Exploiting Visual Motion to Understand Our Visual World

by

Tianfan Xue

B.Eng., Computer Science, Tsinghua University, 2009
M.phil., Information Engineering, Chinese University of Hong Kong, 2011

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in Electrical Engineering and Computer Science
at the Massachusetts Institute of Technology

September 2017

© 2017 Massachusetts Institute of Technology
All Rights Reserved.

Signature redacted

Department of Electrical Engineering and Computer Science
August 14, 2017

Certified by: Signature redacted

Prof. William T. Freeman
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by: Signature redacted

Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Committee for Graduate Students
DISCLAIMER NOTICE

Due to the condition of the original material, there are unavoidable flaws in this reproduction. We have made every effort possible to provide you with the best copy available.

Thank you.

Thesis contains pagination irregularities.
To my parents and to my wife
Exploiting Visual Motion to Understand Our Visual World

by Tianfan Xue

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

Abstract

Motion is important for understanding our visual world. The human visual system relies heavily on motion perception to recognize the movement of objects, to infer the 3D geometry of a scene, and to perceive the emotions of other people. Modern computer vision systems also use motion signals extracted from video sequences to infer high-level visual concepts, including human activities and abnormal events. Both human and computer visual systems try to perceive changes in the 3D physical world through its 2D projection, either on the image plane or on our retinas.

The observed 2D pixel movement is the result of several factors. First, the image sensor might move, inducing egocentric motion, even when the scene is static. Second, the medium between objects and a camera might change and affect how light transmits from the objects to the sensor, like the shimmering in a hot-road mirage. Finally, the objects in a scene might move, either actively, like a person walking along a street, or passively, like a tree branch that is vibrating due to wind. All of these movements reveal information about our visual world.

In this dissertation, we will discuss how to infer physical properties of our visual world from observed 2D movement. First, we show how to infer the depth of a scene from egocentric motion and use this to remove undesired visual obstructions. Second, we relate the slight wiggling motion due to refraction to the movement of hot air and infer the location and velocity of the airflow. Last, we illustrate how to infer the physical properties of objects, such as their deformation space or internal structure, from their motion.
Acknowledgments

I would first like to thank my advisor, Prof. William T. Freeman, for his support and guidance during my Ph.D. I also would like to thank to Prof. Frédo Durand, Dr. Ce liu, and Dr. Miki Rubinstein, who I have closely collaborated when I working on the projects presented in this work. I would also like to thank my other collaborators in this work: Prof. Anat Levin, Prof. Josh Tenenbaum, Dr. Hossein Mobahi, Dr. Neal Wadwa, Katie Bouman, Jiajun Wu, Zhoutong Zhang, and Chengkai Zhang.

I would also like to thank Dr. Richard Szeliski for his inspiring discussions. During three summer internships in Microsoft Research and Facebook Research, I also had the pleasure working or discussion with Dr. Ce Liu, Dr. Miki Rubinstein, Dr. Richard Szeliski, Dr. Johannes Kopf, Dr. Sudipta Sinha, Dr. Sing Bing Kang, Dr. Baoyuan Wang, Dr. Michael Cohen, and Dr. Matt Uyttendaele, and Peter Hedman.

I also had the pleasure work with Prof. Antonio Torralba, Prof. Josh Tenenbaum, Prof. Joseph Lim, Dr. Yuandong Tian, Jiajun Wu, Chengkai Zhang, on separate projects that are not included in this thesis [144, 145].

I would like to acknowledge grants that supported my research throughout my Ph.D: Shell International Exploration & Production Inc. and Motion SensingWi-Fi Sensor Networks Co. under grant No. 6925133, ONR MURI grant N00014-09-1-1051, ONR MURI 6923196.

Finally, I would like to thank my parents and my wife. Without their support and encouragement, I could not have finish this work.
# Contents

Abstract 3

Acknowledgments 5

List of Figures 13

1 Introduction 1

2 Decoupling Layers in Videos Captured by Moving Cameras 5
   2.1 Introduction ............................................................... 5
   2.2 Related Work ............................................................ 7
   2.3 Formulation ............................................................. 8
   2.4 Motion-based Decomposition ......................................... 10
      2.4.1 Formulation ......................................................... 10
      2.4.2 Optimization ....................................................... 12
      2.4.3 Initialization ....................................................... 16
   2.5 Results ................................................................. 18

3 Measuring Fluid Motion from Videos 25
   3.1 Related Work ............................................................ 27
   3.2 Measuring Fluid Depth and Velocity from Video using Refractive Wiggles .................................................. 28
      3.2.1 Refraction Wiggles and Refractive Constancy .................. 31
      3.2.2 Refractive Flow ...................................................... 33
      3.2.3 Probabilistic Refractive Flow ..................................... 34
      3.2.4 Refractive Stereo ................................................... 35
      3.2.5 Experiments .......................................................... 36
      3.2.6 Refractive Flow ...................................................... 36
      3.2.7 Refractive Stereo ................................................... 39
3.3 The Aperture Problem for Refractive Motion ........................................ 41
  3.3.1 A Toy Example ................................................................. 42
  3.3.2 Problem Definition ............................................................ 43
  3.3.3 Aperture Theory for Refractive Motion ..................................... 45
    First Order Observation ......................................................... 47
    Second Order Observation ..................................................... 49
  3.3.4 Experiments ................................................................. 51

4 Estimating the Structure of Objects from the Spectrum of Their Vibration 55
  4.1 Introduction ................................................................. 55
  4.2 Related Work ............................................................... 56
  4.3 Problem Definition .......................................................... 57
    4.3.1 A Physics-Based Link Model .......................................... 58
    4.3.2 ODE of Node Vibration ............................................... 59
    4.3.3 Inferring the Mode of each Sub-branch ............................ 60
  4.4 Extracting Motion and Appearance Features ................................ 62
  4.5 Inference ................................................................. 63
  4.6 Evaluations ............................................................... 65

5 Synthesizing the Movement of Objects in the Future 71
  5.1 Introduction ............................................................... 71
  5.2 Related Work ............................................................... 73
  5.3 Problem Definition .......................................................... 74
    5.3.1 A Toy Example ........................................................ 74
    5.3.2 Conditional Variational Autoencoder ................................ 76
  5.4 Method ................................................................. 77
    5.4.1 Layered Motion Representations and Cross Convolutional Networks 77
    5.4.2 Network Structure ..................................................... 78
  5.5 Evaluations ............................................................... 79
    5.5.1 Movement of 2D Shapes ............................................... 80
    5.5.2 Movement of Video Game Sprites ................................... 82
    5.5.3 Movement in Real Videos Captured in the Wild .................... 83
    5.5.4 Zero-Shot Visual Analogy-Making ................................... 84
    5.5.5 Visualizing Feature Maps ............................................. 85
    5.5.6 Dimension of Latent Representation z ............................... 85
List of Figures

1.1 Understanding our visual world from observed 2D motions caused by different factors. (a) A common setup for capturing. (b) Decoupling different layers based on motion parallax. (c) Measuring the movement of fluid objects from distortion due to refraction. (d) Understanding the deformation space of an object by observing how it moves. (e) Estimating the structure of objects from their movement. .................................................. 2

2.1 We present an algorithm for taking pictures through reflective or occluding elements such as windows and fences. The input to our algorithm is a set of images taken by the user while slightly scanning the scene with a camera/phone (a), and the output is two images: a clean image of the (desired) background scene, and an image of the reflected or occluding content (b). Our algorithm is fully automatic, can run on mobile devices, and allows taking pictures through common visual obstacles, producing images as if they were not there. The full image sequences and a closer comparison between the input images and our results are available in the supplementary material. .................................................. 5

2.2 The image formation model. A camera image of a desired scene though (a) a reflecting surface, and through (b) a partial obstruction. .................................................. 10

2.3 Algorithm pipeline. Our algorithm consists of two steps: initialization and iterative optimization. Initialization: we first calculate the motion vectors on extracted edge pixels from the input images (we thicken the edge mask for a better visualization). Then we fit two perspective transforms (one for each layer) to the estimated edge flow field and assign each edge pixel to either the background layer or the obstruction layer. This results in two sets of sparse flow fields for the two layers (top right), which we then interpolate to produce an initial estimation of the dense motion fields for each layer (bottom right). Optimization: In this stage, we alternate between updating the motion fields, and updating the background and obstruction components, until convergence. .................................................. 11
2.4 The contribution of different components in the algorithm to the result. We use one of our controlled image sequences (a) with ground truth decomposition (b) (see Section 2.5), and compare our algorithm (f) with the following variants: (c) replacing the edge flow initialization with regular optical flow, (d) replacing the dense motion fields with parametric motion (we used projective transforms for both layers), and (e) removing the sparsity prior on the image gradients (i.e. \( \lambda_2 = 0 \) in Equation (2.9)). One pair of ground truth background and reflection images for this 5-frame sequence are shown in (b) for reference. The normalized cross correlation (the higher the better, See Section 2.5 for details) between each recovered layer and the ground truth is shown at the bottom left of each image.

2.5 Decomposition results using three different flow initialization methods: (a) initialized by a random flow field, (b) initialized by the optical flow between frames, and (c) initialized by our edge flow.

2.6 Reflection removal results on four natural sequences. For each sequence we show a representative frame (each input sequence in this figure contain 5 frames), and the background and reflection images recovered automatically by the algorithm (for better visualization, we boost its brightness by four times). Corresponding close-up views are also shown next to the images (on the right of each image for the sequences in the top and middle rows, and below each image for the sequences in the bottom row). More results can be found in the supplementary material.

2.7 Occlusion removal results. For each sequence (row), we show a representative image from the input sequence (left column), the recovered background scene (second column) and the recovered occluding layer (third column). In the right most column, we also show the alpha map, \( A \), as inferred by the algorithm, with colors ranging from black (occlusion) to white (background). The alpha map, as expected, is tightly correlated with the occlusion image, but we show it here for completeness.

2.8 Comparison with “Video De-Fencing” by Mu et al. [92] on sequences from their paper. Left column: two representative frames from each input sequence. Middle column: backgrounds recovered by [92]. Right column: backgrounds recovered by our method.

2.9 Comparison with recent methods for reflection removal. More comparisons can be found in the supplementary material.
2.10 Quantitative evaluation on controlled sequences. **Top:** for each sequence (row), we show a representative frame from the controlled sequence (left column) and our decomposition result (middle and right columns). The normalized cross correlation (NCC) between each recovered layer and the ground truth (not shown, but available on the project web page) is written below the image. **Bottom:** numerical comparison with recent techniques for reflection removal (visual comparisons can be found on the web page).

2.11 **Reflection-free panoramas.** Often when taking panoramas of outdoor scenes through windows, reflections of the indoor scene on the window cannot be avoided. Our algorithm can be used to produce reflection-free panoramas from the same camera motion used to capture the panoramas—i.e. without any additional work needed from the user. (a) The panorama produced with a mobile phone and a state-of-the-art stitching software, where indoor reflection are very apparent. (b) Our reflection-free panorama result. A panorama stitching of the estimated reflection is shown in the inset. On the right are close-up views of corresponding patches in the two panorama images.

3.1 **Illustration of tiny distortion caused by refraction.** The heat rising from two burning candles cause small distortion of the background due to light rays refracting as they travel from the background to the camera passing through the hot air. Such distortion is almost invisible from a single frame (a). However, taking a space-time slice (b) above the candle (the red line in (a)) from the video, it is clear that there is small jittering of background (marked by the white dashed circle) due to the change of refraction. The space-time slice on the right also shows that there is no jittering at places where there is no hot air (the blue line in (a)). Zoomed in for better visualization.

3.2 **Measuring the velocity and depth of imperceptible candle plumes from standard videos.** The heat rising from two burning candles (a, b) cause small distortions of the background due to light rays refracting as they travel from the background to the camera passing through the hot air. Methods such as synthetic Schlieren imaging (c, d) are able to visualize those small disturbances and reveal the heat plume, but are unable to measure its actual motion. We show that, under reasonable conditions, the refraction patterns (observed motions) move coherently with the refracting fluid, allowing to accurately measure the 2D motion of the flow from a monocular video (e), and the depth of the flow from a stereo sequence (f). The full sequence and results are available in our webpage: http://people.csail.mit.edu/tfxue/proj/fluidflow/index.html.
3.3 Refractive distortions (wiggles) in a single view (a) and multiple views (b). A single, thin refractive layer is moving between one or more video cameras and a background. As the refractive fluid moves between time $t_i$ (solid lines) and time $t_i + \Delta t$ (dashed lines), changes in the refractive patterns move points on the background (shown in blue and red) to different positions on the image plane, generating the observed “wiggles” (red and blue arrows). The direction of the wiggles on the image plane can be arbitrary, but they are consistent over short time durations and between close viewpoints as the fluid moves (see text). By tracking the wiggles over time we can recover the projected 2D fluid motion (a), and by stereo-fusing the wiggles between different views, we can recover the fluid’s depth (b). Note: as discussed in the text, wiggle constancy holds if the refraction, the motion of the object and the baseline between the cameras are small. In these illustrations we exaggerated all these quantities for clarity.

3.4 Example result of our refractive flow algorithm (single view) on a sequence of burning candles. Wiggle features (b) are extracted from the input video (a). Notice how the directions of the wiggles (observed motions) are arbitrary and inconsistent with the air flow direction (the right visualization in (b) uses the same wiggle color coding as in Figure 3.2). (c) and (d) show refractive flows calculated by two algorithms, refractive flow and probabilistic refractive flow, respectively.

3.5 Refractive flow results. First row: sample frames from the input videos. Second row: the mean of the optical flow for the same representative frames (using the colormap shown in Figure 1.1), overlayed on the input images. Third row: the mean of the refractive flow, weighted by the variance. Fourth row: the variance of the estimated refractive flow (the square root of the determinant of the covariance matrix for each pixel). For this visualization, variance values above 0.03 were clipped. Check our project website for the full video sequences and results.

3.6 Quantitative evaluation of refractive flow using synthetic sequences. Simulated fluid density (a) and velocity field (b) were generated by Stable Fluids [119], a physics-based fluid flow simulation technique, and rendered on top of three different textures (d). The recovered velocity field from one of the simulations in which the fluid was at 320° Celsius (c) is similar to the ground truth velocity field (b). Quantitative evaluation is given in (e). As expected, larger temperature-related index of refraction differences between the fluid and the background give better flow estimates. The error also increases for backgrounds that do not contain much texture.
3.7 Quantitative evaluation of refractive flow using a controlled experiment. (a) The experiment setup. (b) A representative frame from the captured video. (c) The mean velocity of the hot air blown by the hairdryer, as computed by our algorithm, in m/s. (d) Numerical comparison of our estimated velocities with velocities measured using a velometer, for the four points marked $x_1 - x_4$ in (c). .......................................................... 38

3.8 Refractive stereo results. First and second rows: representative frames from the input videos. The observed wiggles are overlayed on the input frames using the same color coding as in Fig. 1.1. Third row: the estimated disparity maps (weighted by the confidence of the disparity) of the fluid object. Forth row: 3D reconstruction of the scene, where standard stereo is used for solid objects (and the background), and refractive stereo is used for air flows (the depth was scaled for this visualization). Bottom row: Comparison of our estimated refractive flow disparities, with the disparities of the (solid) heat sources that generated them as computed with a standard stereo algorithm, for the points marked as rectangles on the frames in the second row. .......................................................... 40

3.9 Ambiguity in fluid motion estimation. In this figure, upward heat flow is generated by the stove below. Our algorithm correctly recovers the motion of hot air when it is parallel to the background structure (left), but fails when the motion of hot air is perpendicular to the background structure. .......................................................... 41

3.10 The aperture problem for an opaque object. (a) A camera is imaging an opaque moving object (gray). (b) When a vertical edge is observed within the aperture (the white circular mask), we can resolve the horizontal component of the motion. (c) The vertical component of the motion is ambiguous, because when the object moves vertically, no change is observed through this aperture. The aperture problem for a refractive object. (d) A camera is viewing a stationary and planar background (gray and red) through a moving Gaussian-shaped glass (blue). (e) The horizontal motion is ambiguous, because the observed sequence is symmetric. That is, if the glass moves in the opposite direction, the same sequence will be observed. (f) The vertical motion can be recovered, e.g. by tracking the observed tip of the bump. .......................................................... 43

3.11 Capturing setup in presence of refractive fluid. When there is no refractive fluid, the light rays direct hit the image plane and captured image $g(x, t)$ equals the background image $f(x)$. However, due to the refraction, there is a tiny distortion in captured sequences, and we model that distortion as an unknown warping field $r(x, t)$. .......................................................... 44
3.12 The aperture problem for opaque objects. A square object (marked by gray) is moving to the top-right, and we are trying to estimate the motion of this object from small apertures (a)-(c). A smooth region is observed through aperture (a), and we cannot get any information about the object motion, because any motion direction (marked by blue arrows) can explain the observation. A step edge is observed through (b), and we can infer that object is either moving to top-left, top, or top-right (marked by blue arrows). A "L shape" (2D structure) is observed through (c), and we can fully resolve the object motion.

3.13 The ambiguity in motion estimation when only the first order structure is observed. A sequence is captured under the setup shown in (a), and cross sections along the y-direction under two different settings are shown in (b) and (c).

3.14 Ambiguity in motion estimation when a second order structure is observed. See text for more details.

3.15 Experiments on three magnifying lenses. (a) Setup. (b) Captured sequence with different lens and background.

3.16 Experiments on two lenses with significant radial distortion. (a) In both of two sequences, background only contains vertical structure, and both lens are moving to the right. (b) In both of two sequences, background only contains horizontal structure, and both lens are moving to the left.

3.17 Experiments on 3D-printed Gaussian-shaped glasses.

3.18 Experiments on hot air generated by a candle.

4.1 Estimation of the hierarchical tree structure from a video. Inference based on a single frame from a video (a) has inherent ambiguity; figure (b) shows an example, where it is hard to tell from appearance whether the point $P_1$ is connected to $P_2$ (orange curve) or to $P_3$ (blue curve). Using motion information in the temporal domain does not help much as well, as their movements are almost identical (c). We, however, observe that the difference becomes significant in the frequency domain (d), from which we can infer that $P_1$ is more likely to connect to $P_2$ due to their similar spectra. We therefore develop an algorithm to infer tree structure based on both vibration spectra and appearance information. We show results in (e).

4.2 (a) Hierarchical beam structure. (b) Force analysis for one of the branches (the one marked by dashed rectangle in (a)).
4.3 Spectrum of simulated vibration. Each shows one experimental setup. The left column shows the simulated tree structure and the right column shows the temporal spectra of its vibration. (a) shows a tree with two sub-branches \( Y_{2-4} \) and \( Y_{5-7} \). All nodes have the similar power spectrum as their vibrations are dominated by the vibration of the root node \( Y_1 \). To distinguish the spectra of two sub-branches, we calculate the frequency response of each node, which is the ratio between spectrum of the root and the spectrum of each branch, and there is a clear difference between the frequency responses of two branches (b). The modes of frequency response also match the modes of free vibrations of each sub-branches, as if they are detached from the root (c) and (d).

4.4 Overview of our framework. We take a video (a) and a set of keypoints (b) as input (I). We then obtain appearance cues (II) through several intermediate steps (Section 4.4). We use normalized amplitudes (g) and phases (h) of keypoints as our vibration signals (III). Finally, we apply our inference algorithm (Section 4.5) for tree structure estimation.

4.5 Illustration of our hierarchical clustering algorithm. See Section 4.5 for details.

4.6 Evaluation of the algorithm on videos with different frame rates. (a) and (b) shows the power spectra of selected nodes in the input videos captured at different frame rates, and (c) shows the estimated tree structures. See the text for more details.

4.7 Visualization of the mode shapes of the vibration. The top row shows the tree structure (a) and its three mode shapes (b–d), and the bottom row shows the power spectra of the trunk and two branches (e).

4.8 Estimated tree structure on real videos. A1 - A2: results on artificial trees; R1 - R8: results on real trees. In the last row, we show cases where appearance information is not enough for inferring the correct structure. Using vibration signals, our algorithm works well in these cases.

5.1 Predicting the movement of an object from a single snapshot is often ambiguous. For instance, is the girl’s leg in (a) moving up or down? We propose a probabilistic, content-aware motion prediction model (b) that learns the conditional distribution of future frames. Using this model we are able to synthesize various future frames (c) that are all consistent with the observed input (a).

5.2 Illustration of segment-based image synthesis. Given an input, the network first chop it into segments (b), each of which has a consistent motion (here we only show two segments for illustration, and the actual network outputs 64 segments). Then, the network samples the movement of each segment, and synthesize the future frame based on sampled movement (c). Images in (c) are only for illustration. They are not the actual output of our algorithm.
Imagine a world composed of circles that move vertically and squares that move horizontally (a). We consider 3 different models (b-d) in the text to learn the mapping from an image to a motion field. The top row shows graphical models and the bottom row shows corresponding network structures. The deterministic motion prediction structure shown in (b) attempts to learn a one-to-one mapping from appearance to motion, but is unable to model multiple possible motions of an object or to generalize to previously unseen images. The content-agnostic motion prior structure shown in (c) is able to capture a low-dimensional representation of motion, but is unable to leverage cues from image appearance for motion prediction. The content-aware probabilistic motion predictor (d) brings together the advantages of models of (b) and (c) and uses appearance cues along with motion modeling to predict a motion field from a single input image.

Our network consists of five components: (a) a motion encoder, (b) a kernel decoder, (c) an image encoder, (d) a cross convolution layer, and (e) a motion decoder. Our image encoder takes images at four scales as input. For simplicity, we only show two scales in this figure.

Results on the shapes dataset containing circles (C) squares (S) and triangles (T). For each ‘Frame 2’ we show the RGB image along with an overlay of green and magenta versions of the 2 consecutive frames, to help illustrate motion (green is the first frame and magenta is the second frame). See text and our project page for more details and a better visualization: http://visualdynamics.csail.mit.edu/.

Top: for each object, comparison between its ground-truth motion distribution and the distribution predicted by our method. Bottom: KL divergence between ground-truth distributions and distributions predicted by three different algorithms.

Top: Sampling results on the Sprites dataset. Motion is illustrated using the overlay described in Figure 5.5. Bottom: Probability that a synthesized result is labeled as real by humans in Mechanical Turk behavioral experiments.

Top: Sampling results on Exercise dataset. Motion is illustrated using the overlay described in Figure 5.5. Bottom: probability that a synthesized result is labeled as real by humans in Mechanical Turk behavior experiments.

Visual analogy-making (predicted frames are marked in red).

Mean squared pixel error on test analogies, by animation. The first three models (Add, Dis, and Dis+Cls) are from Reed et. al. [103].

Learned feature maps on the shapes dataset (left), the sprites dataset (top right), and the exercise dataset (bottom right).

A.1 Proof of refractive stereo constancy

A.2 Proof of refractive stereo constancy
Chapter 1

Introduction

"We may distinguish rest and motion, absolute and relative, one from the other by their properties, causes, and effects." — Isaac Newton

Motion is ubiquitous. Our ability to perceive motion is important for perceiving and understanding our visual world. For example, our eyes can quickly capture a small moving animal, unnoticeable when static, a baseball player can estimate the speed of the ball and predict where it will land by simply looking at how it moves, and infants learn the concepts of occlusion, containment, and covering by observing the movement of objects [10, 12, 24]. Just as for human vision, motion perception is also a crucial building block for many computer vision tasks, such as video compression [72], object tracking and segmentation [154], event detection and classification [65], video editing [20], and video denoising [80].

Both human visual systems and computer vision systems perceive our world through 2D images captured by either our eyes or cameras. Figure 1.1(a) shows an example of capturing a crowded street. Light rays are first emitted from or reflected by people, trees, lampposts, and other objects on the street, then they travel through the air, and finally they hit the image sensor. During this capturing process, objects on the street might move, air between the camera and the street might waft, and the camera itself might vibrate. All these changes result in apparent motion in the captured frames. By analyzing these observed changes, we can infer properties of the scene being captured, as well as the medium between the scene and the camera.

In this thesis, we study how the observed 2D motion in captured sequences relates to changes in the scene and medium, and then we infer the properties of the scene and medium from the observed 2D motion. The observed motion may be caused by either camera motion, medium motion, or object motion, and different factors reveal different properties of the scene. To understand how each factor relates to the observed 2D motion, we assume that the observed motion in each captured sequence is caused by only one of these three factors, and we show what we can learn from each type of motion. Specifically, we discuss three topics in this thesis, which we describe next.

Learning from camera motion: decoupling different layers in a scene (Chapter 2) First, we discuss what we can learn from camera motion. When the camera moves, the observed 2D motion of stationary objects against a background reveals their relative distance from the camera: if the camera has only
Fig. 1.1: Understanding our visual world from observed 2D motions caused by different factors. (a) A common setup for capturing. (b) Decoupling different layers based on motion parallax. (c) Measuring the movement of fluid objects from distortion due to refraction. (d) Understanding the deformation space of an object by observing how it moves. (e) Estimating the structure of objects from their movement.

translational motion, the magnitude of the observed 2D motion of each object is inversely proportional to the distance between that object and the camera. This effect is also known as motion parallax. Therefore, based on the apparent 2D motion of each object, we can first infer its relative depth and then decompose a scene into different layers based on the estimated depths.

Based on this idea, we developed a new technique to remove undesired layers in videos captured by moving cameras. One example is shown in Fig. 1.1(b), where the fence occludes some details of the tiger that we would like to preserve. Our algorithm takes an image sequence captured by a moving camera as input. It first estimates the motion of each pixel. It then groups pixels into two different clusters based on motion, one for the desired background and one for the unwanted fence layer, and finally it recovers a clean background image, as shown by the right image of Fig. 1.1(c). This technique can also be used to remove other types of visual obstructions, like reflections on a painting when it is imaged through a glass frame, or raindrops on a window.

Learning from medium motion: measuring airflow from refraction (Chapter 3) Next, we show what we can learn from observed motion caused by a change in the medium. Homogenous mediums do not
affect captured frames, as light travels through them in a straight manner. However, inhomogeneous mediums, such as nonuniformly heated air, do affect the capturing process, as light bends when traveling through these mediums due to refraction. As the air moves, small changes in refractive properties appear as small visual distortions (motion) of the background objects in the captured frames, similar to the shimmering effect observed when objects are viewed across hot asphalt or through exhaust gases. Such observed distortions can reveal some properties of the airflow.

In this work, we demonstrate that observed distortions are strongly correlated with the geometry and density of the air fluid. Therefore, the observed distortion serves as a signature that reveals the location of the airflow. Moreover, by tracking the observed distortion, we can recover the fluid’s motion and measure its velocity, as shown in the right image of Figure 1.1(c). Using stereo or multiview sequences as input, we can also estimate the 3D location of the airflow by matching the observed distortion from different viewpoints.

However, observed distortion does not always reveal the movement of fluid. As with estimating the motion of solid objects, there is also ambiguity in fluid motion estimation. Traditional aperture theory shows that, when observing the movement of a one-dimensional image structure through a small aperture, it is impossible to infer motion parallel to the structure. We extend this theory to fluid objects. We show that when fluid objects with a second distortion field move in front of a one-dimensional background structure, it is possible to recover the motion parallel to the background structure, but not the motion perpendicular to it.

Learning from object motions: estimating the objects’ structure (Chapter 5) and deformation space (Chapter 4) Last, we study object motion. The movement of objects reveals some of their physical properties, as most solid objects cannot arbitrarily deform but are constrained by their internal structure. For example, a person can flexibly bend his/her arm at the elbow, but it is almost impossible to bend his/her spine by a large angle, because elbows and spines have different bone structures. Therefore, by observing how objects move, we can infer some of their properties.

We begin with inferring the structure of a tree from its vibrations. When an external force is applied to an object, it has a preferred vibration mode. For example, if the external force has equal energy in all frequencies, the frequency of the maximum vibration equals the natural frequency of the object. This natural frequency is determined by the internal structure of the object, and we can therefore infer the structure of a tree from its natural frequency. To do this, we model a tree-shaped structure as a set of connected beams [93]. Based on this model, we find that each subbranch of the tree is a linear time-invariant system (LTI system), and we can therefore independently estimate the natural frequency of each subbranch and use this information to infer the structure of the tree, as shown in Figure 1.1(e). The details of the algorithm are discussed in Chapter 4.

For objects with more complicated structures, inferring the internal structure is challenging, but we can still predict the possible deformation space. Humans are good at predicting how an object will move without knowing its internal structure. For example, even though very few people have examined the bone structure of human bodies, most people can predict how the girl shown in the left side of Figure 1.1(d) will move in
the future: she will move her leg and keep her arm stable. Humans can make this prediction as we learn the correlation between a human's pose and his/her possible movements by looking at how people move. Inspired by this idea, we propose an algorithm to predict the possible movement of an object from its pose. Our prediction is based on the idea that objects normally move by parts: pixels inside each part move coherently, while pixels from different parts may have different motions. We design a neural network that automatically learns to segment an object into coherently moving parts and learns how each part might move given its current pose. Using this trained network, we can synthesize a person's possible movements from just a single snapshot. The details of the algorithm are presented in Chapter 5.
Chapter 2

Decoupling Layers in Videos Captured by Moving Cameras

2.1 Introduction

Figure 2.1: We present an algorithm for taking pictures through reflective or occluding elements such as windows and fences. The input to our algorithm is a set of images taken by the user while slightly scanning the scene with a camera/phone (a), and the output is two images: a clean image of the (desired) background scene, and an image of the reflected or occluding content (b). Our algorithm is fully automatic, can run on mobile devices, and allows taking pictures through common visual obstacles, producing images as if they were not there. The full image sequences and a closer comparison between the input images and our results are available in the supplementary material.

The principle of parallax shows that one can infer the relative depth of objects from camera motion. When a camera images a static scene from different viewpoints, the difference of the apparent locations of an object at different viewpoints is inversely proportional to the distance between that object and the
camera. This principle is widely used in astronomy and computer vision to infer the geometry of a scene. In astronomy, one way to calculate the distance of closer stars is to measure its relative displacement with respect to a distance background viewing from different angles. In computer vision, to measure the 3D geometry of an object, one can take multiple photos of the same object and reconstruct its 3D geometry from the correspondence between them. In both of these two areas, motion parallax provides an easy way to measure 3D geometry of a scene.

In this section, we propose an algorithm to decouple different layers in a scene using motion parallax. This is very useful to remove undesirable visual obstruction when taking pictures through reflecting or occluding elements. For example, when imaging through glass windows, reflections from indoor objects can obstruct the outdoor scene we wish to capture (Figure 2.1, top row). Similarly, to take pictures of animals in the zoo, we may need to shoot through an enclosure or a fence, leaving an annoying fence pattern on captured images (Figure 2.1, bottom row).

To remove visual obstructions, professional photographers use specialized equipment, such as polarized lenses for reflection removal, which are not accessible to everyday users. On the contrary, our approach requires no additional equipment and the users only need to take a short image sequence while slightly moving the camera—an interaction similar to taking a panorama. Then based on differences in the layers' motions due to motion parallax, our algorithm integrates the space-time information and decoupe the original sequence into two images: an image of the background, and an image of the reflected or occluding content that we want to remove (Figure 1.1).

The use of motion parallax for layer decomposition is not new. Rather, our paper's main contribution is in a more robust and reliable algorithm for motion estimation in the presence of obstructions. Its success comes from mainly: (i) an "edge flow" method that produces a robust initial estimation of the motion of each layer in the presence of visual obstructions, as edges are less affected by the blending of the two layers. Given an input image sequence, we first initialize our algorithm by estimating sparse motion fields on image edges. We then interpolate the sparse edge flows into dense motion fields, and iteratively refine and alternate between computing the motions and estimating the background and obstruction layers in a coarse-to-fine manner (Figure 2.3). (ii) a pixel-wise flow field motion representation for each layer, which, in contrast to many previous image decomposition algorithms that use parametric motion models (affine or homography), is able to handle depth variation as well as small motions within each layer.

Importantly, we also show that the two types of obstructions—reflections and physical occlusions (such as fences)—can be handled by a single framework. Reflections and occlusions may appear different at a glance, and indeed, previous work have used different solutions to address each one. However, we present a unified approach to address both of these two problems with minor difference in formulation. The proposed algorithm achieves results of higher quality than those produced by previous algorithms addressing either subproblem.

\footnote{The camera motion perpendicular to the z-axis is more desired than rotation.}
and users only need to specify the type of obstruction present in the scene (reflective or occluding).

We test our method in various natural and practical scenarios, such as shooting through fences, windows and other reflecting surfaces. For quantitative evaluation, instead of synthetically simulating obstructions by blending or composting images (as commonly done in previous work), we design controlled experiments in which we capture real scenes with ground truth decomposition. Our algorithm is fully automatic, can work with standard phone cameras, and only requires the user to move the camera in a free-form manner to scan the scene. In our experiments, we found that 5 images taken along a small, approximately horizontal baseline of a few centimeters are usually enough to remove the obstructing layer.

2.2 Related Work

Reflection Removal. Separating transmission and reflection in images has been widely studied, both for the purpose of direct decomposition (e.g. [74, 123]), as well as in the context of other graphics and vision applications such as image based rendering [71, 117] and stereo [127].

Previous work on reflection removal can be grouped into three main categories. In the first category are approaches that remove reflection from a single image. As this problem is highly ill-posed, researchers have proposed different priors to make the problem more constrained. For example, Levin et al. [74] proposed to use image priors such as statistics of derivative filters to decompose an image. They later improved their algorithm using patch based priors learned from an external database [75, 76]. However, their method requires a large amount of user input—relying on the user to mark points on the background and reflected content—and does not work well in textured regions. Recently, Li et al.[78] proposed to separate the reflection using a single image focused on the background, under the assumption that the reflection in that case will be blurrier. Even with these priors, single image reflection removal is extremely challenging and hard to make practical for real images.

The second line of work focuses on removing reflections from a set of images taken through polarizers. Using a polarized filter with different orientations, a sequence of images is captured, each of which is a linear combination of the background and reflection layers, \( I_i = a'_i I_B + b'_i I_R \), where the coefficients \( a'_i \) and \( b'_i \) depend on the direction of the polarized filters. This set of images is then used to decompose the background and reflection layers, again, using different image priors for the two layers [33, 70, 111, 116]. These methods perform well, but the requirement of a polarized filter and two images from the exact same capturing position limits their usage.

The third approach is to process an input image sequence where the background and reflection are moving differently. In this setup, both the intensity and the motion of each layer need to be recovered. To simplify the problem, most previous approaches in that category constrained the motion of each layer to follow some parametric model. Be et al.[15] assumed translative motion, and proposed an algorithm to decompose the sequence using a parameterized joint diagonalization. Gai et al.[41] assumed that the motion
of each layer follows an affine transformation, and found a new image prior based on joint patterns of both background and reflectance gradients. Guo et al. [49] assumed that the motion of each layers is a homography, and proposed a low-rank approximation formulation. For many practical scenarios, however, affine and perspective transformations cannot model well enough the motions of the background and reflectance layers. This is manifested as artifacts in the results.

Some authors used dense warp fields, as we do, to model the motions of the background and reflection. Szeliski et al. [123] proposed a min/max alternation algorithm to recover the background and reflectance images, and used optical flow to recover a dense motion field for each layer. In addition to dense motion fields, our algorithm also incorporates image priors that were shown to be instrumental for removing reflections from image sequences [42], which are not used in [123]. Li et al. [77] extended that approach by replacing the optical flow with SIFT flow [81] to calculate the motion field. However, they model the reflection as an independent signal added to each frame, without utilizing the temporal consistency of the reflection layer, thus limiting the quality of their reconstructions.

Occlusion Removal. Occlusion removal is closely related to image and video inpainting [16, 17, 25, 94]. To remove an object from an image or a video, the user first marks some regions to be removed, and then the inpainting algorithm removes those region and fills in the holes by propagating information from other parts of the image or from other frames in the sequence. Image and video inpainting is mainly focused on interactive object removal. On the contrary, our method only requires users to specify whether the obstruction is reflective or opaque.

Several other papers proposed to automatically remove visual obstructions of particular types from either an image or a video. Several algorithms were proposed to remove near-regular structures, like fences, from an image or a video [52, 96, 97, 151]. [92] proposed to detect fence patterns based on visual parallax, and to remove them by pulling the occluded content from other frames. Barnum et al. [13] proposed to detect snow and raindrops based on their unique patterns in the frequency space. Garg et al. [43] remove rain drops based on their physical properties. All these work either focus on particular types of obstacles (e.g. raindrops), or rely on visual properties of the obstruction, such as being comprised of repeating patterns. On the contrary, our goal is develop a general purpose algorithm that could handle common obstructions, without relying on their specific properties.

2.3 Formulation

Let us start with shows our image formation model. As shown in Figure 2.2, a camera is imaging a scene through a visual obstruction, which can be either a reflective object, such as glass, or an opaque object, such as a fence. In order to remove the artifacts—either the obstruction itself, or the reflection introduced by the obstruction—the user captures a sequence of images while moving the camera. In this paper we assume that both the obstruction and background objects remain roughly static during the capture. If the obstruction is
opaque (a fence, for example), we further require that each pixel in the background scene would be visible (not occluded by the obstruction) in at least one frame in the sequence, so that its information could be recovered. We also assume that the obstruction (or the reflected content) is not too close to the background, so that when moving the camera there would be sufficient difference between the motions of the two layers.

In this paper, we use a lower-case letter $a$ to denote a scalar, a normal capital letter $A$ to denote a vector, and a bold capital letter $\mathbf{A}$ to denote a matrix. We denote the matrix product as $AB$, where $A \in \mathbb{R}^{n \times m}$ and $B \in \mathbb{R}^m$, and the element-wise product of two vectors or matrices as $A \circ B$.

If there is no obstruction, we will get a clean image of the background, denoted as $I_B \in \mathbb{R}^n$ (n is the number of pixels). Now, due to the presence of obstructions, the captured image $I \in \mathbb{R}^n$ is a composition of the background image $I_B$ and an additional obstruction layer $I_O$:

$$I = (1 - A) \circ I_O + A \circ I_B, \quad (2.1)$$

where $I_O \in \mathbb{R}^n$ is the obstruction layer we want to remove, $1 \in \mathbb{R}^n$ is a vector with all components equal 1, and $A \in \mathbb{R}^n$ is an alpha blending mask, which assigns a blending factor to each pixel. Notice that $A$ multiplies (element-wise) the background image, not the foreground image (that is, $A = 1$ means a background pixel). This is a less conventional notation for alpha maps, but it will help simplify some of the math later on.

Then if we are imaging through a reflective object, such as a clean window, the obstruction layer $I_O$ is the image of objects on the same side of the camera, as shown in Figure 2.2(a). In this case we assume the alpha blending mask $A$ is a constant, as reflective objects are usually homogeneous (i.e. glass as found in most windows typically reflect light similarly throughout it). This is a common assumption in the reflection separation literature [49, 77, 123]. If, on the other hand, we are imaging through a fence or other opaque objects, the obstruction layer $I_O$ is the opaque object itself, as shown in Figure 2.2(b). In that case, the alpha map, $A$, equals 1 at the region where the background is not occluded, and is between 0 and 1 if the background is partially or fully occluded.

Decomposing an input image $I$ into the obstruction layer $I_O$ and the background layer $I_B$ is ill-posed as there are two unknowns ($I_B$ and $I_O$) but only one observation. We therefore ask the user to move the camera and take a sequence of images. Assuming the obstruction layer $I_O$ is relatively closer (further away) to the camera than the background objects, its projected motion on the image plane will be larger (smaller) than the background objects due to the visual parallax. We utilize this difference in the motions to decompose the input image to the background and obstruction components, similar to [49, 77, 123].

More formally, given an input sequence, we pick one frame $t_0$ from the sequence as the reference frame, and estimate the background component $I_B$ and the obstruction component $I_O$ of that frame, using the information from other frames. Assuming both the background scene and the obstruction layer are static, we can express the background and obstruction components of other frames $t \neq t_0$ as a warped version of the respective components of the reference frame $t_0$. Specifically, let $V_B^t$ and $V_O^t$ denote the motion fields for the obstruction and background layers from the reference frame $t_0$ to the frame $t$, respectively. The observed
image at time $t$ is then

$$I' = (1 - W(V_O^t)A) \circ W(V_O^t)I_O + W(V_B^t)A \circ W(V_B^t)I_B,$$

(2.2)

where $W(V_B^t) \in R^{n \times n}$ is a warping matrix such that $W(V_B^t)I_B$ is the warped background component $I_B$ according to the motion field $V_B^t$. Since the obstruction locates between the camera and background objects, the alpha map shares the same motion of the obstruction component $I_O$, not the background component $I_B$. We can therefore simplify the formulation by defining $I_O = (1 - A) \circ I_O$ (with the abuse of the notation $I_O$) to get the following, simplified equation:

$$I' = W(V_O^t)I_O + W(V_B^t)A \circ W(V_B^t)I_B.$$  

(2.3)

Note that in the case of reflection, since the alpha map is constant, $A = \alpha$, the formulation can be further simplified as $I' = W(V_O^t)I_O + W(V_B^t)I_B$, where $I_O = (1 - \alpha)I_O$ and $I_B = \alpha I_B$. That is, the alpha map is essentially absorbed into the reflection and background components. Except for a small modification in the optimization for that case, which we will describe later on, both types of obstructions (opaque objects and reflective panes) are handled similarly by the algorithm.

Our goal is then to recover the background component $I_B$ and the obstruction component $I_O$ for the reference frame $I^0$, from an input image sequence $\{I'\}$, without knowing the motion fields $V_B^t$ and $V_O^t$, or the alpha map $A$ (in the case of opaque occlusion).

\section*{2.4 Motion-based Decomposition}

\subsection*{2.4.1 Formulation}

Let us now discuss the optimization problem for recovering the background and obstruction components, $I_B$ and $I_O$, from an input image sequence, $\{I'\}$. We will first derive an algorithm for the more general case of an unknown, spatially varying alpha map, $A$, and then show a small simplification that can be used for reflection removal where we assume the alpha map is constant.
Figure 2.3: Algorithm pipeline. Our algorithm consists of two steps: initialization and iterative optimization.

**Initialization:** we first calculate the motion vectors on extracted edge pixels from the input images (we thicken the edge mask for a better visualization). Then we fit two perspective transforms (one for each layer) to the estimated edge flow field and assign each edge pixel to either the background layer or the obstruction layer. This results in two sets of sparse flow fields for the two layers (top right), which we then interpolate to produce an initial estimation of the dense motion fields for each layer (bottom right).

**Optimization:** In this stage, we alternate between updating the motion fields, and updating the background and obstruction components, until convergence.

According to the image formation model (Equation (2.3)), we set our data term to be:

\[ \sum_t ||I^t' - W(V^t_\mathcal{O})Io - W(V^t_\mathcal{B})A \circ W(V^t_\mathcal{B})IB||_1, \]  

(2.4)

where \(\{V^t_\mathcal{O}\}\) and \(\{V^t_\mathcal{B}\}\) are the sets of motion vectors for the obstruction and background components, respectively.

To reduce the ambiguity of the problem, we include additional constraints based on priors on both the decomposed images and their respective motion fields. First, because the obstruction and background components are natural images, we enforce a heavy tailed distribution on their gradients [76], as

\[ \|\nabla I_\mathcal{O}\|_1 + \|\nabla I_\mathcal{B}\|_1, \]  

(2.5)

where \(\nabla I_\mathcal{B}\) are the gradients of the background component \(I_\mathcal{B}\).

We assume that the alpha map is generally smoother than a natural image (smooth transitions in the blending coefficients). Thus, we penalize its \(l_2\)-norm instead of \(l_1\)-norm:

\[ ||\nabla \alpha||^2. \]  

(2.6)
We also assume that the background component and the obstruction component are independent. That is, if we observe a strong gradient in the input image, it most likely belongs either to the background component or the obstruction component, but not to both. To enforce this gradient ownership prior, we penalize the product of the gradients of background and obstruction, as

\[ L(I_O, I_B) = \sum_x ||\nabla I_O(x)||^2 ||\nabla I_B(x)||^2, \]  

where \( x \) is the spatial index and \( \nabla I_B(x) \) is the gradient of image \( I_B \) at position \( x \).

Finally, as a common practice in the optical flow literature [18], we also enforce sparsity on the gradients of the motion fields, seeking to minimize the following term

\[ \sum_t ||\nabla V_{O,t}||_1 + ||\nabla V_{B,t}||_1. \]  

Combining all the terms above, our objective function is:

\[
\begin{align*}
\min_{I_O, I_B, A, \{V_{O,t}\}, \{V_{B,t}\}} & \sum_t ||I^t - W(V_{O,t})I_O - W(V_{B,t})A \circ W(V_{B,t})I_B||_1 \\
& + \lambda_1 ||A||_2^2 + \lambda_2 (||\nabla I_O||_1 + ||\nabla I_B||_1) + \lambda_3 L(I_O, I_B) + \lambda_4 \sum_t ||\nabla V_{O,t}||_1 + ||\nabla V_{B,t}||_1
\end{align*}
\]

Subject to:

\[ 0 \leq I_O, I_B, A \leq 1, \]

where \( \lambda_1, \ldots, \lambda_4 \) are weights for the different terms, which we tuned manually. For all the examples in the paper, we used \( \lambda_1 = 1, \lambda_2 = 0.1, \lambda_3 = 3000, \) and \( \lambda_4 = 0.5. \) Similarly to [123], we also constrain the intensities of both layers to be in the range \([0, 1]\).  

In Figure 2.4, we demonstrate the contribution of different components in our algorithm to the result, using a controlled sequence with ground truth decomposition. One of the main differences between our formulation and previous work in reflection removal is the use of dense motion fields instead of parametric motion. For comparison, we replaced the dense motion fields \( V_{O,t} \) and \( V_{B,t} \) in our formulation with a homography (the results using affine motion were similar). As can be seen in Figure 2.4(c,e), a dense motion representation greatly improves the quality and reduces artifacts. That is because the background/obstruction layer at different frames cannot be aligned well enough using parametric motion. Such misalignments show up as blur and artifacts in the decomposition. The decomposition quality also degrades slightly when not using the gradient sparsity prior, as shown in Figure 2.4(d).

\section*{2.4.2 Optimization}

We use an alternating gradient descent method to solve Equation (2.9). We first fix the motion fields \( \{V_{O,t}\} \) and \( \{V_{B,t}\} \) and solve for \( I_O, I_B \) and \( A \), and then fix \( I_O, I_B \) and \( A \), and solve for \( \{V_{O,t}\} \) and \( \{V_{B,t}\} \). Similar alternating

\[ L(I_B, I_O) \] is significantly smaller than other terms, and so we compensate for that by choosing a larger \( \lambda_3 \).
Figure 2.4: The contribution of different components in the algorithm to the result. We use one of our controlled image sequences (a) with ground truth decomposition (b) (see Section 2.5), and compare our algorithm (f) with the following variants: (c) replacing the edge flow initialization with regular optical flow, (d) replacing the dense motion fields with parametric motion (we used projective transforms for both layers), and (e) removing the sparsity prior on the image gradients (i.e. $\lambda_2 = 0$ in Equation (2.9)). One pair of ground truth background and reflection images for this 5-frame sequence are shown in (b) for reference. The normalized cross correlation (the higher the better., See Section 2.5 for details) between each recovered layer and the ground truth is shown at the bottom left of each image.

gradient descent approach for joint estimation has been used in video super resolution [80].
Decomposition step: fix motion fields \( \{V_O^t\} \) and \( \{V_B^t\} \), and solve for \( I_O, I_B, \) and \( A \). In this step, we ignore all the terms in Equation (2.9) that only consist of \( V_O^t \) and \( V_B^t \):

\[
\min_{(I_O, I_B, A)} \sum_t \|I^t - W_O^t I_O - W_B^t A o W_B^t I_B\|_1 + \lambda_1 \|\nabla A\|_2^2 + \lambda_2 (\|\nabla I_O\|_1 + \|\nabla I_B\|_1) + \lambda_3 L(I_O, I_B),
\]

(2.10)

Subject to

\[
0 \leq I_O, I_B, A \leq 1.
\]

where we use \( W_O^t \) and \( W_B^t \) as short notes for \( W(V_O^t) \) and \( W(V_B^t) \). We solve this problem using a modified version of iterative reweighted least squares (IRLS). The original IRLS algorithm is designed for a non-constrained optimization with only \( l_1 \)-and \( l_2 \)-norms. To get this form, we linearize the higher-order terms in the objective function in Equation (2.10). Let \( \hat{I}_O, \hat{I}_B \) and \( \hat{A} \) be the obstruction component, the background component, and the alpha map of the last iteration, respectively. Then the data term is linearized as

\[
\|I^t - W_O^t I_O - W_B^t A o W_B^t I_B - W_B^t \hat{A} o W_B^t I_B + W_B^t \hat{A} o W_B^t \hat{I}_B\|_1.
\]

(2.11)

We also linearize the edge ownership term as:

\[
\lambda_3 (L(I_O, I_B) + L(I_O, \hat{I}_B) - L(I_O, \hat{I}_B)).
\]

(2.12)

Second, we transform the two inequality constraints to a term in the objective function using the penalty method [85]. For example, for the non-negativity constrain \( I_B \geq 0 \), we include the following penalty function into the objective function:

\[
\lambda_p \min(0, I_B^2),
\]

(2.13)

where \( I_B^2 \) denotes element-wise square and \( \lambda_p \) is the weight for the penalty (we fix \( \lambda_p = 10^5 \)). This function will apply a penalty proportional to the negativity of \( I_B \) (and will be zero if \( I_B \) is nonnegative).

Motion estimation step: fix \( I_O, I_B, A, \) and solve for the motion fields \( V_O^t \) and \( V_B^t \). In this step, we ignore the terms dependent only on \( I_O, I_B, \) and \( A \) in Equation (2.9):

\[
\min_{V_O^t, V_B^t} \|I^t - W(V_O^t) I_O - W(V_O^t) A o W(V_B^t) I_B\|_1 + \lambda_4 (\|\nabla V_O^t\|_1 + \|\nabla V_B^t\|_1).
\]

(2.14)

This equation can again be solved using IRLS, similarly to the decomposition step.

---

3 we make an approximation commonly used in optimization: \( xy \approx x \hat{y} + \hat{x} y \), where \( \hat{x} \) is very close to \( x \) and \( \hat{y} \) is very close to \( y \).

4 \( L(I_O, I_B) = \sum_x \|\nabla I_O\|_1 \|\nabla I_B\|_1 \approx \sum_x \|\nabla I_O\|_1^2 \|\nabla I_B\|_1 + \|\nabla I_O\|_1^2 \|\nabla I_B\|_1^2 - \|\nabla I_O\|_1^2 \|\nabla I_B\|_1^2 = L(I_O, I_B) + L(I_O, \hat{I}_B) - L(I_O, \hat{I}_B). \)
Data: \{I'_t\}_t, initial guess of \(I_0, I_B, A, \{V'_O\}_t\), and \(\{V'_B\}_t\).

Result: \(I_0, I_B, A, \{V'_O\}_t\) and \(\{V'_B\}_t\).

For Scale \(s = 1\) to \(n_s\), do

- \(\{\hat{I}_s\} \leftarrow\) downsample input image sequence \(\{I'_t\}\) to scale \(s\)
- \(I_0, I_B, A, \{V'_O\}_t, \{V'_B\}_t\) \leftarrow\) downsample/upsample \(I_0, I_B, A, \{V'_O\}_t\), and \(\{V'_B\}_t\) to scale \(s\)

For \(i = 1\) to \(n_i\), do

- \(I_0, I_B, A \leftarrow\) Decompose(\(\{\hat{I}_s\}, \{V'_O\}_t, \{V'_B\}_t\))
- \(\{V'_O\}, \{V'_B\} \leftarrow\) EstimateMotion(\(\{\hat{I}_s\}, I_0, I_B, A\))

End

End

Algorithm 1: The motion-based decomposition algorithm.

**Multi-scale Processing.** To accelerate the algorithm, we optimize across multiple scales. We build a Gaussian pyramid for the input image sequence, and first solve all the unknowns—the motions, background and obstruction components, and the alpha blending mask—for the coarsest level. We then propagate the coarse solution to the next level using standard bicubic interpolation, and use it as an initialization to solve for all the unknowns at a finer level. We get the final solution by solving the problem at the original resolution.

At each level, we make a few iterations between solving for the motion and solving for the decomposition.

The final algorithm is summarized in Algorithm 1. The functions Decompose and EstimateMotion are the two steps described above. Scale 1 is the coarsest scale and \(n_s\) is the finest scale (we use \(3 - 4\) scales), and \(n_i\) is the number of iterations, which varies across scales. We typically use 4 iterations for the coarsest scale and 1 iteration for each of the other scales.

**Reflection Removal.** As discussed earlier, in the case of a reflective pane, the alpha map \(A\) is essentially absorbed into the background and reflection images and there is no need to solve for it separately. We thus remove the prior term \(\|\nabla A\|^2\) (Equation (2.6)) from the objective function, and only solve for \(I_0, I_B, \{V'_O\}_t\), and \(\{V'_B\}_t\). The objective function becomes:

\[
\min_{I_0, I_B, \{V'_O\}_t, \{V'_B\}_t} \sum_t ||I'_t - W(V'_O)I_0 - W(V'_B)I_B||_1 + \lambda_2(\|\nabla I_0\|_1 + \|\nabla I_B\|_1) + \lambda_3 L(I_0, I_B) + \lambda_4 \sum_t (\|\nabla V'_O\|_1 + \|\nabla V'_B\|_1)
\]

Subject to:

\[0 \leq I_0, I_B \leq 1,\] \hspace{1cm} (2.15)

and we use the same alternating gradient descent method described above to solve the decomposition.

Currently we distinguish between the two sources of obstructions (opaque and reflective obstruction) manually.
Figure 2.5: Decomposition results using three different flow initialization methods: (a) initialized by a random flow field, (b) initialized by the optical flow between frames, and (c) initialized by our edge flow.

2.4.3 Initialization

A key stage in our algorithm is the initialization of the motion fields and decomposition for the optimization. That part is vital for getting a clean separation as the objective function in Equation (2.9) is nonlinear, and the IRLS algorithms (Section 2.4.2) may get stuck at a local minimum.

To illustrate that, Figure 2.5 shows the results of decomposition method described in Section 2.4.2 using three different flow initialization methods. Just by changing the initialization methods, these three algorithms produce a completely different results. Using random flow fields or optical flow between frames as an initialization (similar to the initialization method described in [77]), the algorithm can barely remove any reflection signal from the input sequence. On the other side, using the initialization method introduce below, our algorithm correctly recover both layers from the input.

Our initialization first estimate an initial motion field for each layer, then calculates the initial decomposition (background component $I_B$, obstruction component $I_O$, and alpha map $A$) from the initial motion fields. We will now describe these two steps in detail.

Initial motion estimation. Motion estimation in videos with reflection/occlusion is challenging. For videos with reflection, for example, each pixel has two motion vectors—one for the background and one for the reflectance. Therefore, we cannot directly use optical flow to estimate the motion fields, as illustrated in Figure 2.5. Notice that previous work in multi-layer optical flow [60, 62, 83, 136] usually assume that the layer in the front occludes the layer on the back, while we assume the captured image is an additive superposition of two layers, so that both layers may be visible at the same location in the image.

Therefore, we propose to get an initial estimate of the motion fields using an “edge flow” algorithm. That is, we estimate a sparse motion field at each edge pixel identified in the image. As discussed before
an observed image gradient often belong to only one of the layers—either the background or the occlusion—but not to both. Indeed, we find that motion vectors estimated from pixels with large image gradients are generally more robust. A similar idea was also used by Kopf et al. [71] for multi-view stereo.

More specifically, for a given input sequence, we first extract the edge map for each frame using the Canny edge detector [23]. Then we calculate the motion of detected edge pixels by solving a discrete Markov random field (MRF):

\[
\min_V \sum_{x \in \text{Edge}(I^1)} -\text{NCC}(I^1(x), I^2(x + V(x))) + \sum_{x, x' \in \text{Edge}(I^1) \text{ and } (x, x') \in \mathcal{N}} S(V(x), V(x')) \tag{2.16}
\]

where \(I^1\) and \(I^2\) are two neighboring input images, \(V\) is the motion field from image \(I^1\) to \(I^2\) that we want to estimate, \(\text{Edge}(I^1)\) is the set of edge pixels in image \(I^1\), and \(\mathcal{N}\) is the 4-connected pixel neighborhood. Notice that different from the motion fields described in previous sections, this motion field is only defined on image edges. We also assume that \(V\) takes only integer values, so that \(x + V(x)\) is also on the grid in the image \(I^2\).

The first term in Equation (2.16) is the data term that describes how well the patch located at position \(x\) in image \(I^1\) matches the patch located at \(x + V(x)\) in image \(I^2\). Here, \(\text{NCC}(I^1(x), I^2(x + V(x)))\) is the normalized cross correlation (NCC) between these two patches, and we minimize the negative NCC between two patches to maximize their similarity. The second term \(S(V(x), V(x'))\) in Equation (2.16) is the smoothness term that enforces neighboring edge pixels to have similar motion. We use the same penalty function described in Equation 1 in [71] for the smoothness term. We solve this Markov random field using belief propagation.

After obtaining the sparse motion field using edge flow, we separate it into two sparse motion fields, one for each layer. To do this, we first fit a perspective transformation to the estimated sparse motion field using RANSAC in the previous step, and assign all the edge pixels that best fit this transformation to the background layer, assuming the background pixels are more dominant. We then fit another perspective transformation to the rest of the edge pixels (again using RANSAC), and assign the pixels best fitting the second transformation to the reflection layer. Finally, we compute the dense motion fields for both layers using visual surface interpolation [124]. An illustration of this process is shown in Figure 2.3.

In Figure 2.4 we compare our result with the proposed edge flow initialization (Figure 2.4(e)) with the one produced when initializing using standard optical flow, as done in [123] (Figure 2.4(b)).

**Initial decomposition.** To get an initial estimation of the decomposition, we first warp all the frames to the reference frame according to the background motion estimated in the previous step. In this warped input sequence, the background pattern should be roughly-aligned, while the obstruction component should move (as the motions of the two components are different).

In the case of an opaque occlusion, we take the per-pixel mean across the warped input frames as an initial estimation of the background image. We also compute a binary alpha map by thresholding the per-pixel difference between the estimated background image and the input images. If the difference is larger than a
threshold (we used 0.1), we set the alpha map to 0 at that location; otherwise we set the alpha map to 1. We then get the obstruction component by plugging the initial estimation of $I_B$ and $A$ to Equation (2.3).

For a reflective pane, we take the initial estimation of the background image to be the minimum intensity across the warped frames$^5$.

2.5 Results

In this chapter, we took most of the image sequences using the cell phone cameras of HTC One M8 and Samsung Galaxy 4, except for the sequence shown in Figure 2.6(c), which was taken using a Canon VIXIA HFG30. We processed the sequences on a desktop computer with Intel Xeon CPU (8 cores) and 64GB memory. With a non-optimized MATLAB implementation, processing a high-resolution image sequence (1152x648) took 20 minutes and required 3GB of RAM, and processing a low-resolution sequence (480x270) took 2 minutes and required 1GB memory. We also implemented a non-optimized Windows phone app prototype, implemented in C++, which produces equivalent results on a desktop computer in less than 2 minutes for low-resolution images (480x270). Most of the sequences contain 5 frames sampled uniformly form the video, except for gallery (Figure 2.9, left) which contains 7 frames.

Removing Obstructions in Natural Sequences. We tested our algorithms under various scenarios, with different background objects, reflecting/occluding elements, and lighting conditions, and it worked consistently well in all these cases.

The top row in Figure 2.1 shows a common scenario when a photographer is taking a picture of an outside view through a window, while self-reflection appears in the captured images. The strong reflection of the shirt covers most of the image and obstructs a large part of the scene. Our algorithm generates a clean separation of the background and reflective components. Notice how the checkerboard pattern on the shirt is completely removed in the recovered background image, while most of the background textures, like the trees and the building, are well-preserved.

Figure 2.6 shows more scenarios where reflections frequently appear in photos. One common case is imaging reflective surfaces, such as a glass-covered billboard of a bus station (Figure 2.6(a)) and a glass-covered panel (Figure 2.6(d)). Notice that in Figure 2.6(a), due to the reflection, many letters on the billboard are difficult to recognize. Even with such strong and textured reflection, our algorithm is able to produce a good separation, and words on the billboard become much clearer after the reflection is removed.

Reflections are also common when imaging outdoor scenes through windows during the night (Figure 2.6(b)) or at dusk (Figure 2.6(c)). Our algorithm is able to separate the background image from the reflection, producing a clean image of the outdoor scene, as well as revealing a lot of information about the indoor scene that is difficult to see in the original image. While the recovered background image is usually of

---

$^5$ we compute the minimum intensity instead of the mean, since the minimum is an upper bound for the background's intensity. See [123] for details.
Figure 2.6: Reflection removal results on four natural sequences. For each sequence we show a representative frame (each input sequence in this figure contain 5 frames), and the background and reflection images recovered automatically by the algorithm (for better visualization, we boost its brightness by four times). Corresponding close-up views are also shown next to the images (on the right of each image for the sequences in the top and middle rows, and below each image for the sequences in the bottom row). More results can be found in the supplementary material.

most interest in this work, our high-quality reconstruction of the reflected scene may also be useful in some cases where more information needs to be extracted from an image sequence or a video.

Figure 2.1 (bottom row) and Figure 2.7 show some common scenarios when photographs are taken through more opaque, occluding elements, such as fences, dirty/textured windows ("simulated" by the SIGGRAPH logo), and surfaces covered by raindrops. In all cases, our algorithm is able to produce good reconstruction of the background scene with the occluding content removed.

Comparison. We compared our algorithm with two state-of-the-art algorithms for removing reflections by Guo et al. [49] and Li et al. [77]. In Figure 2.9 we show side-by-side comparisons of our results with the
results by Guo et al. [49] and Li et al. [77] on two sequences. On the “gallery” sequence\(^6\), both the algorithm by Guo et al. [49] and our algorithm manage to produce a clean background, while there are noticeable artifacts near the boundary of the image in the results by Li et al. [77]. Moreover, the reflection image produced by our algorithm is cleaner than the ones produced by the other methods. On “night”, our algorithm generated a much cleaner separation than the other methods. Notice, for example, how the garbage bin is still visible in the results by the other methods, while our algorithm produces a clean separation in that region.

We also compare our algorithm with one of the state-of-the-art fence removal algorithm from video by Mu et al. [92]. We used the sequences “Tennis” and “Square” provided in [92], shown in Figure 2.8. On the “Tennis” sequence, the two methods produced comparable results, although our algorithm generated a slightly cleaner background image. On the “Square” sequence, since the direction of camera motion is mostly horizontal with a small vertical component (see input frames of “Square”), removing the horizontal part of the fence is challenging. Still, our algorithm takes advantage of this tiny vertical camera motion and remove both horizontal and vertical parts of the fence, while in the result by Mu et al. [92] only the vertical part of the fence is removed.

**Quantitative Evaluation.** To evaluate our results quantitatively, we took three sequences with ground truth background and obstructing layers, as shown in Figure 2.10. The first two sequences were taken through

---

\(^6\)The original sequence is captured by Sinha et al. [117]. We sampled 7 consecutive frames from their video.
glass (reflective layer), while the third sequence was taken through an iron net (occluding layer). We captured a sequence of images over time while moving the camera, where at each time step we captured three images: a standard (composite) image through the obstacle (Figure 2.10, left column), an image of just the background scene, captured by removing the obstacle, and an image of the reflective/occluding layer, which we captured by placing a black sheet of paper behind the obstacle, blocking the background component (and turning the glass into a mirror). All images were taken with a DSLR camera under fully manual control. The full sequences are available on the project web page.

We evaluate our algorithm by calculating the normalized cross correlation (NCC) of our recovered decomposition with the ground truth decomposition. We also compare with the reflection removal algorithms by Li et al. [77] and Guo et al. [49], and our algorithm achieves the highest NCC among three algorithms.

**Reflection-free Panoramas.** Our algorithm can also be used for removing obstructions from panoramic images, for which the image capture already involves camera motion. In Figure 2.11, we show an example of automatically removing window reflections in a panorama of an outdoor scene taken from inside

![Figure 2.8: Comparison with “Video De-Fencing” by Mu et al. [92] on sequences from their paper. Left column: two representative frames from each input sequence. Middle column: backgrounds recovered by [92]. Right column: backgrounds recovered by our method.](image)
Figure 2.9: Comparison with recent methods for reflection removal. More comparisons can be found in the supplementary material.

a building (Figure 2.11(a)). In this case we can use the camera motion that is already introduced by the user for creating the panorama, to also remove the reflections from it. We remove the reflection for each captured image using our algorithm by sliding a temporal window over the captured sequence, and then stitch the results together to produce a reflection-free panorama image, as shown in Figure 2.11(b). Our approach will not work if the camera motion is purely rotational. However, we found that in many practical scenarios—a user taking a panorama with a hand-held camera or a phone—the camera motion will not be purely rotational, and will introduce sufficient parallax for the algorithm to work.
Figure 2.10: Quantitative evaluation on controlled sequences. Top: for each sequence (row), we show a representative frame from the controlled sequence (left column) and our decomposition result (middle and right columns). The normalized cross correlation (NCC) between each recovered layer and the ground truth (not shown, but available on the project web page) is written below the image. Bottom: numerical comparison with recent techniques for reflection removal (visual comparisons can be found on the web page).
Figure 2.11: Reflection-free panoramas. Often when taking panoramas of outdoor scenes through windows, reflections of the indoor scene on the window cannot be avoided. Our algorithm can be used to produce reflection-free panoramas from the same camera motion used to capture the panoramas—i.e. without any additional work needed from the user. (a) The panorama produced with a mobile phone and a state-of-the-art stitching software, where indoor reflection are very apparent. (b) Our reflection-free panorama result. A panorama stitching of the estimated reflection is shown in the inset. On the right are close-up views of corresponding patches in the two panorama images.
Measuring Fluid Motion from Videos

Measuring and visualizing the flow of air and fluids has great importance in many areas of science and technology, such as aeronautical engineering, combustion research, and ballistics. Multiple techniques have been proposed for this purpose, such as sound tomography, Doppler LIDAR and Schlieren photography, but they either rely on complicated and expensive setups or are limited to in-lab use. Our goal is to make the process of measuring and localizing refractive fluids cheaper, more accessible, and applicable in natural settings.

Figure 3.1: Illustration of tiny distortion caused by refraction. The heat rising from two burning candles cause small distortion of the background due to light rays refracting as they travel from the background to the camera passing through the hot air. Such distortion is almost invisible from a single frame (a). However, taking a space-time slice (b) above the candle (the red line in (a)) from the video, it is clear that there is small jittering of background (marked by the white dashed circle) due to the change of refraction. The space-time slice on the right also shows that there is no jittering at places where there is no hot air (the blue line in (a)). Zoomed in for better visualization.

Our techniques utilize the tiny distortion in the captured sequence due to the change of refraction of air (the medium for light propagation). When light rays travel through air of differing densities, they will
bend due to the change of refraction. Such distortion is normally very small and it is almost invisible from a single frame. For example, Figure 3.1 shows a video of two candles emitting hot air above them, and just from a single frame (Figure 3.1(a)), it is hard to tell where are the hot air causing refraction. However, as the air moves, small changes in the refractive properties appear as continuous visual distortions (motions) of the background, similar to the shimmering effect experienced when viewing objects across hot asphalt or through exhaust gases, and it becomes more visible. To illustrate that, we take a space-time slice above the left candle from the original video (Figure 3.1(b)), and it clearly shows that there is small distortion (marked by the white dashed circle) of background caused by the refraction induced by the movement of the hot air. Therefore, by analyzing the tiny distortion due to the change of refraction, it might be possible recover some physical properties of air flow, such its velocity and 3D location.

In this chapter, we tackle this problem from two different angles. First, we propose a new method to recover the velocity and 3D location of air flow from natural videos with textured backgrounds. The main observation is that intensity variations related to movements of refractive fluid elements, as observed by one or more video cameras, are consistent over small space-time volumes. Based on this idea, we develop algorithms to 1) measure the (2D, projected) motion of refractive fluids in monocular videos, and to 2) recover the 3D position of fluid from stereo cameras. Different from previous approaches for measuring refractive flow, our methods operate directly on videos captured with ordinary cameras, do not require auxiliary sensors, light sources or designed backgrounds, and can correctly detect the motion and location of refractive fluids even when they are invisible to naked eyes.

Second, we address this problem from a more theoretical point of view. We ask this fundamental question: does the local image motion due to change of refraction always reviews enough information for motion of refractive elements? This is related to the aperture theory for estimating motion of solid objects. The traditional aperture theory shows that when we view a moving object through a small aperture, we can only recover the component of motion along the local image gradient. We generalize of the traditional aperture problem for moving refractive elements, and show that recovering motion of refractive objects is even more challenging: we cannot infer any information of the motion of a refractive object from the movement of observed 1D structure, and can only recover one component of the motion from 2D structure. This analysis gives an upper bound for any algorithms that recover the fluid motion from natural videos.

The organization of this chapter is as follows. We start by a review of previous works on fluid motion and geometry estimation and their difference to our approach in Section 3.1. We then present detailed algorithms that estimate the velocity and 3D location of fluid flow from the tiny distortion on captured video sequences in Section 3.2. Last, we present our aperture theory for moving refractive elements in Section 3.3.
3.1 Related Work

Techniques to visualize and measure fluid flow can be divided into two categories: those that introduce tracers (dye, smoke or particles) into the fluid, and those that detect the refraction of light rays through the fluid, where variations in index of refraction serve as tracers.

In tracer-based methods, the fluid motion is measured by tracking particles introduced into the fluid, a technique called particle image velocimetry (PIV). Traditional PIV algorithms are based on correlating the particles between image patches [2]. Recently, optical flow algorithms were used with PIV images [108, 109], and different regularization terms were proposed to adapt the optical flow methods to track fluids rather than solid objects [53].

In tracer-free methods, Schlieren photography is a technique to visualize fluid flow that exploits changes in the refractive index of a fluid. It works by using a carefully aligned optical setup to amplify deflections of light rays due to refraction [110, 114, 115]. To measure the velocity of the fluid, researchers have proposed Schlieren PIV [7, 63], in which the motion of a fluid is recovered by tracking vortices in Schlieren photographs using PIV correlation techniques. These methods still require the optical setup for Schlieren photography, which can be expensive and hard to deploy outside a lab.

The initial stage of our approach is most similar to a technique called Background Oriented Schlieren (BOS, a.k.a Synthetic Schlieren) [8, 26, 32, 51, 87, 104]. The optical setup in Schlieren photography is replaced by optical flow calculations on a video of a fluid in front of an in-focus textured background. The refraction due to the fluid motion is recovered by computing optical flow between each frame of the video and an undistorted reference frame. Most previous BOS techniques focus on visualizing, not measuring, the fluid flow, and produce visualizations of the flow similar to the results shown in Figure 3.2(c,d). Atcheson et al. use BOS tomography to recover the volumetric 3D shape of air flow [8]. However, their technique requires a camera array that covers 180° of the air flow of the interest, making the whole system difficult to use outdoors. To the best of our knowledge, ours is the simplest camera-based system that can measure the motion and 3D location of air flow.

While artificial backgrounds are often used when performing BOS [8], Hargather et al. [51] showed that natural backgrounds, such as a forest, are sufficiently textured for BOS, allowing for the visualization of large-scale air flows around planes and boats. To address the fact that the background must be in focus, Wetzstein et al. [139] introduced light field BOS imaging in which the textured background is replaced by a light field probe. However, this comes at the cost of having to build a light field probe as large as the flow of interest.

Another limitation of the aforementioned BOS algorithms is that they require a reference frame, which has no air flow in front of it. To avoid having to capture such a reference frame, Raffel et al. proposed background-oriented stereoscopic Schlieren (BOSS) [102], where images captured by one camera serve as reference frames for the other camera. It is important to note that BOSS uses stereo setup for a different
purpose than our proposed refractive stereo algorithm: the acquisition of a background image, not depth. BOSS uses stereo to achieve a reference-frame-free capture while we use it for depth recovery. Moreover, an important weakness of all of these BOS algorithms is that they require a background that has a strong texture. While texture also helps our algorithms, the probabilistic framework we propose also allows for more general backgrounds.

Complementary to the visualization of BOS, several methods have been developed to recover quantitative information about a scene from refraction. Several authors have shown that it is possible to recover the shape of the surface of an air-water interface by filming a textured background underneath it [31, 91, 153]. Alterman et al. [6] proposed to recover the refraction location and strength from multiple cameras using tomography. Tian et al. [125] showed that atmospheric turbulence provides a depth cue as larger deflections due to refraction typically correspond to greater depths. Wetzstein et al. [140] proposed to recover the shape of a refractive solid object from light field distortions. Alterman et al. [4, 5] showed that it is possible to locate a moving object through refractive atmospheric turbulence or a water interface.

3.2 Measuring Fluid Depth and Velocity from Video using Refractive Wiggles

In this section, we present our technique to measure the velocity and depth of air flow using natural video sequences.

Our techniques are based on visual cues produced by the bending of light rays as they travel through air of differing densities. Such deflections are exploited in various air measurement techniques. As the air moves, small changes in the refractive properties appear as small visual distortions (motions) of the background as shown in Figure 3.1, and we call this distortion “refraction wiggles”.

The main challenge of measuring refractive flow from video is that the air or fluid elements are transparent and cannot be directly observed by a camera. The position of intensity texture features is not directly related to the 3D position of the fluid elements. We therefore cannot apply standard motion analysis and 3D reconstruction techniques directly to the intensity measurements. Previous work also explore the possibility of visualizing fluid flow based on these small visual distortions (refraction wiggles), but none of them can measure velocity and fluid from refraction wiggles.

However, we found that standard motion analysis and 3D reconstruction techniques are still applicable, but in a different feature space. Specifically, while intensity features result from a background layer and their location is not directly related to the fluid layer, motion features (wiggles) correspond to the 3D positions and motion of points on the transparent fluid surface. The movement of those wiggles between consecutive frames (i.e. the movement of the observed motions) is an indicator of the movement of the transparent fluid, and the disparity of these motion features between viewpoints is a good cue for the depth of the fluid surface, as illustrated in Figure 3.2.

Following this observation, we derive algorithms for the following two tasks: 1) tracking the movement
Figure 3.2: **Measuring the velocity and depth of imperceptible candle plumes from standard videos.**

The heat rising from two burning candles (a, b) cause small distortions of the background due to light rays refracting as they travel from the background to the camera passing through the hot air. Methods such as synthetic Schlieren imaging (c, d) are able to visualize those small disturbances and reveal the heat plume, but are unable to measure its actual motion. We show that, under reasonable conditions, the refraction patterns (observed motions) move coherently with the refracting fluid, allowing to accurately measure the 2D motion of the flow from a monocular video (e), and the depth of the flow from a stereo sequence (f). The full sequence and results are available in our webpage: [http://people.csail.mit.edu/tfxue/proj/fluidflow/index.html](http://people.csail.mit.edu/tfxue/proj/fluidflow/index.html).

of refractive fluids in a single video, and 2) recovering the 3D position of points on the fluid surface from stereo sequences. Both these algorithms are based on the *refractive constancy* assumption (analogous to the brightness constancy assumption of ordinary motion analysis): that intensity variations over time (the wiggles) are explained by the motion of a constant, non-uniform refractive field. This distortion is measured by computing the wiggle features in an input video, and then using those features to estimate the motion and depth of the fluid, by matching them across frames and viewpoints. In this work, we focus on estimating the fluid motion and depth from stationary cameras, assuming a single, thin refractive layer between the camera and the background.
Figure 3.3: **Refractive distortions (wiggles) in a single view (a) and multiple views (b).** A single, thin refractive layer is moving between one or more video cameras and a background. As the refractive fluid moves between time $t_1$ (solid lines) and time $t_1 + \Delta t$ (dashed lines), changes in the refractive patterns move points on the background (shown in blue and red) to different positions on the image plane, generating the observed “wiggles” (red and blue arrows). The direction of the wiggles on the image plane can be arbitrary, but they are consistent over short time durations and between close viewpoints as the fluid moves (see text). By tracking the wiggles over time we can recover the projected 2D fluid motion (a), and by stereo-fusing the wiggles between different views, we can recover the fluid’s depth (b). **Note:** as discussed in the text, wiggle constancy holds if the refraction, the motion of the object and the baseline between the cameras are small. In these illustrations we exaggerated all these quantities for clarity.
3.2.1 Refraction Wiggles and Refractive Constancy

Gradients in the refractive properties of air (such as temperature and shape) introduce visual distortions of the background, and changes in the refractive properties over time show up as minute motions in videos. In general, several factors may introduce such changes in refractive properties. For example, the refractive object may be stationary but change in shape or temperature. In this paper, however, we attribute those changes to non-uniform refractive flow elements moving over some short time interval $[t, t + \Delta t]$. We assume that for a small enough $\Delta t$, a refractive object maintains its shape and temperature, such that the observed motions in the video are caused mostly by the motion of the object (and the object having a non-uniform surface, thus introducing variation in the refractive layer). While this is an assumption, we found it to hold well in practice as long as the video frame-rate is high enough, depending on the velocity of the flow being captured (more details in the experimental section below). In this section, we establish the relation between those observed motions in one or more cameras and the motion and depth of refractive fluids in a visual scene.

To understand the basic setup, consider Figure 3.3. A video camera, or multiple video cameras, are observing a static background through a refractive fluid layer (such as hot air). In this paper, we assume that the cameras are stationary, and that a single, thin and moving refractive layer exists between the camera and the background. We use the notation $x_{t,j}$ for points on the $j$'th camera sensor plane at time $t$, $x'_{t,j}$ for points on the (locally planar) fluid, and $x''_{t,j}$ for background points, where $t$ denotes the time index. The depths of these planes are denoted by $z, z', z''$, respectively. We denote the camera centers by $o_j$, and denote by $\alpha_{t,j}, \alpha'_{t,j}$ the angles between the optical axis to the ray from the $j$'th camera center to points on the image plane and background, respectively. For brevity, we will omit the subscript $j$ in the case of a single camera.

Now consider the stereo setup in Figure 3.3(b). An undistorted ray from the camera center $o_j$ to points $x_{t,j}, x'_{t,j}$ on the $j$'th camera plane has angle (assuming small $\alpha_{t,j}$, such that $\tan \alpha_{t,j} \approx \alpha_{t,j}$)

$$\alpha_{t,j} = (x_{t,j} - o_j)/z = (x'_{t,j} - o_j)/z'.$$

(3.1)

This ray is distorted as it passes through the fluid, and transfers into angle $\alpha'_{t,j}$. The exact distortion is determined by Snell’s law. As is common in the optics literature, if the difference between the incident and refraction angles is small, we can use first order paraxial approximations, which imply that Snell’s refraction law is effectively an additive constant,

$$\alpha'_{t,j} \approx \alpha_{t,j} + \Delta \alpha_t,$$

(3.2)

where the angle difference $\Delta \alpha_t$ depends only on the local geometric and refractive properties of the fluid around the intersection point $x'_{t,j}$, and is independent of the incoming ray direction $\alpha'_{t,j}$, and in particular independent of the location of the observing camera $o_j$. The distorted ray then hits the background surface at the point

$$x''_{t,j} = x'_{t,j} + \xi \cdot \alpha'_{t,j} = x'_{t,j} + \xi (\alpha_{t,j} + \Delta \alpha_t)$$

(3.3)

31
where $z' = z'' - z$ is the distance between the fluid object and the background.

At a successive time step $t + \Delta t$ the fluid moves (dashed gray blob in Figure 3.3(b)). We assume this motion introduces an observed motion of the projection of the background point $x''_{t,j}$ on the image plane, from point $x_{t,j}$ to point $x_{t+\Delta t,j}$. We call that motion a "refraction wiggle", or "wiggle" for short. The geometry of the fluid can be complex and predicting the exact path of the light ray mathematically is not straightforward. Nevertheless, we define as $x'_{t+\Delta t,j}$ the point at which a ray connecting $x''_{t,j}$ to the camera center $o_j$ intersects and refracts at the fluid layer.

Let us now fix a point $x'_t = x'_{t,j}$ on the fluid surface, and refer to the image $x_{t,j}$ of that same physical point in all cameras. The rays from each camera to that point have different angles $\alpha'_{t,j}$, and as a result they intersect the background layer at different points $x''_{t,j}$. Thus, the texture intensity observed by each camera at the points $x_{t,j}$ can be arbitrarily different. Therefore, we cannot match intensities between the cameras if we wish to match features on the fluid layer and recover their depth. However, we observe that the wiggles correspond to features on the fluid layer rather than on the background, which allows us to stereo-fuse them to estimate the fluid depth. For this we need to show that despite the fact that, without loss of generality, $x''_{t,1} \neq x''_{t,2}$, the two points are refracted at time $t + \Delta t$ via the same fluid point $x'_{t+\Delta t,1} = x'_{t+\Delta t,2}$. To see that, we use Equation (3.3) to express

$$x'_{t+\Delta t,j} - x'_{t,j} = -\xi (\alpha_{t+\Delta t,j} - \alpha_{t,j}) - \xi (\Delta \alpha_{t+\Delta t} - \Delta \alpha_{t}),$$

(3.4)

and from Equation (3.1), we have

$$\alpha_{t+\Delta t,j} - \alpha_{t,j} = (x'_{t+\Delta t,j} - x'_{t,j}) / z'. $$

(3.5)

Plugging Equation (3.5) in Equation (3.4) we thus have

$$x'_{t+\Delta t,j} - x'_{t,j} = c \cdot (\Delta \alpha_{t+\Delta t} - \Delta \alpha_{t}),$$

(3.6)

where $c = -(z'' - z')z'/z''$. Therefore, since the terms on the RHS of Equation (3.6) are all camera-independent (under our setup and assumptions), if $x'_{t,1} = x'_{t,2} = x'_t$, we conclude that $x'_{t+\Delta t,1} = x'_{t+\Delta t,2}$ and the wiggles in both cameras are equal.

This refraction constancy can be shown similarly for the monocular case (Figure 3.3(a)). That is, if the fluid object is moving at constant velocity over a short spatio-temporal window, then the observed refraction wiggles move coherently with the fluid object between consecutive frames. This stands in contrast to the fact that the observed motions themselves (the wiggles) are unrelated to the actual fluid motion, and in fact can point in opposite directions (Figure 3.2, Figure 3.4). Also note that in Equation (3.1) and Equation (3.3) we assumed the viewing angle $\alpha$ is small to simplify the derivation, and it is easy to show that refraction constancy is not restricted by the viewing angle (not to be confused with $\Delta \alpha$, which does need to be small).

The practical implication of the observations we made in this section is that if we wish to match the projection of a point on the fluid object across frames or viewpoints, we can match the observed wiggle
features. In practice, we use the optical flow algorithm to calculate the refractive wiggle between neighboring frames. Then we calculate the velocity or the depth of refractive fluid by matching wiggle features (optical flow) over time or across viewpoints.

### 3.2.2 Refractive Flow

The goal of fluid motion estimation is to recover the projected 2D velocity, $u(x, y, t)$, of a refractive fluid object from an observed image sequence, $I(x, y, t)$ (Figure 3.3(a)). As discussed in the previous section, the wiggle features $v(x, y, t)$, not the image intensities $I(x, y, t)$, move with the refractive object. Thus, there are two steps in estimating the fluid's motion: 1) computing the wiggle features $v(x, y, t)$ from an input image sequence $I(x, y, t)$, and 2) estimating the fluid motion $u(x, y, t)$ from the wiggle features $v(x, y, t)$. We will now discuss each of these steps in turn.

#### Computing wiggle features.

We use optical flow to compute the wiggles $v(x, y, t)$ from an input image sequence. Recall the brightness constancy assumption in optical flow is that any changes in pixel intensities in a short duration $\Delta t$, $I$, are assumed to be caused by a translation $v = (v_x, v_y)$ over spatial horizontal or vertical positions, $x$ and $y$, of the image intensities, where $v_x$ and $v_y$ are the $x$ and $y$ components of the velocity, respectively. That is,

$$I(x, y, t + dt) = I(x - v_x \Delta t, y - v_y \Delta t, t).$$  \hfill (3.7)

Based on this brightness constancy equation, a traditional way to calculate the motion vector $v$ is to minimize the following optical flow equation [11]:

$$\bar{v} = \arg \min_v \sum_x \alpha_1 \left( \frac{\partial I}{\partial x} v_x + \frac{\partial I}{\partial y} v_y + \frac{\partial I}{\partial t} \right)^2 + \alpha_2 \left( \left\| \frac{\partial v}{\partial x} \right\|^2 + \left\| \frac{\partial v}{\partial y} \right\|^2 \right).$$  \hfill (3.8)

where $\alpha_1$ and $\alpha_2$ are weights for the data and smoothness terms, respectively.

#### Estimating fluid motion.

Let $u_x$ and $u_y$ be the $x$ and $y$ components of the fluid's velocity as seen in the image. Following Section 3.2.1, we thus define the refractive constancy equation for a single view sequence as

$$v(x, y, t + \Delta t) = v(x - u_x \Delta t, y - u_y \Delta t, t).$$  \hfill (3.9)

Notice that refractive constancy has the exact same form as brightness constancy (Equation (3.7)), except that the features are the wiggles, $v$, rather than the image intensities, $I$. This implies that running an optical flow algorithm on the wiggle features $v$ (i.e. the motion of the motion features), will yield the fluid motion $u$.

Formally, we calculate the fluid motion $u$ by minimizing the following equation:

$$\bar{u} = \arg \min_u \sum_x \beta_1 \left( \frac{\partial \bar{v}}{\partial x} u_x + \frac{\partial \bar{v}}{\partial y} u_y + \frac{\partial \bar{v}}{\partial t} \right)^2 + \beta_2 \left( \left\| \frac{\partial u}{\partial x} \right\|^2 + \left\| \frac{\partial u}{\partial y} \right\|^2 \right).$$  \hfill (3.10)
Figure 3.4: *Example result of our refractive flow algorithm (single view) on a sequence of burning candles.* Wiggle features (b) are extracted from the input video (a). Notice how the directions of the wiggles (observed motions) are arbitrary and inconsistent with the air flow direction (the right visualization in (b) uses the same wiggle color coding as in Figure 3.2). (c) and (d) show refractive flows calculated by two algorithms, refractive flow and probabilistic refractive flow, respectively.

This is similar to the Horn-Schunck optical flow formulation, except that we use $l_2$-norm for regularization, as opposed to robust penalty functions such as $l_1$-norm traditionally used by optical flow methods. This is because fluid objects, especially hot air or gas, do not have clear and sharp boundaries like solid objects. We use a multi-scale iterative algorithm to solve Equation (3.10), as a common practice in the optical flow literature.

Figure 3.4 demonstrates a result of this refractive flow algorithm, when applied to a video of burning candles. First, wiggle features (Figure 3.4(b)) are extracted from the input video (Figure 3.4(a)). Since wiggle features move coherently with the air flow, the algorithm correctly estimates the motion of the thermal plume rising from the candle. Notice that the observed motions (wiggles) have arbitrary directions, yet the estimated refractive flow is much more coherent.

Such processing is very sensitive to noise, however, as can be seen in Figure 3.4(b) (the noise is more obvious in an enlarged image). The problem is even more severe for less textured backgrounds. This motivates the probabilistic formulation, which we will now describe.

### 3.2.3 Probabilistic Refractive Flow

We seek to estimate both the refractive flow, and its uncertainty. Consider a background that is smooth in the $x$ direction and textured in the $y$ direction. Due to the aperture problem [36], the flow in the $x$ direction may be dominated by noise, while the optical flow in the $y$ direction can be clean. Knowing the uncertainty in the flow allows uncertain estimates to be down-weighted, increasing the robustness of the algorithm.
To find the variance of the optical flow, let us reformulate Equation (3.8) as a posterior distribution:

$$P(v|I) = \exp\left(-\sum_x a_1 \left[ \frac{\partial l}{\partial x} u_x + \frac{\partial l}{\partial y} u_y + \frac{\partial l}{\partial t} \right]^2 + a_2 \left[ \frac{\partial v}{\partial x} \right]^2 + a_2 \left[ \frac{\partial v}{\partial y} \right]^2 \right).$$ (3.11)

Here, $P(v|I)$ is a Gaussian distribution, and the mean of $P(v|I)$ is equal to the solution of the original optical flow equation (3.8). With this formulation, we can also calculate the variance of the optical flow (the wiggle features). Please refer to Appendix A for the detailed calculation. Let $\bar{v}$ and $\Sigma_v$ be the mean and covariance, respectively, of the wiggle features computed from Eq. (3.11). Then, with the variance of the wiggle features, we can reweight the fluid flow equation as follows:

$$\bar{u} = \arg\min_u \sum_x \beta_1 \left[ \frac{\partial \bar{v}}{\partial x} u_x + \frac{\partial \bar{v}}{\partial y} u_y + \frac{\partial \bar{v}}{\partial t} \right]_x + \beta_2 \left( \left[ \frac{\partial \bar{u}}{\partial x} \right]^2 + \left[ \frac{\partial \bar{u}}{\partial y} \right]^2 \right) + \beta_3 ||u||^2,$$ (3.12)

where $||x||^2 = x^T \Sigma^{-1} x$ is the squared Mahalanobis distance. In this formulation, the data term is reweighted by the variance of the optical flow to robustly estimate the fluid motion: wiggle features with less certainty, such as motions measured in regions of low-contrast, or of flat or one-dimensional structure, will have lower weight in the fluid flow equation. To increase the robustness, we also penalize the magnitude of $u$ to avoid estimating spurious large flows. Including the uncertainty information leads to more accurate estimation of the fluid motion, as shown in Figure 3.4(d).

In practice, calculating the covariance matrix precisely for each pixel is computationally intractable, as we need to compute the marginal probability distribution for each optical flow vector. To avoid this calculation, we concatenate all the optical flow vectors into a single vector and compute its covariance. Also, notice that the fluid flow equation (3.12) still has a quadratic form, so we can model the posterior distribution of the fluid flow $u$ as a Gaussian distribution, and compute its variance. This variance serves as a confidence measure in the estimated fluid motion.

### 3.2.4 Refractive Stereo

The goal of fluid depth estimation is to recover the depth $z'(x, y)$ of a refractive fluid object from a stereo sequence $I_L(x, y, t)$ and $I_R(x, y, t)$ (Fig. 3.3(b)). Following Section 3.2.1, we can find the depth of the fluid object by stereo-matching the refraction wiggles from the left and right views. Therefore, we first use the algorithm discussed in the previous section to calculate the mean and variance of the optical flows in the left and right views, $v_L \sim N(\bar{v}_L, \Sigma_L)$ and $v_R \sim N(\bar{v}_R, \Sigma_R)$, respectively. We then use a discrete Markov Random Field (MRF), commonly used in the stereo literature [112], to regularize the depth estimates.

Formally, let $x_L$ and $x_R$ be the projection of a point on the fluid object onto the left and right image plane, respectively, and define disparity as $d = x_L - x_R$. We first solve the disparity map by minimizing the objective function

$$\ddot{d} = \min_d \sum_{x, y} f(v_R(x, y), v_L(x + d(x, y), y)) + \alpha \left( \left[ \frac{\partial d}{\partial x} \right]^2 + \left[ \frac{\partial d}{\partial y} \right]^2 \right),$$ (3.13)
where \( f(v_R, v_L) \) is the data term based on the observed wiggles \( v_R \) and \( v_L \), and the last two terms regularize the disparity field. We found that using the \( l_2 \)-norm for regularization generates better results overall, better explaining the fuzzy boundaries of fluid refractive objects (similar to what we observed for estimating the optical flow in Section 3.2.2).

As with the refractive flow, we weigh the data term by the variance of the optical flow to make the depth estimation robust to points in a scene where the extracted wiggles are not as reliable. To achieve this, we define the data term, \( f(v_R, v_L) \), as the log of the covariance between the two optical flows from the left and right views,

\[
f(\tilde{v}_R, \tilde{v}_L) = \log \text{cov}(v_R, v_L) = \frac{1}{2} \log |\Sigma_L + \Sigma_R| + \frac{1}{2} \|\tilde{v}_R - \tilde{v}_L\|_{\Sigma_L + \Sigma_R}^2,
\]  

(3.14)

where \( \|\tilde{v}_R - \tilde{v}_L\|_{\Sigma_L + \Sigma_R}^2 = (\tilde{v}_R - \tilde{v}_L)(\Sigma_L + \Sigma_R)^{-1}(\tilde{v}_R - \tilde{v}_L) \). This data term will assign a higher penalty to inconsistencies in the wiggle matching where the wiggles are more reliable, and a lower penalty where the wiggles are less reliable (typically where the background is less textured and the optical flow is noisy). The choice of the log of the covariance as the data term is not arbitrary. It is the log of the conditional marginal distribution of \( v_L \) and \( v_R \), given that \( v_L \) and \( v_R \) match. See Appendix B for a detailed derivation.

With calibrated cameras, we can then compute the depth map, \( z'(x, y) \), from the disparity map, \( d(x, y) \).

### 3.2.5 Experiments

We show several examples of measuring and localizing hot air radiating from several heat sources, such as people, fans, and heating vents.

All the videos were recorded in raw format to avoid compression artifacts. To deal with small camera motions or background motions, we subtracted the mean flow for each frame from the optical flow result. For each sequence we captured, we first applied a temporal Gaussian blur to the input sequence to increase SNR. For fast-moving flow, we recorded high-speed videos using a Phantom v10 high-speed camera. In some of the indoor high-speed sequences that required additional lighting, we used a temporal band-stop filter to remove intensity variations from the lighting due to AC power.

### 3.2.6 Refractive Flow

**Qualitative Results.** Figure 3.5 shows several results of refractive flow analysis from a single camera. All results are available at our project website: [http://people.csail.mit.edu/tfxue/proj/fluidflow/index.html](http://people.csail.mit.edu/tfxue/proj/fluidflow/index.html).

We first tested our algorithm in a controlled setting, using a textured background. In **hand**, we took a 30fps video of a person’s hand after he held a cup of hot water. The algorithm was able to recover heat radiating upward from the hand. In **hairdryers**, we took a 1000fps high-speed video of two hairdryers placed in opposite directions (the two dark shapes in the top left and bottom right are the front ends of the hairdryers),
Figure 3.5: **Refractive flow results.** First row: sample frames from the input videos. Second row: the mean of the optical flow for the same representative frames (using the colormap shown in Figure 1.1), overlayed on the input images. Third row: the mean of the refractive flow, weighted by the variance. Fourth row: the variance of the estimated refractive flow (the square root of the determinant of the covariance matrix for each pixel). For this visualization, variance values above 0.03 were clipped. Check our project website for the full video sequences and results.

and our algorithm detected two opposite streams of hot air flows.

*kettle* and *vents* demonstrate the result on videos with more natural backgrounds. In *vents* (700fps), the background is very challenging for traditional Background Oriented Schlieren algorithms (BOS), as some parts of the background are very smooth, such as the sky, or contain edges in one direction, such as the top of the dome or the boundary of the buildings. BOS algorithms rely on the motion calculated from input videos, similar to the wiggle features shown in the second row of Figure 3.5, which is very noisy due to the
insufficiently textured background. In contrast, the fluid flow (third row of Figure 3.5) clearly shows several flows of hot air coming out from heating vents. By modeling the variance of wiggles in the probabilistic refractive flow (bottom row of Figure 3.5), most of noise in the motion is suppressed.

Figure 3.6: Quantitative evaluation of refractive flow using synthetic sequences. Simulated fluid density (a) and velocity field (b) were generated by Stable Fluids [119], a physics-based fluid flow simulation technique, and rendered on top of three different textures (d). The recovered velocity field from one of the simulations in which the fluid was at 320° Celsius (c) is similar to the ground truth velocity field (b). Quantitative evaluation is given in (e). As expected, larger temperature-related index of refraction differences between the fluid and the background give better flow estimates. The error also increases for backgrounds that do not contain much texture.

Figure 3.7: Quantitative evaluation of refractive flow using a controlled experiment. (a) The experiment setup. (b) A representative frame from the captured video. (c) The mean velocity of the hot air blown by the hairdryer, as computed by our algorithm, in m/s. (d) Numerical comparison of our estimated velocities with velocities measured using a velometer, for the four points marked x1 - x4 in (c).

Quantitative Evaluation. To quantitatively evaluate the fluid velocity recovered by the proposed algorithm, we tested it on simulated sequences with precise ground truth. We generated a set of realistic simulations of
dynamic refracting fluid using Stable Fluids [119], a physics-based fluid flow simulation technique, resulting in fluid densities and (2D) ground truth velocities at each pixel over time (Figure 3.6(a-b)). We then used the simulated densities to render refraction patterns over several types of background textures with varying spatial smoothness (Figure 3.6(d)). To render realistic refractions, we assumed the simulated flow is at a constant distance between the camera and background, with thickness depending linearly on its density (given by the simulation). For a given scene temperature (we used 20°C Celsius) and a temperature of the fluid, we can compute exactly the amount of refraction at every point in the image. We then apply our algorithm to the refraction sequences. The angular error of the recovered fluid motion at different temperature is shown in (e).

All the sequences and results are available in the supplemental material.

To further demonstrate that the magnitude of the motion computed by the refractive flow algorithm is correct, we also performed a controlled experiment shown in Figure 3.7. We use a velocimeter to measure the velocity of hot air blown from a hairdryer, and compare it with the velocities extracted by the algorithm. To convert the velocity on the image plane to the velocity in the real world, we set the hairdryer parallel to the imaging plane and put a ruler on the same plane of the hairdryer. We measured the velocity of the flow at four different locations as shown in Figure 3.7(c). Although our estimated flow does not match the velocimeter flow exactly, it is highly correlated, and we believe the agreement is accurate to within the experimental uncertainties.

3.2.7 Refractive Stereo

Qualitative Results. We captured several stereo sequences using our own stereo rig, comprised of two Point Grey Grasshopper 3 cameras recording at 50fps. The two cameras are synchronized via Genlock and global shutter is used to avoid temporal misalignment. All videos were captured in 16-bit grayscale.

Several results are shown in Figure 3.8. The third row in Figure 3.8 shows the disparity map of air flow as estimated by our refractive stereo algorithm, and the forth row shows a 3D reconstruction of the scene according to the estimated depths of the solid and refractive objects. For the 3D reconstructions, we first used a standard stereo method to reconstruct the (solid) objects and the background, and then rendered fluid pixels according to their depth as estimated by our refractive stereo algorithm, colored by their disparities (and weighted by the confidence of the disparity). In candle, two plumes of hot airs at different depth are recovered. In lamps, three lights were positioned at different distances from the camera: the left one was the closest to the camera and the middle one was the furthest. The disparities of three plumes recovered by the algorithm match the locations of the lights. In monitor, the algorithm recovers the location of hot air radiating from the center top of a monitor. We intentionally tilted the monitor such that its right side was closer to the camera to introduce variation in the depth of the air flow. The algorithm successfully detects this gradual change in disparities, as shown in the right column of Figure 3.8.
Figure 3.8: **Refractive stereo results.** First and second rows: representative frames from the input videos. The observed wiggles are overlayed on the input frames using the same color coding as in Fig. 1.1. Third row: the estimated disparity maps (weighted by the confidence of the disparity) of the fluid object. Forth row: 3D reconstruction of the scene, where standard stereo is used for solid objects (and the background), and refractive stereo is used for air flows (the depth was scaled for this visualization). Bottom row: Comparison of our estimated refractive flow disparities, with the disparities of the (solid) heat sources that generated them as computed with a standard stereo algorithm, for the points marked as rectangles on the frames in the second row.
Quantitative Evaluation. We compared the recovered depths of the refractive fluid layers with that of the heat sources generating them, as computed using a standard stereo algorithm (since the depth of the actual heat sources, being solid objects, can be estimated well using existing stereo techniques). More specifically, we picked a region on the heat source and picked another region of hot air right above the heat source (second row in Figure 3.8), and compared the average disparities in these two regions. The recovered depth map of the (refractive) hot air matched well the recovered depth map of the (solid) heat sources, with an average error of less than a few pixels (bottom row in Figure 3.8).

3.3 The Aperture Problem for Refractive Motion

In Section 3.2, we demonstrate how to infer the velocity and 3D location of refractive fluid from observed distortion (refraction wiggles) in the captured sequences. However, we cannot always correctly recover the fluid motion just from observed refractive distortion. For example, Figure 3.9 shows a sequence taken through the upward heat flow generated by the stove below. The motion this hot air flow is estimated from two regions (red rectangles) in this sequence, using the refractive flow algorithm described in Section 3.2. When the background structure is vertical (parallel to the direction of air flow), the algorithm can correctly recover the air motion, as shown on the left. However, when the background structure is horizontal (perpendicular to the direction of air flow), the algorithm fails to find the correct air motion, as shown on the right.

Figure 3.9: Ambiguity in fluid motion estimation. In this figure, upward heat flow is generated by the stove below. Our algorithm correctly recovers the motion of hot air when it is parallel to the background structure (left), but fails when the motion of hot air is perpendicular to the background structure.

In this section, we try to understand whether the failure of our algorithm is due to inappropriate design of the algorithm, or due to the intrinsic ambiguity of the refraction flow estimation problem itself.

To understand this, we study the aperture problem of refractive objects: when a refractive object moves between a static camera and a static and opaque background, is it possible to recover the movement of this object from the observed distortion of the background pattern due to refraction? Following the tradition of the aperture problem, we choose the aperture small enough so that the background within that aperture is
either uniform (purely black or white) or contains a single straight edge. Due to the non-uniform refraction, the observed shape through the aperture might be distorted. Our study shows that if the distortion is strong enough so that we observe a curved shape, we can recover the component of the motion parallel to the background structure, but we cannot recover the component perpendicular to it. Otherwise, we cannot infer anything about the motion. This demonstrates that there is ambiguity in fluid motion estimation and errors shown in Figure 3.9 is unavoidable, unless a much stronger prior on fluid motion is used.

The rest of this section is organized as follows. We first demonstrate the intrinsic ambiguity in fluid motion estimation in Section 3.3.1. We then formally define the refractive fluid estimation problem in Section 3.3.2 and derive our aperture theory for refractive fluid in Section 3.3.3. In Section 3.3.4, we end this section by results from both synthetic and real videos to justify our claim.

### 3.3.1 A Toy Example

Let us start with a toy example. The traditional aperture shows that a moving image provides incomplete information about the local motion: only the component of motion along the local image gradient is constrained. Consider a camera imaging an opaque moving object (Figure 3.10(a)). When a vertical edge is observed within the aperture (the white circular mask), we can resolve the horizontal component of the motion (Figure 3.10(b)). However, no change is observed through this aperture when object moves vertically, so it is impossible to recover the vertical component of the motion (Figure 3.10(b)).

Now let us extent this to refractive objects. Consider a camera is viewing a static background, and Gaussian-shaped glass moves in between with an unknown velocity, as shown in Figure 3.10(d). The task is to estimate the velocity the glass from the local distortion observed through a small aperture (the white mask). When the glass moves perpendicular to the edge boundary, as in Figure 3.10(e), the observed boundary first bulges to the left ($t = 1, 2, 3$), and then returns to the original shape ($t = 3, 4, 5$). Due to the symmetry in this observed sequence, reversing the motion of the glass produces the same observation. Therefore, from the observed sequence, we cannot infer whether the glass is moving toward or away from the edge boundary. On the other hand, here we can recover the vertical motion of the glass, as in Figure 3.10(f), by tracking the observed tip of the bump.

This shows that motion analysis of refractive objects based on the traditional aperture theory can be quite misleading. The traditional aperture theory states that we can only recover the component of the motion perpendicular to the edge direction, while in this toy example, we can actually measure the component parallel to the edge, but cannot recover the component perpendicular. This suggests that we need to derive a new aperture theory for refractive element, which will be discussed in the rest of this section.
Figure 3.10: **The aperture problem for an opaque object.** (a) A camera is imaging an opaque moving object (gray). (b) When a vertical edge is observed within the aperture (the white circular mask), we can resolve the horizontal component of the motion. (c) The vertical component of the motion is ambiguous, because when the object moves vertically, no change is observed through this aperture. **The aperture problem for a refractive object.** (d) A camera is viewing a stationary and planar background (gray and red) through a moving Gaussian-shaped glass (blue). (e) The horizontal motion is ambiguous, because the observed sequence is symmetric. That is, if the glass moves in the opposite direction, the same sequence will be observed. (f) The vertical motion can be recovered, e.g. by tracking the observed tip of the bump.

### 3.3.2 Problem Definition

In the rest of this section, we use the following notation convention. We denote a scalar as $a$, vector as $\mathbf{a}$, matrix as $A$, and a set as $\mathcal{A}$. We let $\mathcal{N}(a)$ be a small neighborhood of $a$ and denote the 2D coordinate of a point on the image plane by the vector $\mathbf{x} = (x, y)$. All vectors, if not specified, are column vectors.

Let us revisit the capturing process in presence of refractive distortions, as discussed in Section 3.2.1. Consider any planar and static background pattern $f(\mathbf{x})$ and a camera observing it through a refractive layer, as shown in Figure 3.11. Denote observed images (over time) through the camera by $g(\mathbf{x}, t)$. For simplicity,
let image coordinates be such that the observed image on the camera plane is the same as the background image when there is no refractive object, that is \( g(x, t) = f(x) \).

Now we add a refractive object between the camera and the background. This causes a distorted observation of the background through the camera. We model this distortion by an unknown warping field \( r(x, t) = (r_x(x, t), r_y(x, t)) \), which depends on the shape and index of refraction of the object. The distorted background captured by the camera obeys the following identity (see Figure 3.11),

\[
g(x, t) = f(x - r(x, t)).
\] (3.15)

Under the refractive constancy assumption described in Section 3.2.1, the warping field \( r \) can be expressed as:

\[
\forall t \in \mathcal{N}(t_0); \ r(x, t) = r_0(x - ut).
\] (3.16)

where \( r_0 \) is the refractive field at time \( t_0 \), i.e. \( r_0(x) = r(x, t_0) \). By plugging Equation (3.16) to Equation (3.15), we obtain the image formation equation:

\[
\forall (x, t) \in \mathcal{N}(x_0, t_0); \ g(x, t) = f(x - r_0(x - ut)).
\] (3.17)

The motion estimation task is to recover the motion \( u \) of the refractive object from the observed image sequence \( g(x, t) \), without knowing either the background image \( f(x) \) or the refractive field \( r_0(x) \). Formally, given the observed image sequence \( g(x, t) \) within a small space-time window around \( (x_0, t_0) \), we want to find the solution space of the refractive motion:

\[
\mathcal{U} = \{ u | \exists f, r_0, \forall x, t ; g(x, t) = f(x - r_0(x - ut)) \}.
\] (3.18)
Ideally, \(|U|\) would contain a single element. Generally, however, the observed sequence may be explained by multiple motion vectors \(|U| > 1\), meaning the problem is ill-posed. Identifying such ambiguous cases and their associated solution space \(U\) is the goal of this paper. In the next section, we present a theory to improve our understanding about such ambiguities.

Also, we assume that the background is only a step edge within the aperture (a spatiotemporal window around \((x_0, t_0)\)), that is:

\[
f(x) = \begin{cases} 
0, & \text{if } n^\top x \leq 0 \\
1, & \text{otherwise}
\end{cases}
\]  

where \(n\) is the direction of background gradient. For simplicity, we use the following notation to denote the black-and-white background pattern described in Equation (3.19):

\[
f(x) : n^\top x \leq 0.
\]

Finally, we assume the refractive field is quadratic within the aperture:

\[
r_0(x) = \bar{r} + Jx + \frac{1}{2} \begin{pmatrix} x^\top H_x x \\ x^\top H_y x \end{pmatrix},
\]

where \(\bar{r} \in \mathbb{R}^2\) is the constant term, \(J \in \mathbb{R}^{2 \times 2}\) is the Jacobian matrix of the refractive field \(r\), and \(H_x, H_y \in \mathbb{R}^{2 \times 2}\) are Hessian matrices of \(x\) and \(y\) components of it.

### 3.3.3 Aperture Theory for Refractive Motion

The "aperture problem" describes the intrinsic ambiguity of perceiving the motion of an object through a local observation. Such ambiguity depends on the complexity of the structure observed through an aperture. This section presents a theory for characterizing the ambiguity space (Equation (3.18)) for the refractive motion.

**The aperture problem for surface motion.** In order to provide some context, we first review the traditional aperture problem for surface motion. The traditional aperture problem studies the motion of an *opaque* object observed through a small aperture. The associated ambiguity of recovering motion is demonstrated in Figure 3.12. If the observed region is totally plain, as in Figure 3.12(a), then we cannot infer the motion of the object, and the solution space\(^2\) of motion \(U = \mathbb{R}^2\) is the whole velocity space. If only a straight edge (first order structure) is observed within the aperture, as shown in Figure 3.12(b), we can only resolve the component of the motion that is perpendicular to this edge, and the solution space \(U \subset \mathbb{R}^2\) is a 2D line in the velocity space. Finally, if a higher order structure is observed within the aperture (in Figure 3.12(c)), the ambiguity is fully resolved. This is summarized in the middle row of Table 3.1.

---

1 The general equation of linear background should be \(n^\top x + c \leq 0\). For simplicity, we ignore the bias term \(c\). See the supplementary material for the full derivation.

2 With the abused of notation, here we also denote the solution space of the traditional aperture problem as \(U\).
Table 3.1: Comparison between the traditional aperture problem and the refractive aperture problem. $\mathcal{U}$ is the set of all possible solutions to the object motion (either opaque objects or refractive objects) from the input sequence, and $u^*$ is the ground truth motion.

<table>
<thead>
<tr>
<th>The observed image sequence through the aperture</th>
<th>The traditional aperture problem</th>
<th>The refractive aperture problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constant</td>
<td>$\mathcal{U} = \mathbb{R}^2$ (No information)</td>
<td>$\mathcal{U} = \mathbb{R}^2$ (No information)</td>
</tr>
<tr>
<td>First order structure (a straight edge)</td>
<td>$\mathcal{U}$ is a line in $\mathbb{R}^2$</td>
<td>$\mathcal{U} = {u^*}$</td>
</tr>
<tr>
<td>Second order structure (a conic curve)</td>
<td>$\mathcal{U} = \mathbb{R}^2$ (No information)</td>
<td>$\mathcal{U}$ is a line in $\mathbb{R}^2$</td>
</tr>
</tbody>
</table>

Figure 3.12: The aperture problem for opaque objects. A square object (marked by gray) is moving to the top-right, and we are trying to estimate the motion of this object from small apertures (a)-(c). A smooth region is observed through aperture (a), and we cannot get any information about the object motion, because any motion direction (marked by blue arrows) can explain the observation. A step edge is observed through (b), and we can infer that object is either moving to top-left, top, or top-right (marked by blue arrows). A “L shape” (2D structure) is observes through (c), and we can fully resolve the object motion.

The refractive aperture problem. For refractive motion, we still discuss its ambiguity under three different classes of local observations: flat regions, first order structure (the boundary is a straight edge), and second order structure (the boundary is a conic curve).

When the background is totally plain, no matter how refractive object moves, we observe no change in the captured sequence. Therefore, 1. Observing a plain pattern (pure black or pure white) does not reveal any information about object’s motion.

The other two cases are more complicated, and in the following two sections we will show that: 2. the
Figure 3.13: The ambiguity in motion estimation when only the first order structure is observed. A sequence is captured under the setup shown in (a), and cross sections along the y-direction under two different settings are shown in (b) and (c).

\textit{movement of a 1D structure in the observed sequence still does not reveal any information about object’s motion, and 3. the movement of a 2D structure in the observed sequence reveals the motion in only one direction.}

\textbf{First Order Observation}

When a first order structure is observed, no information about refractive motion can be recovered. To illustrate this, consider the example shown in Figure 3.13. A camera is imaging a static edge-shaped background through a moving prism. The prism is oriented in the way that it has no variation along the x-direction. Therefore, the motion in x-direction does not affect the observed sequence, which is similar to the traditional aperture problem.

In addition, the magnitude of the motion along the y-direction is also lost. To see that, consider a cross section along the y-direction (Figure 3.13(b,c)). The projection of the background boundary moves from $A$ to $B$ due to the change of refraction.\textsuperscript{3} The magnitude and the direction of this observed motion (red arrow in Figure 3.13) is related to both the motion of the refractive object and the angle of the prism. The same observed motion might be due to either a large motion of the prism and a small refraction (small angle) as in Figure 3.13(b), or to a small motion and a large refraction (large angle) as in Figure 3.13(c). Since both the motion and the shape of the refractive object are unknown, there is no way to resolve this ambiguity. Therefore, the ambiguity space $U$ of refractive motion is the whole velocity space $\mathbb{R}^2$.

\textbf{Mathematical derivation} Now we will illustrate this ambiguity mathematically. Because only a first order structure is observed through the aperture, we can drop the second order term\textsuperscript{4} in $r$. Plugging the equation of

\textsuperscript{3} Notice that the observed motion of image boundary on the image plane (marked by the red arrow) does not necessarily have the same direction as the motion of the refractive object (marked by the blue arrow).

\textsuperscript{4} Second order term in $r$ can be ignored because otherwise a curved pattern would have been observed instead of a straight edge.
Figure 3.14: Ambiguity in motion estimation when a second order structure is observed. See text for more details.

background Equation (3.19) and the equation of refractive field Equation (3.21) into the image formation Equation (3.17) (constant terms are omitted for simplicity):

\[ g(x, t) : x^T(I - J^T)n - n^T\bar{r} + n^TJu \tag{3.22} \]

where \( I \) is the \( 2 \times 2 \) identity matrix.

Equation (3.22) is an equation of a line w.r.t. variable \( x \). Note that unknowns in Equation (3.22) are \( r, J, n, \) and \( u \). It is easy to show that for any motion vector \( u \), we can design \( r, J, \) and \( n \) such that they satisfy this line equation and hence generate the same observed sequence for the boundary (see Appendix C for detailed proof). Moreover, the speed of line is proportional to \( n^TJu \), which shows the same observed movement can either due to a large refraction (large \( J \)) and a small motion (small \( u \)), or a small refraction (small \( u \)) and a large motion (large \( u \)). Therefore, we cannot recover any information about refractive motion \( u \) in this case.
Second Order Observation

For a refractive object, when we observe a second order through the aperture, we can only recover the motion in one direction. That means, the solution space \( \mathcal{U} \) for the motion is reduced to a line in \( \mathbb{R}^2 \) velocity space.

To illustrate this, we revisit the Gaussian-shaped glass example discussed in the introduction. First it is impossible to recover its the component of the motion perpendicular to the background structure. Figure 3.14(a) illustrates this ambiguity. Two different Gaussian glasses move towards opposite directions (the left column of Figure 3.14(a)). Due to the refraction, the background boundary moves away from its original location, and such distortion is illustrated by the dark blue arrow in Figure 3.14(c). Also, one glass is a complementary of the other, so that they will bend the light passing towards opposite directions. Because the shapes of glasses are complementary and the motions of glasses are also opposite, the observed distortion within the aperture are actually similar (the right column of Figure 3.14(a)). Thus it is hard to infer the motion of the glass from observed distortion. The cause of this ambiguity is similar to the case of first order observation.

On the other hand, when the glass moves parallel to the background structure (Figure 3.14(b)), the observe boundary moves at the same speed as the glass. Now the movement of observed boundary does not depend on the shape of the glass. Thus, we can infer the component of the glass motion that is parallel to the background structure by tracking the tip of the observed boundary (the red dot in Figure 3.14(c)).

Mathematical derivation

Now we will explain the cause of such ambiguity. For simplicity, we assume that the Jacobian and Hessian of the refractive field are scalar matrices, which is true when the refractive object have a centroid-symmetric shape (like Gaussian-shape glasses shown in Figure 3.14). We will present the refractive flow equation for general case at the end of this sub-section.

In the simplified case, the refractive field \( r \) can be represented as:

\[
\begin{equation}
 r_0(x) = \mathbf{r} + jx + \frac{1}{2} \begin{pmatrix}
 h_x x^T x \\
 h_y y^T x
\end{pmatrix},
\end{equation}
\]

where \( J = jI, H_x = h_x I, H_y = h_y I \). Then the image formation Equation (3.17) becomes (plugs Equation (3.19) and Equation (3.23) into Equation (3.17)):

\[
\begin{equation}
 g(x, t) : = \frac{h}{2} x^T x + (hu + n - jn)^T x + jn^T ut + d - \frac{h}{2} u^T ut^2,
\end{equation}
\]

where \( n = (n_x, n_y), h = h_x n_x + h_y n_y, \) and \( d = -n^T \mathbf{r} \).

Equation (3.24) is an equation of a circle \( a(t)x^T x + q(t)^T x + 1 = 0 \), and we can get its parameters \( a(t) \) and \( q(t) \) by circle fitting:

\[
\begin{align*}
 a(t) &= \frac{h}{2(jn^T ut + d - \frac{h}{2} u^T ut^2)}, \\
 q(t) &= \frac{hut + n - jn}{jn^T ut + d - \frac{h}{2} u^T ut^2}.
\end{align*}
\]

\(^5 I \) is the \( 2 \times 2 \) identity matrix.
Note that except \( a(t) \) and \( q(t) \), all other variables in Eq 3.25 are unknown, and our task is to recover the fluid motion \( u \). Considering:

\[
\frac{dq(0)/dt}{2a(0)} = u + \eta n. \tag{3.26}
\]

where \( \eta \) is an unknown scalar defined as \( \eta = -\frac{\sqrt{(1-j)u}}{dk} \).

Note that the parameters of the observed circle (that is the LHS of Equation (3.26)) are known from observed sequences. Therefore, we can recover the motion perpendicular to the background gradient \( n \), but cannot recover the motion parallel to the background gradient \( n \), because the scalar \( \eta \) in front of \( n \) is unknown. Moreover, let \( n_\perp \) be the unit vector that is parallel to the background structure (so it is perpendicular to \( n \)). Then by multiplying \(^6 n_\perp \) on both sides of Equation (3.26), we get the simplified refractive flow equation:

\[
-n_\perp \frac{dq/dt}{2a} = n_\perp u. \tag{3.27}
\]

It is easy to show that \( -\frac{dq/dt}{2a} \) is the speed of the center of the observed circular boundary. Thus the simplified refractive flow equation (3.27) shows that the motion of the observed image boundary and the motion of the refractive object have the same projection on the direction of background structure \( n_\perp \). Thus we can get the component of refractive motion that is parallel to the background structure by tracking the observed boundary.

This equation also shows that we cannot recover the component of the motion perpendicular to background structure\(^7 \). This is different from the traditional aperture problem, where it is the component of the motion parallel to the background structure that cannot be estimated.

In a general setup, the Jacobian and Hessian of the refractive field are not scalar matrices, and we have the following refractive flow equation (see Appendix C for the proof).

**Proposition 1 (Refractive flow equation (general)).** At each time point \( t \), we fit a conic curve \( x^\top A x + q^\top x + 1 = 0 \) to the observed boundary within the aperture, where \( A \in \mathbb{R}^{2 \times 2} \) and \( q \in \mathbb{R}^2 \) are the coefficients of this curve. Then the motion \( u \) satisfies the equation,

\[
-q_\perp \frac{dq/dt}{2} = (Aq_\perp)^\top u, \tag{3.28}
\]

where \( q_\perp \) is a vector perpendicular to \( q \). Moreover, we cannot recover the motion along with the vector \( RAq_\perp \) just from the observation, where \( R \) is the \( 90^\circ \) rotation matrix.

\(^6 \)We consider \( n_\perp \) as a known variable, as it we can recover the direction of \( n_\perp \) from the observation, since \( q(0) = d^{-1}(1-j)n \) and \( n_\perp \perp n \).

\(^7 \)Although we only show Equation (3.27) is a necessary condition that \( u \) is a solution of the refractive flow problem, it is also a sufficient condition. See Appendix C for the proof of the sufficiency.
The ambiguous direction $\hat{R}Aq$ in the general refractive flow equation is along the vector connecting the center of the aperture $o$ to the center of the observed conic curve $o'$. This is illustrated in Figure 3.14(c) where the dark red arrow indicating the vector $\overrightarrow{o'o}$ aligns with the blue arrow showing the direction of observed distortion due to the refraction.

When the refraction effect is mild, this ambiguous direction also approximately equals to the direction of background structure rotated by the Hessian $H$ of refractive field. Note that this ambiguous direction is not always aligned with the background gradient. In Figure 3.14(c), they are happened to be aligned because $H$ is a scalar matrix.

### 3.3.4 Experiments

In this section, we present experiments that verify the ambiguity theory.

**First order observation** First, the experiment shown in Figure 3.15 confirms the claim that by observing a first order structure, it is impossible to infer the motion of a refractive object. In this experiment,
Figure 3.16: Experiments on two lenses with significant radial distortion. (a) In both of two sequences, background only contains vertical structure, and both lens are moving to the right. (b) In both of two sequences, background only contains horizontal structure, and both lens are moving to the left.

<table>
<thead>
<tr>
<th>Lens 1</th>
<th>Lens 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lens speed cm/s</td>
<td>1.43 cm/s</td>
</tr>
</tbody>
</table>

(a) Motion of the lens is perpendicular to the background structure

<table>
<thead>
<tr>
<th>Lens 1</th>
<th>Lens 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lens speed cm/s</td>
<td>2.00 cm/s</td>
</tr>
</tbody>
</table>

(b) Motion of the lens is parallel to the background structure

Second order observation In the previous experiment, the aperture is chosen to be very close to the center of the lens, so its refractive field is approximately linear in this region. Thus radial distortion is hardly observed (the observed boundary is a straight line).
To illustrate the ambiguity when the refractive field has non-negligible second-order component, we now use two thick lenses and pick the aperture away from the lens center, so that a significant radial distortion is observed, as shown in Figure 3.16. Lens 1 has a longer focal length, so that its refractive field has a weaker second order component than that of lens 2.

First, we move these lenses perpendicularly to the background structure (Figure 3.16(a)). Although they move at different speeds, the observed sequences are quiet similar. This indicates that it is hard to recover the horizontal component of the motion. Next, we move them parallel to the background structure (Figure 3.16(b)). Now the observed boundary moves together with the lenses, and to generate two similar sequences, these two lenses must move in the same speed. This experiment shows that we can recover the component of the motion parallel to the background structure, but not the component of the motion perpendicular to it, which is again consistent with our theory.

**3D-printed glasses.** We also illustrate the ambiguity using Gaussian-shaped glasses, similar to the synthetic experiment shown in Figure 3.10. We 3D-printed two 3D Gaussian-shaped glasses shown in the first row of Figure 3.17 and captured four sequences using the same setup as described in Figure 3.10. We also get the same result as described in Figure 3.10, that is, the component of the motion perpendicular to the
background is hard to estimate (Figure 3.17(a)), but the component of the motion parallel to the background can be recovered (Figure 3.17(b)).

**Hot air from a candle.** At last, we verify our theory by two sequences captured through hot air generated by a burning candle, as shown in Figure 3.18. The background texture consists of patterns from two color channels. The blue channel of the background is fully-textured, from which we can correctly recover the upward air motion (Figure 3.18(b,d)), and we consider the recovered motion from this channel as the ground truth (using the refractive flow algorithm described in Section 3.2.2). The red channel of background contains texture only in one direction (Figure 3.18(a,c)). When the texture in red channel is perpendicular to the direction of air motion (Figure 3.18(a)), the estimated motion is incorrect (different from the ground truth motion in Figure 3.18(b)), and when the background texture is parallel to the direction of air motion (Figure 3.18(c)), the estimated motion is roughly correct (similar to the ground truth motion Figure 3.18(d)).
Chapter 4

Estimating the Structure of Objects from the Spectrum of Their Vibration

4.1 Introduction

Motion information is widely used to understand the object segments from videos. Adelson et al. [1] proposed a pioneering work on “layered image presentation”, which shows that an image sequence can be represented as a set of layers, each of which has a consistent motion. This representation is later used for video segmentation [27, 66, 135] and motion analysis [9, 60, 122, 134]. Recently, Tsai et al. also showed that motion analysis helps to propagate the foreground segmentation map in the first frame to the rest of a video [126]. All these methods are based on the assumption that points on different parts of an object have different motions so that they can be differentiated based on observed absolute pixel displacement.

However, the pixel displacement might not always be enough to infer the structure of an object. Take the branches shown in Figure 4.1(b) as an example. Three points, P1, P2, and P3, are on two occluding branches, marked by the orange and light blue lines respectively. There are two plausible explanations of their connections: P1 either connects to P2 or P3. Due to the self-occlusion, it is hard to infer which explanation is correct just from their appearance (Figure 4.1(a)). It is also very hard to infer which explanation is correct just from absolute pixel movement of these three points, because, the movement of three nodes are dominated by the vibration of the root branch, so they share almost the same movement, as shown in Figure 4.1(c).

To resolve this ambiguity, we propose a structure estimation algorithm based on spectral analysis. This is inspired by the observation that pixels on different branches, though sharing similar trajectories, often have distinctive modes in the frequency domain. As shown in Figure 4.1(d), P3 has a distinct amplitude at certain frequencies, compared to P1 and P2. Thus, P3 is more likely to be on a separate branch.

To understand how the structure of a tree relates to the spectrum of its vibration, we use a physics-based link model from the field of botany [93]. Based on this model, we find the key property of tree structure: each branch is a linear time-invariant (LTI) system with respect to the vibration of root. With this property, we can infer the natural frequencies of each sub-branch in a tree from its frequency response, and group nodes based on the inferred natural frequencies.
Figure 4.1: Estimation of the hierarchical tree structure from a video. Inference based on a single frame from a video (a) has inherent ambiguity; figure (b) shows an example, where it is hard to tell from appearance whether the point $P_1$ is connected to $P_2$ (orange curve) or to $P_3$ (blue curve). Using motion information in the temporal domain does not help much as well, as their movements are almost identical (c). We, however, observe that the difference becomes significant in the frequency domain (d), from which we can infer that $P_1$ is more likely to connect to $P_2$ due to their similar spectra. We therefore develop an algorithm to infer tree structure based on both vibration spectra and appearance information. We show results in (e).

Based on this idea, we then develop a hierarchical grouping algorithm to infer tree structure, using both spectral motion signals and appearance cues. As each node in a tree may connect to an indefinite number of children, our algorithm employs nonparametric Bayesian methods for inference.

For evaluation, we collect videos of both artificial and real-world trees. We demonstrate that our algorithm works well in recognizing tree structure, using both appearance cues and spectra of vibration. We also conduct ablation studies to reveal how each component in our algorithm contributes to its final performance. Besides, we also justify the LTI assumption through theoretical analysis, numerical simulation, and empirical observations.

4.2 Related Work

Motion for structured prediction Researchers in computer vision have been using motion signals for various tasks [14, 98, 120, 150]. For structured prediction in particular, the layered motion representations [134]
Figure 4.2: (a) Hierarchical beam structure. (b) Force analysis for one of the branches (the one marked by dashed rectangle in (a)).

have been studied and applied extensively [61, 122]. These works model motion signals in the temporal domain, and are therefore not designed for scenarios where objects may only have subtle motion differences.

Regarding spectral analysis of motion, the pioneer work of Fleet and Jepson [37] discussed how phase signals could help to estimate object velocity. Gautama and Van [44] extended the work, proposing a phase-based approach for optical flow estimation. Zhou et al. [155] also discussed how phase information helps recognizing object motion. Recently, there have also been a number of works on visualizing and magnifying subtle motion signals from video [28, 142], and Rubinstein et al. did a thorough review in [106].

Modeling tree vibration Tree vibration is an important research area in the field of botany [58, 89]. Moore and Maguire [89] reviewed the concepts and dynamic studies by examining the natural frequencies and damping ratios of trees in winds. Recently, James et al. [58] reviewed tree bio-mechanics studies using dynamic methods of analysis.

Our formulation of tree vibration is based on the lumped-mass procedure. Related literature include spring-mass-damper models for trees as a single mass point [88], or as a complex system of coupled masses that represent the trunk and branches [57, 93]. Our formulation also considers a tree as a system of coupled masses, but different from [93] which only studied one-layer structure, we model hierarchical tree structure of multiple layers.

Bayesian theory of perception Researchers have developed Bayesian theories for human visual perception in general [69, 73, 90], and for object motion perception in particular [22, 138]. Our inference algorithm draws inspirations from the recent hierarchical Bayesian model for object motion from Gershman et al. [46], which employs the nested Chinese restaurant process (nCRP) [19] as the prior of object structure.

4.3 Problem Definition

In this section, we discuss how the temporal complex spectra of all nodes in a tree relate to its structure.

We start by introducing the physics-based hierarchical link model, a straight-forward extension of the
physics-based link model proposed by Murphy et al. [93], which models a tree as a set of beams with certain mass and stiffness (see Figure 4.2a). Unlike Murphy et al. [93], we do not solve the mass and stiffness of all the beams, which is very challenging in the presence of noise and occlusion. Rather, based on this hierarchical physics-based link model, we derive a set of ordinary differential equations (ODE) of node vibrations (Section 4.3.2). From these equations, we will then derive an important property of vibration (Section 4.3.3): each sub-branch of a tree is a linear time-invariant system under certain assumptions. Based on this property, we then designed a Bayesian inference algorithm that estimates tree structure from its vibration (Section 4.4 and Section 4.5).

### 4.3.1 A Physics-Based Link Model

Let us start with the physical model of a tree. Following Murphy et al. [93], we use a rigid link model to describe the vibration of a tree, as shown in Figure 4.2a. In this model, each branch $i$ of the tree is modeled as a rigid beam with a certain mass $m_i$ and length $l_i$. Under the uniform mass assumption, the center of mass of a branch is at $\frac{l_i}{2}$. Each branch connects to its parent through a torsional spring with stiffness $k_i$. Based on this model, we derive a set of non-linear ordinary derivative equations (ODEs) that describe the relationship between the vibration of a tree and its structure and physical properties.

We describe the vibration of a tree by deviation angles $\theta_i$ of branches. As shown in Figure 4.2b, let $\hat{\theta}_i$ be the angle between a branch and vertical line when the tree is static (no external forces except gravity), and let $\theta_i$ be the deviation angle from its static location when the tree is vibrating ($\theta_i$ is a variable changes over time). To derive the governing equations for $\theta_i$, we start from the Newton’s law applied to each branch $i$ (see Figure 4.2b)$^1$:

$$ma_i = -r_i + \sum_{c \in C_i} r_c + mg, \quad (4.1)$$

where $r_c \in \mathbb{R}^2$ is defined as the force exerted by branch $c$ on its parent, and $C_i$ is the set of children of branch $i$. Note the negative sign before $r_i$ due to our definition and Newton’s third law. Branch $i$’s acceleration $\alpha_i \in \mathbb{R}^2$ is defined as the acceleration of the branch’s center of mass.

In addition, we have the rotation equation:

$$I_i \dot{\omega}_i = -k_i \theta_i + \sum_{c \in C_i} k_c \dot{\theta}_c + r_i \times \omega_i + \sum_{c \in C_i} r_c \times x_i, \quad (4.2)$$

where $I_i$ is the moment of inertia of branch $i$ when it rotates around its center. Also, the branch acceleration $\alpha_i$ also has the following relationship with the acceleration of its endpoint $\alpha_{i_0}$:

$$\alpha_i = \alpha_{i_0} + \dot{\omega}_i \times x_i + \omega_i \times (\dot{\omega}_i \times x_i), \quad (4.3)$$

$^1$In this chapter, we use a lower-case letter $a$ to denote a scalar, a bold lower-case letter $\mathbf{a}$ to denote a vector, and a capital letter $\mathbf{A}$ to denote a matrix. We denote the matrix product as $\mathbf{A} \mathbf{b}$, where $\mathbf{A} \in \mathbb{R}^{n \times m}$ and $\mathbf{b} \in \mathbb{R}^m$. 57
Note that all the equations above do not include any fictitious force, all quantities are expressed as global values under the reference frame.

Therefore, the angular velocity and angular acceleration of branch \( i \) is:

\[
\omega_i = \dot{\theta}_i + \sum_{p \in P_i} \dot{\theta}_p, \quad \dot{\omega}_i = \ddot{\theta}_i + \sum_{p \in P_i} \ddot{\theta}_p, \tag{4.4}
\]

where \( P_i \) is the set of ancestors of branch \( i \).

At last, replacing the angular acceleration \( (a_i \text{ and } a_{i,a}) \) and angular speed \( \omega_i \) in Equations (4.2) and (4.1) using Equations (4.3) and (4.4), and eliminate forces between branches \( r_i \), we get the ordinary derivative equation with respect to the deviation angles of all branches \( \theta_i \):

\[
I_i f_i(\overline{\theta}) = -k_i \dot{\theta}_i + \sum_{c \in C_i} k_c \dot{\theta}_c + r_i(\theta, \dot{\theta}, \overline{\theta}) \times x_i + \sum_{c \in C_i} r_c(\theta, \dot{\theta}, \overline{\theta}) \times x_i, \tag{4.5}
\]

where \( r_i(\theta, \dot{\theta}, \overline{\theta}) \) is a vector functions of \( \theta, \dot{\theta}, \text{and } \overline{\theta} \). See the appendix for the detailed definition of that function.

### 4.3.2 ODE of Node Vibration

The ODE 4.5 is highly nonlinear as it contains sinusoid and quadratic terms that \( m \). In order to solve this ODE, we first linearize it around its stabilized solution. First, we assume that the deviation angle \( \theta_i \) of each branch \( i \) is small and ignore all \( O(\theta^2) \) terms. Under this assumption, the quadratic term of angular velocity \( O(\dot{\theta}^2) \) can also be ignored, because according to the conservation of energy, the potential energy \( \frac{1}{2}k\theta^2 \) of a branch is on the same scale of its kinetic energy \( \frac{1}{2}I_i \dot{\theta}^2 \). Since \( O(\theta^4) \) is negligible, \( O(\theta^6) \) is also negligible.

Then, omitting all quadratic terms of deviation angles and angular speed, we get a linear system of angular vibrations \( \theta \) of all branches:

\[
M \ddot{\theta} + K \theta = 0, \tag{4.6}
\]

where \( M \) and \( K \) are two matrices depending on the structure of a tree and its physical properties, including the moment of inertia \( (I) \), mass \( (m) \), and stiffness \( (k) \) of all branches. Note that the constant term must be zero, as when the tree is stable, we have \( y = \ddot{y} = 0 \). See the Appendix E for the detailed derivation.

At last, in practice, from an input video, it is easier to measure the 2D shift of each node, rather than the rotation of each branch. To derive the ordinary differential equation of 2D shifts of all nodes from Equation (4.6), we denote the 2D location of \( i \)-th node when the tree is stable as \( \hat{y}_i \), and denote the 2D shifts from its stable location as \( y_i \). Thus the 2D location of \( i \)-th node satisfies:

\[
y_i + \hat{y}_i = \sum_{j \in P_i} l_r n(\theta_j + \dot{\theta}_j), \tag{4.7}
\]

where \( n(\theta) = (\cos(\theta), \sin(\theta)) \) (recall that \( P_i \) is the set of ancestors of branch \( i \)). Let \( y \) be the concatenation of 2D shifts of all the nodes. Plugging Equation (4.7) to Equation (4.6), we have:

\[
Ny + Ly = 0, \tag{4.8}
\]
where $N$ and $L$ are matrices depending on $M$, $K$, $l_j$, and $\theta_j$.

### 4.3.3 Inferring the Mode of each Sub-branch

Based on the second order ODE derived in the previous section, we can infer the modes of each sub-branch and use that information to assign nodes to different sub-branches.

**Proposition 4.3.1 (Each sub-branch is a LTI-system).** Consider a sub-branch connected to a root node undergoing a forced vibration. In one experiment, the displacement of the root node at time $t$ is $y_{root}^1(t)$ and the displacement of one of the leave node in that branch is $y_{leaf}^1(t)$, where $t$ is the temporal index. For the same tree, in another experiment, the displacement of the root node is $y_{root}^2(t)$ and the displacement of the same leave node in that branch is $y_{leaf}^2(t)$. Then if the displacement of the root node is $\alpha_1 \cdot y_{root}^1(t) + \alpha_2 \cdot y_{root}^2(t)$, where $\alpha_1, \alpha_2 \in \mathbb{R}$, the vibration of the same leaf node must be:

$$\alpha_1 \cdot y_{leaf}^1(t) + \alpha_2 \cdot y_{leaf}^2(t)$$

This is a straightforward corollary of Equation (4.8), which is a linear second order ODE. The system is also time-invariant, since all matrices in Equation (4.8) are constants and do not change in time.

The practical implication of this property is that we can infer the mode of free vibration of each sub-branch as if that sub-branch is disconnected from the rest of the tree. Let $S$ be a set of nodes in a sub-branch, and $Y_i(\eta)$ be the temporal spectrum of the displacement of the $i$-th node in that branch ($i \in S$), where $\eta$ is the frequency index. Let $Y_{root}$ be the temporal spectrum of the root displacement. Because each sub-branch is a LTI-system, the frequency response of that sub-branch is

$$\overline{Y}_i(\eta) = \frac{Y_i(\eta)}{Y_{root}(\eta)}, \forall \eta.$$  \hspace{1cm} (4.9)

It is well known that when there is no damping, the natural frequencies of an oscillating system coincide with its resonance frequency [39, Chapter 4]. In our case, this suggests that the natural frequencies of a sub-branch are the same as the modes of frequency response of that branch\(^2\).

To illustrate this point, we create a synthetic tree. Figure 4.3(a) shows a tree with two sub-branches ($Y_{2-4}$ and $Y_{5-7}$). All nodes have the similar power spectrum as their vibrations are dominated by the vibration of the root node ($Y_1$). To distinguish the spectra of two sub-branches, we calculate the frequency response of each node, which is the ratio between spectrum of the root and the spectrum of each branch, and there is a clear difference between the frequency responses of two branches, as shown in Figure 4.3(b). The modes of frequency response also match the modes of free vibrations of each sub-branches, as if they are detached from the root (see Figure 4.3(c) and (d)).

\(^2\)Even in the presence of damping, the difference between the modes of frequency response and the modes of free vibration are small, as damping is small during our experiments.
Figure 4.3: Spectrum of simulated vibration. Each shows one experimental setup. The left column shows the simulated tree structure and the right column shows the temporal spectra of its vibration. (a) shows a tree with two sub-branches ($Y_2$-$4$ and $Y_5$-$7$). All nodes have the similar power spectrum as their vibrations are dominated by the vibration of the root node ($Y_1$). To distinguish the spectra of two sub-branches, we calculate the frequency response of each node, which is the ratio between spectrum of the root and the spectrum of each branch, and there is a clear difference between the frequency responses of two branches (b). The modes of frequency response also match the modes of free vibrations of each sub-branches, as if they are detached from the root (c) and (d).

Once we computationally separate all nodes from the root by calculating their spectrum response, we can then group them into different sub-branches. This is because the natural frequencies of each sub-branch depend on the structure, mass, and stiffness of that sub-branch. Different sub-branches normally have different natural frequency distribution, and therefore they normally have different different frequency responses and we can group nodes to different branches based on their natural frequencies, a.k.a, the modes of frequency response (Equation (4.9)). In practice, the modes of frequency response are not a robust measure in the presence of noise and damping. Therefore, we group nodes based on their the normalized power spectra and phases instead.
Figure 4.4: Overview of our framework. We take a video (a) and a set of keypoints (b) as input (I). We then obtain appearance cues (II) through several intermediate steps (Section 4.4). We use normalized amplitudes (g) and phases (h) of keypoints as our vibration signals (III). Finally, we apply our inference algorithm (Section 4.5) for tree structure estimation.

4.4 Extracting Motion and Appearance Features

We discuss our structure estimation algorithm based on the tree models we introduce in the previous section in this and next section. We first present the method to robustly estimate the power spectrum of each node (motion information) from an input video. To improve the robustness of our algorithm, we also use the appearance of the tree to help the structure estimation. In the next section, we present the inference algorithm based on the extracted motion and appearance feature.

Motion Given an input video, we first manually label all nodes in the first frame and then track them over time, and then we use the optical flow algorithm to track nodes. There are many tracking algorithms that can extract trajectories of sparse keypoints [50, 54, 84, 107], but we choose to calculate the dense motion field for two reasons. First, most vibrations are small, and optical flow is known to perform well to capture the small motion with subpixel accuracy. Second, sparse tracking algorithms, like the KLT tracker [84], might suffer the aperture problem, as most of branches only contain one-dimensional local structure. On the other hand, dense optical flow algorithms aggregate the information from other locations, so it would be more robust to the aperture problem.

Specifically, we first use Liu's optical flow package [82] to compute a dense flow field from the first frame to one of the frame t in the sequence. We then get trajectory of each node in the sequences from dense motion fields through interpolation. At last, we apply Fourier transform to the trajectory of each node independently to get its complex spectrum Y, and extract its modes from the 5-th order spectral envelope [40].
Appearance We use an overcomplete connectivity matrix of interest points as our appearance cues. As shown in Figure 4.4-II, we compute the matrix via the following steps: obtaining a contour map, computing the closure of each interest point, flood-filling the contour map from all closures, and adding edges to junctions (Figure 4.4f).

Given the first frame from an input video, we first use Canny edge detector [23] with threshold 0.5 to obtain an initial contour map (Figure 4.4c). Then, for each interest point i, we consider all contour pixels $S_i$ whose distance to i is no larger than $r_i$. We search for the minimum $r_i$ such that if we connect i to all pixels in $S_i$, the angle between each two adjacent lines is no larger than 30°. We call $S_i$ the closure for point i (Figure 4.4d).

We then apply a shortest-path algorithm to obtain the connectivity map of all interest points. Our algorithm is a variant of the Dijkstra's algorithm [30], where there is a hypothetical starting point connecting to pixels in the union of all closures with cost 0. The cost between two 8-way adjacent pixels is 0, if they're both on the contour map, or 1 otherwise. The algorithm is then in essence expanding all closures simultaneously. When it finishes, we connect two keypoints if their corresponding closures are adjacent after expansion (Figure 4.4e).

To balance the expansion rate of each closure, we use a tuple $(c_i, d_i)$ as the entry for any pixel i in the priority queue, where the primary key $c_i$ is the traditional term for the distance on the graph from i to the origin, and the secondary key $d_i$ is the Chebyshev ($L_\infty$) distance between i to the center of the closure it belongs to.

Finally, junctions in the image may be because of a tree fork, where the junction point should connect to all its neighbors, or two overlapping branches, where it should only connect to points nearby that share similar motion patterns. To deal with the case, for all points that has 4 or more neighbors, we add an edge between each pair of its neighbors whose angle is no smaller than 135°. This leads to an overcomplete connectivity matrix $E$ (Figure 4.4f), which we use as our appearance cues.

4.5 Inference

A toy example We start with a high-level overview of our hierarchical inference algorithm along with a toy tree with three levels of hierarchy (Figure 4.5). As shown in Algorithm 2, given the root, our algorithm first
Algorithm 2: Our hierarchical clustering algorithm

Data: A set of nodes with complex spectra \( Y = \{Y_i\} \)

Calculate the free vibration of each node in this tree

for each node \( i \) do
    \( Y_i \leftarrow Y_i / Y_r \)
end

Cluster nodes based on their appearance and frequency using Gibbs sampling.

Let \( \{S_j\}_{j=1,...,k} \) be all \( k \) clusters

for \( j = 1, \cdots, k \) do
    Select subroot \( r_j \)
    Call \( \text{cluster}(Y_{S_j}, r_j) \) recursively
end

Algorithm 2: Our hierarchical clustering algorithm

computes the free vibration of the rest of nodes (Step I), groups them into several clusters (Step II), and then recursively finds tree structure for each cluster (Step III).

In this toy tree with \( vi \) as the root, the algorithm groups the other nodes into two clusters: \((v_2, v_4, v_5)\) and \((v_3, v_6, v_7, \ldots, v_{11})\), as shown in Figure 4.5b. For each subtree, the algorithm recursively applies itself for finer-level tree structure. Here in the right branch, we get two level-2 subtrees \((v_6, v_8, v_9)\) and \((v_7, v_{10}, v_{11})\) (Figure 4.5c).

Step I: Computing free vibration

We first compute the vibration of each node given the root. Based on Equation (4.9), we divide the complex spectrum of each leaf node by the complex spectrum of the root. Note that under certain frequency, the complex spectrum of the root might be close to zero. Therefore, a direct division might magnify the noise.

In order to deal with this problem, we use the following equation to calculate the spectrum of each node \( i \) after removing the root \( r \):

\[
\frac{Y_i \cdot Y_r^*}{|Y_r|^2 + \epsilon^2} \tag{4.10}
\]

where \( Y_r^* \) is the complex conjugate of \( Y_r \), and \( \epsilon \) controls the noise level. Note that when \( \epsilon = 0 \), Equation (4.10) reduces to normal division as

\[
\frac{Y_i \cdot Y_r^*}{|Y_r|^2} = \frac{Y_i \cdot Y_r^*}{Y_r \cdot Y_r^*} = \frac{Y_i}{Y_r} \tag{4.11}
\]

Step II: Grouping nodes

We group nodes into clusters \( \{S_j\} \) under the assumption that nodes in each cluster share similar vibration patterns (complex frequencies) and appearance cues. Each node has an unknown number of children, we use a Chinese Restaurant Process (CRP) prior [19] over the tree structure. Let \( z_i \) be the index of cluster that node
i is assigned to, and let \( Z = \{ z_i \} \) be the assignment of all nodes. The joint probability of assignment is

\[
P(Z|E, Y) = P_{\text{CRP}}(Z) \cdot P_a(Z|E) \cdot P_m(Z|Y),
\]

where \( P_{\text{CRP}}(\cdot) \) is the CRP prior, \( P_a(\cdot) \) is the likelihood based on appearance, and \( P_m(\cdot) \) is the likelihood based on motion.

**Appearance term:** We expect all the nodes in a sub-branch connect to each other, and they should also connect to the root. To ensure this, we define the appearance term as

\[
\log P_a(Z|E) = \sum_{x_i = x_j} \alpha \cdot \mathbb{1}(i, j|Z, E) + \sum_i \beta \cdot \mathbb{1}(i, r|Z, E),
\]

where \( \mathbb{1}(i, j) \) is the indicator function of whether there exists a path between nodes \( i \) and \( j \) given the current assignment \( Z \) and the estimated connectivity matrix \( E \) (see Section 4.4).

**Motion term:** We use two statistics of the spectrum in the probabilistic model: the normalized amplitude \( Y_i^n = |Y_i|/||Y_i||_2 \) and the phase \( Y_i^p = \angle(Y_i) \).

\[
\log P_m(Z|Y) = \sum_i -\sigma_n^2||Y_i^n - C^n_k||^2_2 - \sigma_p^2||Y_i^p - C^p_k||^2_2,
\]

where \( C^n_k \) and \( C^p_k \) are the mean normalized amplitudes and phases of all nodes in cluster \( k \).

Given the joint probability defined in Equation (4.12), we run Gibbs sampling [45] for 20 iterations over each node's assignment \( z_i \).

**Step III: Recursion**

As illustrated in the toy example (Figure 4.5), for each cluster \( S_j \), our algorithm selects the node that is closest to the root \( r \) in the Euclidean space as the subroot \( r_j \). It then infers subtree structure for the cluster \( S_j \) recursively.

### 4.6 Evaluations

In this section, we discuss how we use simulation to verify our formulation presented in Section 4.3. We then show both qualitative and quantitative results of our method on videos of artificial and real trees.

**Simulation** Based on formulation described in 4.3.1, we implemented a tree simulator by solving Equation (4.5) using Euler Method [34]. As shown in Section 4.4, the analytic form of ODE is very complicated. Therefore, we do not eliminate all the redundant variables, including the acceleration of the branch \( (a_i \text{ and } a_{i\theta}) \), forces between branches \( (ri) \), and angular speed of each branch \( (\omega_i) \). Instead, we directly solve Equation (4.1) and 4.2 numerically. Here we manually specify the structure a tree and physical properties of each branch, including its mass, stiffness, and length, and numerically solve for the rotation angle of each branch. See Appendix E for the detailed derivation.

Since the Euler method is likely to introduce numerical error for time-stepping, the energy of the simulated system might increase, resulting in unnaturally large vibrations. To prevent our simulation from growing in
energy, we force the system to have constant total energy for every time-stepping update. If the system’s energy were to increase during an update, we rescale the kinetic and potential energy of each branch to ensure that the total energy of the system is constant. This makes our simulation robust and stable.

Figure 4.7 shows the vibration modes of the simulated tree structure. A shows the initial geometry of the tree; B,C,D are the three mode shapes. On the bottom, the mode of power spectra (the natural frequencies) of the trunk and two branches matches the three mode shapes of the tree.

Real videos We record videos of both artificial and real trees. For artificial trees, we take 5 videos in an indoor lab environment, where we use a fan to simulate the effects of wind. We also take 5 videos of outdoor real trees. All videos are taken at 24 frames per second by a Canon EOS 6D DSLR camera, with resolution 1920x1080.

To understand and analyze motions, we also take high-speed videos of trees using an Edgertronic high-speed camera. We captured 1 normal-speed video (30 FPS) and 5 high-speed videos with a frame rate varying from 60 to 500. For each video, we manually label around 100 interest points, including endpoints of branches, junctions, etc.

Evaluation We compare our full model, which makes use of appearance and vibration cues jointly (appearance + motion), with a simplified variant, which uses only appearance information, but ignores all motion signals during inference.

Results on normal-speed videos Figure 4.8 shows the qualitative results. Our algorithm works well on real videos of both artificial and real trees. In the last row, we show our algorithm can deal with challenging cases; for occlusion, twigs that are indistinguishable from pure visual appearance, our algorithm
Figure 4.7: Visualization of the mode shapes of the vibration. The top row shows the tree structure (a) and its three mode shapes (b–d), and the bottom row shows the power spectra of the trunk and two branches (e).

Table 4.1: Percentage (%) of nodes whose parents are correctly recovered. We compare our algorithm with a variant using appearance cues only. On all videos, including motion cues consistently improves the accuracy of structure estimation.

<table>
<thead>
<tr>
<th>Methods</th>
<th>Artificial trees</th>
<th>Real trees</th>
<th>High-speed videos of trees</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>A1</td>
<td>A2</td>
<td>A3</td>
<td>R1</td>
</tr>
<tr>
<td>Appearance</td>
<td>40</td>
<td>31</td>
<td>90</td>
<td>67</td>
</tr>
<tr>
<td>Appearance + motion</td>
<td>54</td>
<td>45</td>
<td>100</td>
<td>81</td>
</tr>
</tbody>
</table>

still correctly recovers tree structure via motion signals.

To quantitatively evaluate the proposed algorithm, we manually label the parents of each node and use it as ground truth. Table 4.1 shows the results, where we compare different methods by the percentage of nodes whose parents are correctly recovered. Our algorithm achieves good performance in general, and on videos of all types, including motion cues consistently improves the accuracy of the inference.
Figure 4.8: Estimated tree structure on real videos. A1 - A2: results on artificial trees; R1 - R8: results on real trees. In the last row, we show cases where appearance information is not enough for inferring the correct structure. Using vibration signals, our algorithm works well in these cases.

Results on high-speed videos At last, to illustrate which frequencies are the most informative for structure estimation, we test our algorithm on videos captured at six different frame rates, as shown in Figure 4.6. All the videos contain 1000 frames. Intuitively, the root branches have higher stiffness and lower natural frequencies. Therefore, low-frame-rate videos, from which the estimation of low-frequency modes is more accurate, are better for recovering the major structures of a tree. On the other side, high-frame-rate videos provide more information on high-frequency vibration, so it is better to differentiate tiny structures.

To illustrate this point, we first pick two points (P1 and P2 in Figure 4.6(c)) on two major branches of the tree, and compare their power spectra as shown in Figure 4.6(a). At 60 FPS, the power spectra of these two nodes are different in the whole spectrum, while at 500 FPS, they are only different in lower spectrum, as the natural frequencies of the main branches are low. Then we pick two points (P3 and P4 in Figure 4.6(c)) on two small branches of the tree, and compared their power spectra (see Figure 4.6(b)). Now in both 60 FPS and 200 FPS videos, they have the similar spectrum, and the difference in modes only become significant in 500 FPS video. Also in Figure 4.6(c), we show that the estimation errors from low-frame videos (60 or 100 FPS) on the top-right corner are eliminated when using 500 FPS as input. This also shows that high-speed
videos are better for detailed-structure estimation.

The quantitative evaluation of the results on high-speed videos are shown in the last few columns of Table 4.1, where H1 to H6 refer to videos captured at 30, 60, 100, 200, 300, 500 FPS, respectively.
Chapter 5

Synthesizing the Movement of Objects in the Future

5.1 Introduction

In Chapter 4, we showed that it is possible to infer the structure of a tree-shaped object from its vibration. However, many objects have much more complicated structure than trees, and just from their movement, it is hard or even impossible to recover their internal structures. For example, the deformation of a human body is constrained by hundreds of bones, joints, and muscles, and it is very challenging to infer their internal structure unless using specialized equipments such as X-ray examiners.

![Figure 5.1](image)

Figure 5.1: Predicting the movement of an object from a single snapshot is often ambiguous. For instance, is the girl’s leg in (a) moving up or down? We propose a probabilistic, content-aware motion prediction model (b) that learns the conditional distribution of future frames. Using this model we are able to synthesize various future frames (c) that are all consistent with the observed input (a).

Still, even without knowing the exact internal structure of objects, humans are good at predicting how they will change over time. For example, very few people actually studies the detailed bone structure of our body, but can still predict the girl shown in Figure 5.1 will move her leg and keep her arms stationary. People can make this prediction because they have observed how human moves for many years, and empirically learn its possible deformation space of our body. This shows that we can also teach our computer to learn the
deformation space of an object and synthesize how it might move in the future, without knowing the physics behind them.

In this work, we study the problem of predicting and synthesizing the movement of an object from its pose. This is an interesting problem, as it demonstrates the ability of a model to learn the correlation between motion and appearance. In addition, the model also needs to recognize different parts of an object, such as legs and arms of a human body, in order to predict their movement.

There is an intrinsic ambiguity in motion prediction. For example, given the single image of the girl shown Figure 5.1(a), she might either move her leg up or down. Therefore, we need to model the conditional distribution of future frames given an observed image (Figure 5.1(b)). Samples from the distribution also allows us to visualize the many possible ways that an input image is likely to change over time (Figure 5.1(c)).

Modeling the distribution of future frames given only a single image as input is a very challenging task for a number of reasons. First, natural images come from a very high dimensional distribution that is difficult to model, which makes it hard to design a generative model for realistic images. Second, the correlation between the appearance of an object and its possible movement is not deterministic: there are multiple possible movement of an object given its current pose.

Previous works on future frame prediction and synthesis can be grouped to two categories: pixel-based synthesis and flow-base synthesis. Pixel-based synthesis methods \[35, 86, 128\] directly synthesize the future frame from the observed input. However, because the dimension of a natural image is very high, pixel-based synthesis methods cannot capture every details of an image, and often generate blurry results. Flow-based synthesis methods \[81, 99, 132\] partially resolve this problem by first predicting the flow field between the input and the future frames and then warping the input frame based on the predicted flow field. However, most of flow-based synthesis methods are deterministic and cannot model the uncertainty of future frames.

To solve these problem, we proposed a segment-based synthesis network. The main idea is that pixels often move in groups. For example, in Figure 5.1, all the pixels on the leg of the girl have a consistent motion. Therefore, instead of predicting how pixels move between frames, we only need to predict how each segment moves. Based on this idea, the proposed segment-based synthesis network consists of two parts: segmentation network and transformation network. Given an input image (Figure 5.2(a)), the segmentation network first finds all segments of the input object (for the ease of illustration, we only show two segments in Figure 5.2(b)). Then the transformation network samples the motion of each segment, and deforms each segment based on sampled motion and synthesizes the future frame (Figure 5.2(c)).

There are several advantages of this model. First, the network does not directly synthesize the output frame, which normally results in a blurry output, but moves pixels in the input image based on the sampled motion. Second, the network only samples the movement of each segment, instead of all the pixels or a dense flow field. Since the motion of segments has lower dimension than an output image or a dense motion field, its distribution is easier to model and the network can sample more diverse and realistic movement of an
The leg moves up
Sampled motion of segments
The leg moves down
(c) Desired synthesized frames

Figure 5.2: Illustration of segment-based image synthesis. Given an input, the network first chop it into segments (b), each of which has a consistent motion (here we only show two segments for illustration, and the actual network outputs 64 segments). Then, the network samples the movement of each segment, and synthesize the future frame based on sampled movement (c). Images in (c) are only for illustration. They are not the actual output of our algorithm.

We test the proposed model on two synthetic datasets as well as a dataset generated from real videos. We show that, given an RGB input image, the algorithm can successfully model a distribution of possible future frames, and generate different samples that cover a variety of realistic motions. In addition, we demonstrate that our model can be easily applied to tasks such as visual analogy-making, and present an analysis of the learned network representations.

5.2 Related Work

Motion priors  Research studying the human visual system and motion priors provides evidence for low-level statistics of object motion. Pioneering work by Weiss and Adelson [137] found that the human visual system prefers slow and smooth motion fields. More recent work by Roth and Black [105] analyzed the response of spatial filters applied to optical flow fields. Fleet et al. [38] also found that a local motion field can be represented by a linear combination of a small number of bases. All these works focus on the distribution of a motion field itself without considering any image information. On the contrary, our context-aware model captures the relationship between an observed image and its motion field.

Motion or future prediction  Our problem is closely related to the motion or feature prediction problem. Given an observed image or a short video sequence, models have been proposed to predict a future motion field [81, 99, 132, 133, 148], a future trajectory of objects [131, 143], or a future visual representation [129]. Most of these works use deterministic prediction models [99, 129]. Recently, and concurrently with our own work, [133] found that there is an intrinsic ambiguity in deterministic prediction, and propose a probabilistic prediction framework. Our model is also a probabilistic prediction model, but it...
directly predicts the pixel values, rather than motion fields or image features.

**Parametric image synthesis** Early work in parametric image synthesis mostly focus on texture synthesis using hand-crafted features [100]. More recently, works in image synthesis have begun to produce impressive results by training variants of neural network structures to produce novel images [48, 146, 147, 156]. Generative adversarial networks [29, 47, 101] and variational autoencoders [67, 152] have been used to model and sample from natural image distributions. Our proposed algorithm is also based on the variational autoencoder, but unlike in this previous work, we also model temporal consistency.

**Video synthesis** Techniques that exploit the periodic structure of motion in videos have also been successful at generating novel frames from an input sequence. Early work in video textures by Schödl et al. [113] proposed to shuffle frames from an existing video to generate a temporally consistent, looping image sequence. These ideas were later extended to generate cinemagraphies [64], seamlessly looping videos containing a variety of objects with different motion patterns [3, 79], or video inpainting [141]. While high-resolution and realistic looking videos are generated using these techniques, they are often limited to periodic motion and require an input reference video. In contrast, we build an image generation model that does not require a reference video at test time.

Recently, several network structures have been proposed to synthesize a new frame from observed frames. They infer the future motion either from multiple previous frames [86, 118], user-supplied action labels [35, 95], or a random vector [128]. In contrast to these approaches, our network takes a single frame as input and learns the distribution of future frames without any supervision.

### 5.3 Problem Definition

In this section, we describe how to sample future frames from a current observation image. Here we focus on next frame synthesis; given an RGB image $I$ observed at time $t$, our goal is to model the conditional distribution of possible frames observed at time $t + 1$.

Formally, let $\{(I^{(1)}, J^{(1)}), \ldots, (I^{(n)}, J^{(n)})\}$ be the set of image pairs in the training set, where $I^{(t)}$ and $J^{(t)}$ are images observed at two consecutive time steps. Using this data, our task is to model the distribution $p_\theta(J|I)$ of all possible next frames $J$ for a new, previously unseen test image $I$, and then to sample new images from this distribution. In practice, we choose not to directly predict the next frame, but instead to predict the difference image $v = J - I$, also known as the Eulerian motion, between the observed frame $I$ and the future frame $J$; these two problems are equivalent. The task is then to learn the conditional distribution $p_\theta(v|I)$ from a set of training pairs $\{(I^{(1)}, v^{(1)}), \ldots, (I^{(n)}, v^{(n)})\}$.

### 5.3.1 A Toy Example

Consider a simple toy world that only consists of circles and squares. All circles move vertically, while all squares move horizontally, as shown in the Figure 5.3(a). Although in practice we choose $v$ to be the RGB
Figure 5.3: Imagine a world composed of circles that move vertically and squares that move horizontally (a). We consider 3 different models (b-d) in the text to learn the mapping from an image to a motion field. The top row shows graphical models and the bottom row shows corresponding network structures. The deterministic motion prediction structure shown in (b) attempts to learn a one-to-one mapping from appearance to motion, but is unable to model multiple possible motions of an object or to generalize to previously unseen images. The content-agnostic motion prior structure shown in (c) is able to capture a low-dimensional representation of motion, but is unable to leverage cues from image appearance for motion prediction. The content-aware probabilistic motion predictor (d) brings together the advantages of models of (b) and (c) and uses appearance cues along with motion modeling to predict a motion field from a single input image.

intensity difference between consecutive frames \( v = I - J \), for this toy example we define \( v \) as the 2D motion field. Consider the three models shown in Figure 5.3.

**Deterministic motion prediction** In this structure, the model tries to find a deterministic relationship between the input image and object motion (Figure 5.3(b)). To do this, it attempts to find a function that maps an image to its motion field: \( v = f(I) \), by minimizing the reconstruction error \( \sum_i \| v(i) - f(I(i)) \| \) on a training set. However, because the model does not formulate motions in a probabilistic manner, it cannot capture the multiple possible motions that a shape can have. The algorithm can only learn a mean motion for each object. In the case of zero-mean, symmetric motion distributions, the algorithm would predict objects are mostly static.

**Motion prior** A simple way to model the multiple possible motions of future frames is to use a variational autoencoder [67], as shown in Figure 5.3(c). This model contains a latent representation, \( z \), which encodes the intrinsic dimensionality of the motion fields. The network that learns this intrinsic representation \( z \) consists of two parts: an encoder network that maps the motion field \( v \) to an intrinsic representation \( z \) (the gray network in Figure 5.3(c), which corresponds to \( p(z|v) \)), and a decoder network that maps the intrinsic representation \( z \) to the motion field \( v \) (the yellow network, which corresponds to \( p(v|z) \)). At the training time, the network learns the latent representation \( z \) by minimizing the reconstruction error on the training set \( \sum_i \| v(i) - g(f(v(i))) \| \). A shortcoming of this model is that it does not see the input image during inference.
Therefore, it will only learn a global motion field of both circles and squares, without distinguishing the particular motion pattern for each class of objects.

**Probabilistic frame predictor** In this work, we combine the deterministic motion prediction structure with a motion prior, to model the uncertainty in a motion field and the correlation between motion and image content. We extend the decoder in (2) to take two inputs, the intrinsic motion representation $z$ and an image $I$ (see the yellow network in Figure 5.3(d), which corresponds to $p(v|l, z)$). Therefore, instead of modeling a joint distribution of motion $v$, it will learn a conditional distribution of motion given the input image $I$.

In this toy example, since squares and circles only move in one (although different) direction, we would only need a scalar $z \in \mathbb{R}$ for encoding the velocity of the object. The model is then able to infer the location and direction of motion conditioned on the shape that appears in the input image.

### 5.3.2 Conditional Variational Autoencoder

In this section, we will formally derive the training objective of our model, following the similar derivations as those in [67, 68, 152]. Consider the following generative process that samples a future frame conditioned on an observed image, $I$. First, the algorithm samples the hidden variable $z$ from a prior distribution $p_z(z)$; in this work, we assume $p_z(z)$ is a multivariate Gaussian distribution where each dimension is i.i.d. with zero-mean and unit-variance. Then, given a the sampled hidden variable $z$, the algorithm samples the intensity difference image $v$ from the conditional distribution $p_\theta(v|I, z)$. The final image, $J = I + v$, is then returned as output.

In the training stage, the algorithm attempts to maximize the log-likelihood of the conditional marginal distribution $\sum_i \log p(v(i)|I(i))$. Assuming $I$ and $z$ are independent, the marginal distribution is expanded as $\sum_i \log \int_z p(v(i)|I(i), z)p_z(z)dz$. Directly maximizing this marginal distribution is hard, thus we instead maximize its variational upper-bound, as proposed by [67]. Each term in the marginal distribution is upper-bounded by

$$\mathcal{L}(\theta, \phi, v(i)|I(i)) \approx -D_{KL}(q_\phi(z|v(i), I(i))\|p_z(z)) + \frac{1}{L} \sum_{i=1}^{L} \left[ \log p_\theta(v(i)|z^{(i)}, I(i)) \right],$$

where $D_{KL}$ is the KL-divergence, $q_\phi(z|v(i), I(i))$ is the variational distribution that approximates the posterior $p(z|v(i), I(i))$, and $z^{(i)}$ are samples from the variational distribution. For simplicity, we refer to the conditional data distribution, $p_\theta(\cdot)$, as the *generative model*, and the variational distribution, $q_\phi(\cdot)$, as the *recognition model*.

In practice, we always choose $L = 1$. Therefore, the upper bound of the KL-divergence can be simplified as:

$$\mathcal{L}(\theta, \phi, v(i)|I(i)) \approx -D_{KL}(q_\phi(z|v(i), I(i))\|p_z(z)) + \left[ \log p_\theta(v(i)|z^{(i)}, I(i)) \right],$$

76
We assume Gaussian distributions for both the generative model and recognition model*, where the mean and variance of the distributions are functions specified by neural networks, which are:

\[
p_\theta(z^{(i)}|x^{(i)}, l^{(i)}) = \mathcal{N}(z^{(i)}; \mu_{\text{mean}}(z^{(i)}, l^{(i)}), \sigma^2 I),
\]

\[
q_\phi(z^{(i)}|x^{(i)}, l^{(i)}) = \mathcal{N}(z^{(i)}; \mu_{\text{mean}}(z^{(i)}, l^{(i)}), \sigma^2),
\]

where \(\mathcal{N}(\cdot; a, b)\) is a Gaussian distribution with mean \(a\) and variance \(b\). \(\mu_{\text{mean}}\) is a function that predicts the mean of the generative model, defined by the generative network (the yellow network in Figure 5.3(d)). \(\mu_{\text{mean}}\) and \(\sigma^2\) are functions that predict the mean and variance of the recognition model, respectively, defined by the recognition network (the gray network in Figure 5.3(d)). Here we assume that all dimensions of the generative model have the same variance \(\sigma^2\), where \(\sigma\) is a hand-tuned hyper parameter. In the next section, we will describe the details of both network structures.

**5.4 Method**

In this section we present a trainable neural network structure, which defines the generative function \(f_{\text{mean}}\) and recognition functions \(g_{\mu_{\text{mean}}}\) and \(g_{\sigma_{\text{var}}}\). Once trained, these functions can be used in conjunction with an input image to sample future frames. We first describe our newly proposed cross convolutional layer, which naturally characterizes a layered motion representation [134]. We then explain our network structure and demonstrate how we integrate the cross convolutional layer into the network for future frame synthesis.

### 5.4.1 Layered Motion Representations and Cross Convolutional Networks

Motion can often be decomposed in a layer-wise manner [134]. Intuitively, different semantic segments in an image should have different distributions over all possible motions; for example, a building is often static, but a river flows.

To model layered motion, we propose a novel cross convolutional network (Figure 5.4). The network first decomposes an input image pyramid into multiple feature maps through an image encoder (Figure 5.4(c)). It then convolves these maps with different kernels (Figure 5.4(d)), and uses the outputs to synthesize a difference image (Figure 5.4(e)). This network structure naturally fits a layered motion representation, as each feature map characterizes an image layer (note this is different from a network layer) and the corresponding kernel characterizes the motion of that layer. In other words, we model motions as convolutional kernels, which are applied to feature maps of images at multiple scales.

Unlike a traditional convolutional network, these kernels should not be identical for all inputs, as different images typically have different motions (kernels). We therefore propose a cross convolutional layer to tackle

---

* A complicated distribution can be approximated by a function of a simple distribution, e.g. Gaussian, which is referred as the reparameterization trick in [67].

† Here the bold \(I\) denotes an identity matrix, whereas the normal-font \(I\) denotes the observed image.
Figure 5.4: Our network consists of five components: (a) a motion encoder, (b) a kernel decoder, (c) an image encoder, (d) a cross convolution layer, and (e) a motion decoder. Our image encoder takes images at four scales as input. For simplicity, we only show two scales in this figure.

This problem. The cross convolutional layer does not learn the weights of the kernels itself. Instead, it takes both kernel weights and feature maps as input and performs convolution during a forward pass; for back propagation, it computes the gradients of both convolutional kernels and feature maps. Concurrent works by Finn et al. [35] and Brabandere et al. [21] also explored similar ideas. While they applied the learned kernels on input images, we jointly learn feature maps and kernels without direct supervision.

5.4.2 Network Structure

As shown in Figure 5.4, our network consists of five components: (a) a motion encoder, (b) a kernel decoder, (c) an image encoder, (d) a cross convolutional layer, and (e) a motion decoder. The recognition functions $g_{\text{mean}}$ and $g_{\text{var}}$ are defined by the motion encoder, whereas the generative function $f_{\text{mean}}$ is defined by the remaining network.

During training, our variational motion encoder (Figure 5.4(a)) takes two adjacent frames as input, both at a resolution of 128 x 128, and outputs a 3,200-dimensional mean vector and a 3,200-dimensional variance vector. The network samples the latent motion representation $z$ using these mean and variance vectors. Next, the kernel decoder (Figure 5.4(b)) sends the 3,200-dimension tensor into two additional convolutional layers, producing four sets of 32 motion kernels of size 5 x 5 ($3,200 = 128 \times 5 \times 5$). Our image encoder (Figure 5.4(c)) operates on four different scaled versions of the input image $I$ (256 x 256, 128 x 128, 64 x 64, and 32 x 32).
Figure 5.5: Results on the shapes dataset containing circles (C), squares (S), and triangles (T). For each ‘Frame 2’ we show the RGB image along with an overlay of green and magenta versions of the 2 consecutive frames, to help illustrate motion (green is the first frame and magenta is the second frame). See text and our project page for more details and a better visualization: http://visualdynamics.csail.mit.edu/.

The output sizes of the feature maps in these four channels are $32 \times 64 \times 64, 32 \times 32 \times 32, 32 \times 16 \times 16,$ and $32 \times 8 \times 8$, respectively. This multi-scale convolutional network allows us to model both global and local structures in the image, which may have different motions.

The core of our network is a cross convolutional layer (Figure 5.4(d)) which, as discussed in Section 5.4.1, applies the kernels learned by the kernel decoder to the feature maps learned by the image encoder, respectively. The output size of the cross convolutional layer is identical to that of the image encoder. Finally, our motion decoder (Figure 5.4(e)) uses the output of the cross convolutional layer to regress the output difference image.

**Training and testing details**

During training, the image encoder takes a single frame $I^{(i)}$ as input, and the motion encoder takes both $I^{(i)}$ and the difference image $d^{(i)} = j^{(i)} - I^{(i)}$ as input, where $j^{(i)}$ is the next frame. The network aims to regress the difference image that minimizes the $\ell^2$ difference to the ground truth.

During testing, the image encoder still takes a single frame $I$ as input; however, instead of using a motion encoder, we directly sample motion vectors $z^{(J)}$ from the prior distribution $p_z(z)$. In practice, we use an empirical distribution of $z$ over all training samples as an approximation to the prior $p_z(z)$, as we find it produces better synthesis results. The network synthesizes possible difference images $d^{(J)}$ by taking the sampled latent representation $z^{(J)}$ and an RGB image $I$ as input. We then generate a set of future frames $\{j^{(J)}\}$ from these difference images: $j^{(J)} = I + d^{(J)}$.

### 5.5 Evaluations

We now present a series of experiments to evaluate our method. All experimental results, along with additional visualizations, are also available on our project page http://visualdynamics.csail.mit.edu.
KL divergence ($D_{KL}(p_{gt} \parallel p_{pred})$) between predicted and ground truth distributions

<table>
<thead>
<tr>
<th>Method</th>
<th>Shapes</th>
<th>C.</th>
<th>S.</th>
<th>T.</th>
<th>C.-T.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow</td>
<td></td>
<td>6.77</td>
<td>7.07</td>
<td>6.07</td>
<td>8.42</td>
</tr>
<tr>
<td>Autoencoder</td>
<td></td>
<td>8.76</td>
<td>12.37</td>
<td>10.36</td>
<td>10.58</td>
</tr>
<tr>
<td>Ours (variational autoencoder)</td>
<td></td>
<td><strong>1.70</strong></td>
<td><strong>2.48</strong></td>
<td><strong>1.14</strong></td>
<td><strong>2.46</strong></td>
</tr>
</tbody>
</table>

Figure 5.6: Top: for each object, comparison between its ground-truth motion distribution and the distribution predicted by our method. Bottom: KL divergence between ground-truth distributions and distributions predicted by three different algorithms.

### 5.5.1 Movement of 2D Shapes

We first evaluate our method using a dataset of synthetic 2D shapes. This dataset serves to benchmark our model on objects with simple, yet nontrivial, motion distributions. It contains three types of objects: circles, squares, and triangles. Circles always move vertically, squares horizontally, and triangles diagonally (when the triangle moves up, the circle moves down). The motion of circles and squares are independent, while the motion of circles and triangles are correlated. The shapes can be heavily occluded, and their sizes, positions, and colors are chosen randomly. There are 20,000 pairs for training, and 500 for testing.

Results are shown in Figure 5.5. Figure 5.5(a) and (b) show a sample of consecutive frames in the dataset, and Figure 5.5(c) shows the reconstruction of the second frame after encoding and decoding with the ground truth images. Figure 5.5(d) and (e) show samples of the second frame; in these results the network only takes the first image as input, and the compact motion representation, $z$, is randomly sampled. Note that the
Figure 5.7: Top: Sampling results on the Sprites dataset. Motion is illustrated using the overlay described in Figure 5.5. Bottom: Probability that a synthesized result is labeled as real by humans in Mechanical Turk behavioral experiments.

To quantitatively evaluate our algorithm, we compare the displacement distributions of circles, squares, and triangles in the sampled images with their ground truth distributions. We sampled 50,000 images and used the optical flow package by Ce Liu [82] to calculate the movement of each object. We compare our algorithm with a simple baseline that copies the optical flow field of the closest image pairs from the training set ("Flow" in Figure 5.6 right); for each test image, we find its 10-nearest neighbors in the training set (the retrieval is based on $\ell^2$ distance between query image and images in the training dataset), and randomly transfer one of the corresponding optical flow fields. To illustrate the advantage of using a variational autoencoder over a standard autoencoder, we also modify our network by removing the KL-divergence loss and sampling layer ("autoencoder" in Figure 5.6 right). Figure 5.6 shows our predicted distribution is very close to the ground-truth distribution. It also shows that a variational autoencoder helps to capture the true distribution of future frames.

<table>
<thead>
<tr>
<th>Method</th>
<th>Resolution</th>
<th>Labeled real (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>32x32</td>
<td>64x64</td>
</tr>
<tr>
<td>Flow</td>
<td>29.7</td>
<td>21.0</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>41.2</strong></td>
<td><strong>35.7</strong></td>
</tr>
</tbody>
</table>
Figure 5.8: Results on Exercise dataset. Top: Sampling results on Exercise dataset. Motion is illustrated using the overlay described in Figure 5.5. Bottom: probability that a synthesized result is labeled as real by humans in Mechanical Turk behavior experiments.

### 5.5.2 Movement of Video Game Sprites

We evaluate our framework on a video game sprites dataset\(^1\), also used by Reed et al. [103]. The dataset consists of 672 unique characters, and for each character there are 5 animations (spellcast, thrust, walk, slash, shoot) from 4 different viewpoints. The length of each animation ranges from 6 to 13 frames. We collect 102,364 pairs of neighboring frames for training, and 3,140 pairs for testing. The same character does not appear in both the training and test sets. Synthesized sample frames are shown in Figure 5.7. The results show that from a single input frame, our method can capture various possible motions that are consistent with those in the training set.

For a quantitative evaluation, we conduct behavioral experiments on Amazon Mechanical Turk. We randomly select 200 images, sample possible next frames using our algorithm, and show them to multiple human subjects as an animation side by side with the ground truth animation. We then ask the subject to choose which animation is real (not synthesized). An ideal algorithm should achieve a success rate of 50%. In our experiments, we present the animation in both the original resolution (64 x 64) and a lower

\(^1\)Liberated pixel cup: [http://tpc.opengameart.org](http://tpc.opengameart.org)
### Figure 5.9: Visual analogy-making (predicted frames are marked in red)

<table>
<thead>
<tr>
<th>Model</th>
<th>spellcast</th>
<th>thrust</th>
<th>walk</th>
<th>slash</th>
<th>shoot</th>
<th>Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add</td>
<td>41.0</td>
<td>53.8</td>
<td>55.7</td>
<td>52.1</td>
<td>77.6</td>
<td>56.0</td>
</tr>
<tr>
<td>Dis</td>
<td>40.8</td>
<td>55.8</td>
<td>52.6</td>
<td>53.5</td>
<td>79.8</td>
<td>56.5</td>
</tr>
<tr>
<td>Dis+Cls</td>
<td>13.3</td>
<td>24.6</td>
<td>17.2</td>
<td>18.9</td>
<td>40.8</td>
<td>23.0</td>
</tr>
<tr>
<td>Ours</td>
<td>9.5</td>
<td>11.5</td>
<td>11.1</td>
<td>28.2</td>
<td>19.0</td>
<td>15.9</td>
</tr>
</tbody>
</table>

Figure 5.10: Mean squared pixel error on test analogies, by animation. The first three models (Add, Dis, and Dis+Cls) are from Reed et al. [103].

resolution (32 × 32). We only evaluate on subjects that have a past approval rating of > 95% and also pass our qualification tests. Figure 5.7 shows that our algorithm significantly out-performs a baseline algorithm that warps an input image by transferring a randomly selected flow field from the training set. Subjects are less likely fooled by the 64 × 64 pixel images, as it is harder to hallucinate realistic details in high-resolution images.

### 5.5.3 Movement in Real Videos Captured in the Wild

To demonstrate that our algorithm can also handle real videos, we collect 20 workout videos from YouTube, each about 30 to 60 minutes long. We first apply motion stabilization to the training data as a pre-processing step to remove camera motion. We then extract 56,838 pairs of frames for training and 6,243 pairs for testing. The training and testing pairs come from different video sequences. Figure 5.8 shows that our framework works well in predicting the movement of the legs and torso. Additionally, Mechanical Turk behavioral experiments show that the synthesized frames are visually realistic.
5.5.4 Zero-Shot Visual Analogy-Making

Recently, Reed et al. [103] studied the problem of inferring the relationship between a pair of reference images and synthesizing a new analogy-image by applying the inferred relationship to a test image. For example, the character shown in the top row Figure 5.9(a) leans toward the right and apply such motion to
the target image (the left bottom image of Figure 5.9(a)) to generate the same motion (the right bottom image of Figure 5.9(a)). The method by Reed et al. requires a set of quadruples as supervision (two source images and two target images). Our network is also able to perform this task, without even requiring quadruple as supervision. Specifically, we extract the motion vector, $z$, from two reference frames using our motion encoder (Figure 5.4(a)). We then use the extracted motion vector $z$ to synthesize an analogy-image given a new test image. In this way, our network learns to transfer motion from a source pair to the target image, without even supervision on whether two pairs share the same motion or not.

Our network successfully transfers the motion in reference pairs to a test image. For example, in Figure 5.9(a), it learns that the character leans toward to the right, and in Figure 5.9(b) it learns that the girl spreads her feet apart. A quantitative evaluation is also shown in Table 5.10. Even without supervision, our method out-performs the algorithm by [103], which requires visual analogy labels during training.

### 5.5.5 Visualizing Feature Maps

We visualize the learned feature maps (see the output of image encoder, in Figure 5.4(c)) in Figure 5.11. Even without supervision, our network learns to detect objects or contours in the image. For example, we see that the network automatically learns object detectors and edge detectors on the shape dataset. It also learns a hair detector and a body detector on the sprites and exercise datasets, respectively.

### 5.5.6 Dimension of Latent Representation $z$

Although our latent motion representation, $z$, has 3,200 dimensions, its intrinsic dimensionality is much smaller. First, $z_{\text{mean}}$ is very sparse. The non-zero elements of $z_{\text{mean}}$ for each dataset are 299 in shapes, 54 in sprites, and 978 in exercise. Second, the independent components of $z$ are even fewer. We run principle component analysis (PCA) on the $z_{\text{mean}}$s obtained from a set of training images, and find that for each dataset, a small fraction of components cover at least 95% of the variance in $z_{\text{mean}}$ (5 dimensions in shapes, 2 dimensions in sprites, and 27 dimensions in exercise). This indicates that our network has learned a compact representation of motion in an unsupervised fashion, and encodes high-level knowledge using a small number of bits, rather than simply memorizing training samples. The KL-divergence criterion in Eq. 5.2 forces the latent representation, $z$, to carry minimal information, as discussed by [56] and concurrently by [55].
In the previous chapters, we show how observed 2D motion in captured sequences helps to infer different properties of our visual world. Below, we will summarize the contribution of our work, its relation to previous literature, and potential future directions.

**Decoupling Layers in Videos Captured by Moving Cameras (Chapter 2).** In that chapter, we have shown how to separate a obstructing foreground layer from a background layer in a captured scene using motion parallax. Although there are many works on layer decomposition using motion parallax [44, 51, 79, 114], finding a good initialization of motion field is still an unsolved problem. The main challenge is that the obstructing foreground layer overlays the background layer, and traditional flow estimation algorithm cannot estimate the motion of a scene with multiple overlapping layers. To solve this problem, we propose a novel edge flow algorithm that simultaneously tracks both the foreground and background layers and obtains a roughly correct initial estimation of both motion fields. Our experiment shows that our edge flow significantly outperforms other initialization methods, e.g. initialization by a random flow or the optical flow (see Figure 2.4).

We also show that dense nonparametric motion fields more precisely models the movement of each layer compared with parametric motion fields, such as affine motion, and reduces the artifacts in decomposition, especially when the depth variation of either layers is not negligible.

Motion based layer decomposition also relates to other problems in computer vision. One set of closest problems is structure from motion (SfM) [61] and multi-view stereo (MVS) [132], which is to estimate the full 3D geometry of a scene, either sparse (SfM) or dense (MVS), from a set of images captured by a moving camera. Unlike SfM or MVS, our algorithm does not estimate the full 3D geometry of a scene, but solves a comparatively simpler problem, which is to compose a scene to two layers at different depth. By doing so, we save most of computational power on computing exact 3D geometries and camera movement. The disadvantage of our method is that it only works when two layers are far from each other in order to separate them in the motion fields. Luckily, in many situations, the obstructing layer is far away from the background layer, and we can separate them by motion.

Another closest problem is to segment a scene from its movement [122, 123], also know as layer-motion
representation. In this problem, a common assumption is that a scene consists of several opaque chunks (segments or objects), each of which has consistent motion inside. However, in our case, foreground objects are either transparent (reflection) or have a lot of holes (fence), which breaks the assumption of the traditional layered models. Therefore, we use an alternative model that assumes a scene consists of two overlapping layers.

Despite all the differences, there are still a lot of relation between these problems. In both structure from motion and layer-motion representation, one challenge is to deal with the reflection in a scene, like reflection from a lake or a glass, and using the reflection removal algorithm proposed in our work might help to resolve those problems.

Measuring Fluid Motion from Videos (Chapter 3). In that chapter, we have shown how to estimate velocity and depth of fluid from the tiny distortions caused by refraction. There are many works discussed about how to detect fluid and visualize them using observed distortion [53, 89], but most of these works do not discuss how the observed distortion relates to the actual movement of fluid, and how to recover the velocity and depth of fluid from the distortion. In this work, we find a simple but power principle that helps us to locate and track fluid: observed distortions related to movements of refractive fluid elements, as observed by one or more video cameras, are consistent over small space-time volumes. Based on this observation, to recover the depth of fluid, we can stereo fuse the distortion fields estimated from two video streams, similar to the way that we stereo fuse the intensity values in the standard stereo matching. We can also track the observed distortion over time to recover the velocity of fluid.

There are a few algorithms in the literature that can also recover the 3D geometry of air flow from visual distortion [6, 8]. These methods recover a full 3D density field of air flow and can also handle multiple-layer fluid, with the cost that the system is normally complicated and a lot of cameras are required. For example, the algorithm by Atcheson et al. [8] needs a camera array that covers 180° of the air flow of the interest. This is because that there are many unknown variables when solving the full 3D density field, and there is not enough constrains to recover those variables if there are only two input video streams. On the contrary, our approach only requires either a single or stereo cameras, which is a much simpler and portable setup than a camera array, as we use additional assumptions to air flow. One assumption is the air flow must consist of only one thin layer. If there are more than one layers, the refractive constancy that our algorithm relies on does not hold any more. Second, to recover the velocity of air flow, the frame rate of the camera need to high enough (faster air flow requires higher frame rate), otherwise the air flow might move a lot between two neighboring frames and it is hard to track them. Besides, our fluid stereo algorithm only recovers the depth of air flow, not the full 3D density field.

Estimating the Structure of Objects from the Spectrum of Their Vibration (Chapter 4). In that chapter, we have shown how to estimate the structure of objects from their vibration. Our main contribution is to demonstrate that the spectrum of vibration can help to resolve the ambiguity in structure estimation, in
addition to appearance cues.

The relationship between the physical properties of objects and the spectrum of its vibration has widely studied in sound simulation [40] and physical property analysis [25, 29], but our work is the first attempt to estimate the structure of a complicated object using spectrum analysis. The main challenging prevents most of previous works to estimate the structure of an object is that the vibration mode of an object relates to many different factors, including density, stiffness, and the structure of an object, and the magnitude and the direction of external forces applied to it. Inferring all these factors from the vibration of an object is very challenging if there is no strong prior on these factors.

To alleviate this problem, we focus on tree-structured objects in this work, and found a simple principle that makes the structure analysis possible: the frequency response of a branch are independent to other branches when the vibration is small (see Proposition 4.3.1). This makes it possible to estimate the structure of objects in an efficient greedy manner, that is to estimate the structure each sub-branch independently, which greatly reduces the search space of the problem. This is the first attempt in this direction, and in the future, it is also interesting to explore what other priors can make the structure estimation from vibration solvable.

**Synthesizing the Movement of Objects in the Future (Chapter 5).** In that chapter, we have shown that how to sample future frames from a single input image. Most of previous work either directly synthesizes the future frame from an input image, or synthesizes the motion field between them, while we propose a segment-based synthesis algorithm. The proposed algorithm has two advantages. First, the network does not directly synthesize the output image, which is normally blurry and in a low-resolution, but transforms pixels in the input frame in the way that keeps details in the input. Second, the network only samples the movement of each segment, instead of a full-resolution motion field. In this way, the network can easily model and sample the movement of objects, as the movement of segments has much lower dimensionality than a full-resolution motion field.

One limitation of current sampling framework is that it does not explicitly handle occlusion, so it can only synthesize the movement of an object with a simple clean background. When the background is textured, the algorithm might fail, as the foreground objects will occlude different parts of the background when it moves, and our model cannot handle that. It would be interesting to extend it to model a scene with multiple layers occluding each other.
Appendix A

Proof of Refractive Constancy

Recall that in the paper, we define the refractive wiggle as:

**Define (Refraction wiggle).** Let \( x' \) be a point on the refractive fluid layer, \( x'' \) be the intersection between the background and the light ray passing through \( x' \) and the center of projection at time \( t \), and \( \Delta t \) be a short time interval. Then the wiggle of \( x' \) from time \( t \) to \( t + \Delta t \) is the shift of the projection of \( x'' \) on the image plane during this time.

Then, let us prove the following lemma:

**Lemma (Refractive wiggle).** Let \( z, z', \) and \( z'' \) be the focal length of the camera, the depth of fluid, and the depth of background. Assuming that the fluid object is moving parallel to the camera plane, then the wiggle feature equals to:

\[
\nu = -\frac{(z'' - z')z}{z''} \sec^2 \alpha_t \beta_t (\alpha_{t+\Delta t} - \alpha_t) \tag{A.1}
\]

**Proof.** Because wiggles are defined by shifts in the image plane, we first trace rays to determine which points in the image plane correspond to \( x'_t \) and \( x'_{t+\Delta t} \) on the fluid object. At time \( t \), an undistorted ray is emitted from the center of the projection \( o \) to point \( x'_t \) with angle \( \beta_t \), with respect to vertical (solid lines in Fig. A.1). This ray is bent by \( \beta_t \) due to the refraction when it passes through the fluid and finally hits the image plane at \( x''_t \).

At successive time \( t + \Delta t \), the fluid object moves to a new location (dashed gray blob in Fig. A.1). The background point \( x''_t \) now correspond to a different point \( x'_{t+\Delta t} \) on the fluid object and a different point \( x_{t+\Delta t} \) on the image point. The ray from the background to the fluid object is now bent by \( \alpha_{t+\Delta t} \) when passing through the fluid, and finally hits the image plane with angle of \( \beta_{t+\Delta t} \) with respect to the vertical. The wiggle at time \( t \) is the distance between \( x_t \) and \( x_{t+\Delta t} \), denoted as \( x_t x_{t+\Delta t} \) (blue or red arrows in Fig. A.1). A simple geometric calculation shows:

\[
x_{t+\Delta t} - x_t = z(\tan \beta_{t+\Delta t} - \tan \beta_t) \tag{A.2}
\]

\[
x'_t - o = z' \tan \beta_{t+\Delta t} + (z'' - z') \tan(\alpha_{t+\Delta t} + \beta_{t+\Delta t})
\]

\[
= z' \tan \beta_t + (z'' - z') \tan(\alpha_t + \beta_t) \tag{A.3}
\]
From Equation (A.3), we have:

\[
0 = z' (\tan \beta_{t+\Delta t} - \tan \beta_t) + (z'' - z') \left( \tan(\alpha_{t+\Delta t} + \beta_{t+\Delta t}) - \tan(\alpha_t + \beta_t) \right)
\]

\[
\approx \sec^2 \beta_t (\beta_{t+\Delta t} - \beta_t) z' + (z'' - z') \sec^2 (\beta_t + \alpha_t)(\beta_{t+\Delta t} - \beta_t + \alpha_{t+\Delta t} - \alpha_t)
\]

\[
= \sec^2 \beta_t (\beta_{t+\Delta t} - \beta_t) z' + (z'' - z') \sec^2 \beta_t (\beta_{t+\Delta t} - \beta_t + \alpha_{t+\Delta t} - \alpha_t),
\]

where the first line to the second line is based on the Taylor expansion, and second line to the third line is based on the assumption that \(\alpha\) is small, because the index of refraction is close to 1. Then from Equation (A.4), we have:

\[
\beta_{t+\Delta t} - \beta_t = -\frac{z'' - z'}{z''}(\alpha_{t+\Delta t} - \alpha_t).
\]

Finally, combining Equation (A.2) and Equation (A.5), we have:

\[
x_{t+\Delta t} - x_t = z (\tan \beta_{t+\Delta t} - \tan \beta_t)
\]

\[
= z (\beta_{t+\Delta t} - \beta_t) \sec^2 \beta_t
\]

\[
= -\frac{(z'' - z')\alpha_{t+\Delta t} - \alpha_t}{z''}
\]

This completes the proof.

Based on this lemma, we can prove the refractive flow constancy and the refractive stereo constancy.

**Theorem (Refractive flow constancy).** Suppose the fluid object does not change its shape and index of refraction during a short time interval \([t_1, t_2]\). Then for any point on the fluid object, its wiggle \(v(t_1)\) at \(t_1\) equals its wiggle \(v(t_2)\) at \(t_2\).
Proof. As shown in Fig. A.1, let \( o, x_t, x'_t, x''_t \) be the light ray that hits the background \( x''_t \) at time \( t \), and \( o, x_{t+\Delta t}, x'_{t+\Delta t}, x''_{t+\Delta t} \) be the light ray that hits the same background point \( x''_t \) at time \( t + \Delta t \), \( i = 1, 2 \). According to the definition of the refractive wiggle:

\[
v(t_1) = \overrightarrow{x_{t_1}x_{t_1+\Delta t}}, \quad v(t_2) = \overrightarrow{x_{t_2}x_{t_2+\Delta t}}
\]

To prove the refractive flow constancy, first we will show that \( x'_{t_1+\Delta t} \) and \( x'_{t_2+\Delta t} \) are the same point on the fluid object, or equivalently, the shifts \( x'_{t_1+\Delta t} \) and \( x'_{t_2+\Delta t} \) are equal. By the definition of wiggle, we only know that \( x'_{t_1+\Delta t} \) and \( x'_{t_2+\Delta t} \) are equal.

Based on Lemma A, we know that:

\[
\frac{x_{t_1}x_{t_1+\Delta t}}{x_{t_2}x_{t_2+\Delta t}} = -\frac{(z'' - z')z \sec^2 \beta}{z'}(\alpha_{t_1+\Delta t} - \alpha_{t_1})
\]

(A.8)

By the similar triangle formula, we have:

\[
\frac{x'_{t_1+\Delta t}}{x'_{t_2+\Delta t}} = -\frac{(z'' - z')z' \sec^2 \beta}{z'}(\alpha_{t_2+\Delta t} - \alpha_{t_1})
\]

(A.9)

\[
\frac{x''_{t_1+\Delta t}}{x''_{t_2+\Delta t}} = -\frac{(z'' - z')z' \sec^2 \beta}{z'}(\alpha_{t_2+\Delta t} - \alpha_{t_1})
\]

(A.10)

As discussed in the paper, for a short duration, the refractive angle of a same point on the fluid object will remain constant, so \( \alpha_{t_1} = \alpha_{t_2} \). Subtracting Eq. A.9 by Eq. A.10, we have:

\[
\frac{x'_{t_1+\Delta t}}{x'_{t_2+\Delta t}} - \frac{x''_{t_1+\Delta t}}{x''_{t_2+\Delta t}} = -\frac{(z'' - z')z' \sec^2 \beta}{z'}(\alpha_{t_1+\Delta t} - \alpha_{t_2+\Delta t})
\]

(A.11)

Then we can solve \( x_{t_2+\Delta t} \) from Equation (A.11). If \( x'_{t_1+\Delta t} \) and \( x'_{t_2+\Delta t} \) are also the same point on the fluid object, then LHS of Equation (A.11) is zero as \( x'_{t_1+\Delta t} = x_{t_2+\Delta t} - x'_t \), and the RHS of Equation (A.11) is also zero because the refractive angle of the same point on the fluid object are the same, that is \( \alpha_{t_1+\Delta t} = \alpha_{t_2+\Delta t} \).

This shows \( x'_{t_1+\Delta t} = x'_{t_2+\Delta t} \) is one solution to Equation (A.11). Assuming that there is only one light ray that emits from \( x''_t \) and hits at the center of projection, then \( x'_{t_1+\Delta t} = x'_{t_2+\Delta t} \) is also the only solution to Equation (A.11). This proves \( x'_{t_1+\Delta t} = x'_{t_2+\Delta t} \) and consequently \( \alpha_{t_1+\Delta t} = \alpha_{t_2+\Delta t} \).

Finally, we prove that wiggles at \( t_1 \) and \( t_2 \) are equal. Plugging

\[
\alpha_{t_1+\Delta t} - \alpha_{t_1} = \alpha_{t_2+\Delta t} - \alpha_{t_2}
\]

(A.12)

to Equation (A.8), we have:

\[
\frac{x_{t_1}x_{t_1+\Delta t}}{x_{t_2}x_{t_2+\Delta t}} = -\frac{(z'' - z')z \sec^2 \beta}{z'}(\alpha_{t_1+\Delta t} - \alpha_{t_1})
\]

\[
= -\frac{(z'' - z')z' \sec^2 \beta}{z'}(\alpha_{t_1+\Delta t} - \alpha_{t_1}) = \frac{x_{t_1}x_{t_1+\Delta t}}{x_{t_2}x_{t_2+\Delta t}}
\]

(A.13)
**Theorem** (Refractive stereo constancy). Suppose there are \( n \geq 2 \) cameras imaging a refractive fluid object, and they are all parallel and close to each other. Then at any time \( t \), and for any point on the fluid object, the corresponding wiggle in all the cameras is the same.

**Proof.** Similar to the previous proof, let \( x'_j \) be a point on the fluid object. Tracing rays, at time \( t \), an undistorted ray is emitted from the center of the projection \( o_j (j = 1, 2) \) to the point point \( x'_j \). It is refracted by the fluid object and hits the background at \( x''_j \). At a successive time \( t + \Delta t \), the fluid object moves, and the light ray from the points on the background to the center of the projection now goes through points \( x_{t+\Delta t,j} \) on the fluid object and \( x_{t+\Delta t,j} \) on the image plane. We will show that the wiggle features \( x'_{t+\Delta t,j} \) and \( x'_{t+\Delta t,j} \) are equal in both views.

Like in the previous proof, we first will show that \( x'_{t+\Delta t,1} = x'_{t+\Delta t,2} \). Following a similar derivation as in the previous proof, we have:

\[
x'_{t+\Delta t,1} - x'_t = -\frac{z''(z'' - z')}{z'} (\Delta \alpha_{t+\Delta t}(x'_{t+\Delta t,1}) - \Delta \alpha_t(x'_t)). \tag{A.14}
\]

Then, subtracting Equation (A.14) with \( j = 2 \) from Equation (A.14) with \( j = 1 \), we have:

\[
x'_{t+\Delta t,1} - x'_{t+\Delta t,2} = -\frac{z''(z'' - z')}{z'} (\Delta \alpha_{t+\Delta t}(x'_{t+\Delta t,1}) - \Delta \alpha_{t+\Delta t}(x'_{t+\Delta t,2})). \tag{A.15}
\]

By the same logic as in the previous proof, when \( x'_{t+\Delta t,1} = x'_{t+\Delta t,2} \), both the LHS and RHS of Equation (A.15) are equal to 0. Therefore, \( x'_{t+\Delta t,1} = x'_{t+\Delta t,2} \) is the solution to Equation (A.15). Thus, \( x'_{t+\Delta t,1} \) and \( x'_{t+\Delta t,2} \) are the same point.

Finally, we prove that wiggles from two views are equal:

\[
\frac{z}{z'} x'_{t+\Delta t,1} x'_t \xrightarrow{\rightarrow} \frac{z}{z'} x'_{t+\Delta t,2} x'_t = \frac{z}{z'} x'_{t+\Delta t,2} x'_{t+\Delta t,2}. \tag{A.16}
\]
This completes the proof. ■
Appendix B

Calculating Fluid Flow Efficiently

Recall that in Section 3.2.3, the probabilistic refractive flow algorithm consists of two steps. First, we solve for the mean $\mu$ and the variance $\Sigma_u$ of the wiggle features $v$ from the following Gaussian distribution:

$$P(v|l) = \exp \left( -\sum_x \alpha_1 \left\| \frac{\partial l}{\partial x} u_x + \frac{\partial l}{\partial y} u_y + \frac{\partial l}{\partial t} \right\|_2^2 + \alpha_2 \left\| \frac{\partial v}{\partial x} \right\|_2^2 + \alpha_2 \left\| \frac{\partial v}{\partial y} \right\|_2^2 \right). \tag{B.1}$$

To solve for the mean and variance of flow from Equation (B.1), let the $V$ be the vector formed by concatenating all the optical flow vectors in one frame. That is, $V = (\cdots, v(x), \cdots)$. Also, let us represent Equation (B.1) in information form $P(v|l) = \exp(-\frac{1}{2} v^T J v + h^T v)$, where $h$ and $J$ can be calculated from (B.1). Then the mean of $V$ is $\bar{V} = J^{-1} h$ and covariance of $V$ is $\Sigma = J^{-1}$.

In the second step, the fluid flow is calculated by minimizing the following optimization problem based on the mean and variance of the wiggle features computed in the first step.

$$\hat{u} = \arg \min_u \sum_x \beta_1 \left\| \frac{\partial \bar{v}}{\partial x} u_x + \frac{\partial \bar{v}}{\partial y} u_y + \frac{\partial \bar{v}}{\partial t} \right\|_2^2 + \beta_2 \left( \left\| \frac{\partial u}{\partial x} \right\|_2^2 + \left\| \frac{\partial u}{\partial y} \right\|_2^2 \right) + \beta_3 \|u\|^2. \tag{B.2}$$

Calculating the covariance of each wiggle feature from Equation (B.1) requires inverting the information matrix $J$. This step will be slow if the matrix is large. To avoid this time-consuming inversion, we make a slight change to the fluid flow objective function. Let $V_x$, $V_y$, and $V_t$ be the vectors formed by concatenating all the partial derivatives of mean wiggle features in a frame, that is $V_x = (\cdots, \frac{\partial v}{\partial x}(x), \cdots)$, $V_y = (\cdots, \frac{\partial v}{\partial y}(x), \cdots)$, and $V_t = (\cdots, \frac{\partial v}{\partial t}(x), \cdots)$. Similarly, let $U_x$, $U_y$ be the vectors formed by concatenating all the x-components and y-components of $u$ in a frame respectively. Then we can calculate the refractive flow as follows:

$$\min_u f_1 (V_x \cdot U_x + V_y \cdot U_y + V_t)^T J (V_x \cdot U_x + V_y \cdot U_y + V_t)$$

$$+ \beta_2 (||D_x U_x||^2 + ||D_y U_y||^2 + ||D_x U_x||^2 + ||D_y U_y||^2) + \beta_3 ||U||^2 \tag{B.3}$$

where $D_x$ and $D_y$ are the partial derivative matrices to $x$ and $y$ respectively. The smoothness term of (B.3) is exactly the same as that in (B.2), and the data term of (B.2) is

$$(V_x \cdot U_x + V_y \cdot U_y + V_t)^T J (V_x \cdot U_x + V_y \cdot U_y + V_t)$$

$$= ||V_x \cdot U_x + V_y \cdot U_y + V_t||_J = ||V_x \cdot U_x + V_y \cdot U_y + V_t||_\Sigma. \tag{B.4}$$
which is also similar to the data term in Equation (B.2) except that it jointly considers all the wiggle vectors in a frame. Therefore, this change will not affect the result too much, but the algorithm is more computationally efficient as we never need to compute $J^{-1}$. The term never appears in Equation (B.3).
Appendix C

Proof of Refractive Aperture

C.1 Proof of the Claim in Section 4.1 (observing 1D structure)

In Section 4.1 of the main paper, we claim that when background is a straight edge and refractive field are linear within the aperture, we cannot recover the refractive motion \( u \) from the observed sequence \( g(x, t) \). We will give a formal proof of this claim in this section.

Consider a small space-time window around \( x = x_0 \) and \( t = t_0 \). Without loss of generality, we suppose \( t_0 = 0 \). Recall that within this window (aperture), the background image \( f(x) \) is assumed to be approximately equal to,

\[
f(x) = \begin{cases} 
1 & n^\top x + c > 0, \\
0 & n^\top x + c \leq 0,
\end{cases}
\]  
(C.1)

and the refractive field \( r_0(x) \) approximately equal to

\[
r_0(x) = r(x_0) + J(x_0)^\top \Delta x,
\]  
(C.2)

where \( x = x_0 + \Delta x \). Here, \( n \in \mathbb{R}^2 \) and \( c \in \mathbb{R} \) are the normal and intercept of the background edge, \( J \in \mathbb{R}^{2 \times 2} \) is the linear coefficient (Jacobian matrix) of the refractive field, and \( r(x_0) \in \mathbb{R}^2 \) is the bias term of the refractive field. Then according to the image formation equation (Eq. 5 in the main paper), we get:

\[
g(x, t) = \begin{cases} 
1 & n^\top \Delta x - n^\top J(x_0) \Delta x + n^\top J(x_0) u \Delta t + n^\top x_0 + c - n^\top r(x_0) < 0 \\
0 & \text{otherwise}
\end{cases}
\]  
(C.3)

For the ease of description, we abuse notations \( f = f(x_0) \) and \( r = r(x_0) \), then the boundary of the observed image at time \( t \) is

\[
n^\top (I - J) \Delta x + n^\top J u t + c + n^\top x_0 - n^\top r = 0,
\]  
(C.4)

*Note that from Eq. C.3 to Eq. C.4, we ignore the information that which side of boundary is black. This ignorance will not lose any information, because we assume that background pattern is unknown and we can always switch the black and white region in observed sequence by switching the black and white region in the background.
where $I$ is the $2 \times 2$ identity matrix.

Since we know $g(x, t)$ is a linear function with respect to $x$ and $t$, so the only information we get from $g(x, t)$ are those linear coefficients. Let us summarize the observation and the unknown in our problem:

- **Observation:** The linear coefficients in Eq. C.4.

- **Unknown:** $n, c$ (coefficients of background), $r, J$ (coefficients of refractive field), and $u$ (refractive motion).

Since there are 3 independent observed quantities $^\dagger$, but 11 unknowns $^\ddagger$, intuitively it is impossible to recover the refractive motion. To rigorously prove it, we state the problem formally as follows.

**Claim:** Let $\overline{n}$ and $\overline{c}$ be the coefficients of the ground truth background, $\overline{r}$ and $\overline{J}$ be the coefficients of the ground truth refractive field, and $\overline{u}$ be the ground truth motion of the refractive object. Let $g(x, t)$ be the sequence observed under this setup. Then for any non-zero motion vector $u \in \mathbb{R}^2$, there exists a linear background pattern and a refractive field such that the same sequence $g(x, t)$ will be observed in this setup. More specifically, given any $\overline{n}, \overline{c}, \overline{J}, \overline{r}$, and $u$, $\forall u^* \neq 0 \in \mathbb{R}^2$, $\exists n^*, c^*, r^*, J^*, s.t., \forall t$, following two equations describe the same line in $\mathbb{R}^2$:

\[
\begin{align*}
\overline{n}^\top (I - \overline{J})Ax + \overline{n}^\top \overline{J}u + \overline{c} + \overline{n}^\top x_0 - \overline{n}^\top \overline{r} &= 0, \\
n^* (I - J^*)Ax + n^* J^* u^* t + c^* + n^* x_0 - n^* r^* &= 0.
\end{align*}
\]

**Proof.** Because $u^* \neq 0$, then there exists a $2 \times 2$ non-singular matrix $U$ such that

\[
U u^* = \overline{u}.
\]

Then we define a new background and refractive field as:

\[
\begin{cases}
J^* = A^{-1} \overline{J} U, \\
n^* = A^\top \overline{n},
\end{cases}
\]

where $A = \overline{J} U + I - \overline{J}$.

---

$^\dagger$ $g(x, t)$ has 2 linear coefficients for $x$, 1 linear coefficients for $t$, and 1 bias term. However, there is a 1 dimensional redundancy among these terms, because by multiplying arbitrary scalar to both sides of Eq. C.4, we still get the same linear equation. Thus, there are only 3 independent parameters.

$^\ddagger$ Because $n \in \mathbb{R}^2, c \in \mathbb{R}, r \in \mathbb{R}^2, J \in \mathbb{R}^{2 \times 2}$, and $u \in \mathbb{R}^2$, so that there are 10 unknowns in total.
Then the left hand side of Eq. C.6 becomes:

\[
\text{LHS of Eq. C.6} = n^* (I - J^t) \Delta x + n^* J^t u^t + c^* + n^* x_0 - n^* r^*
\]

\[
= \bar{n}^t A(I - A^{-1} J) \Delta x + \bar{n}^t A A^{-1} J U U^{-1} u t + (\bar{c} - n^* x_0 + n^* r + \bar{n}^t x_0 - \bar{n}^t \bar{r}) + n^* x_0 - n^* r
\]

\[
= \bar{n}^t (A - J U) \Delta x + \bar{n}^t J U u t + \bar{c} + \bar{n}^t x_0 - \bar{n}^t \bar{r}
\]

which is exactly the same as the left hand side of Eq. C.5. Therefore, Eq. C.5 and Eq. C.6 are equal. This completes the proof.

\[\Box\]

\section*{C.2 Proof of Proposition 1 and 2 in Section 4.2 (observing 2D structure)}

Before proving the proposition 1 and 2 in Section 4.2, let us first review the image formation model. In the main paper, we approximate the refractive field within the aperture by the second order Taylor expansion as follows:

\[
r_0(x) = r_0(x_0) + J(x_0)^t \Delta x + \frac{1}{2} \left( \frac{\Delta x^t H_x(x_0) \Delta x}{\Delta x^t H_y(x_0) \Delta x} \right) + O(||\Delta x||^3),
\]

where \( J \in \mathbb{R}^{2 \times 2} \) is the Jacobian matrix of the refractive field \( r \) and \( H_x, H_y \in \mathbb{R}^{2 \times 2} \) are Hessian matrices of \( x \) and \( y \) components of the refractive field. The background is still a single edge defined in Eq. C.1. Note that for rigor, we include all the high order term in the term \( O(||\Delta x||^3) \). That is why in the above equation exact equality is used.

According to the image formation equation (Eq. 5 in the main paper), the observed image under this setup is:

\[
g(x, t) = \begin{cases} 
1 & n^t (\Delta x + x_0 - r_0(x_0) - J(x_0)^t \Delta x) - \frac{1}{2} \left( \frac{\Delta x - ut)^t H_x(x_0)(\Delta x - ut)}{\Delta x^t H_y(x_0) \Delta x} \right) + O(||\Delta x - ut||^3) + c < 0 \\
0 & \text{otherwise}
\end{cases}
\]

\[\text{(C.11)}\]

So the boundary of the observed sequence at time \( t \) is

\[
1 - \frac{1}{2} \Delta x^t (H + O(t)) \Delta x + (-J^t n + n + H u t + O(t^2))^t \Delta x - \frac{1}{2} u^t H u t^2 + n^t J u t + d + O(t^3) + O(||\Delta x||^3) = 0
\]

\[\text{(C.12)}\]

where

\[
H = n_x H_x(x_0) + n_y H_y(x_0),
\]

\[\text{(C.13)}\]

\[
d = n^t x_0 - n^t r(x_0) + c
\]

\[\text{(C.14)}\]
where \( n = (n_x, n_y) \). Also, with the abuse the notation, let \( J = f(x_0) \). Note that \( H \) is symmetric matrix, because \( H_x \) and \( H_y \) are symmetric matrices.

From Eq. C.12, the boundary of the observed sequence at each time \( t \) is a conic curve. By fitting the equation \( x^\top Q(t)x + q(t)^\top x + 1 = 0 \) to the observed boundary, we can recover the coefficient \( Q(t) \in \mathbb{R}^{2 \times 2} \) and \( q(t) \in \mathbb{R}^2 \) of this conic curve \( (Q \) is a symmetric matrix). From Eq. C.12, we have:

\[
\begin{align*}
Q(t) &= \frac{H + O(t)}{\frac{1}{2} u^\top Hu + n^\top f u + d + O(t^2)}, \\
q(t) &= \frac{\bar{J}^\top n + n + Hu + O(t^2)}{\frac{1}{2} u^\top Hu + n^\top f u + d + O(t^2)}.
\end{align*}
\]

Although we can recover coefficients \( Q(t) \) and \( q(t) \) in all neighboring frames around \( t = 0 \), the only information we can get are the following three statistics:

\[
\begin{align*}
Q(0) &= d^{-1} H, \\
q(0) &= d^{-1}(\bar{J}^\top n + n), \\
\frac{\partial q(0)}{\partial t} &= d^{-1} Hu - d^{-2}(n^\top Ju)(\bar{J}^\top n + n).
\end{align*}
\]

Other higher order derivatives of the \( Q(t) \) and \( q(t) \) will contain the unknown term \( O(\cdot) \) and cannot be used. For example, \( \frac{\partial Q(0)}{\partial t} = O(\{n^\top x_0 - n^\top r\} - n^\top fJuH) \) contains the unknown term \( O(1) \). Since \( O(1) \) here can be an arbitrary term, it does not reveal any information about the unknowns \( (n, d, \bar{J}, H, u) \). Therefore, the only information we get from the observed sequence are the values \( Q(0), q(0) \), and \( \frac{\partial q(0)}{\partial t} \).

Besides, Eq. C.17–Eq C.19 is based on the assumption that the spatial temporal window is centered at \( t = 0 \). If the spatial temporal window is centered at arbitrary temporal location \( t \), we have:

\[
\begin{align*}
Q(t) &= d^{-1} H, \\
q(t) &= d^{-1}(\bar{J}^\top n + n), \\
\frac{\partial q(t)}{\partial t} &= d^{-1} Hu - d^{-2}(n^\top Ju)(\bar{J}^\top n + n).
\end{align*}
\]

Therefore, let us again summarize the observation and unknown:

- **Observation:** \( Q(t), q(t), \frac{\partial q(t)}{\partial t} \).
- **Unknown:** \( H, \bar{J}, n, d, u \).

Compared with the setup where only a single edge is observed (Section 4.1), now we have more observations, and we might be able to recover some information about refractive motion. This is stated the proposition 1 in the main paper (repeat it as follows).

\footnote{The Hessian matrix of a real function is always symmetric.}

\footnote{Here we abuse the notation \( t \) and Eq. C.20 should not be confused with Eq. C.15. The former holds for \( t \) around \( t = 0 \), but the latter is true for any \( t \).}
Proposition. Given the observed sequence $g(x, t)$. At each time point $t$, we can fit a conic curve $x^\top Q(t)x + q(t)^\top x + 1 = 0$ to the observed boundary shape, where $Q(t) \in \mathbb{R}^{2 \times 2}$ and $q(t) \in \mathbb{R}^2$ are the coefficients of this curve. Then the motion $u$ satisfies the equation,

$$q_\perp(t)^\top \frac{\partial q(t)}{\partial t} = -2u^\top (Q(t)q_\perp(t)).$$  \hspace{1cm} (C.23)

where, for each $t$, $q_\perp(t)$ is a vector perpendicular to $q(t)$. Therefore, we cannot recover the motion perpendicular to the vector $Q(t)q_\perp(t)$.

Proof. First, we will show the ground truth motion $u$ satisfy Eq. C.23. Let $q_\perp(t)$ be a non-zero vector that is perpendicular to $q(t)$. Then from Eq. C.21, we have:

$$q_\perp(t)^\top (-J^\top n + n) = 0.$$  \hspace{1cm} (C.24)

Multiply $q_\perp$ to both sides of Eq. C.22, we have:

$$q_\perp^\top \frac{\partial q(0)}{\partial t} = d^{-1}q_\perp^\top Hu - d^{-2}(n^\top Ju)q_\perp^\top (-J^\top n + n) = d^{-1}q_\perp^\top Hu$$  \hspace{1cm} (C.25)

Plugging Eq. C.20 to Eq. C.25, we obtain the equation:

$$q_\perp^\top \frac{\partial q(0)}{\partial t} = q_\perp^\top Q(0)u$$  \hspace{1cm} (C.26)

Second, we will show that, it is impossible to recover the motion perpendicular to $Q(t)q_\perp(t)$. To prove it, we show that for any motion vector $u^* \in \mathbb{R}^2$ that satisfies Eq. C.23, there exists a refractive field and background pattern that generate the same sequence. More specifically, we need to prove, for any $u^* \in \mathbb{R}^2$, $\exists H^* \in \mathbb{R}^{2 \times 2}, J^* \in \mathbb{R}^2, n^* \in \mathbb{R}^2, d^* \in \mathbb{R}$ such that

$$\begin{cases}  
Q(t) = d^{*-1}H^*,  
\hfill (C.27) 
q(t) = d^{*-1}(-J^\top n^* + n^*),  
\hfill (C.28) 
d^*q(t) \quad d^*t = d^{-1}H^*u^* - d^{-2}(n^*^\top J^*u^*)(-J^\top n^* + n^*).  
\hfill (C.29)
\end{cases}$$

First, let $\overline{u}$ be the ground truth refractive motion and let $\overline{n}, \overline{d}, \overline{J}, \overline{H}$ be the ground truth coefficients for the background and refractive object. These ground truth parameter should satisfy Eq. C.20–Eq. C.22, that is:

$$\begin{cases}  
Q(t) = \overline{d}^{-1}\overline{H},  
\hfill (C.30) 
q(t) = \overline{d}^{-1}(-\overline{J}^\top \overline{n} + \overline{n}),  
\hfill (C.31) 
d^*q(t) \quad d^*t = \overline{d}^{-1}\overline{H}\overline{u} - \overline{d}^{-2}(\overline{n}^\top \overline{J}\overline{u})(-\overline{J}^\top \overline{n} + \overline{n}).  
\hfill (C.32)
\end{cases}$$

For any nonzero vector $u^*$ that satisfies Eq. C.23, we will construct coefficient $H^*, J^*, n^*$, and $d^*$ that satisfy Eq. C.27–C.29. First, because both $u^*$ and $\overline{u}$ satisfy Eq. C.23, there exists $\lambda \in \mathbb{R}$, such that:

$$u^* = \overline{u} + \lambda R Q(t) q_\perp(t).$$  \hspace{1cm} (C.33)
where \( R = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} \).

Since \( R \) is a 90° rotation matrix, then we have \( q_\perp(t)^\top Q(t)^\top R Q(t) q_\perp(t) = 0 \). Therefore there exists \( \lambda' \) such that (recall that \( q_\perp(t) \perp q(t) \)):

\[
\lambda' q(t) = \lambda Q(t)^\top R Q(t) q_\perp(t).
\]  

(C.34)

Because \( u^* \) is non-zero, there exists a \( 2 \times 2 \) non-singular matrix \( U \), such that:

\[
U u^* = \left( 1 - \frac{\lambda'}{\lambda^\top J u} \right) u.
\]  

(C.35)

At last, define the parameters of background and refractive field as:

\[
\begin{align*}
    n^* &= A^\top \bar{n} \\
    d^* &= \bar{d} \\
    J^* &= A^\top \bar{J} U \\
    H^* &= \bar{H}
\end{align*}
\]

(C.36) (C.37) (C.38) (C.39)

where \( A = \bar{J} U + I - \bar{J} \). Under this definition, we have:

\[
-J^\top n^* + n^* = (I - J^*)^\top n^* = (I - A^\top \bar{J} U)^\top A^\top \bar{n} = (A - \bar{J} U)^\top \bar{n} = (I - \bar{J})^\top \bar{n} = -\bar{J}^\top \bar{n} + \bar{n}
\]

(C.40)

\[
\lambda' - n^\top J^* u^* = \lambda' - \bar{n}^\top A A^{-1} \bar{J} U u^* = \lambda' - \bar{n}^\top \bar{J} \left( 1 - \frac{\lambda'}{\bar{n}^\top \bar{J} u} \right) \bar{u} = -\bar{n}^\top \bar{J} u
\]

(C.41)

We now prove that the first and second order terms of the observation under the new setup \( u^*, n^*, d^*, J^* \), and \( H^* \) are the same as the first and second terms of the observation under the ground truth setup:

\[
Q^*(0) = d'^{-1} H^* = \bar{d}^{-1} \bar{H}
\]

(C.42)

\[
q^*(0) = d'^{-1} (-J^\top n^* + n^*) = \bar{d}^{-1} (-\bar{J}^\top \bar{n} + \bar{n}) \quad \text{(by Eq. C.40): \ -J^\top n^* + n^* = -\bar{J}^\top \bar{n} + \bar{n})}
\]

(C.43)

\[
\frac{\partial q^*(0)}{\partial t} = d'^{-1} H^* u^* - d'^{-2} n^\top J^* u^* (-J^\top n^* + n^*)
\]

\[
= \bar{d}^{-1} \bar{H} \bar{u} + \lambda R Q(t) q_\perp(t) - \bar{d}^{-2} n^\top J^* u^* (-J^\top n^* + n^*) \quad \text{(by Eq. C.20: \( Q(t) = d'^{-1} \bar{H} \))}
\]

\[
= \bar{d}^{-1} \bar{H} \bar{u} + \bar{d} \frac{\lambda}{\bar{d}} q(t) - \bar{d}^{-2} n^\top J^* u^* (-J^\top n^* + n^*) \quad \text{(by Eq. C.34 \( \lambda R Q(t) q_\perp(t) = \frac{\lambda}{\bar{d}} q(t) \))}
\]

\[
= \bar{d}^{-1} \bar{H} \bar{u} + \frac{\lambda'}{\bar{d}} q(t) - \bar{d}^{-2} n^\top J^* u^* (-J^\top n^* + n^*) \quad \text{(by Eq. C.40: \ -J^\top n^* + n^* = -\bar{J}^\top \bar{n} + \bar{n} \text{ and Eq. C.40: \ \lambda' - n^\top J^* u^* = -\bar{n}^\top \bar{J} u)}}
\]

\[
= \frac{\partial q(0)}{\partial t} \quad \text{(Because the ground truth setup satisfy Eq. C.22)}
\]

(C.44)
This proves the correctness of Eq. C.30–Eq. C.32. Therefore, for any motion vector \( \mathbf{u}^* \) of the refractive object that satisfies the refractive flow equation Eq. C.23 is a possible explanation of the observed sequence. This shows the ambiguity of the problem.

\[ Q(t)q_1(t) = aHR(I - J^\top)n, \]  
\[(C.45)\]

Proposition. The following identity holds,

where \( a \) is a scalar, and \( R \) is a 90° rotation matrix, that is \( R = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \). This means that motion perpendicular to \( HR(I - J^\top)n \) is not recoverable.

Proof. From Eq. C.20 and Eq. C.21, we have:

\[ Q(t)q_1(t) = Q(t)Rq(t) = d^{-2}HR(I - J)^\top n \]  
\[(C.46)\]

Let \( a = d^{-2} \). This proves the correctness of Eq. C.45
Appendix D

Derivation of Iterative Re-weighted Least Square

The task of obstruction removal is to decompose an input sequence \( \{I^t\} \) into the background and obstruction components, \( I_B \) and \( I_O \). We will first derive an algorithm for the more general case of an unknown, spatially varying alpha map, \( A \), and then show a small simplification that can be used for reflection removal where we assume the alpha map is constant. Let \( \{V_O^t\} \) and \( \{V_B^t\} \) are the sets of motion vectors for the obstruction and background components, respectively. According to the Section 4.1 of [149],

\[
\min_{I_O, I_B, A, \{V_O^t\}, \{V_B^t\}} \sum_t \left[ ||I^t - W(V_O^t)I_O - W(V_B^t)A \circ W(V_B^t)I_B||_1 + \lambda_1 ||\nabla A||_2^2 + \lambda_2 (||\nabla I_O||_1 + ||\nabla I_B||_1) + \lambda_3 L(I_O, I_B) + \lambda_4 \sum_t ||\nabla V_O^t||_1 + ||\nabla V_B^t||_1 \right]
\]

Subject to:

\[0 \leq I_O, I_B, A \leq 1.
\]

where \( I^t, I_B, I_O, A \in \mathbb{R}^n \), temporal index \( t \in \{1, 2, ..., T\} \), and \( n \) is number of pixels per frame. \( V_O, V_B \in \mathbb{R}^{2N} \), and \( W(V_B^t) \in \mathbb{R}^{nxn} \) is a warping matrix such that \( W(V_B^t)I_B \) is the warped background component \( I_B \) according to the motion field \( V_B^t \). We define \( L(I_O, I_B) = \sum_x ||\nabla I_O(x)||^2 ||\nabla I_B(x)||^2 \), where \( x \) is the spatial index and \( \nabla I_B(x) \) is the gradient of image \( I_B \) at position \( x \). Also, recall that we use \( \circ \) to denote element-wise product.

We use an alternating gradient descent method to solve Eq. D.1. We first fix the motion fields \( \{V_O^t\} \) and \( \{V_B^t\} \) and solve for \( I_O, I_B \) and \( A \), and then fix \( I_O, I_B \) and \( A \), and solve for \( \{V_O^t\} \) and \( \{V_B^t\} \). Similar alternating gradient descent approach for joint estimation has been used in video super resolution [80].
D.1 Decomposition step: fix motion fields and solve for images of each layer.

In this step, we fix motion fields \{V_O\} and \{V_B\}, and solve for \(I_O\), \(I_B\), and \(A\). We ignore all the terms in Eq. D.1 that only consist of \(V_O^t\) and \(V_B^t\): Ignoring all the terms that only contain \(v_O^t\) and \(v_B^t\) in Eq. D.1, we get:

\[
\min_{I_O, I_B, A} \sum_t \left[ ||I^t - W_O^t I_O - W_O^t A \circ W_B^t I_B||_1 + \lambda_1 ||\nabla A||^2 + \lambda_2 (||\nabla I_O||_1 + ||\nabla I_B||_1) + \lambda_3 L(I_O, I_B) \right]. \tag{D.2}
\]

Subject to

\[
0 \leq I_O, I_B, A \leq 1 .
\]

We solve this problem using a modified version of iterative reweighted least squares (IRLS). The original IRLS algorithm is designed for a non-constrained optimization with only \(l_1\)-and \(l_2\)-norms. To get this form, we linearize the higher-order terms in the objective function in Eq. D.2.

First, we approximate the non-smooth \(l_1\)-norm by the smooth function \(\phi(x) = \sqrt{x + \epsilon^2}\), where \(\epsilon\) is a very small number:

\[
\min_{I_O, I_B, A} \sum_t \phi(||I^t - W_O^t I_O - W_O^t A \circ W_B^t I_B||_1^2) + \lambda_1 ||\nabla A||^2 + \lambda_2 (\phi(||D_x I_O||^2) + \phi(||D_y I_O||^2)) + \lambda_3 L(I_O, I_B). \tag{D.3}
\]

Subject to

\[
0 \leq I_O, I_B, A \leq 1 . \tag{D.4}
\]

where \(D_x, D_y \in \mathbb{R}^{\mathbb{R}^{nxn}}\) are the derivative matrices defined as: \((D_x I_B)(x) = \frac{\partial I_B(x)}{\partial x}\) and \((D_y I_B)(x) = \frac{\partial I_B(x)}{\partial y}\)

To achieve the minimum, set the derivative of Eq. D.4 with respect to \(I_B\) and \(I_R\) to 0, we get:

\[
\begin{align*}
\sum_{x,t} w^t(x) \psi_1^t(x) \left( (I^t - W_B^t(x) I_B - W_B^t(x) I_R) W_B^t(x)^T + \sum_x \psi_2(x) (D_x(x)^T D_x(x) + D_y(x)^T D_y(x)) \right) I_B &= 0 \\
\sum_{x,t} w^t(x) \psi_1^t(x) \left( (I^t - W_B^t(x) I_B - W_B^t(x) I_R) W_B^t(x)^T + \sum_x \psi_3(x) (D_x(x)^T D_x(x) + D_y(x)^T D_y(x)) \right) I_R &= 0
\end{align*}
\]

where

\[
\psi_1^t(x) = \phi'(I^t - W_B^t(x)\tilde{I}_B - W_B^t(x)\tilde{I}_R)^2, \tag{D.6}
\]

\[
\psi_2(x) = \lambda_1 \phi'((D_x(x)\tilde{I}_B)^2 + (D_y(x)\tilde{I}_B)^2) + \lambda_2 ((D_x(x)\tilde{I}_R)^2 + (D_y(x)\tilde{I}_R)^2), \tag{D.7}
\]

\[
\psi_3(x) = \lambda_1 \phi'((D_x(x)\tilde{I}_B)^2 + (D_y(x)\tilde{I}_B)^2) + \lambda_2 ((D_x(x)\tilde{I}_R)^2 + (D_y(x)\tilde{I}_R)^2), \tag{D.8}
\]

\[
\phi'(x) = \frac{\partial \phi}{\partial x} = \frac{1}{2\sqrt{x + \epsilon}}. \tag{D.9}
\]

where \(\tilde{I}_B\) and \(\tilde{I}_R\) are the background and reflectance image from the last iteration.

Eq. D.5 is a linear equation respect to \(I_B\) and \(I_R\), so it can be directly solved.
Including the Inequality constraints  So far, we totally ignore the inequality constraints in Eq. D.2. Now we will modify the formulation to include these constraints. Here we use the Barrier method to move the inequality constraints into the objective function:

\[
\min_{\{I_B^t, I_R^t\}_t} \sum_{t=1}^{T-1} \sum_x w^t(x)\|l^t(x) - I_B(x - v_B^t(x)) - I_R(x - v_R^t(x))\|_1 \\
+ \lambda_1 \sum_x (\|\nabla I_B^t(x)\|_1 + \|\nabla I_R^t(x)\|_1) + \lambda_2 \sum_x (\|\nabla I_B^t(x)\|_2^2 + \|\nabla I_R^t(x)\|_2^2) \\
+ \beta \sum_x (\min(0, I_B(x)))^2 + (\min(0, I_R(x)))^2 + (\min(0, 1 - I_B(x)))^2 + (\min(0, 1 - I_R(x)))^2 \\
+ \beta \sum_x (1 - w^t(x)) (\min(0, I_B(x) - v_B^t(x)) + I_R(x - v_R^t(x)) - 1)^2 
\]  

(D.10)

Here the term \(\min(0, I_B(x))^2\) corresponds to the constraint \(I_B(x) \geq 0\). To illustrate that, considering following two cases. 1) When \(I_B(x) \geq 0\), \(\min(0, I_B(x)) = 0\) and there is no penalty. And 2) when \(I_B(x) < 0\), \(\min(0, I_B(x))^2 = I_B(x)^2\) and this term penalize the deviation of \(I_B(x)\) from 0. Therefore, by minimizing the term \(\min(0, I_B(x))^2\), we roughly should a positive \(I_B\).

For the similar reason, \((\min(0, I_R(x)))^2\), \((\min(0, 1-I_B(x)))^2\), \((\min(0, 1-I_R(x)))^2\), and \((1-w^t(x))(\min(0, I_B(x) - v_B^t(x)) + I_R(x - v_R^t(x)) - 1))^2\) correspond to the inequality constraints \(I_B(x) \leq 1\), \(I_R(x) \geq 0\), \(I_R(x) \leq 1\), and \(I_B(x - v_B^t(x)) + I_R(x - v_R^t(x)) > 1\) respectively.

Taking the derivatives of this new object function with respect to \(I_B\) and \(I_R\), we get:

\[
\sum_{x,t} w^t(x)\psi_1^t(x) \left( I^t - W_B^t(x)I_B - W_R^t(x)I_R \right) W_B^t(x)^\top + \sum_x \psi_2(x) \left( D_x(x)^\top D_x(x)I_B + D_y(x)^\top D_y(x)I_B \right) \\
+ \beta \sum_x \left( a_{0B}(x)I_B(x) - a_{1B}(x)(1 - I_B(x)) \right) + \beta \sum_{x,t} (1 - w^t(x))(I_B(x - v_B^t(x)) + I_R(x - v_R^t(x)) - 1)W_B^t(x) = 0, \\
\sum_{x,t} w^t(x)\psi_1^t(x) \left( I^t - W_B^t(x)I_B - W_R^t(x)I_R \right) W_R^t(x)^\top + \sum_x \psi_3(x) \left( D_x(x)^\top D_x(x)I_R + D_y(x)^\top D_y(x)I_R \right) \\
+ \beta \sum_x \left( a_{0R}(x)I_R(x) - a_{1R}(x)(1 - I_R(x)) \right) + \beta \sum_{x,t} (1 - w^t(x))(I_B(x - v_B^t(x)) + I_R(x - v_R^t(x)) - 1)W_R^t(x) = 0, 
\]

(D.11)

where

\[
a_{0B}(x) = 1(I_B(x) < 0), a_{1B}(x) = 1(I_B(x) > 1), \\
a_{0R}(x) = 1(I_R(x) < 0), a_{1R}(x) = 1(I_R(x) > 1), 
\]

(D.12)

(D.13)

where \(1(A)\) is the indicator function that only takes 1 when the event \(A\) happens, and takes 0 otherwise.

**D.2 Motion Estimation Step: Fix images of each layer and solve for motion fields**

In this step, we will fix \(I_B\) and \(I_R\) and solve for motion fields \(v_B\) and \(v_R\). This become a joint optical flow problem. Ignoring all the terms that only contain \(I_B\) and \(I_R\) in Eq. D.1, we get (ignoring constraints \(v_B^t < T\)
and \( \nu^t_R > T \):

\[
\min_{\nu^t_B, \nu^t_R} \sum_x w^t(x) \| I^t(x) - I_B(x - \nu^t_B(x)) - \nu^t_R(x) \|_1 + \lambda_3 \sum_x \| \nabla \nu^t_B(x) \|_1 + \| \nabla \nu^t_R(x) \|_1 \tag{D.14}
\]

Note that when we solve the motion field, there is no dependency between frames, so that we can solve each frame independently.

Again, we solve this problem using IRLS. First linearizing \( I_B(x - \nu^t_B(x)) \) and \( I_R(x - \nu^t_R(x)) \), we get:

\[
\min_{\nu^t_B, \nu^t_R} \sum_{t=1}^{T-1} \sum_x w^t(x) \| I^t(x) - I_B(x) - I_R(x) + \nabla I_B(x)^\top \nu^t_B(x) + \nabla I_R(x)^\top \nu^t_R(x) \|_1 + \lambda_3 \sum_{t=1}^T \sum_x \| \nabla \nu^t_B(x) \|_1 + \| \nabla \nu^t_R(x) \|_1. \tag{D.15}
\]

Also, replacing the non-linear \( l_1 \)-norm by the function \( \phi(x) = \sqrt{x + \epsilon^2} \), we get:

\[
\min_{\nu^t_B, \nu^t_R} \sum_{t=1}^{T-1} \sum_x w^t(x) \phi \left( (I^t(x) - I_B(x) - I_R(x) + \nabla I_B(x)^\top \nu^t_B(x) + \nabla I_R(x)^\top \nu^t_R(x))^2 \right) + \lambda_3 \sum_{t=1}^T \sum_x \phi \left( (D_x \nu_x)^2 + (D_y \nu_x)^2 \right) + \phi \left( (D_x \nu_y)^2 + (D_y \nu_y)^2 \right) \tag{D.16}
\]

Similar to the Section D.1, this can also be solved using IRLS.

*Note that to simplify the derivation, here we did the first order Taylor expansion at \( \nu_B(x) = 0 \) and \( \nu_R(x) = 0 \). In practice, we did the Taylor expansion at \( \nu_B(x) = \overline{\nu}_B(x) \) and \( \nu_R(x) = \overline{\nu}_R(x) \), where \( \overline{\nu}_B(x) \) and \( \overline{\nu}_R(x) \) are the motion vectors from previous iterations.
Appendix E

Hierarchical Beam Model

E.1 Deriving the ODE

As discussed in Section 3.1 in the main paper, each branch $i$ has to satisfy the following 5 equations:

1. $m_i a_i = -r_i + \sum_{c \in C_i} r_c + m_i g_i$, \hfill (E.1)
2. $I_i \dot{\omega}_i = -k_i \dot{\theta}_i + \sum_{c \in C_i} k_c \dot{\theta}_c + r_i \times x_i + \sum_{c \in C_i} r_c \times x_i$, \hfill (E.2)
3. $a_i = a_{i,0} + \dot{\omega}_i \times x_i + \omega_i \times (\omega_i \times x_i)$, \hfill (E.3)
4. $\omega_i = \dot{\theta}_i + \sum_{p \in P_i} \dot{\theta}_p$, \hfill (E.4)
5. $\dot{\omega}_i = \ddot{\theta}_i + \sum_{p \in P_i} \ddot{\theta}_p$, \hfill (E.5)

where all vectors are defined in a fixed global coordinate shown in Figure E.1 below, and the unknowns are $a_i$, $a_{i,0}$, $r_i$, $\omega_i$, $\dot{\omega}_i$, $\theta_i$, $\dot{\theta}_i$, $\ddot{\theta}_i$. $a_i$ represents the acceleration of branch $i$ at its center of mass. $a_{i,0}$ is the acceleration of branch $i$'s junction with its parent. $r_i$ is the force branch $i$ applied to its parent. $\omega_i$, $\dot{\omega}_i$ are branch $i$'s angular velocity and angular acceleration respectively. $\theta_i$ is the deviation angle of branch $i$ from the equilibrium position. $\dot{\theta}_i$ and $\ddot{\theta}_i$ are $\theta$'s first and second order derivative with respect to time, respectively.

As we are only considering the 2D motion of the tree, all rotation vectors (angular velocity $\omega$ and angular acceleration $\dot{\omega}$) in Equation (E.2) are perpendicular to the plane of the tree (image plane), and align with the direction of $r \times x$. With Equations (E.1)-(E.5), we aim to derive the ODE in the form of Equation (5) in Section 3.1 of the main paper

$$g(\theta, \dot{\theta}, \ddot{\theta}) = 0.$$ \hfill (E.6)

To start with, we could substitute all $\omega_i$ and $\dot{\omega}_i$ with $\dot{\theta}_i$ and $\ddot{\theta}_i$, respectively, using Equations (E.4) and (E.5). For the ease of description, we define

$$f_i(\theta) \overset{\text{def}}{=} \theta_i + \sum_{p \in P_i} \theta_p,$$ \hfill (E.7)
Then, one could recursively substitute $a_i$ using

$$a_{io} = a_{po} + f_p(\ddot{\theta}) \times h_i + f_p(\dot{\theta}) \times (f_p(\theta) \times h_i),$$

(E.9)

where $h_i$ is the vector that starts from its parent's junction and points to its own junction point. As the junction of the trunk has zero acceleration, all $a_{io}$ and $a_i$ can be replaced by a function of $\theta$, $\dot{\theta}$, and $\ddot{\theta}$, note

$$a_i(\theta, \dot{\theta}, \ddot{\theta}) \overset{\text{def}}{=} a_i.$$ (E.10)

To substitute all $r_i$, one could start with any leaf branch $l$, so that Equation (E.1) becomes

$$m_i a_l(\theta, \dot{\theta}, \ddot{\theta}) = -r_l + m_i g.$$ (E.11)

For all the leaf branches, we define

$$r_l(\theta, \dot{\theta}, \ddot{\theta}) \overset{\text{def}}{=} -m_i a_l(\theta, \dot{\theta}, \ddot{\theta}) + m_i g.$$ (E.12)

so we could write $r_l$ as $r_l(\theta, \dot{\theta}, \ddot{\theta})$.

Then we could iteratively substitute all $r_i$ from leaf branches to the trunk, i.e.,

$$m_i a_i(\theta, \dot{\theta}, \ddot{\theta}) = -r_i + \sum_{c \in C_i} r_c(\theta, \dot{\theta}, \ddot{\theta}) + m_i g.$$ (E.13)

and we define

$$r_i(\theta, \dot{\theta}, \ddot{\theta}) = \sum_{c \in C_i} r_c(\theta, \dot{\theta}, \ddot{\theta}) + m_i g - m_i a_i(\theta, \dot{\theta}, \ddot{\theta}),$$ (E.14)

\footnote{With the abuse of notation, here $a_i$ denotes a 2D vector of the acceleration of branch $i$, and $a_i(\theta, \dot{\theta}, \ddot{\theta})$ denotes a function that maps $\theta$, $\dot{\theta}$, and $\ddot{\theta}$ to the acceleration.}
for all non-leaf branches. Then we have the forces \( r_i \) written as a function \( r_i(\theta, \dot{\theta}, \ddot{\theta}) \) of \( \theta, \dot{\theta}, \) and \( \ddot{\theta} \) for all the branches.

Substituting Equation (E.2) with Equations (E.8) and (E.14), we have

\[
I_i \ddot{f}_i(\ddot{\theta}) = -k_i \dot{\theta}_i + \sum_{c \in C_i} k_c \dot{\theta}_c + r_i(\theta, \dot{\theta}, \ddot{\theta}) \times x_i + \sum_{c \in C_i} r_c(\theta, \dot{\theta}, \ddot{\theta}) \times x_i,
\]

which gives us the desired ODE system of the form

\[
g(\theta, \dot{\theta}, \ddot{\theta}) = 0.
\]

**E.2 Linearizing the ODE**

In this section, we provide details on how to linearize Equation (E.6) with the assumption that \( \theta \) is small. As mentioned in our paper, we assume small vibrations, i.e., \( O(\theta^2) \), \( O(\ddot{\theta}) \), and \( O(\ddot{\theta}^2) \) are negligible. We start by investigating all the non-linear terms in Equations (E.1) to (E.5).

By writing the scalar form of Equations (E.1) to (E.5), one could find non-linear terms only in Equations (E.2) and (E.3). Specifically, for Equation (E.2), we have:

\[
I_i \dot{\theta}_i = -k_i \dot{\theta}_i + \sum_{c \in C_i} k_c \dot{\theta}_c + (r_{ix} + r_{cx}) \frac{l_i}{2} \cos \left( \theta_i + \dot{\theta}_i + \Theta_i + \dot{\Theta}_i \right),
\]

\[
+ (r_{iy} + r_{cy}) \frac{l_i}{2} \sin \left( \theta_i + \dot{\theta}_i + \Theta_i + \dot{\Theta}_i \right),
\]

where

\[
\Theta_i = \sum_{p \in P_i} \theta_p, \quad \hat{\Theta}_i = \sum_{p \in P_i} \hat{\theta}_p,
\]

\[
r_i = r_{ix} e_x + r_{iy} e_y, \quad r_c = r_{cx} e_x + r_{cy} e_y.
\]

Using our assumption, we could linearize the \( \sin(\cdot) \) and \( \cos(\cdot) \) terms through the first-order Taylor expansion (note that both \( \Theta_i \) and \( \theta_i \) are very small and their higher-order terms are negligible):

\[
\sin \left( \theta_i + \dot{\theta}_i + \Theta_i + \dot{\Theta}_i \right) \approx \sin \left( \dot{\Theta}_i + \dot{\theta}_i \right) + \cos \left( \dot{\Theta}_i + \dot{\theta}_i \right) (\Theta_i + \theta_i),
\]

\[
\cos \left( \theta_i + \dot{\theta}_i + \Theta_i + \dot{\Theta}_i \right) \approx \cos \left( \dot{\Theta}_i + \dot{\theta}_i \right) - \sin \left( \dot{\Theta}_i + \dot{\theta}_i \right) (\Theta_i + \theta_i).
\]

For Equation (E.3), we have:

\[
a_{ix} = a_{ix} + \dot{\omega}_i \left[ -\frac{l_i}{2} \cos \left( \theta_i + \dot{\theta}_i + \Theta_i + \dot{\Theta}_i \right) \right] + \omega_i^2 \left[ -\frac{l_i}{2} \sin \left( \theta_i + \dot{\theta}_i + \Theta_i + \dot{\Theta}_i \right) \right],
\]

\[
a_{iy} = a_{iy} + \dot{\omega}_i \left[ -\frac{l_i}{2} \sin \left( \theta_i + \dot{\theta}_i + \Theta_i + \dot{\Theta}_i \right) \right] + \omega_i^2 \left[ -\frac{l_i}{2} \cos \left( \theta_i + \dot{\theta}_i + \Theta_i + \dot{\Theta}_i \right) \right].
\]
Taking the first-order Taylor expansion to the equations above, we have

\[
\begin{align*}
a_{ix} & \approx a_{ix} + a_i \dot{a}_i \approx a_{ix} + \omega_i \left[ -\frac{l_i}{2} \cos (\hat{\theta}_i + \hat{\phi}_i) \right], \quad (E.19) \\
a_{iy} & \approx a_{iy} + a_i \dot{a}_i \approx a_{iy} + \omega_i \left[ -\frac{l_i}{2} \sin (\hat{\theta}_i + \hat{\phi}_i) \right]. \quad (E.20)
\end{align*}
\]

where terms of the form \( \omega \theta_i \) are omitted as \( \omega \theta_i \leq \frac{1}{2} (\dot{\omega}_i^2 + \theta_i^2) \) is a second-order term and is negligible. Also, the last term in Equations (E.17) and (E.18) contains a second-order factor \( \omega_i^2 \), and is also ignored. Then, using the same elimination techniques from the previous section, we get a set of linear equations with respect to \( \theta \) and \( \dot{\theta} \), i.e.,

\[
M \ddot{\theta} + K \theta = 0,
\]

which is Equation (6) in the main manuscript.

### E.3 Simulation Details

We built a tree simulator by solving Equation (E.6) using the Euler method. Let \( t \) be the temporal index, and \( \theta(t) \) be all the rotations observed at time \( t \). The idea is that given the initial condition \((\theta(0), \dot{\theta}(0))\), we can solve \( \dot{\theta}(0) \) by inverting a linear system. Then, \( \dot{\theta}(\Delta t) \) is approximated by \( \dot{\theta}(0) + \dot{\theta}(0) \Delta t \), and \( \theta(\Delta t) \) by \( \theta(0) + \dot{\theta}(0) \Delta t \).

Theoretically, one could explicitly write out the linear system by performing the substitution described earlier. However, it is much more convenient to introduce intermediate variables \( a_i, a_{io}, r_i, \) and \( \dot{\omega}_i \), and solve the linear system built from Equations (E.1) to (E.5) and Equation (E.9). By gathering the coefficients at time \( t \), one could write all linear equations above as

\[
A \begin{bmatrix} \dot{\theta}(t) \\ r(t) \\ a(t) \\ a_{io}(t) \\ \dot{\omega}(t) \end{bmatrix} = b. \quad (E.21)
\]

Then we can get \( \dot{\theta}(t) \) by solving the linear system above.
Bibliography


