch-based synthesis (d) w/o flow update

Figure 4.9: Contribution of different components of the proposed algorithm to the final results. (b) Our result. (c) Without patch-based synthesis, the algorithm cannot hallucinate regions that are not visible in the image sequence (see the blue box). (d) Disabling the flow update introduces visible artifacts.

Figure 4.10: The effect of using Poisson blending for compensating the photometric inconsistency. (a) Input + hole. (b) Our result w/o blending. (c) Our result with Poisson blending.

Figure 4.11: The advantage of video completion over image completion algorithms. Our video completion algorithm faithfully recover the missing region by taking all the frames into consideration.

Figure 4.12: Our algorithm may fail to hallucinate large missing areas. Here the artifacts are visible with a closer examination.

CHAPTER 5

DETECTING MIGRATING BIRDS AT NIGHT

5.1 Introduction

Bird migration is the regular seasonal, large-scale, often long-distance movement between breeding and wintering grounds. Many species of bird migrate. Migration behavior is a critical indicator for evaluating environmental health [108]. By identifying important stopover and wintering locations, one can take action to save these key locations to protect endangered species. Scientists use a variety of methods to monitor bird migration, including satellite tracking, weather radar, moon-watching, or attaching geolocators on captured birds. However, these methods are either expensive (e.g., satellite tracking), inaccurate because they are indirect (e.g., weather surveillance radars), labor-intensive and error-prone (e.g., moon-watching), or intrusive (e.g., geolocators). Moreover, these techniques only crudely estimate the bulk density of migrating birds aloft.

We propose to use a vision-based approach as a complementary sensing modality to build a bird migration monitoring system. By setting up stereo cameras facing up to the night sky, we can detect and track migrating birds in flight illuminated from below by light pollution in the recorded videos, as shown in Figure 5.1. Vision-based solutions offer several advantages over existing techniques. First, we can automatically and accurately count the number of individual birds aloft along with detailed trajectory estimation such as orientation, speed, and altitude. Such unprecedented accuracy in the reconstructed trajectories of individual birds may help re-evaluate migration, locomotion and navigation theories. Second, the estimated statistics could be used to calibrate other sensing modalities such as weather radar. Third, low-cost digital cameras allow us to build large-scale, distributed monitoring systems that cover broad areas.

There are three main challenges in developing a robust bird detection algorithm from videos. First, as migration usually occurs at night, the recorded videos inevitably contain substantial noise because of the low-light environment — the birds are generally invisible to the naked eye in the sky unless they pass in front of an illuminated object such as the moon. We illustrate this using sample frames from three video sequences in Figure 5.2. Second, depending on the species, migrating birds fly at altitudes ranging from several

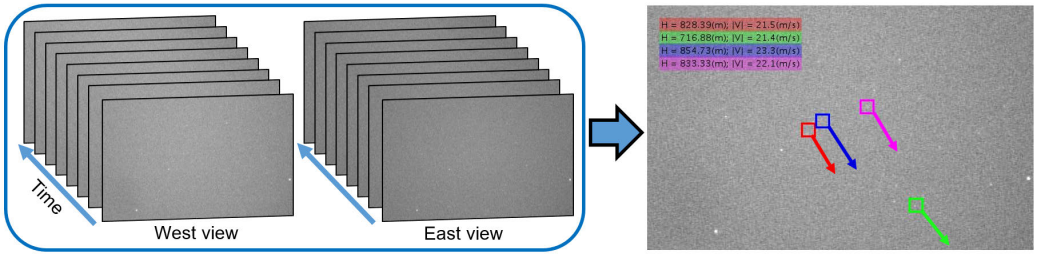


Figure 5.1: An example of automatic bird detection in stereo sequences. Our system takes stereo videos of the night sky as inputs, detects migrating birds in flight, and infers their orientation, speed, and altitude in very low SNR.

hundred feet to two miles. If the lens and camera provide an adequate field of view, the imaged bird may span only 1-2 pixels in a frame. This suggests that motion is the only reliable cue for detecting a bird. Third, efficient algorithms are required for large-scale deployment.

Several methods have been proposed to detect small objects in image sequences under different problem contexts. In Automatic Target Recognition (ATR) [109, 110] the presence of the target is detected using either simple frame differencing, filter responses, or matching against a known template and then tracking over time. Similarly, in ridge detection in three-dimensional volumetric data (e.g., vessel extraction [111]), the ridge is often detected using a pre-defined set of oriented filters. The common drawback of these approaches is that the detection is mostly performed *locally*. These techniques are thus not directly applicable to our problem due to the extremely low SNRs in our case. Recent methods address this issue by designing filter banks to improve detection of faint signals [112, 113, 114, 115] or by searching a large set of curves [116, 117]. However, most of these algorithms, designed specifically for 2D images, are computationally infeasible for 3D image data. In multi-object tracking, several algorithms have been proposed to track objects in 3D using stereoscopy [118, 119].

In general, the problem of target tracking can be divided into four main categories: (1) Large objects in bright light (e.g., tracking cars, pedestrians, faces in daylight). (2) Small objects in bright light (e.g., meteor streaks in sky surveys, planes with lights at night or in daylight at a great distance, rockets/missiles that are bright in IR). (3) Large objects in dim light (e.g., people detection and tracking at night under surveillance illumination). (4) Small objects in low light (e.g., birds flying over 1 mile high at night illuminated by light pollution). Unfortunately, a direct application of existing techniques does not suffice for our problem (category 4). These techniques often pose tracking and trajectory reconstruction as independent problems of frame-level target localization and cross-frame and cross-view data-association. The target size and SNR in our case are so low that targets cannot be reliably detected in individual frames.

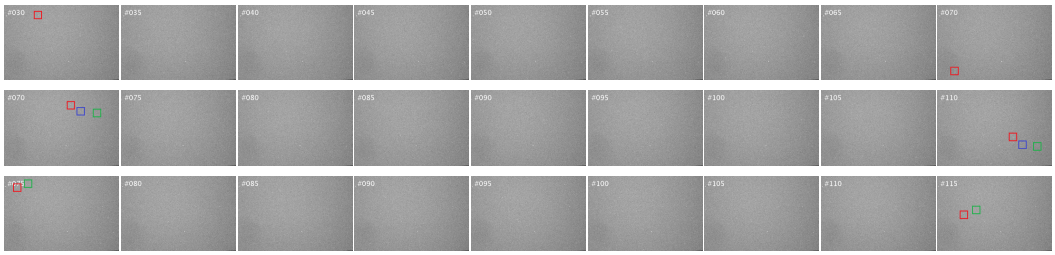
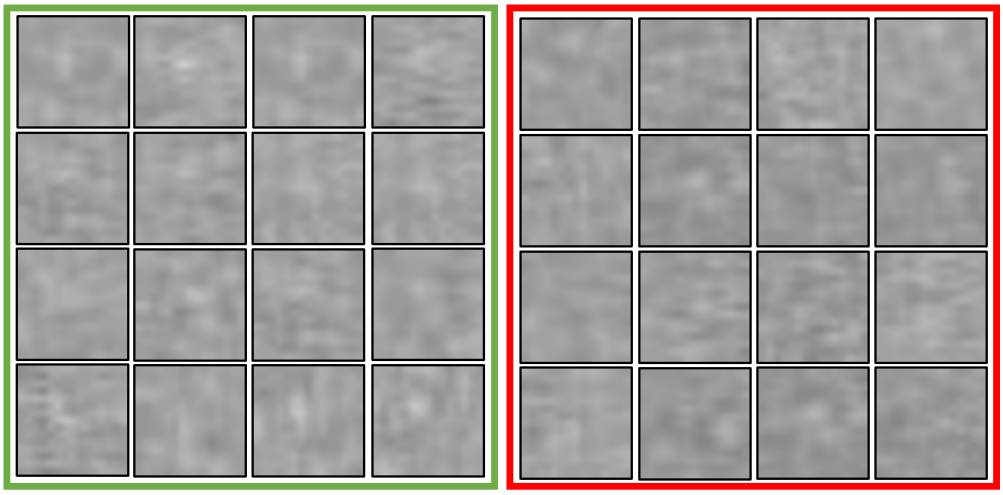


Figure 5.2: Detecting migrating birds from noisy image sequences. Each row shows a set of frames from a video sequence. From top to bottom, the sequences shown here have increasing levels of difficulty. Most of the bright spots in the images are stars. Color boxes indicate the birds in the first and the last frame of each sequence. Because of the low SNR and small size of high-flying birds (1-2 pixels), detection is very difficult, and often impossible, when looking at individual frames. It is only by detecting motion in the video stream that the human perceptual system can identify and track most birds. Similarly, the detection algorithm can only detect the more difficult high-flying birds by looking at the full video sequence and by simultaneously using stereo constraints from both cameras. Results are best viewed on a high-resolution display with adequate zoom level.

In this chapter, we tackle this problem using a two-stage robust model fitting approach. In contrast to prior work that aims at local detections in each frame, we aim at detecting using domain knowledge and global reasoning. Our fundamental assumption is that the migrating birds do not significantly change course and speed over short temporal spans (e.g., 5 seconds). We can thus cast the bird detection as finding curved 3D ridges in a spatiotemporal volume. The core detection algorithm consists of two main stages:

(1) *Geometric verification*: Given a large collection of noisy local detections, we extend the RANSAC-based 3D line fitting algorithm by explicitly incorporating stereo vision constraints. Specifically, we fit the model to both views *jointly*, which offers several advantages over a straightforward application of RANSAC independently in each view. First, the sample subset is used to determine the full bird model including altitude, speed, orientation, and position. Second, we can quickly reject a large number of physically implausible model hypotheses by checking the disparity, the temporal alignment, and extreme speed and altitude. Third, our model hypothesis allows us to exploit simultaneously the detected foreground points from both views by compensating the disparity. We set a loose threshold for line fitting so that birds flying at time-varying speed or directions could also be detected. To the best of our knowledge, while RANSAC has been extensively applied to two-view robust correspondence problems (e.g., solving the fundamental matrix, homography), it is less explored in robust model fitting (e.g., fitting 3D lines in volumetric data) by incorporating multi-view inputs and constraints.



(a) Bird patches (b) Background patches

Figure 5.3: The difficulty of detection based on local image patches. (a) 16 cropped local image patch along a manually labeled bird trajectory. (b) 16 cropped random background patches. These patches are virtually indistinguishable by the naked eye.

(2) *Trajectory verification*: In this step, we aim at verifying the presence of the bird using guidance from geometric verification. Given a small set of 3D line hypotheses, we integrate the signals along the direction of the coarse 3D trajectory while accounting for spatial uncertainties due to time-varying speed, direction, and altitude. This is technically realized using the generalized distance transform to efficiently search over all possible spatial deformations. The trajectory verification allows us to integrate all of the local responses along the predicted trajectory, resulting in a more discriminative signal for separating birds from noisy background night sky and ranking hypothesis. This step is critical for handling challenging low-SNR scenarios.

We make the following contributions in this chapter:

1. We address a novel application domain using computer vision algorithms. The vision-based system provides a low-cost, accurate, and new sensing modality for monitoring and studying bird migration.
2. We propose a RANSAC-based 3D line fitting algorithm that explicitly incorporates stereo vision constraints. We demonstrate that such constraints are crucial for robust model fitting in very low SNRs.
3. We account for birds flying with time-varying speeds and directions using deformable part modeling. The trajectory verification step allows us to gather all the local responses along the predicted trajectory, resulting in a discriminative signal for separating birds from the noisy background night sky.

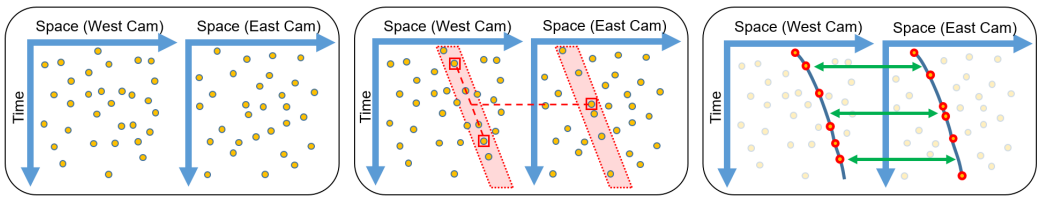


Figure 5.4: Overview of the bird detection algorithm. Our algorithm consists of three main modules: (a) Foreground detection: using statistical background modeling for moving object detection. (b) Geometric verification: RANSAC-based line fitting with stereo vision constraints. The three red boxes indicate the selected hypothetical inliers. This strategy naturally handles disparity estimation and offers computational efficiency by rejecting a large number of physically implausible configurations. (c) Trajectory verification: with the coarse 3D line fitting, we integrate weak signals along the predicted trajectory for both videos to verify if there is a bird. To account for birds flying at time-varying speed and directions, we interpret the motion compensated local image patch as a "part" of an object and use the generalized distance transform [120] for handling such spatial uncertainty. We detect the birds by thresholding the final response map.

5.2 Related Work

Bird migration monitoring techniques. Scientists use methods such as weather radar [121, 122, 123] and acoustic sensors [124, 125, 126] to monitor migrating birds [127, 128, 129]. Radar networks can provide wide area coverage over 1000s of kilometers, but radar reflectivity data is difficult to interpret and requires careful calibration as the data contain many biological (birds, bats, and insects) and meteorological phenomena. Calibration often is based on a traditional method for counting migrating birds: the use of a telescope to count birds as they pass across the full moon. Although moon-watching [130, 131] can provide direct visual bird counts, it is labor-intensive, error-prone (e.g., when multiple birds fly across), and only covers a very small portion of the night sky (the moon is about 0.5 deg wide in the sky). In contrast, our vision-based approach can accurately detect birds, infer their orientations, speeds, altitudes, and cover a large portion of the sky — a 5-10 degree FOV covers 250 to 1000$\times$ larger area than the moon.

Small target detection in image sequences. Detecting and tracking small targets in infrared image sequences is a long-standing problem in computer vision with numerous military applications. These methods typically rely on detecting the small targets locally, e.g., using frame-differencing [132], max-mean/max-median filter [133], top-hat transformation [134], or directional filters [135]. Local detections are then linked over time using sequential hypothesis testing or motion models such as Kalman, particle filters, or

global optimization approaches [119, 118]. As our videos contain a substantial amount of noise, local detections are not reliable (as shown in Figure 5.3). Unlike previous approaches that aim at getting correct *local* detections, we leverage top-down models with *global* reasoning for robust detection.

The work most related to our work is that of Ballerini et al. [136], which uses stereo vision to reconstruct 3D positions of individual birds to study the collective behavior of flocks of birds during the day. Our problem differs from theirs because many birds migrate *at night*. The challenge thus lies in how to detect birds in very low SNRs reliably. We can perform detection only by doing detection and tracking simultaneously so that detection is enabled by additional constraints coming from tracking, and vice versa.

Ridge detection in three-dimensions. We can view our problem as ridge detection in three-dimensional volumetric data (i.e., spatiotemporal volume). Ridge detection techniques often detect ridges using a pre-defined set of oriented filters at multiple scales. However, the local filters are not optimal for detecting faint signals in low SNR settings. Recent efforts include designing image representation for facilitating faint signal detection [112, 113, 115] or detecting faint curved edges in images [116, 117].

Geometric model fitting. Our work is related to classical parametric shape fitting techniques in computer vision such as RANSAC [137] and generalized Hough transform [138]. In our problem context, Hough transform would need to construct a 5-D parameter space, making the memory cost prohibitively high. Our method uses a RANSAC-based algorithm to perform line fitting in 3D point clouds (2D space + 1D time). The novelty lies in that we propose a sample selection approach for generating hypothetical inliers by leveraging the stereo vision constraints.

5.3 Overview

Figure 5.4 illustrates the three main steps for detecting migrating birds in flight. Given a pair of stereo videos, we first use classical statistical background modeling to detect foreground candidates (Section 5.4.3). As shown in Figure 5.4, the substantial number of outliers obscure the hidden curved line. Second, we use a RANSAC-based 3D line fitting algorithm to generate and verify hypotheses (Section 5.4.4). We propose a sampling strategy that explicitly incorporates stereo vision constraints. Such constraints are powerful because they allow us to reject a large portion of physically implausible configurations, and thereby offer computational efficiency when a large number of random samples are required due to the unusually high outliers ratio. We use a coarse threshold to maintain

high recall in detection. Third, we use trajectory verification (Section 5.4.4) to integrate the faint signals along the predicted trajectory from geometric verification while accounting for spatial uncertainties. Unlike RANSAC-based detection methods that use *sparse* detection data (i.e., 3D point clouds), we exploit *dense* information across the spatiotemporal volume. Through gathering local evidence across a long temporal span, we get a clean and discriminative signal that allows us to separate birds from the noisy background with high precision.

5.4 Stereo-based Bird Detection

In this section, we describe the proposed method in detail. We first present the local bird trajectory model by assuming a weak perspective camera model. We then briefly describe pre-processing steps for rectifying videos of the night sky by registering stars, followed by the core detection algorithm: (1) foreground detection, (2) geometry verification, and (3) trajectory verification.

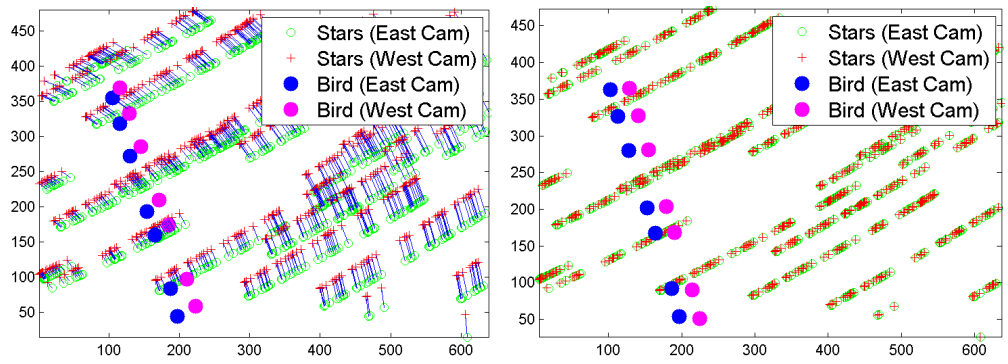
5.4.1 Bird trajectory modeling

To model the coarse bird trajectory in a video, we make the following two assumptions. First, we assume affine camera models because the migrating birds in flight are reasonably far away from the camera (with altitudes ranging from several hundred feet to two miles) compared with the size of a bird. Second, we assume that birds fly at relatively constant speed, orientation, and altitude during a short time-frame (e.g., 5 seconds).

Denoting the three-dimensional position in space of a bird at time t as $\mathbf{P}_t = [X_t, Y_t, Z_t]^\top$, we can express the imaged position of the bird $\mathbf{p}_t = [x_t, y_t]^\top$ as $\mathbf{p}_t = \mathbf{M}[\mathbf{P}_t^\top, 1]^\top$, where $\mathbf{M} \in \mathbb{R}^{2\times 4}$ is the camera projection matrix. Using the constant speed, orientation, and altitude assumptions, we simplify the 3D position $\mathbf{P}_t$ as $\mathbf{P}_t = \mathbf{P}_0 + t[V_x, V_y, 0]^\top$, where $\mathbf{P}_0$ indicates the position at time $t = 0$, and V_x, V_y are the physical speeds in space. We can write down the imaged position $\mathbf{p}_t = \mathbf{p}_0 + t[v_x, v_y]^\top$, where v_x, v_y are the speed in the image space. We can thus view this *idealized* bird trajectory as a thin, straight ridge in the spatio-temporal video cube.

5.4.2 Stereo image rectification

Our system uses stereo vision to determine the altitude of a flying bird from correspondence. To simplify the 2D correspondence search to 1D, we first rectify the images from two views so that all epipolar lines are parallel to the horizontal axis. We follow the



(a) Correspondence (b) Stereo rectification

Figure 5.5: Stereo image rectification using star registration. (a) Correspondence, (b) Stereo image rectification.

standard procedure for stereo image rectification: (1) finding a set of correspondences in the stereo pair of videos, (2) estimate the fundamental matrix [139], and (3) compute the rectifying matrices using [140].

For night sky images, we cannot apply the commonly used local interest point and feature descriptor matching approaches to establish correspondences. Fortunately, mother nature provides stars as markers. The two cameras are set up to capture roughly the same patch of the sky, so we exploit the imaged star positions for image registration. For each video, we first apply a moving average filter over the temporal axis to suppress the background noise. We then apply a single-scale 2D Laplacian of the Gaussian (LoG) to locate bright blob structures. After thresholding the LoG filter response and non-maximum suppression, we obtain a set of star positions (i.e., 2D point cloud) for each video.

With the detected star positions, we use the Iterative Closest Point (ICP) algorithm [141] with an affine motion model to find the transformation and inlier matches. However, as the stars are infinitely far away from the camera, the correspondences from stars give rise to a degenerate case in fundamental matrix estimation. To eliminate this degeneracy, we manually label the position of a flying bird in several frames. We only need to do this manual labeling *once* because we assume the cameras remain fixed through the videos. It is possible to use the proposed automatic flying bird detection to perform self-calibration (e.g., for cases where the stereo camera setup cannot remain fixed over time), but we leave that for future work.

We show in Figure 5.5(a) the detected stars in two views (Red and Green) the correspondence from ICP in Blue line. Figure 5.5(b) shows the rectified positions for the stars and the manually labeled bird. The stars from two views align accurately (as they are infinitely far) and the labeled birds fall on horizontal lines. Note that the results shown here contain star positions over 20 mins. The purpose of using this "star trail" is to provide

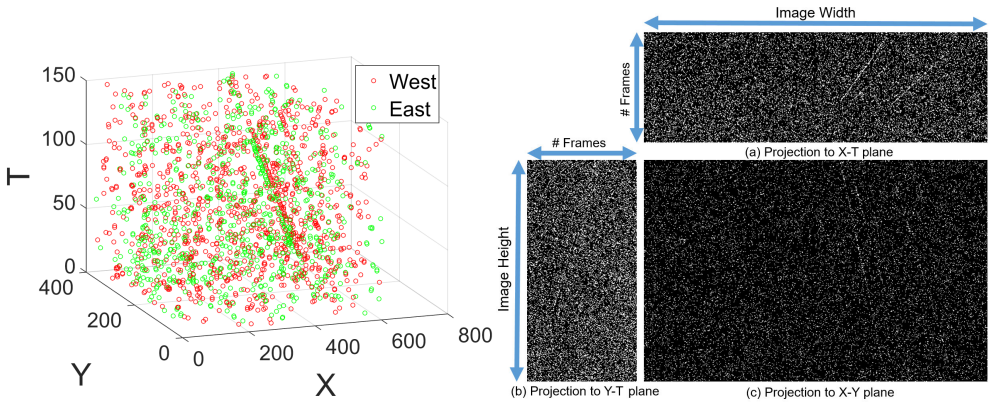


Figure 5.6: Foreground detection. (a) Sample foreground detection plots. Flying birds in a video appear like curved lines in the spatio-temporal volume. In this scattered plot, there are three curved lines. (b) Projection of foreground detection onto X-Y, X-T, and Y-T planes.

additional accuracy for registration.

5.4.3 Foreground detection

In this step, we look for *local* evidences for detecting flying birds. As imaged flying birds appear brighter than the surrounding background (illuminated from below by light pollution), the imaged bird trajectory can be seen as an intensity ridge in the video sequence. The problem of bird detection could be naturally cast as a *ridge detection* task in a 3D spatiotemporal video cube. Ridge (and valley) detection have been extensively studied in computer vision and image analysis with typical applications for detecting road in aerial images and for detecting blood vessels in 2D retinal or 3D magnetic resonance images. These methods often rely on designing filters that respond to locally linear intensity features followed by linking processes. However, these methods cannot directly be applied to our problem. As our videos have very low SNR, achieving accurate local detection would require evaluating a large collection of oriented filters with large kernel sizes, and thus would not scale well with large-scale video datasets.

For efficiency, we rely on top-down knowledge and global reasoning for detecting dim flying birds and resort to a simple statistical background modeling approach for local foreground pixel detection. Specifically, we build a per-pixel Gaussian model and compute the response of a pixel by measuring the intensity deviation from the learned model. We detect foreground pixels by thresholding the local responses. We estimate the parameters of the per-pixel Gaussian model (mean and variance) online using a pre-defined learning rate. Note that while other more sophisticated background modeling and subtraction tech-

niques are available, we did not observe substantial improvement. Figure 5.6 shows the three-dimensional (X-Y-T) scattered plot of the foreground detection on the West and East cameras on a video with a flock of three birds. Figure 5.6(b) shows the projections of the 3D point cloud onto X-Y, Y-T, and X-T planes, respectively. We could visually spot the three flying birds. The challenge, however, lies in how to handle the high outliers ratio.

5.4.4 Geometric verification

Our coarse bird model (i.e., a straight line in a 3D video cube) consists of 5 parameters, including an initial spatial position in the image plane (2D), constant motion vectors (2D), and disparity from stereo vision (1D). The goal of geometric verification is to fit coarse bird models to the 3D point clouds with a significant portion of outliers from the foreground detection step. The most widely used robust fitting algorithms are (1) Generalized Hough Transform (GHT) and (2) RANSAC. We choose to perform geometric verification using RANSAC because of the demanding memory complexity in GHT for estimating 5D models.

A straightforward approach would be using RANSAC-based 3D line fitting method independently for each video and then solve the disparity by matching fitted lines in two views *after* the models in each video are found. However, such an approach does not exploit the available physical constraints presented in the stereo videos. For example, the two corresponding 3D lines in the stereo pair should be parallel, having the same y-coordinate at all frames, and with positive disparity values. To incorporate these constraints, we propose a stereo-based 3D line fitting algorithm. Specifically, of the detected foreground points from the stereo pair, we select random subsets of three detected points to estimate the bird model, where two points are drawn from one video, and one point is drawn from the other video.

Figure 5.4(b) illustrates the three-point hypothetical inlier. The proposed three-point subset sampling strategy offers several advantages. First, we can fully determine the 5D bird model using the selected three points. Second, we can quickly reject a large collection of model hypotheses that are not physically plausible by checking the disparity and temporal alignment. Third, as we also have disparity in the estimated model, we can simultaneously exploit the detected foreground points from both videos by compensating for the disparity.

We follow the standard RANSAC algorithm and count the number of inliers (number of foreground points fall inside the 3D tube). We then apply the J-linkage clustering algorithm [142] to group repeatedly sampled hypotheses. Once we have the grouped model hypothesis, we perform least squares fitting using all the inlier foreground points from both videos to compute a more reliable bird model estimation. We solve this refinement

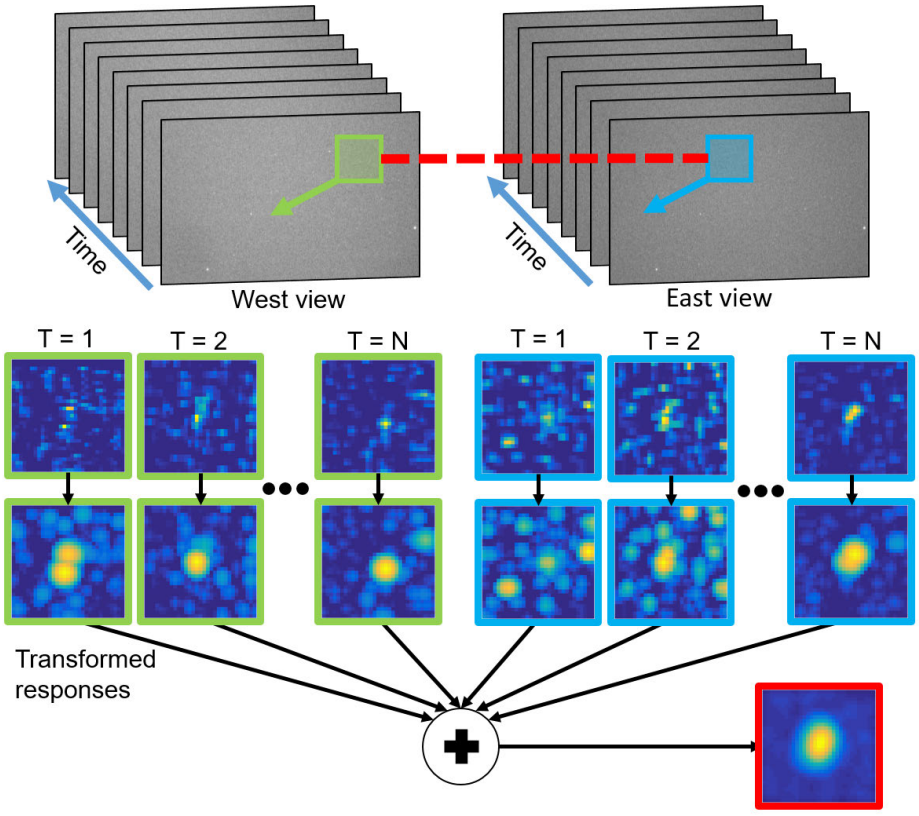


Figure 5.7: Trajectory verification. Given a 3D line model, we gather the spatial patches along the coarse trajectory from $T = 1$ (when the bird enters the frame) to $T = N$ (when the bird leaves the frame). These local responses are noisy and misaligned due to time-varying speed and directions. We transform the responses to account for spatial uncertainty.

step iteratively. Given an estimated disparity, we can solve the orientation using Singular Value Decomposition. In turn, we fix the orientation and update the disparity using least-square fitting.

5.4.5 Trajectory verification

While geometric verification can efficiently detect flying birds by exploiting the stereo vision constraints, we observe a high false positive rate due to inevitable noisy foreground detections. We address this issue by integrating signals along the bird's trajectory. Unlike geometric verification that fit models to *sparse* foreground candidates, trajectory verification exploits *dense* information across the entire video cube.

One way to achieve this is to use the corresponding matched filter that computes the average local response along its trajectory. However, the actual bird trajectory may not

be a perfect 3D straight line in the video because the bird may not fly along the same direction or maintain constant speed and altitude. Simply using the estimated coarse bird model to filter the videos is clearly sub-optimal as spatial misalignments lead to blurry accumulated response.

We address this problem by allowing the bird trajectory to be "deformed" as illustrated in Figure 5.7. We interpret the bird response at the predicted position using the coarse model at a frame as the local response for a "part". The detection of the bird can then be cast as the detection of a deformable part model. Specifically, we evaluate the score of a small window (e.g., 15×15) as

$$\text{score}(x,y) = \sum_{t=1}^{N_t} \max_{dx,dy} \left[R_t\left(x_t + dx, y_t + dy\right) - \alpha\left(dx^2 + dy^2\right) \right],$$

where R_t is the response map for foreground object, (x_t, y_t) is the predicted position at time t using the hypothesized 3D line from the geometric verification step, and α is the weights for allowing different levels of spatial deformation. As we also have the disparity estimation, we aggregate the scores from two views. We use the Generalized Distance Transform [120] to efficiently search over all possible deformations through time. These transformed responses can then be added together and ranked for verification.

5.5 Experimental Results

5.5.1 Implementation details

In foreground detection, we classify a pixel as a foreground if its intensity is greater than the mean background intensity by 2.75 standard deviations. In geometric verification, we keep model hypotheses with at least 5 inliers and reject the rest. In trajectory verification, we use 15×15 windows and set the spatial deformation parameter $\alpha = 0.5$. We fix these parameters throughout all experiments.

We process a video in a mini-batch mode, by dividing a long video sequence into a set of overlapping five-second sequences with a one-second interval. We detect birds in each video clip and cluster these detections in the nearby clips to generate our final results. In a video with frame rate 30 fps, we have in total 150 frames. For processing one 5 second video clip, our MATLAB implementation takes 7 seconds on a PC with 1.9GHz and 8 GB memory. The data and source code are available on the project website.[1]

[1] `https://sites.google.com/site/jbhuang0604`

Table 5.1: Quantitative performance

Method	Precision	Recall
Geometric verification only	6.08%	83.10%
Geometric and Trajectory verification	97.30%	83.72%

5.5.2 Evaluation on real videos

To evaluate the proposed method, we have developed a prototype stereo video system to capture videos of migrating birds at night. In what follows, we present the data collection steps and our results on real videos.

Data collection We use two low-light near-IR mono industrial VGA cameras to capture the stereo video. We chose the cameras because of their superior low-light sensitivity (10k–100k $\times$ more sensitive than consumer video cameras). The cameras have a spatial resolution of 640×480 pixels. We use a pair of 50 mm lenses and set the two cameras on tripods facing the sky with a two-meter baseline. We captured hours of stereo video on different nights and selected a 40-minutes long video from Spring migration for testing.

Quantitative results To evaluate performance, we developed a Graphical User Interface to allow experts to annotate the birds flying across video frames. In total, 86 birds were found in the video. A majority of the birds head North $+-20$ degrees. Among the 86 annotated birds, our method detects 74 of them, with two false positives and 12 missed detections. In Table 5.1, we show the quantitative performance of our algorithm. When using geometric verification only, we achieve 83.10% in recall. However, precision is very poor, with only 6.08%. Coupled with trajectory verification, precision rises above 95% with 83% recall. The automatic system detects 9 birds missed by the experts.

We further evaluate the relative contributions from (1) fusing information from two views and (2) using the deformable part model to account for the inevitable spatial uncertainty when using real videos of birds migrating at night. Specifically, we report the precision and recall values using the four variants. *One View*: use only the video from the West camera. *Two Views*: use both West and East videos. *Without deformation*: did not transform the scores in each local image patch. *With deformation*: use the generalized distance transform to allow spatial uncertainty.

To make the contribution of each term clear, Figure 5.8 shows the precision and recall of these four variants using a version of the system that does no post-processing to reduce false detections. In cases of integrating signals along the estimated trajectory (from geometric verification) in one view, both the precision and recall improve when we account for the spatial deformation. When using two views *without* accounting for

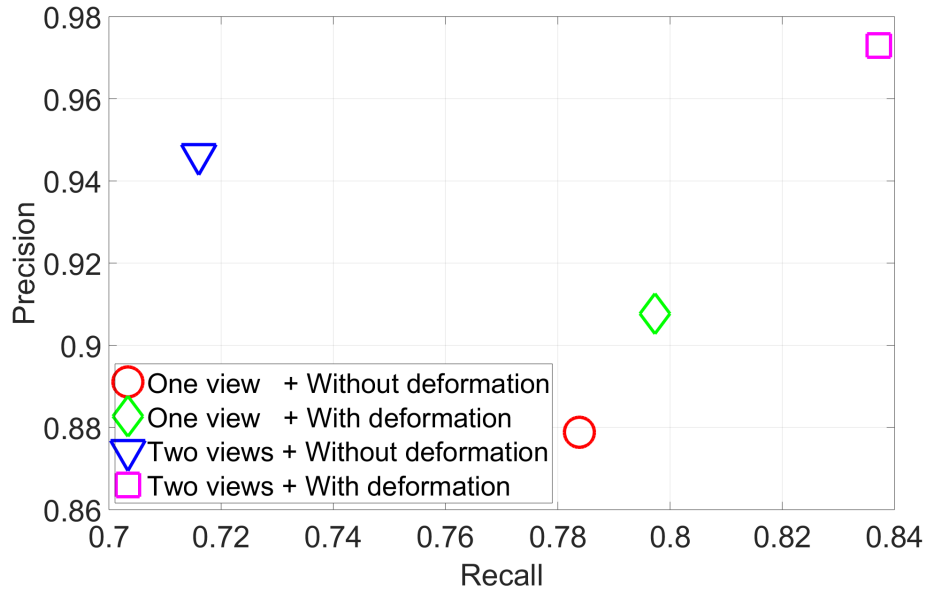


Figure 5.8: Precision and recall of four variants of the proposed trajectory verification approach on real videos.

the spatial deformation, we found that the recall drops significantly. We attribute the performance degradation to the imperfect disparity estimation between the two views. Integrating scores from two views without taking the spatial uncertainty into account, the results suggest that the algorithm may not be able to accumulate the weak signals due to the misalignment, and, therefore, fails to detect dim birds. Overall, the best performance is achieved by taking advantage of the stereo constraint while also allowing for deformation to account for spatial uncertainty.

Qualitative results In Figure 5.9, we show detection results in a variety of scenarios to demonstrate the effectiveness of the proposed approach. For example, our method can detect birds flying at altitudes ranging from 200 meters to more than 2,500 meters as well as at different directions and speeds. We can also handle multiple birds flying across the video frame. Unlike existing techniques that can only detect the presence of the birds, the direct visual analysis provides detailed measurements about the trajectory of *individual* birds. We believe such information may provide valuable insights about the behavior of migrating flocks.

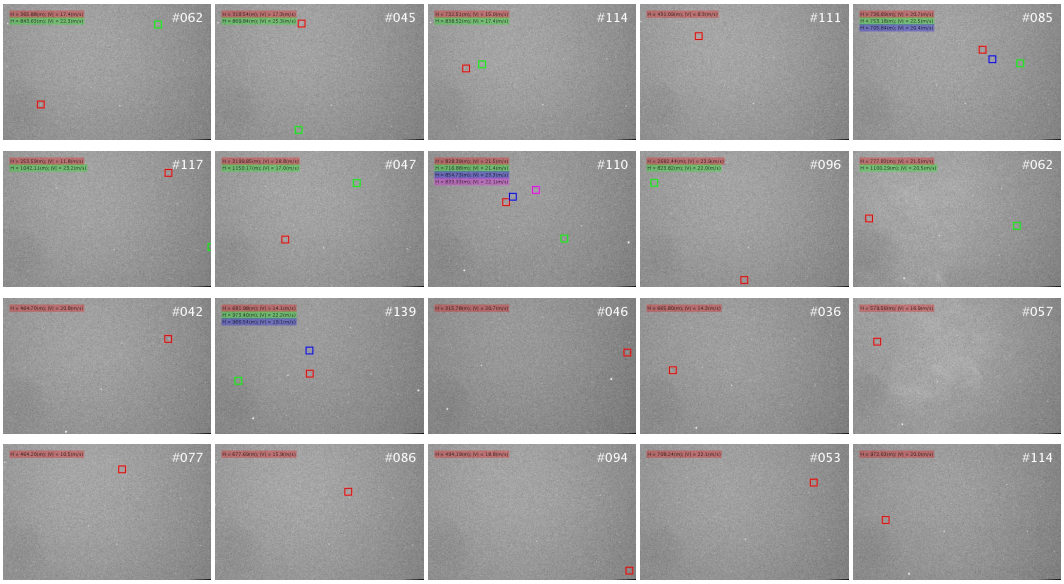
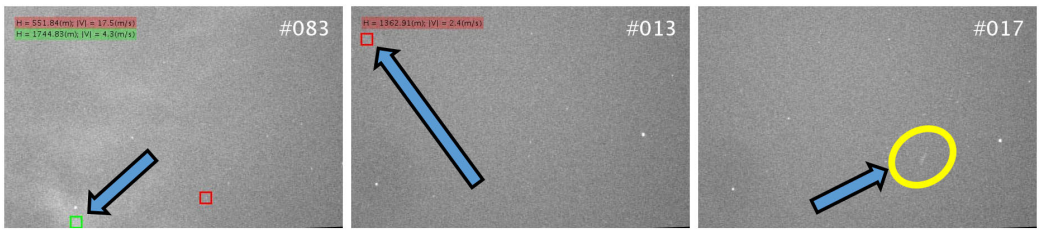


Figure 5.9: Detection results on real videos. Our system can handle diverse scenarios, e.g., single, multiple birds, birds flying parallel with each other, or birds flying at very different altitudes.

5.5.3 Discussion

Limitations One potential problem and limitation in evaluating the performance on real videos is that the groundtruth annotations are not available, and the human visual system may not be able to detect very dim, high-flying birds from the video. In the future, we plan to investigate a multi-modal solution (e.g., vision-based, acoustics-based, and weather radar) to this problem. Figure 5.10 shows a few of the limitations of our method. First, as our foreground detection is based on a statistical background modeling approach, we are not able to handle dynamic background or sudden illumination changes. For example, in Figure 5.10(a), our method falsely detects the moving cloud as a bird. Second, even with the use of stereo-based constraints for rejecting physically implausible detections (e.g., Figure 5.10(b)), our method may sometimes produce false positives due to the substantial noise in the video. One potential solution is to use three or more cameras covering the same patch of the night sky. Our framework could be extended to multi-camera settings to further improve the detection performance. Third, in Figure 5.10(c) we show that our method is robust to other types of flying objects. The altitude estimation provides important cues for separating migrating birds from high-flying objects (satellites or airplanes) and low-flying objects (insects).



(a) FP (cloud) (b) FP (noises) (c) TN (insect)

Figure 5.10: Interesting cases: (a) A false positive detection due to a moving cloud. (b) A false positive detection due to noise. (c) A true negative — the moving blob is an insect. Our system uses the estimated altitude to avoid confusion with high-flying objects (e.g., above 3000 meters) such as satellites or planes and low-flying objects (e.g., under 50 meters) such as insects.

5.6 Conclusions

We presented the first stereo-vision-based approach for monitoring migrating birds at night. From a pair of stereo videos, we perform stereo image rectification by detecting and registering stars. The core bird detection algorithm then consists of three main steps. First, we use statistical background modeling for foreground detection for each video. This produces a noisy three-dimensional point cloud. Second, we propose a novel RANSAC-based 3D line fitting that explicitly takes into account stereo vision constraints. Third, we apply deformable part modeling for handling the spatial uncertainty of birds due to time-varying speed and orientation. Through evaluation on real videos captured from a physical setup, we demonstrate the effectiveness of the proposed method. We believe the new capabilities will make a significant impact on computational ecology.

While our work addresses a particular application, the approach for detecting and tracking multiple small targets in 3D volumetric data with very low SNR using multiple cameras is general and potentially can be applied to many other important problems. In this work, we show how to leverage the underlying physical constraints and domain knowledge to achieve physically plausible detection that otherwise would not be feasible due to the high level of noise.

CHAPTER 6

CONCLUSIONS

In this dissertation, we have studied the use of physically grounded visual constraints for addressing visual analysis and synthesis problems. First, we propose to incorporate scene-specific geometric constraints as well as translational symmetry into a patch-based optimization framework. Using this framework, we demonstrate two important applications: (1) filling unknown pixels of an image caused by the removal of a foreground or background object and (2) predicting the missing high-frequency components that are not resolvable in the low-resolution observation. We show that such approaches are not only conceptually simple, but also outperform existing approaches by a large margin. Second, we investigate the role of physically grounded constraints in the spatio-temporal case. Specifically, we propose to jointly estimate the missing colors and motion fields for temporally coherent completion of dynamic video. Our algorithm achieves convincing synthesis results without making any assumptions about the input video (such as static cameras or periodic moving objects). For the visual analysis task, we exploit the stereo camera constraints for detecting and tracking multiple small targets in low-SNR videos. We apply the technique to a practical application, migrating bird monitoring, as the first video-based solution for detecting migrating birds at night.

The main contribution of this thesis is in incorporating physically grounded constraints for processing, synthesizing, and extracting information from images and videos. We demonstrate such techniques through applications in image/video completion, super-resolution, and small target detection and tracking. We hope that the thesis work not only extends the capacity of algorithms to individual problems, but also sheds light on using physically grounded visual constraints in addressing other problems in computer vision.

REFERENCES

[1] J.-B. Huang, S. B. Kang, N. Ahuja, and J. Kopf, "Image completion using planar structure guidance," *ACM Trans. on Graphics*, vol. 33, no. 4, p. 129, 2014.

[2] J.-B. Huang, A. Singh, and N. Ahuja, "Single image super-resolution from transformed self-exemplars," in *CVPR*, 2015.

[3] J.-B. Huang, R. Caruana, A. Farnsworth, S. Kelling, and N. Ahuja, "Detecting migrating birds at night," in *CVPR*, 2016.

[4] J.-B. Huang and C.-S. Chen, "Moving cast shadow detection using physics-based features," in *CVPR*, 2009.

[5] J.-B. Huang and M.-H. Yang, "Fast sparse representation with prototypes." in *CVPR*, 2010.

[6] Z. Hu, J.-B. Huang, and M.-H. Yang, "Single image deblurring with adaptive dictionary learning," in *ICIP*, 2010.

[7] C.-Y. Yang, J.-B. Huang, and M.-H. Yang, "Exploiting self-similarities for single frame super-resolution," in *ACCV*, 2010.

[8] J.-B. Huang and N. Ahuja, "Saliency detection via divergence analysis: A unified perspective," in *icpr*, 2012.

[9] J.-B. Huang, J. Kopf, N. Ahuja, and S. B. Kang, "Transformation guided image completion," in *ICCP*, 2013.

[10] J.-B. Huang, Q. Cai, Z. Liu, N. Ahuja, and Z. Zhang, "Towards accurate and robust cross-ratio based gaze trackers through learning from simulation," in *Proceedings of the Symposium on Eye Tracking Research and Applications*, 2014.

[11] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *ICCV*, 2015.

[12] D. Li, J.-B. Huang, Y. Li, S. Wang, and M.-H. Yang, "Weakly supervised object localization with progressive domain adaptation," in *CVPR*, 2016.

[13] W.-S. Lai, J.-B. Huang, Z. Hu, N. Ahuja, and M.-H. Yang, "A comparative study for single image blind deblurring," in *CVPR*, 2016.

[14] D. Li, W.-C. Hung, J.-B. Huang, S. Wang, N. Ahuja, and M.-H. Yang, "Unsupervised visual representation learning by graph-based consistent constraints," in *ECCV*, 2016.

[15] Y. Li, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep joint image filter," in *ECCV*, 2016.

[16] S. Zhang, Y. Gong, J.-B. Huang, J. Lim, J. Wang, N. Ahuja, and M.-H. Yang, "Tracking persons-of-interest via adaptive discriminative features," in *ECCV*, 2016.

[17] S. Darabi, E. Shechtman, C. Barnes, D. B. Goldman, and P. Sen, "Image Melding: Combining inconsistent images using patch-based synthesis," *ACM Trans. on Graphics*, vol. 31, no. 4, 2012.

[18] K. He and J. Sun, "Statistics of patch offsets for image completion," in *ECCV*, 2012.

[19] Y. Wexler, E. Shechtman, and M. Irani, "Space-time completion of video," *IEEE TPAMI*, vol. 29, no. 3, pp. 463–476, 2007.

[20] C. Barnes, E. Shechtman, A. Finkelstein, and D. Goldman, "PatchMatch: a randomized correspondence algorithm for structural image editing," *ACM Trans. on Graphics*, vol. 28, no. 3, p. 24, 2009.

[21] A. Mansfield, M. Prasad, C. Rother, T. Sharp, P. Kohli, and L. Van Gool, "Transforming image completion," in *BMVC*, 2011.

[22] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 2000, pp. 417–424.

[23] C. Ballester, M. Bertalmio, V. Caselles, G. Sapiro, and J. Verdera, "Filling-in by joint interpolation of vector fields and gray levels," *IEEE TIP*, vol. 10, no. 8, pp. 1200–1211, 2001.

[24] M. Bertalmio, L. Vese, G. Sapiro, and S. Osher, "Simultaneous structure and texture image inpainting," *IEEE TIP*, vol. 12, no. 8, pp. 882–889, 2003.

[25] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *ICCV*, vol. 2, 1999.

[26] A. A. Efros and W. T. Freeman, "Image quilting for texture synthesis and transfer," *ACM Trans. on Graphics*, vol. 20, no. 3, pp. 341–346, 2001.

[27] A. Criminisi, P. Pérez, and K. Toyama, "Region filling and object removal by exemplar-based image inpainting," *IEEE TIP*, vol. 13, no. 9, pp. 1200–1212, 2004.

[28] V. Kwatra, I. Essa, A. Bobick, and N. Kwatra, "Texture optimization for example-based synthesis," in *ACM Trans. on Graphics*, vol. 24, no. 3, 2005, pp. 795–802.

[29] N. Komodakis and G. Tziritas, "Image completion using efficient belief propagation via priority scheduling and dynamic pruning," *IEEE TIP*, vol. 16, no. 11, pp. 2649–2661, 2007.

[30] Y. Pritch, E. Kav-Venaki, and S. Peleg, "Shift-map image editing," in *ICCV*, 2009.

[31] J. Hays and A. A. Efros, "Scene completion using millions of photographs," in *ACM Trans. on Graphics*, vol. 26, no. 3, 2007, p. 4.

[32] Y. Zhang, J. Xiao, J. Hays, and P. Tan, "Framebreak: Dramatic image extrapolation by guided shift-maps," in *CVPR*, 2013, pp. 1171–1178.

[33] O. Whyte, J. Sivic, and A. Zisserman, "Get out of my picture! internet-based inpainting," in *BMVC*, 2009.

[34] J. Jia and C. Tang, "Image repairing: Robust image synthesis by adaptive nd tensor voting," in *CVPR*, 2003.

[35] H. Huang, K.Yin, M. Gong, D. Lischinski, D. Cohen-Or, U. Ascher, and B. Chen, "Mind the gap: Tele-registration for structure-driven image completion," *ACM Trans. on Graphics*, vol. 32, pp. 174:1–174:10, 2013.

[36] J. Kopf, W. Kienzle, S. Drucker, and S. B. Kang, "Quality prediction for image completion," *ACM Trans. on Graphics*, vol. 31, no. 6, p. 131, 2012.

[37] A. Hertzmann, C. E. Jacobs, N. Oliver, B. Curless, and D. H. Salesin, "Image analogies," *ACM Trans. on Graphics*, vol. 20, no. 3, pp. 327–340, 2001.

[38] J. Sun, L. Yuan, J. Jia, and H. Shum, "Image completion with structure propagation," *ACM Trans. on Graphics*, vol. 24, no. 3, pp. 861–868, 2005.

[39] D. Pavić, V. Schönefeld, and L. Kobbelt, "Interactive image completion with perspective correction," *The Visual Computer*, vol. 22, no. 9, pp. 671–681, 2006.

[40] Y. Liu, W.-C. Lin, and J. Hays, "Near-regular texture analysis and manipulation," *ACM Trans. on Graphics*, vol. 23, no. 3, pp. 368–376, 2004.

[41] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[42] O. Chum and J. Matas, "Planar affine rectification from change of scale," in *ACCV*, 2010.

[43] Z. Zhang, A. Ganesh, X. Liang, and Y. Ma, "TILT: Transform invariant low-rank textures," *International Journal of Computer Vision*, vol. 99, no. 1, pp. 1–24, 2012.

[44] D. Aiger, D. Cohen-Or, and N. J. Mitra, "Repetition maximization based texture rectification," *Computer Graphics Forum*, vol. 31, no. 2pt2, pp. 439–448, 2012.

[45] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, 2004.

[46] O. Barinova, V. Konushin, A. Yakubenko, K. Lee, H. Lim, and A. Konushin, "Fast automatic single-view 3-d reconstruction of urban scenes," in *ECCV*, 2008.

[47] Y. Liu, H. Hel-Or, and C. Kaplan, *Computational symmetry in computer vision and computer graphics*. Now Publishers, 2010.

[48] D. Comaniciu and P. Meer, "Mean Shift: A robust approach toward feature space analysis," *IEEE TPAMI*, vol. 24, no. 5, pp. 603–619, 2002.

[49] W. T. Freeman, T. Jones, and E. Pasztor, "Example-based super-resolution," *IEEE CG&A*, vol. 22, no. 2, pp. 56–65, 2002.

[50] H. Chang, D.-Y. Yeung, and Y. Xiong, "Super-resolution through neighbor embedding," in *CVPR*, 2004.

[51] J. Yang, J. Wright, T. S. Huang, and Y. Ma, "Image super-resolution via sparse representation," *IEEE TIP*, vol. 19, no. 11, pp. 2861–2873, 2010.

[52] C.-Y. Yang and M.-H. Yang, "Fast direct super-resolution by simple functions," in *ICCV*, 2013.

[53] R. Timofte, V. De, and L. V. Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *ICCV*, 2013.

[54] R. Timofte, V. De Smet, and L. Van Gool, "A+: Adjusted anchored neighborhood regression for fast super-resolution," in *ACCV*, 2014.

[55] C. Dong, C. C. Loy, K. He, and X. Tang, "Learning a deep convolutional network for image super-resolution," in *ECCV*, 2014.

[56] M. Ebrahimi and E. R. Vrscay, "Solving the inverse problem of image zooming using self-examples," in *Image analysis and Recognition*, 2007.

[57] D. Glasner, S. Bagon, and M. Irani, "Super-resolution from a single image," in *ICCV*, 2009.

[58] G. Freedman and R. Fattal, "Image and video upscaling from local self-examples," *ACM Trans. on Graphics*, vol. 30, no. 2, p. 12, 2011.

[59] A. Singh and N. Ahuja, "Super-resolution using sub-band self-similarity," in *ACCV*, 2014.

[60] M. Barnsley, *Fractals Everywhere*. Academic Press Professional, Inc., 1988.

[61] M. Zontak and M. Irani, "Internal statistics of a single natural image," in *CVPR*, 2011.

[62] M. Irani and S. Peleg, "Improving resolution by image registration," *CVGIP: Graphical models and image processing*, vol. 53, no. 3, pp. 231–239, 1991.

[63] K. I. Kim and Y. Kwon, "Single-image super-resolution using sparse regression and natural image prior," *IEEE TPAMI*, vol. 32, no. 6, pp. 1127–1133, 2010.

[64] J. Yang, Z. Wang, Z. Lin, S. Cohen, and T. Huang, "Coupled dictionary training for image super-resolution," *IEEE TIP*, vol. 21, no. 8, pp. 3467–3478, 2012.

[65] R. Zeyde, M. Elad, and M. Protter, "On single image scale-up using sparse-representations," in *International Conference on Curves and Surfaces*, 2012.

[66] J. Sun, Z. Xu, and H.-Y. Shum, "Gradient profile prior and its applications in image super-resolution and enhancement," *IEEE TIP*, vol. 20, no. 6, pp. 1529–1542, 2011.

[67] R. Fattal, "Image upsampling via imposed edge statistics," *ACM Trans. on Graphics*, vol. 26, no. 3, p. 95, 2007.

[68] Y. HaCohen, R. Fattal, and D. Lischinski, "Image upsampling via texture hallucination," in *ICCP*, 2010, pp. 1–8.

[69] J. Sun, J. Zhu, and M. Tappen, "Context-constrained hallucination for image super-resolution," in *CVPR*, 2010.

[70] L. Sun and J. Hays, "Super-resolution from Internet-scale scene matching," in *ICCP*, 2012.

[71] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *CVPR*, 2005.

[72] J. Yang, Z. Lin, and S. Cohen, "Fast image super-resolution based on in-place example regression," in *CVPR*, 2013.

[73] T. Michaeli and M. Irani, "Nonparametric blind super-resolution," in *ICCV*, 2013.

[74] A. Singh, F. Porikli, and N. Ahuja, "Super-resolving noisy images," in *CVPR*, 2014.

[75] A. Singh and N. Ahuja, "Sub-band energy constraints for self-similarity based super-resolution," in *ICPR*, 2014.

[76] Y. Zhu, Y. Zhang, and A. L. Yuille, "Single image super-resolution using deformable patches," in *CVPR*, 2014.

[77] C. Fernandez-Granda and E. J. Candes, "Super-resolution via transform-invariant group-sparse regularization," in *ICCV*, 2013.

[78] M. Bleyer, C. Rhemann, and C. Rother, "PatchMatch stereo-stereo matching with slanted support windows." in *BMVC*, 2011.

[79] M. Hornáček, C. Rhemann, M. Gelautz, and C. Rother, "Depth super resolution by rigid body self-similarity in 3d," in *CVPR*, 2013.

[80] M. Hornáček, F. Besse, J. Kautz, A. Fitzgibbon, and C. Rother, "Highly overparameterized optical flow using patchmatch belief propagation," in *ECCV*, 2014.

[81] C. Barnes, E. Shechtman, D. Goldman, and A. Finkelstein, "The generalized patchmatch correspondence algorithm," in *ECCV*, 2010.

[82] Y. HaCohen, E. Shechtman, D. Goldman, and D. Lischinski, "Non-rigid dense correspondence with applications for image enhancement," *ACM Trans. on Graphics*, vol. 30, no. 4, p. 70, 2011.

[83] J. J. Koenderink, A. J. Van Doorn et al., "Affine structure from motion," *JOSA A*, vol. 8, no. 2, pp. 377–385, 1991.

[84] O. Chum and J. Matas, "Planar affine rectification from change of scale," in *ACCV*, 2010.

[85] C.-Y. Yang, C. Ma, and M.-H. Yang, "Single-image super-resolution: A benchmark," in *ECCV*, 2014.

[86] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *ICCV*, 2001.

[87] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE TIP*, vol. 13, no. 4, pp. 600–612, 2004.

[88] A. Newson, A. Almansa, M. Fradet, Y. Gousseau, P. Pérez et al., "Video inpainting of complex scenes," *SIAM Journal on Imaging Sciences*, 2014.

[89] M. Granados, K. I. Kim, J. Tompkin, J. Kautz, and C. Theobalt, "Background inpainting for videos with dynamic objects and a free-moving camera," in *ECCV*, 2012.

[90] M. Granados, J. Tompkin, K. Kim, O. Grau, J. Kautz, and C. Theobalt, "How not to be seenobject removal from videos of crowded scenes," in *Computer Graphics Forum*, vol. 31, no. 2pt1, 2012, pp. 219–228.

[91] M. Strobel, J. Diebold, and D. Cremers, "Flow and color inpainting for video completion," in *German Conference on Pattern Recognition*, 2014.

[92] J. Jia, Y.-W. Tai, T.-P. Wu, and C.-K. Tang, "Video repairing under variable illumination using cyclic motions," *IEEE TPAMI*, vol. 28, no. 5, pp. 832–839, 2006.

[93] M. Roxas, T. Shiratori, and K. Ikeuchi, "Video completion via spatio-temporally consistent motion inpainting," *Information and Media Technologies*, vol. 9, no. 4, pp. 500–504, 2014.

[94] C. Guillemot and O. Le Meur, "Image inpainting: Overview and recent advances," *IEEE Signal Processing Magazine*, vol. 31, no. 1, pp. 127–144, 2014.

[95] S. Ilan and A. Shamir, "A survey on data-driven video completion," *Computer Graphics Forum*, 2014.

[96] I. Drori, D. Cohen-Or, and H. Yeshurun, "Fragment-based image completion," in *ACM Trans. on Graphics*, vol. 22, no. 3, 2003, pp. 303–312.

[97] K. A. Patwardhan, G. Sapiro, and M. Bertalmio, "Video inpainting of occluding and occluded objects," in *ICIP*, 2005.

[98] K. A. Patwardhan, G. Sapiro, and M. Bertalmío, "Video inpainting under constrained camera motion," *IEEE TIP*, vol. 16, no. 2, pp. 545–553, 2007.

[99] V. Kwatra, A. Schödl, I. Essa, G. Turk, and A. Bobick, "Graphcut textures: Image and video synthesis using graph cuts," in *ACM Trans. on Graphics*, vol. 22, no. 3, 2003, pp. 277–286.

[100] T. Shiratori, Y. Matsushita, X. Tang, and S. B. Kang, "Video completion by motion field transfer," in *CVPR*, 2006.

[101] O. Jamriška, J. Fišer, P. Asente, J. Lu, E. Shechtman, and D. Sỳkora, "Lazyfluids: Appearance transfer for fluid animations," *ACM Trans. on Graphics*, vol. 34, no. 4, p. 92, 2015.

[102] K. S. Bhat, S. M. Seitz, J. K. Hodgins, and P. K. Khosla, "Flow-based video synthesis and editing," in *ACM Trans. on Graphics*, vol. 23, no. 3, 2004, pp. 360–363.

[103] N. K. Kalantari, E. Shechtman, C. Barnes, S. Darabi, D. B. Goldman, and P. Sen, "Patch-based high dynamic range video." *ACM Trans. on Graphics*, vol. 32, no. 6, pp. 202–1, 2013.

[104] T. Xue, M. Rubinstein, C. Liu, and W. T. Freeman, "A computational approach for obstruction-free photography," *ACM Trans. on Graphics*, vol. 34, no. 4, p. 79, 2015.

[105] C. Liu, "Beyond pixels: Exploring new representations and applications for motion analysis," Ph.D. dissertation, Massachusetts Institute of Technology, 2009.

[106] O. Mac Aodha, A. Humayun, M. Pollefeys, and G. J. Brostow, "Learning a confidence measure for optical flow," *IEEE TPAMI*, vol. 35, no. 5, pp. 1107–1120, 2013.

[107] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung, "A benchmark dataset and evaluation methodology for video object segmentation," in *CVPR*, 2016.

[108] H. Higuchi, "Bird migration and the conservation of the global environment," *Journal of Ornithology*, vol. 153, no. 1, pp. 3–14, 2012.

[109] B. Bhanu, "Automatic target recognition: State of the art survey," *IEEE Transactions on Aerospace and Electronic Systems*, no. 4, pp. 364–379, 1986.

[110] W. Zhang, M. Cong, and L. Wang, "Algorithms for optical weak small targets detection and tracking: Review," in *International Conference on Neural Networks and Signal Processing*, 2003.

[111] C. Kirbas and F. Quek, "A review of vessel extraction techniques and algorithms," *ACM Computing Surveys*, vol. 36, no. 2, pp. 81–121, 2004.

[112] M. Galun, R. Basri, and A. Brandt, "Multiscale edge detection and fiber enhancement using differences of oriented means," in *ICCV*, 2007.

[113] K. Guo and D. Labate, "Optimally sparse multidimensional representation using shearlets," *SIAM journal on mathematical analysis*, vol. 39, no. 1, pp. 298–318, 2007.

[114] D. L. Donoho and X. Huo, "Beamlets and multiscale image analysis," *Multiscale and Multiresolution Methods: Theory and Applications*, vol. 20, p. 149, 2002.

[115] D. L. Donoho and O. Levi, "Fast x-ray and beamlet transforms for three-dimensional data," *Modern signal processing*, vol. 46, 2002.

[116] S. Alpert, M. Galun, B. Nadler, and R. Basri, "Detecting faint curved edges in noisy images," in *ECCV*, 2010.

[117] N. Ofir, M. Galun, B. Nadler, and R. Basri, "Fast detection of curved edges at low snr," *arXiv:1505.06600*, 2015.

[118] Z. Wu and M. Betke, "Global optimization for coupled detection and data association in multiple object tracking," vol. 143, pp. 25–37, 2016.

[119] A. Attanasi, A. Cavagna, L. Del Castello, I. Giardina, A. Jelic, S. Melillo, L. Parisi, F. Pellacini, E. Shen, E. Silvestri et al., "Greta-a novel global and recursive tracking algorithm in three dimensions," *IEEE TPAMI*, vol. 37, no. 12, pp. 2451–2463, 2015.

[120] P. Felzenszwalb and D. Huttenlocher, "Distance transforms of sampled functions," Cornell University, Tech. Rep., 2004.

[121] A. M. Dokter, F. Liechti, H. Stark, L. Delobbe, P. Tabary, and I. Holleman, "Bird migration flight altitudes studied by a network of operational weather radars," *Journal of The Royal Society Interface*, vol. 8, no. 54, pp. 30–43, 2011.

[122] D. Sheldon, A. Farnsworth, J. Irvine, B. Van Doren, K. Webb, T. G. Dietterich, and S. Kelling, "Approximate bayesian inference for reconstructing velocities of migrating birds from weather radar," in *AAAI Conference on Artificial Intelligence*, 2013.

[123] A. Farnsworth, B. M. Van Doren, W. M. Hochachka, D. Sheldon, K. Winner, J. Irvine, J. Geevarghese, and S. Kelling, "A characterization of autumn nocturnal migration detected by weather surveillance radars in the northeastern us," *Ecological Applications*, 2015.

[124] T. A. Marques, L. Thomas, S. W. Martin, D. K. Mellinger, J. A. Ward, D. J. Moretti, D. Harris, and P. L. Tyack, "Estimating animal population density using passive acoustics," *Biological Reviews*, 2012.

[125] R. Bardeli, D. Wolff, F. Kurth, M. Koch, K.-H. Tauchert, and K.-H. Frommolt, "Detecting bird sounds in a complex acoustic environment and application to bioacoustic monitoring," *Pattern Recognition Letters*, vol. 31, no. 12, pp. 1524–1534, 2010.

[126] W. R. Evans and K. V. Rosenberg, "Acoustic monitoring of night-migrating birds: A progress report," *Strategies for bird conservation: The Partners in Flight planning process*, pp. 1–5, 2000.

[127] P. Berthold, *Bird Migration: A General Survey*. Oxford University Press, 2001, vol. 12.

[128] T. Alerstam, A. Hedenström, and S. Åkesson, "Long-distance migration: Evolution and determinants," *Oikos*, vol. 103, no. 2, pp. 247–260, 2003.

[129] E. S. Bridge, K. Thorup, M. S. Bowlin, P. B. Chilson, R. H. Diehl, R. W. Fléron, P. Hartl, K. Roland, J. F. Kelly, W. D. Robinson et al., "Technology on the move: Recent and forthcoming innovations for tracking migratory birds," *BioScience*, vol. 61, no. 9, pp. 689–698, 2011.

[130] F. Liechti, B. Bruderer, and H. Paproth, "Quantification of nocturnal bird migration by moonwatching: Comparison with radar and infrared observations (cuantificación de la migración nocturna de aves observando la luna: Comparación con observaciones de radar e intrarrojas)," *Journal of Field Ornithology*, pp. 457–468, 1995.

[131] K. G. Horton, W. G. Shriver, and J. J. Buler, "A comparison of traffic estimates of nocturnal flying animals using radar, thermal imaging, and acoustic recording," *Ecological Applications*, vol. 25, no. 2, pp. 390–401, 2015.

[132] S. D. Blostein and T. S. Huang, "Detecting small, moving objects in image sequences using sequential hypothesis testing," *IEEE Transactions on Signal Processing*, vol. 39, no. 7, pp. 1611–1629, 1991.

[133] S. D. Deshpande, H. E. Meng, R. Venkateswarlu, and P. Chan, "Max-mean and max-median filters for detection of small targets," in *SPIE's International Symposium on Optical Science, Engineering, and Instrumentation*, 1999.

[134] X. Bai and F. Zhou, "Analysis of new top-hat transformation and the application for infrared dim small target detection," *Pattern Recognition*, vol. 43, no. 6, pp. 2145–2156, 2010.

[135] T.-W. Bae, F. Zhang, and I.-S. Kweon, "Edge directional 2d lms filter for infrared small target detection," *Infrared Physics & Technology*, vol. 55, no. 1, pp. 137–145, 2012.

[136] M. Ballerini et al., "Interaction ruling animal collective behavior depends on topological rather than metric distance: Evidence from a field study," *Proceedings of the National Academy of Sciences*, vol. 105, no. 4, pp. 1232–1237, 2008.

[137] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[138] D. H. Ballard, "Generalizing the hough transform to detect arbitrary shapes," *Pattern recognition*, vol. 13, no. 2, pp. 111–122, 1981.

[139] Z. Zhang, "Determining the epipolar geometry and its uncertainty: A review," *IJCV*, vol. 27, no. 2, pp. 161–195, 1998.

[140] C. Loop and Z. Zhang, "Computing rectifying homographies for stereo vision," in *CVPR*, 1999.

[141] P. J. Besl and N. D. McKay, "A method for registration of 3-d shapes," *IEEE TPAMI*, vol. 14, no. 2, pp. 239–256, 1992.

[142] R. Toldo and A. Fusiello, "Robust multiple structures estimation with j-linkage," in *ECCV*, 2008.