Summer 8-15-2016

Geometric Inference with Microlens Arrays

Ian Schillebeeckx
Washington University in St. Louis

WASHINGTON UNIVERSITY IN SAINT LOUIS

School of Engineering and Applied Science

Department of Computer Science and Engineering

Dissertation Examination Committee:
Robert Pless, Chair
Bryan Clair
Yasutaka Furukawa
Viktor Gruev
Tao Ju
Joseph O'Sullivan

Geometric Inference with Microlens Arrays

by

Ian Peter Schillebeeckx

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2016
Saint Louis, Missouri

Contents

List of Figures

List of Tables

Acknowledgments

A PhD is a journey of personal growth that requires the support of many people. My journey is no exception, and I owe a great deal of gratitude to those who have directly and indirectly supported me, providing a diversity of assistance from academic guidance to encouragement. There are many people who deserve thanks, and I apologize for anyone who I have inadvertently omitted in these acknowledgements. However, I hope all of those who supported me understand my appreciation and their importance in my training as a scientist and growth as an individual.

First, I owe my entire training to my advisor, Robert Pless. I am grateful to have had a mentor who was not only interested in my academic success, but also my personal growth. Robert's ubiquitous enthusiasm, creativity, and guidance for topics in and outside the lab made research enjoyable and inspiring. I recognize how fortunate I am to have an advisor that supported my interests and goals outside of the lab, particularly with IDEA Labs STL. Robert trusted me to balance my time with this interest, giving me the confidence to lead IDEA Labs STL. Robert has taught me a pragmatic style of analysis and problem solving that I will forever cherish.

I have valued the openness and advice from those on my committee. I thank Yasutaka Furukawa, Viktor Gruev, Tao Ju, and Joseph O'Sullivan who have provided a diversity of perspectives and access to equipment that have enabled my research. I also would like to thank Bryan Clair, who helped me gain a curiosity and passion for Computer Science during my undergraduate studies.

I have had the tremendous privilege of being part of IDEA Labs STL while being a PhD student. This organization and the people in it have inspired me to change the world and impact people's lives. In addition, their support has allowed me to balance the challenges of pursuing interests outside of the lab. I am grateful to all those who have helped make IDEA Labs STL the impactful organization it is today.

I have enjoyed a wealth of support from my friends. I am thankful for Zheng Chen and Austin Abrams for being valuable role models and mentors during my initial adjustment to the research lifestyle. I thank Samuel Powell, whose fastidious conversations inspired me to be diligent in my work. I appreciate the fun and energetic members of Pless lab, especially Abby Stylianou and Brendan Kelly. I would also like to thank other friends in the Computer Science program including Jon Shidal and Missael Garcia. Outside of Washington University, I enjoyed the support of countless friends including Bryan McDaniel, Michael Rooney, and Melissa Grim.

My family has shaped me as the individual I am today. My father, Marc, and mother, Roselyne, have instilled in me the pride and diligence in doing rigorous work. My brothers,

Titou and Maxim, have always been my best friends and role models in a foreign world and have inspired me to be better that i thought possible.

Foremost, I am inexpressibly grateful for the love and support of my champion, best friend, and girlfriend Jeanette Gehrig. Her unwavering patience, understanding, and kindness have made my journey achievable and her presence and companionship inspire me everyday. I dedicate this work to You.

Ian Peter Schillebeeckx

Washington University in Saint Louis

August 2016

ABSTRACT OF THE DISSERTATION

Geometric Inference with Microlens Arrays

by

Ian Peter Schillebeeckx

Doctor of Philosophy in Computer Science

Washington University in St. Louis, August 2016

Research Advisor: Robert Pless

This dissertation explores an alternative to traditional fiducial markers where geometric information is inferred from the observed position of 3D points seen in an image. We offer an alternative approach which enables geometric inference based on the relative *orientation* of markers in an image. We present markers fabricated from microlenses whose appearance changes depending on the marker's orientation relative to the camera. First, we show how to manufacture and calibrate chromo-coding lenticular arrays to create a known relationship between the observed hue and orientation of the array. Second, we use 2 small chromo-coding lenticular arrays to estimate the pose of an object. Third, we use 3 large chromo-coding lenticular arrays to calibrate a camera with a single image. Finally, we create another type of fiducial marker from lenslet arrays that encode orientation with discrete black and white appearances. Collectively, these approaches offer new opportunities for pose estimation and camera calibration that are relevant for robotics, virtual reality, and augmented reality.

Chapter 1

Introduction

Understanding 3D positions is vital for measurement, navigation, grasping, and applications like 3D surgery. Visual sensors offer convenient methods to describe the environment but require geometric inference to understand 3D position.

Geometric inference most often starts by finding the image of known 3D points. To simplify this process, 3D points are often designed to be easy to find and these easy to find points are called fiducial markers. Fiducial markers are the standard for many applications, including understanding the pose of an object relative to a camera or calibrating the optics of a camera. Traditional fiducial markers create a relationship between their position and their projected image location. Sometimes this is limiting because positional constraints must be combined to give orientation constraints, and this is sometimes a weak relationship. For example, two images of 3D points on a plane nearly perpendicular to the camera look very similar and this ambiguity results in errors in inferring their 3D location.

Therefore, this dissertation considers an alternative fiducial marker design. We show how to manufacture and use new fiducial markers that give extra orientation information for geometric inference in pose estimation and camera calibration. These novel fiducial markers

are created from small or large microlens arrays. More direct geometric inference is possible with these objects because they produce an appearance based on relative orientation.

1.1 Creating Orientation Specific Appearances with Microlens Arrays

Figure 1.1 depicts a cross section of how a microlens array is able to create different appearances for specific relative orientations. All the optical elements that comprise the microlens array focus parallel light rays onto a designed pattern behind each microlens. The pattern can be designed such that a) each microlens may focus on a similar local pattern and thus create a single macro appearance for the entire array, or b) each microlens may focus on a different local pattern and thus create a unique appearance for each microlens of the array. Regardless of how the pattern is designed, the appearance of the microlens array depends on the viewing angle and on the pattern adhered to the back of the array. Therefore, we explore how to create microlens arrays by designing patterns that affect appearance so that the array has useful and interesting geometric properties.

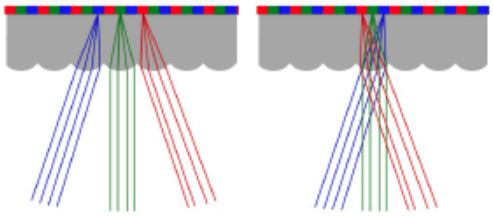
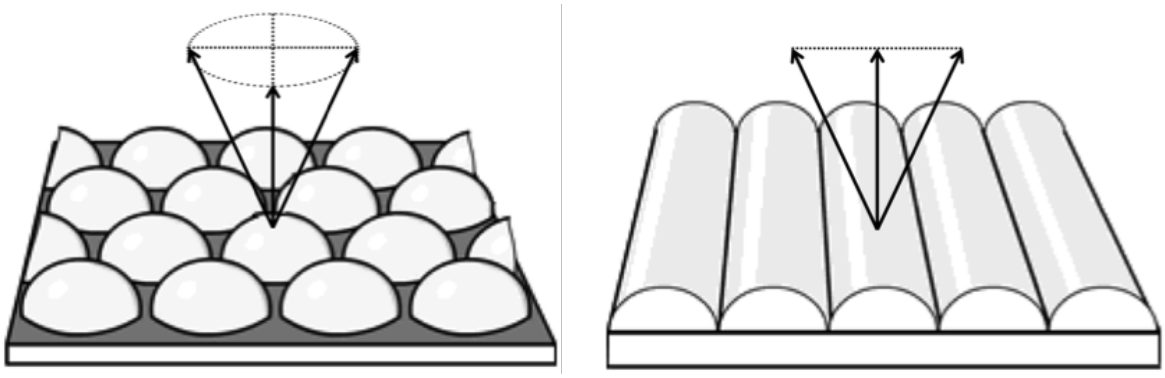


Figure 1.1: We show cross sections of a microlens array to detail the individual microlens elements. (Left) Parallel light rays are focused onto the back focal plane of the array. (Right) Local patterns behind each lens makes a lenticular pattern whose appearance depends on the viewing angle. One can think of each microlens as a light source that projects a different appearance, such as color, in each direction.



(a) Lenslet Array (b) Lenticular Array

Figure 1.2: In this dissertation, we use 2 types of microlens arrays for geometric inference. The lenslet array (left) will have varying appearance for viewpoints in any direction in 2d spherical coordinates. A special case of the lenslet array, the lenticular array (right) only varies appearance for viewpoints rotated around the length of the individual optical elements.

In this work, we use microlens arrays whose focal length is equal to the thickness of the array. Microlens arrays can be divided into 2 classes which describe the shape of the individual optical elements comprising the array: spherical and cylindrical. The work presented in this dissertation uses both classes of microlens arrays and we visualize the two types in Figure 1.2. The spherical microlens arrays are typically used as light diffusers or collimators. When we seek to specifically refer to these spherical microlens arrays, we will call them lenslet arrays. These lenslet arrays can be designed such that their appearance changes for viewpoint changes in spherical coordinates. The cylindrical microlens arrays, often used in children's toys and corporate promotional materials, are commonly referred to as lenticular arrays. Because of their shape, these arrays change appearance as one rotates the lenticular array around a single axis. In this dissertation, we use both types of microlens arrays and we adopt this terminology to differentiate between the two.

1.2 Problem Domain

Lenticular arrays and microlens arrays give orientation cues to solve two inter-related and fundamental Computer Vision problems: pose estimation and camera calibration. Briefly, pose estimation seeks to use images to understand the position and orientation, or the pose, of an object relative to a camera in the physical world. This is important in manufacturing and robotics applications, in order to manipulate and navigate in the physical world. However, to accomplish accurate pose estimation, the geometric properties of a camera and its optics, such as focal length, must be known. Camera calibration is the process to procure this knowledge. Together, pose estimation and camera calibration seek to define how a 3D object in the physical world is projected onto a 2D image taken by a given camera. Standard calibration approaches with traditional fiducial markers often involves simultaneous estimation of the pose of a reference object, as well as the camera properties. One of the advantages of the view dependent fiducial markers considered here is that the camera calibration cues are partially decoupled from pose estimation, leading to simpler approaches that use a single image. In the following section, we detail the contributions of this dissertation.

1.3 Contributions

To support the research aim of creating new fiducial markers with microlens arrays, we have developed prototypes, described geometric constraints, created algorithms, and characterised pose estimation and camera calibration performance. Here, we give a brief overview of the novel contributions presented in this dissertation, visualized in Figure 1.3.

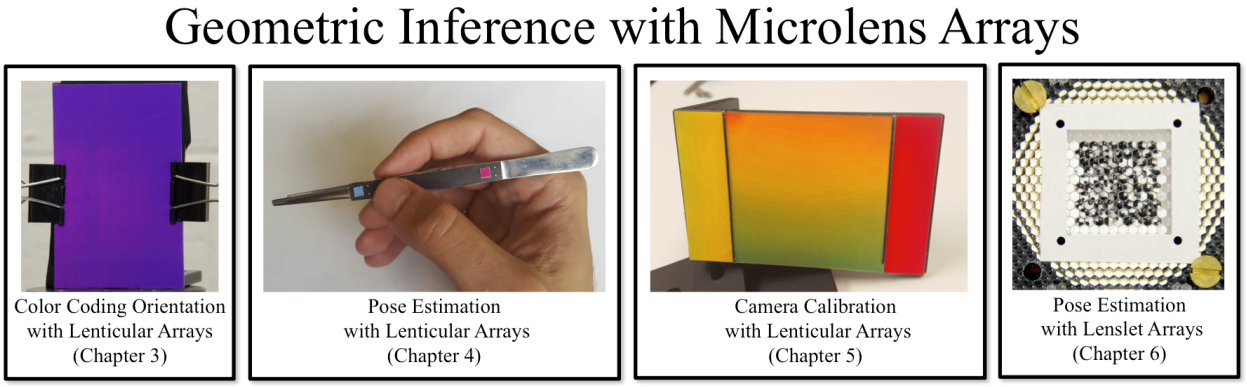


Figure 1.3: Organization of this dissertation. We use 2 types of microlens arrays for geometric inference. We show how to create lenticular arrays which encode orientation by color in Chapter 3. These lenticular arrays are used as fiducial markers for pose estimation in Chapter 4 and camera calibration in Chapter 5. Lenslet arrays, the 2D analog to lenticular arrays, are used to create fiducial markers in Chapter 6 for pose estimation.

1.3.1 Color-coding Orientation with Lenticular Arrays

In Chapter 3, we describe how to create lenticular arrays which encode orientation by hue and explore their properties. We detail how to assemble these chromo-coding lenticular from blank lenticular arrays of a particular manufacturer. We document some of the major challenges with encoding orientation with hue with lenticular arrays and show how to address a major assembly challenge. These chromo-coding lenticular arrays are used for pose estimation and camera calibration.

1.3.2 Pose Estimation with Lenticular Arrays

In Chapter 4, we create small fiducial markers from chromo-coding lenticular arrays, called chromo-coded markers, to estimate the pose of an object. The chromo-coding lenticular array has different hue appearance for different orientations and we can derive constraints to solve for the orientation of the lenticular array relative to the camera independent of its

relative position. This constraint eliminates some limitations that challenge traditional fiducial markers. With 2 chromo-coded markers, we can estimate the full pose of an object, and with more, we can make our method tolerant to changing environmental lighting conditions.

1.3.3 Camera Calibration with Lenticular Arrays

In Chapter 5, we use 3 large chromo-coding lenticular arrays to create a calibration object. This object derives constraints similar to those used for pose estimation, but because the lenticular arrays are large, we can relate the appearance of all parts of the object to the camera parameters. Depending on the focal length of an image, the rays imaging the calibration object will having varying incident angles, resulting in more or less color change across the large chromo-coding lenticular arrays. These cues enable the method to estimate the focal length, in addition to pose from a single image of the calibration object.

1.3.4 Pose Estimation with Lenslet Arrays

In Chapter 6, we create a fiducial marker from a lenslet array to estimate the pose of an object from a single image. This fiducial marker encodes viewpoint using discrete black or white appearances of individual lenslets. When combined with traditional fiducial markers, one can solve for the full pose of an object. This approach gives the best reported pose estimation results and avoids issues with measuring color.

1.4 Dissemination of Work

Much of the work in this dissertation has appeared in peer reviewd conference articles, each of which I was the first and primary contributor. All code, experiments, and figures are my work, except where explicitly noted. In each work, co-authors helped with text. The following lists the publications used and in which chapters their content appears.

- "Structured Light Field Design for Correspondence Free Rotation Estimation" is a paper that was presented at the International Conference on Computation Photography 2015 [55] and is co-authored with Robert Pless. The work characterizing the chromo-coding lenticular arrays in this paper is the basis for Chapter 3.

- "The Geometry of Colorful, Lenticular Fiducial Markers" is a paper that was presented at the International Conference on 3D Vision 2015 [54] and is co-authored with Joshua Little, Brendan Kelly, and Robert Pless. In this work, Little wrote the code that optimizes for pose using traditional markers and Brendan Kelly helped annotate marker locations in experimental video. Chapter 4 presents the material from this paper.

- "Using Chromo-coded Light Fields for Augmented Reality" is a poster that was presented at the IEEE International Conference on Virtual Reality 2016 [58] and is co-authored with Robert Pless. This work is included as the applications in Chapters 4 and 5.

- "Single Image Camera Calibration with Lenticular Arrays for Augmented Reality" is a paper that was presented at the Conference on Computer Vision and Pattern Recognition 2016 [57] and is co-authored with Robert Pless. Most of this work was presented

in Chapter 5, but work identifying and addressing some manufacturing challenges were included in Chapter 3.

- "Pose Hashing with Microlens Arrays" is a paper that will be presented at the European Conference for Computer Vision 2016 [56] and is co-authored with Robert Pless. This work and other unpublished work on back-plane texture analysis is presented in Chapter 6.

Chapter 2

Background

The work in this dissertation is related to a variety of topics in Computer Vision that attempt geometric inference for pose estimation and camera calibration. Because geometric inference is a core problem, there is a rich literature in pose estimation and fiducial marker design. Traditional fiducial markers derive geometric constraints from point correspondences or the relationship between known 3D points with known positions and the 2D projected location of those points onto an image. As such, these markers are designed to be easy to detect for any camera exposure setting or marker position. Our fiducial markers are fundamentally different from traditional markers in that they present orientation cues from their varying appearance. Some existing work has proposed using materials that also have different appearances for different orientations for geometric inference. In fact, lenticular arrays and microlens arrays have been used in the past to understand how light refracts through transparent objects in order to reconstruct that object's shape. Our work uses the changing appearance of microlens arrays to directly solve for pose estimation and camera calibration. The most similar work to ours uses lenticular arrays and microlens array to create a visual effect that changes the position of a cross or line to indicate orientation for pose estimation. This technique has some of the same challenges as other point correspondence methods. Our fiducial markers

build on this method with explicit orientation cues based on a hue or grayscale value at a pixel. In the following section, we review these topics in more depth to provide context and contrast to our work.

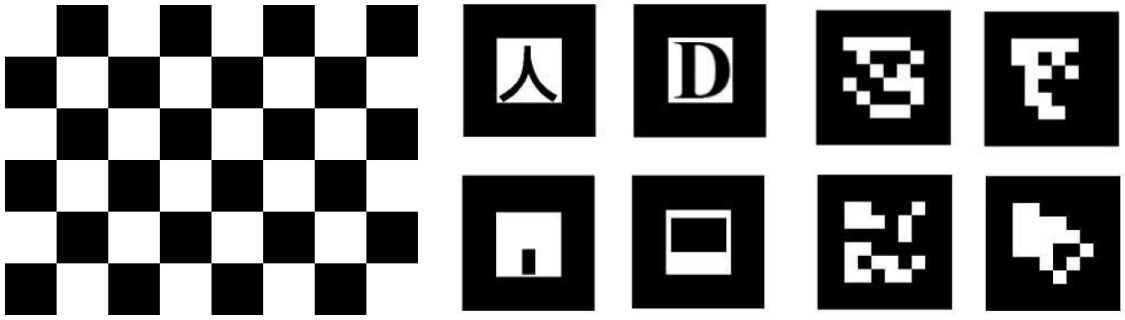
2.1 Traditional Fiducial Markers

Most current fiducial marker approaches rely on correspondences between the image and the apparent position of fiducial markers for pose estimation and camera calibration. These fiducial markers typically have the following characteristics:

- known intra-marker 3D geometries for geometric inference
- high contrast for easy geometric interpretation and identification
- inter-marker differentiability
- robust to varying imaging conditions
- design to enable sub-pixel localization of marker position

All of these characteristics facilitate the consistent and precise use of point correspondences for geometric inference.

For the pose of a plane to be uniquely determined from a single image, at least four point correspondences are needed, and so most fiducial markers are based off of using a square where each corner serves as a point correspondence. These squares are typically black on a white background for easy identification. Because of the high contrast edges and corners of



(a) Checkerboard (b) ARTtoolkit Markers (c) ARTag Markers

Figure 2.1: The most popular fiducial markers are based off of black squares, whose corners give point correspondences for geometric inference in pose estimation and camera calibration.

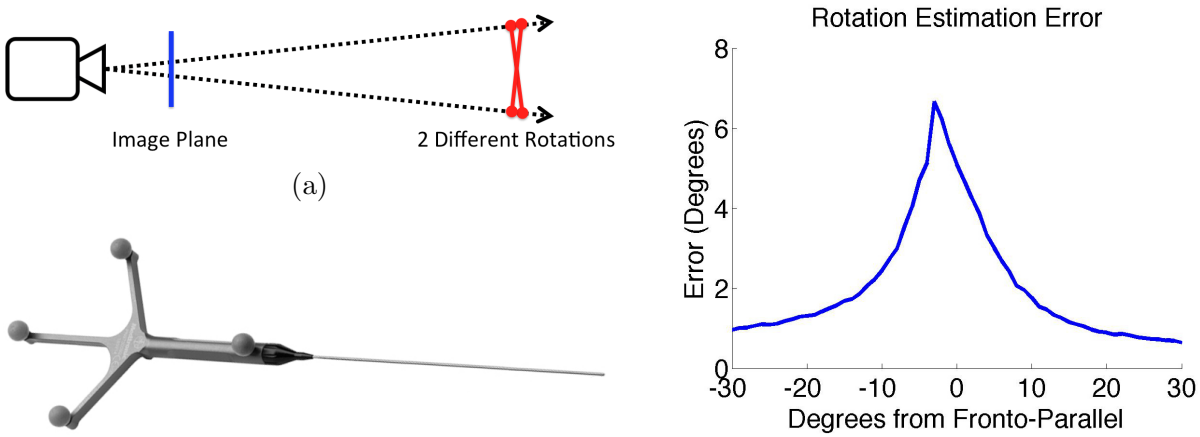
the square, line-fitting approaches can give sub-pixel image coordinates to the corners of the square. A black square on white background is advantageous as it retains its high contrast in various camera exposure settings. In this subsection, we highlight some of the more popular planar fiducial markers which rely on point correspondences from a black and white square design. Examples of some of these markers can be seen in Figure 2.1.

For pose estimation, many planar fiducial markers build off a simple black square to provide additional capabilities by including unique patterns inside the square. The unique interior patterns allows differentiation of several markers in an image. This interior pattern changes for different types of markers and provides different attributes and capabilities. ARtoolkit [28, 72] uses symbols and image matching to differentiate markers. To increase the possible number of uniquely identified markers, ARTag [14, 15] uses binary check-sums, with the added benefit of fault tolerance during image matching. Some work has been done to automatically create interior patterns which maximize the inter-marker code distance while minimizing the complexity of the pattern [19]. CALTag [3] is able to robustly handle occlusions by individually identifying markers similar to ARTag in a checkerboard and locating any missed calibration points using prior knowledge of the checkerboard layout. These black

and white planar markers are frequently used for pose estimation in Augmented Reality applications because they are easy to print with commodity printers.

The most popular fiducial marker for camera calibration is the planar checkerboard. As the checkerboard is comprised of many squares, this method shares the advantages of square fiducial markers, including easy fabrication, detection, and processing. In addition, many more point correspondences, and thus constraints, can be derived from the dense number of corners of the checkers, facilitating camera calibration. The checkerboard fiducial marker was made popular in Zhang's widely used camera calibration method [81], and continues to be widely used and supported, for example in unofficial [7] and official [39] MATLAB toolboxes and C++ libraries [8]. Despite the many correspondences that can be derived from a single image of a checkerboard, many multiple images are needed for single camera calibration because of the additional parameters for the camera, for example focal length. However, with 2 checkerboards on non-parallel planes, one can calibrate a camera from a single image [59].

Other research goes beyond the standard black and white planar fiducial markers to provide additional characteristics. Some markers have been developed to provide robustness to varying scale caused by significantly changing the distance of the camera from the fiducial marker. Fractal Marker Fields [24] use a fractal approach to ensure that features can be found at any scale. Similarly, Nested Markers [69] have a recursively layered structure of smaller elements that can be identified and used at larger scales. Other markers have been designed to be unnoticeable by the human eye. VRCodes [76] can create fiducial markers on active displays by using rolling-shutter cameras to detect signals that are presented by the display at a high frequency unperceivable by human vision.



(a) (b) (c)

Figure 2.2: Traditional fiducial markers that use point-correspondences suffer from a well-understood ambiguity for orientations near front-parallel. We show the intuition in a): points of 2 planes near fronto-parallel to the camera but rotated slightly differently will project to nearly the same locations on the image. c) As a result, orientation estimations have larger error for rotations near fronto-parallel views. To limit this effect, points are moved farther away, making potential applications cumbersome to handle, like the wand used in stereotactic surgery, shown in b).

The traditional black square markers, including instantiations presented above, have been widely used and deployed for applications in augmented reality, robotics, and manufacturing. However, with coplanar point correspondences, as is the case with all the above cited work, there are well-documented ambiguities for relative orientations close to fronto-parallel to the camera [59, 70, 1, 48]. Consider two images of a square that are nearly fronto-parallel to the camera, but differ by a 1 degree rotation around one local axis: the projection of the points on an image have approximately the same appearance. We visualize this intuition in Figure 2.2a. Hence, pose estimations, especially for rotation, are challenged with large error around fronto-parallel views. As shown in Figure 2.2c, this is especially pertinent for orientation estimations within 10 degrees of fronto-parallel. A way to mitigate these errors is by physically separating the fiducial points are far away as possible, such as the stereotactic surgery tool shown in Figure 2.2b. By deriving orientation constraints from lenticular arrays

and microlens arrays, our method does not face this traditional ambiguity and therefore our methods have consistent performance across all possible orientations of the fiducial marker. This is an important distinction and characteristic of our markers since most applications of pose estimation encounter fiducial markers that are fronto-parallel to the camera.

2.2 Geometric Inference without Fiducial Markers

There are many research works in the Photogrammetry and Computer Vision communities that cover geometric inference, particularly camera calibration, using various cues. We refer the reader to more comprehensive works for a full scope of work done [2, 83, 51, 18, 52]. In this section, we cover a few topics to give a sense of the variety of different techniques.

The Perspective-n-Point (PnP) problem tries to estimate the pose of a calibrated camera given n 3D points with known relative position and the corresponding n 2D points in an image. The version with 4 points, P4P, is the generalized algorithm to solve an object's pose based on cues using 4 point correspondences, for example the 4 corners of a square fiducial marker. The P3P problem has been shown to have at most 4 solutions [16, 21, 17], while the P2P and P1P problem have infinitely many solutions. However, with additional assumptions, one can constrain the solutions for PnP problems with $n \leq 4$, and thus the minimum number of point correspondences needed. In situations where the vertical direction is known for the camera (e.g. from an Inertial Measurement Unit), it is possible to solve for the pose of a camera with 2 corresponding points [32, 53]. Recent work also derives a two point solution if the observed points in the world have a known direction that projects to the image (e.g. building corners where the vertical edge is visible), and characterizes the degeneracies of these constraints [6].

The PnP problem assumes the configuration of 3D points in the world is known. Some work considers only partial information of the points lying on regular shapes in the physical world. Methods such as [42] use 5 point correspondences on 2 orthogonal 1D lines for single image camera calibration. Other methods use co-planar circles to calibrate a camera with a single image. One method [11] derives geometric constraints from conic surfaces. A similar work uses vanishing lines of a plane containing two circles in order to calibrate a camera [12]. These techniques leverage regular shapes that exist in the physical world, instead of fabricated markers.

There are some camera calibration approaches that are not based on identifying exactly corresponding points in a scene. Calibration patterns that consist of patches or parallel lines can be used for intrinsic camera calibration and extrinsic calibration (including rotation and translation relative to other cameras). Approaches that do not require correspondence between specific lines are based on seeing the orientation and spacing of those parallel lines on the image including those based on the prismatic line constraint [5, 4], and an approach using the Radon transform as a filtering operator [38, 37]. In some circumstances there are scenes that have large numbers of parallel lines with a known orientation, such as vertical edges of buildings or plumb-lines; the orientation and position of those lines in an image provide constraints for both intrinsic and extrinsic calibration even without matching the pixel to a specific line in the world [41, 6].

Other approaches have used textures for camera calibration. The authors of [82] assume that natural urban textures have low rank (for example, grids of windows on the exterior of a building). Using this assumption, the authors solve for the calibration and lens distortion that minimizes the rank of the textures in the image, using all pixels, not just point locations of geometric points.

Our approach of using microlens arrays as fiducial markers is interesting because we can leverage existing research done for point correspondence based calibration methods and the PnP problem. A microlens array can give point correspondence cues by identifying its location in the image. However, we can build on this by inferring orientation information at this point as well. Therefore, fiducial markers created from microlens arrays have a higher density of geometric constraints than methods listed in this section. In fact, by using microlens arrays, we show how to use a single image for pose estimation and camera calibration; in Chapter 4, only two different measurements of 2 lenticular markers in an image are needed for pose estimation; in Chapter 5, a single image of a lenticular array calibration object is required for camera calibration; and in Chapter 6, 1 microlens array is sufficient for pose estimation.

2.3 Fiducial Markers with Orientation Cues

A large majority of the Computer Vision literature around fiducial markers, pose estimation, and camera calibration relies on point correspondences to infer the geometry of the imaged physical world. However, there are a few prior works that have explicitly created fiducial markers whose relative appearance depends on the direction from which they are viewed. Agam markers [9, 10], inspired by the kinetic art of Yaacov Agam, are comprised of many adjacent, long triangular prisms. Shown in Figure 2.3, it is comprised of many triangular prisms, each colored white on one side, and black on the other. When viewed from afar, the darkness of this pattern relates to the angle at which the prisms are viewed. Some work has explored using Agam Fiducials and report how the markers change in appearance as the camera is moved in simulation and empirically [47, 46]. Another type of marker called

BoKodes [43], creates a highly structured pattern of light projected away from one point in a scene. This structured pattern is based on thousands of small QR-codes. When a defocused camera takes a picture of a scene, this pattern is visible, and the identity of QR-codes in view indicates the relative direction of the camera relative to the bokode marker.

Both the BoKode markers and the AGAM markers both vary appearance depending on orientation. However in both cases, the set of orientations is limited to a pencil of ray directions that intersect the BoKode marker or AGAM marker. The fiducial markers in this dissertation build on this previous work by varying appearance for a very large set of orientations: as much as 40 degrees from fronto-parallel.

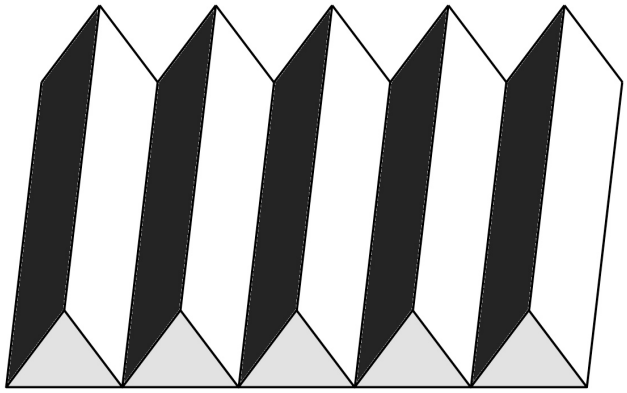


Figure 2.3: Agam fiducial markers are similar to lenticular arrays in that many small elements produce a similar appearance and thus create a macro scale appearance for a viewer. Agam Markers are constructed from many parallel, small triangular prism, where two sides of the prism are colored black and white. Therefore, at different viewpoints, different amounts of each side will show, resolving as a gray with varying intensity. Similar to lenticular arrays, this appearance could be conceivably used to infer the relative orientation of a camera.

2.4 Geometric Inference with Microlens Arrays

Micro-optic arrays have been used for their unique optical properties in a variety of ways. An early example includes the Integral Image" in 1908, when Gabriel Lippmann first proposed using lenticular arrays to capture and display an autostereoscopic image [36]. Since then, microlens arrays have been used to capture the full plenoptic representation of a scene with lightfield cameras [44, 35, 31, 49, 20] while lenticular arrays have been used

to create 3D TVs without the need of additional equipment like special glasses for users to wear [40, 25]. Microlens arrays have a wide variety of uses, but for this document, we review only the literature which uses microlens arrays for geometric inference.

One use of microlens arrays is to create light-field probes which use color to encode the relative path of a ray of light. As illustrated in Figure 2.4, this is done with a light field probe which encodes the 2D origin location and 2D direction of light rays by illuminating patterns on transparencies through a lenslet array. Light field probes have been used to detect the change of the index of refraction in transparent mediums for Schlieren photography [74] and to reconstruct the refractive index field of gas flows [26]. In addition, light field probes have been used to reconstruct the surface geometry of thin transparent objects [75]. Similar to these works, the BoKode [43], mentioned in Section 2.3, has been used to reconstruct dynamic 3D fluid surfaces [79].

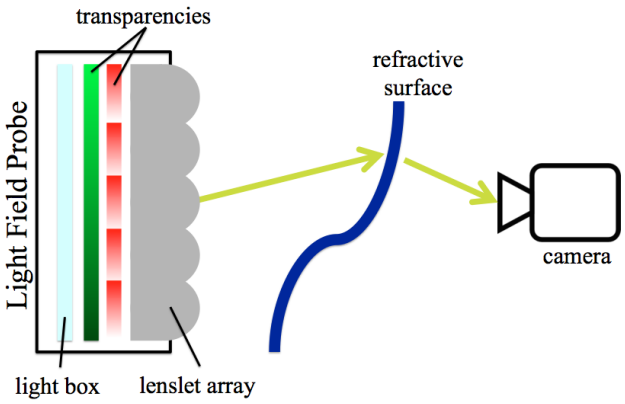


Figure 2.4: A light field probe is able to encode the 2D location and 2D orientation of a ray of light leaving the probe using the green and red color channels (respectively) of an image. As these colored rays of light refract through a transparent medium, they change direction which is detected by a calibrated camera.

The light field probe shares a core characteristic of the fiducial markers described in this dissertation: it encodes orientation using color. In fact, our fiducial markers can be described as generating a light field as well. The above work uses a light field to observe how these encoded rays change direction through a refractive medium. For this geometric inference, the location and orientation of the light field probe as well as the camera properties must be

known. The authors used a well calibrated system and use lightfield probes to understand the reflective and refractive object. We use similar objects, but in our work, we describe how to determine the optical properties, as well as the relative location of an uncalibrated camera.

2.5 Microlens Arrays as Fiducial Markers

The works most similar to this dissertation are a few cases were microlens arrays have been explicitly used as fiducial markers. Most recently, a patent from Sony [33] proposes using lenticular arrays to track head position. The patent proposes the concept of using lenticular arrays that vary in color based on the orientation and describes an application to pose estimation in general. This idea is very similar to the work presented in Chapters 4 and 5. However, our work goes on to describe the necessary constraints to determine pose from lenticular arrays and experimentally validate the concept.

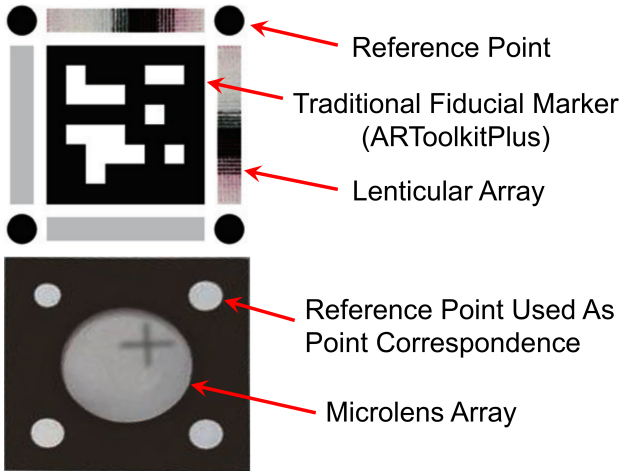


Figure 2.5: (Top) A LentiMark is comprised of a traditional fiducial marker with the addition of a lenticular array which shows orientation cues by the position of a black mark relative to reference points. (Bottom) Similarly, an ArrayMark is comprised of 4 reference points that are used as traditional point correspondences with the addition of a microlens array which gives orientation cues by the position of a black cross relative to the 4 point correspondences.

One group has used both lenticular arrays and microlens arrays to augment existing

traditional markers and 4-point correspondence methods. Shown in Figure 2.5, these markers, called Lentimark [64, 67] and Arraymark [62, 65, 66], change appearance based on orientation and have been used in applications including Augmented Reality [68, 63, 61], Robotics [29, 30, 71], and Motion Capture [60]. This appearance change manifests as a black mark or cross that translates relative to the rest of the fiducial marker. The relative translation of this black mark indicates the viewpoint orientation of the marker relative to the camera. This orientation information given by the lenticular array (of the Lentimark) or the lenslet array (of the Arraymark) improves orientation estimations of existing traditional fiducial markers, especially for fronto-parallel marker orientations. Because the orientation cues are found by comparing the relative position of points on a fiducial marker, however, they suffer from some of the challenges of traditional fiducial markers. In particular, as a Lentimark or Arraymark is moved farther away from the camera, the accuracy of the additional orientation information will decay. This is because as a Lentimark or Arraymark is moved farther away from the camera, there are fewer pixels to measure the relative location of the orientation mark on the lenticular array or microlens array relative to the rest of the marker. Therefore, errors in locating these landmarks result in faulty orientation cues. Our method also embraces visually encoding orientation by appearance, but builds on this by explicitly encoding orientation information via hue or grayscale intensity. As a result, our markers can be moved very far away and, as long as one pixel can image our fiducial markers, an image will still be able to capture strong orientation constraints from the fiducial markers.

2.6 Summary

The fiducial markers presented in this dissertation are related to other research in Computer Vision, especially in the topic of fiducial markers and tasks of pose estimation and camera calibration. Unlike traditional fiducial markers, fiducial markers made from microlens arrays do not suffer from pose ambiguity. That is because these markers explicitly encode orientation information by their appearance. Our work is similar in this way to other research that uses microlens arrays for general geometric inference. However, ours is the first to be designed and used for pose estimation and camera calibration. In addition, we derive geometric constraints on orientation purely based on appearance and show a working prototype and experimentally characterize performance.

Chapter 3

Color Coding Orientation with Lenticular Arrays

In this chapter, we show how lenticular arrays can be constructed to encode orientation with color. Lenticular arrays are sheets of plastic comprised of many small cylindrical surfaces. We discuss how the lenticular array can project patterns adhered to the back of the lenticular array to a create orientation specific appearances. We design a lenticular array that has different hue appearances and we provide details on how to fabricate a prototype. The resulting prototype can encode orientation by hue through a a 1-to-1 function. This calibrated relationship enables geometric inference for pose estimation in Chapter 4 and camera calibration in Chapter 5.

3.1 The Lenticular Array

Lenticular arrays are plastic sheets which are able to produce different appearances for different viewpoints in ambient light. Common children's toys use lenticular arrays to show different images to a viewer as the toy is rotated to give the illusion of an animated object [45]. Lenticular arrays can also be designed to show two images of the same scene undergoing parallax to two particular viewpoints. These two appearances are designed to be perceived by both eyes of a user to give the perception of depth without any external apparatus. This has been exploited in some modern TVs to bring 3D video into the home any extra equipment like special glasses [40, 25].

Physically, lenticular arrays are sheets of plastic and have a flat surface, called the back-plane, on one side, and a front surface comprised of many parallel rows of small half cylindrical surfaces, on the other. These cylinders, called lenticules, act as lenses and are designed such that the focal length is equal to the thickness of the lenticular array. Because of its repeating shape, a lenticular array can have an orientation. We define the direction parallel to the lenticules as the major axis and the direction coplanar but orthogonal to this, the minor axis. We show this geometry in Figure 3.1. A specific pattern can be designed such that when adhered

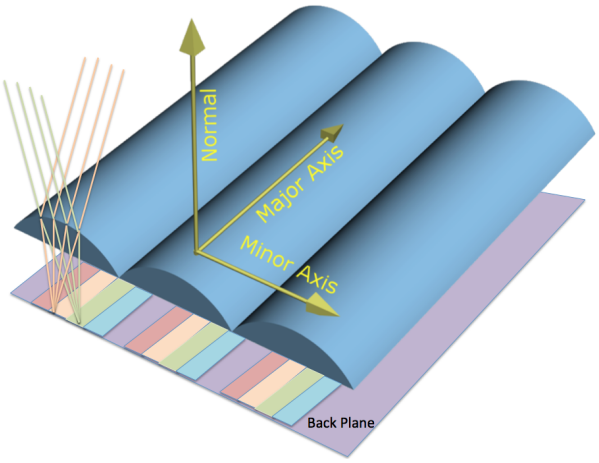


Figure 3.1: A lenticular array is comprised of a series of parallel cylindrical surfaces that focus parallel rays onto particular rows of a back-plane. The geometry of the lenticular array is described in terms of its major axis and minor axis.

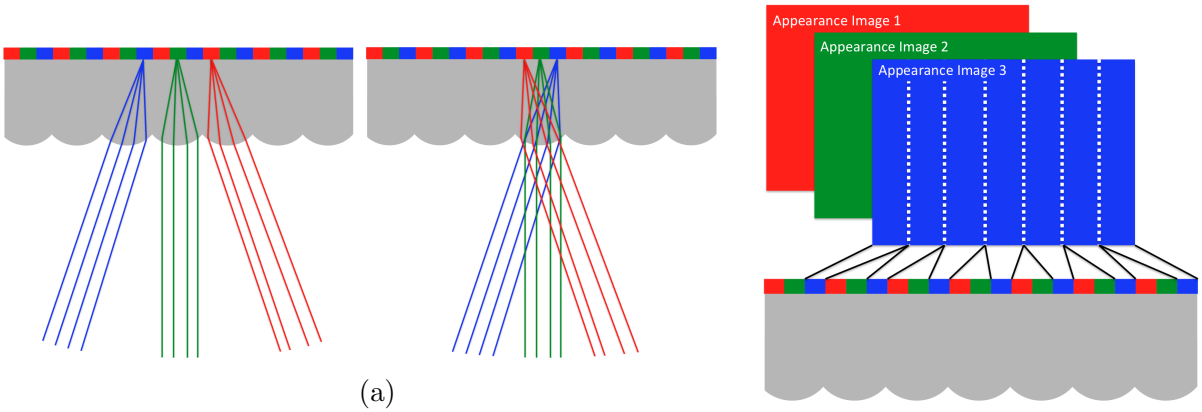


Figure 3.2: a) A cross-view of the lenticular array shows how lenticules focus parallel light onto the back plane. This focus point translates across the back-plane for different incident angles of parallel light. b) Therefore, to create different appearances, images of the desired appearance are chopped up spatially and interleaved behind each lenticule. At a particular incident angle, parallel light will focus on a single appearance image across all lenticules.

to the back-plane, the lenticular array will produce different appearances. We refer to this pattern as the back-plane texture.

Lenticular arrays are able to create different appearances for different viewpoints because the lenticules act as lenses which show the same local pattern adhered to the back of the lenticular array. The lenticules focus parallel rays of light from a specific direction onto the back-plane. Figure 3.2a shows this happening in a cross section of a lenticular array. However, whereas a common spherical lens would focus entering parallel light onto a single point, each lenticule focuses parallel light onto a line along the major axis. As the parallel rays of light change angle incident to the lenticules, the focus line translates across the back-plane along the direction of the minor axis. The focus line only translates for views that change by rotating around the major axis. One could also think of the lenticules as mini projectors which display some pattern along a line on the back-plane texture. Then,

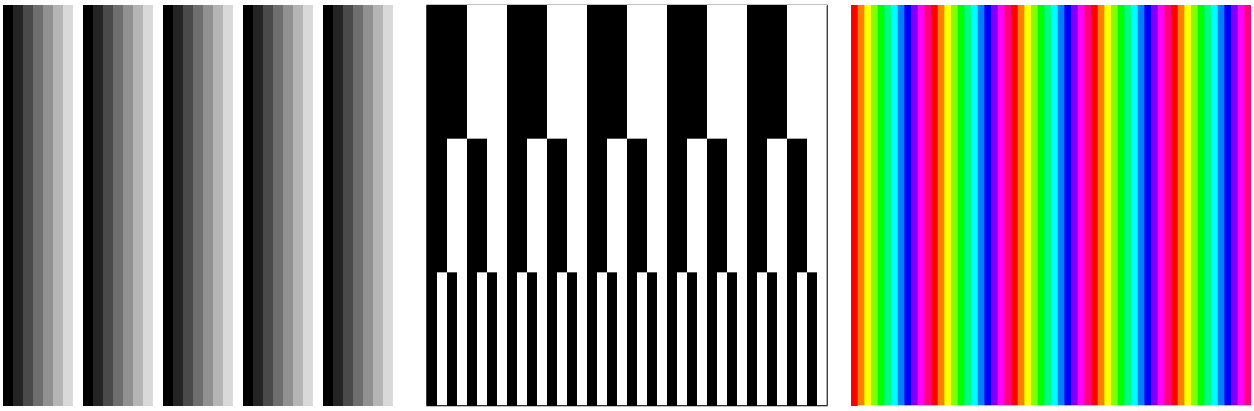
whatever local pattern is repeated under every lenticule on the back-plane texture will be projected as various appearances out the lenticular array.

The back-plane texture, thus, is created by interleaving a series of images of specific appearances. What this means is that each image is spatially segmented into strips and the same spatial strip from each image is under each lenticule. We demonstrate this in Figure 3.2b. Thus, for a viewer at a given orientation, all lenticules focus onto strips of a single frame to give the viewer the macro appearance of that appearance image. In the case of children's toys that give the illusion of an animation, a series of frames of an animation are interleaved to create the back-plane texture. Because the back-plane texture dictates what the set of appearances a lenticular array will create, the back-plane texture defines the appearance of the lenticular array.

3.2 Encoding Orientation with Lenticular Arrays

In this section, we leverage these properties to create a lenticular array that can encode orientation with specific appearance. We can design a set of known appearance images and interleave them to create a back-plane texture. This would allow us to design and calibrate a visual encoding of the incident angle of a camera relative to the lenticular array. Then, for images of the lenticular array, we could infer the orientation of the lenticular array relative to the camera using its appearance.

The orientation of the lenticular array could be encoded in a number of visual ways. Images or patterns with high details, like the animations of a common child's toy, would be difficult to use as this would require a process just to detect an appearance. Instead, some solid



(a) (b) (c)

Figure 3.3: This figure shows different back-plane textures that could be used to encode orientation from an appearance for a lenticular array with 5 lenticules. Similar to the Agam markers, we could interleave images of various darkness to encode orientation with grayscale. b) An alternative, could be to use a binary coding across one array, or many arrays. c) We choose to use interleaved hues because hue appearance is invariant to scene light or camera exposure settings. These pictures are dramatically enlarged; when placed under the lenticular array each of the five repetitions of this color spectrum is scaled to fit under one lenticule, which is typically < 1 mm wide.

appearance would be easier to detect and measure. Following the Agam fiducial marker [9], we could interleave different shades of gray (shown in Figure 3.3a). Then, we could relate orientation to the grayscale value measured from the lenticular array. Grayscale, however, is variant to light intensity in the scene and exposure settings of a camera. As such, the appearance of the lenticular array with such a back-plane texture could vary drastically for the same viewpoint, for example by increasing the exposure time of the image. One way to address this could be to use just black and white appearances to encode orientation. Shown in Figure 3.3b, one could design a back-plane texture with some binary encoding that has various frequencies of black and white stripes. This would require having a higher spatial understanding of where measurements are coming from on the lenticular array, or would require different frequencies to be divided among multiple lenticular arrays. Either of these strategies would complicate any inference using an image of one or more lenticular

arrays. Instead of using a grayscale appearance, we could use a hue appearance to encode orientation. Hue is invariant to the amount of light that is captured by the camera, and so would be a better orientation cue.

We thus propose to use hue to encode orientation with a lenticular array. Ideally, we would like to encode for a wide range of orientations, while still having high contrast for appearances from similar orientations. A binary encoding of two different hues could be used, but as described above, would suffer from complexities in identifying different regions of a lenticular array or require many lenticular arrays. A better approach would be to use appearances of the different colors of the saturated hue color wheel to encode orientation. With this approach, the back-plane texture would have a discrete sampling of the hue color wheel under each lenticule. We show this in Figure 3.3c. By using the whole color wheel, we maximize the number of orientations that could be encoded by different colors. In addition, because the set of hues is continuous, the different appearances will be change continuously as the lenticular array is rotated.

3.3 Chromo-coding Lenticular Array

By using a back-plane texture of a continuous set of hues, we can create a lenticular array that explicitly changes hue appearance smoothly as the lenticular array is rotated around its major axis. Therefore, this lenticular array encodes the orientation of the array relative to a viewer or camera with hue. Because of this, we say that the lenticular array is chromo-coding its orientation. Unless otherwise stated, we will will assume that all lenticular arrays discussed in this dissertation have this quality. The lenticular array is unlike most imaging

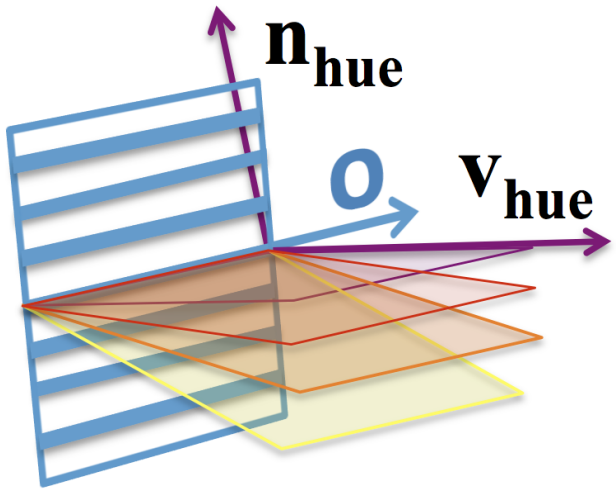


Figure 3.4: Here we detail the chromo-coded lightfield produced from a lenticular array. We define the major orientation to be the direction along the lenticular lenses. Depending on the angle at which the pattern is viewed, it appears to have a different hue. For notation, we define the major axis of the lenticular pattern to be $\vec{o}$, and an example ray that would be seen for each hue as $\vec{v}_{hue}$. Any ray that views the lenticular sheet in the plane spanned by $\vec{o}$ and $\vec{v}_{hue}$ has the same color, our derivations are written most simply in terms of $\vec{n}_{hue}$, surface normal to that plane.

situations; for example, in Lambertian scenes, all rays coming from a given location are assumed to have the same color and intensity.

Relative to the lenticular array, the lenticular array will have different hue appearances for different directions to a viewer. Because of the geometry of the lenticular array, there will be different appearances for different viewpoint directions that rotate around the major axis. Then, all the viewing directions that have a particular hue will lie on a plane that is parallel to the major axis. Therefore, all the set of views that have a particular hue will lie on one of the respective hue planes that radiate out of the major axis.

Figure 3.4 visualizes this geometry. In the coordinate system of the lenticular array, $\vec{o}$ is the direction of major axis, $\vec{v}_{hue}$ is an example ray observed at a given color and $\vec{n}_{hue} = \vec{v}_{hue} \times \vec{o}$ is

the surface normal to the plane containing all rays of that hue. This constraint is leveraged in Chapters 4 and 5 for geometric inference.

3.4 Manufacture of the Chromo-coding Lenticular Array

In this section, we introduce the physical chromo-coding lenticular array prototype that is used in this dissertation. The prototype is based on the EcoLens Lenticular Array, purchased from Pacur, Inc. This lenticular array has a design that includes lenticules that are elliptical instead of cylindrical, eliminating some of the optical aberrations that happen at each lenticule [27]. We use arrays where each lenticule was 0.34 mm wide, and the thickness of the array was 0.4 mm.

To create orientation dependent hue appearance, we use a back-plane texture that includes 12 discrete samples of the hue, sampled every 30 degrees around a color wheel. Our pattern is shown in Figure 3.3c. A pattern with 12 colors aligned along the major axis every 0.34 mm requires printing a texture at $\approx$ 900 dpi to get a stripe of each color 1 dot wide. This back-plane texture is adhered to the back of a blank lenticular array with optical adhesive in order to make the final object. Due to an exclusive availability of blank EcoLens lenticular arrays, the back-plane texture was printed and mounted by professionals at Pixalen Studio.

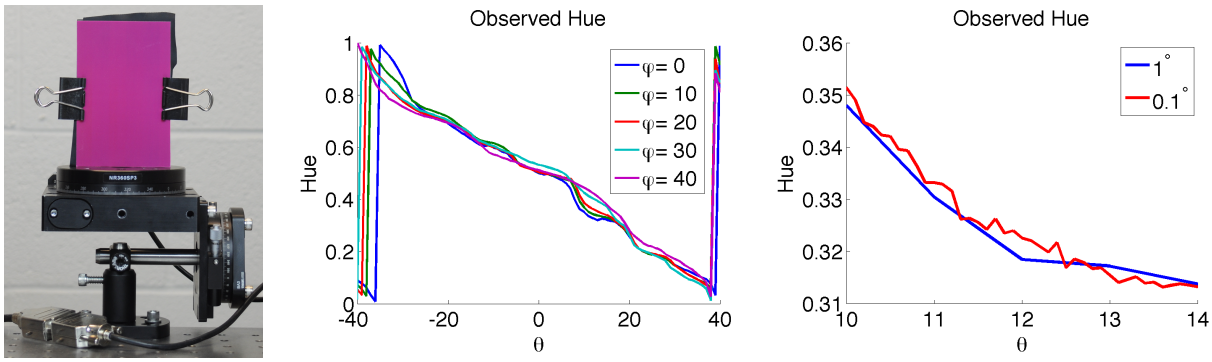
Blank lenticular arrays are cheap commodity items and the appropriate back-plane textures can be printed on commodity printers, so this prototype is broadly feasible to manufacture even in application domains where cost is a factor.

3.5 Hue Response Function

The relationship of the apparent viewing direction to the observed hue depends on the orientation of the lenticular array relative to the camera, in particular the rotation angle of the viewing direction around the major axis of the lenticular array. Therefore, the appearance of the chromo-coded lightfield to an observer only depends on this single rotation parameter, which we indicate as θ. The relationship between this rotation and observed color is captured in the Hue Response Function (HRF), which is a 1-to-1 relationship for incident angles, or θ, of up to $\approx$ 40 degrees (after which the colors repeat). In this section, we explore this relationship. The HRF is useful for geometric inference because it defines the relative orientation of a viewpoint given a measured hue.

Using a chromo-coded lenticular array described in Section 3.3, we test the hypothesis that the hue appearance of the lenticular array only depends only on the rotation around the major axis of the array. We measure the apparent hue of the lenticular array at various orientations using two motorized rotation stages from ThorLabs. These motors are setup to move the lenticular array in a controlled way around its major and minor axes. Images were captured with a Nikon D60, with a 300mm lens to minimize perspective effects. In addition, camera settings were set at ISO 100 and a static white balance level to ensure consistent and saturated color measurements.

In the first experiment, we rotate the mounted lenticular array around its major axis at 1 degree increments ($\theta \in \{-40^\circ, \ldots, 40^\circ\}$) and image the pattern at each angle. This set of angles is repeated with the lenticular array tilted at 5 rotations around its minor axis ($\phi \in \{0^\circ, 10^\circ, 20^\circ, 30^\circ, 40^\circ\}$). Figure 3.5a shows the setup for the experiment: a large lenticular



(a) (b) (c)

Figure 3.5: (a) We experimentally measure the HRF in the laboratory using a precision rotation controller. The observed hue (b) has a nearly linear response as a function of rotation around the major axis θ, even for different rotations around the minor axis (shown for 5 different angles ϕ). The drastic jumps in hue at wide angles of θ are a consequence of the design of our BFP texture. (c) The HRF resolution with our current prototype suggests that accuracy is possible to about $\frac{1}{4}$ to $\frac{1}{2}$ of a degree. At smaller scales, noise in the measurement process makes the hue/angle function non-monotonic.

array was mounted square on a right angled bracket attached to a motorized stage to vary θ, while another motorized stage could be tilted to vary ϕ.

We take a series of images where the lenticular array is centered in the image. For each image, we average a small 20 pixel region (5 lenticules wide) at the center of the image/lenticular array to capture a representative hue value for each viewpoint. Because we sample at the center of the image, we eliminate any perspective that would change the angle of incidence of rays from the camera imaging the lenticular array. Figure 3.5b shows that for θ in the range of $[-35, 35]$, all five sets of images show an approximately linear HRF giving a 1-to-1 relationship between the hue and the angle rotated around the major axis. Furthermore, the HRF does not vary greatly between different rotations around the minor axis (ϕ). This shows that it is appropriate to model the chromo-coded lightfield by a single rotation parameter via the HRF.

For values of θ outside the range of $[-40, 40]$, the optics of the lenticular array focus the rays hitting one lenticule on the back-focal plane texture of the adjacent lenticule. Since the hue spectrum is repeated under each lenticule on the BFP texture, the hue appearance of the lenticular array rolls over on the hue color wheel (and jumps from 0 to 1 on the standard mapping of hues to numbers from 0 to 1).

The next experiment explores the question: how small of angular changes can be differentiated with an HRF? Here we set $\phi = 0$ and change θ in increments of $\frac{1}{10}^\circ$. Figure 3.5c shows the observed hues for θ values between 10 and 14. The blue line shows a previous measurement series measured at each integer degree, and the more jagged red line shows a different set of measurements done every $\frac{1}{10}^\circ$. The measurements done every $\frac{1}{10}^\circ$ have some non-monotonic behavior that suggests that the HRF would be useful to predict orientation angle only up to $\frac{1}{4}$ to $\frac{1}{2}^\circ$. Since our input hues are from averaging the hues in a small region of the lenticular array, this error may be from camera noise in precisely measuring color.

In Chapters 4 and 5, the HRF of a lenticular array is leveraged for geometric inference. To do this, first the HRF must be pre-calibrated using known rotations around the major axis, using, for example, the motorized rotation method mentioned above. We have shown that the chromo-coded light field produced by our designed lenticular array can be well modeled by the HRF. Thus, two 1-to-1 functions can be fit in order to do hue $\rightarrow \theta$ or $\theta \rightarrow$ hue lookups. Then, given an image with an unknown relative orientation of the lenticular array to the camera, we could lookup the θ to constrain a direct estimation, or lookup the hue to fit a model for estimation.

The remainder of this chapter explores the challenges and limits of using an HRF to determine orientation from hue appearance.

3.6 Challenges

As we have seen, chromo-coding lenticular arrays can give useful cues on orientation relative to a camera. However, this method has a few challenges. Strong directional light and colored ambient light can change the appearance of the lenticular arrays for some or all viewpoints. In addition, the small size of the lenticules of the array makes it difficult to fabricate a back-plane texture that creates a consistent HRF across the lenticular array. Finally, the fact that cameras make discrete RGB measurements of hue limits the precision of encoding orientation with hue. In this section, we discuss these challenges.

3.6.1 Scene Light Color

The appearance of an object is the product of the spectrum of the albedo, or light independent color of an object, and the spectrum of the light illuminating the scene. The same is true for a lenticular array: the observed hue appearance will depend on what hues were printed on the back-plane texture, as well as what color light is illuminating the back-plane texture of the lenticular array.

For extreme light environments where only a narrow band of color is illuminating the scene, our method will only be able to encode a few orientations. To explain with an example, consider an environmental light that is pure blue. The only color that will reflect off of the back-plane texture will be blue. Therefore, only a small range of correct viewpoints will reflect blue light off of the blue section of the back-plane texture to create a blue appearance. Other viewpoints will be very dark, and if measurable, will erroneously appear to be blue.

This means that the lenticular array will no longer be able to reliably encode a large useful set of viewpoints.

For the more typical case, the spectrum of environmental light is fairly uniform, but has a shift in power towards certain colors. For example, incandescent lights have more power in the higher wavelength of the spectrum and so appear yellowish. In this case, there is a broad enough spectrum to reflect all the hues from the back-plane texture, but the higher power colors will cause higher reflectivity in the back-plane albedo matching and near the peaked color. So in an environment with incandescent light, the lenticular array will be yellowish for more sets of orientations. This sort of stretching of the set of reflected hues from the lenticular array can be handled by calibrating the HRF in this specific environment. In Section 4.4.3, we introduce an inference-time color-calibration strategy to address the case where the HRF may be calibrated in one lighting environment and moved to another.

3.6.2 Directed Light

Our lenticular arrays depend on ambient light to illuminate the back-plane texture. In effect, scattered light from many different directions enters the lenticular array and illuminates the back-plane texture evenly. Then, as the rays of light reflect off of a given hue strip, some rays collimate as they exit the lenticule and are observed by a viewer. The rays from different hues will collimate at different angles around the major axis, allowing the lenticular array to encode orientation by hue.

If a strong collinear light, for example a desk lamp, is the only light source illuminating the lenticular array, the lenticules will focus the collinear light only onto specific locations and hues on the back-plane texture. As a result, only the views that are in plane with

the collimated light source will see a strong hue appearance. Other viewpoints may have a different hue appearance due to lenticule imperfections and internal light scattering, but the appearance will be very dark. In most environments, this isn't a challenge, however when it is, this problem is easily addressed with a diffuse back-light attached behind the back-plane texture aiming towards the cylindrical side of the lenticular array. This solution would also enable the use of lenticular arrays in dark scenes or at night. Because the solution requires only simple engineering, we do not explore this problem or solution.

3.6.3 Back-plane Texture Fabrication Challenges

The process of creating a lenticular array that encodes orientation with hue involves a few simple steps. First, the back-plane texture must be designed to match the geometry of a particular lenticular array. Second, this back-plane texture is printed and attached to the back of that blank lenticular array. However, because of the size of lenticules of lenticular arrays, one of the largest challenges is ensuring that hues are precisely repeating under each lenticule. This is a challenge that spans the design and printing of the back-plane texture to the array.

The largest challenge in designing the back-plane texture is matching the printing resolution of an image to the physical properties of the lenticular array. Consider the case of trying to print 12 discrete hues under each lenticule that is 0.34 mm wide. If each hue was a pixel, then the image of the back-plane texture would have to be printed at 896.47 dpi to ensure that the same set of hues were under each lenticule. This resolution is a property of the back-plane texture image, and may not match the native resolution of a printer. If the resolution was rounded or the texture was re-sampled, it would not be guaranteed that

each color is precisely repeated under each lenticule. As an example, if we were to round the resolution to 897 dpi, this means that we fit $\frac{1}{2}$ of a discrete color strip into the same unit length (an inch). In the scale of the lenticular array, this means 1 extra strip of color every ≈ 5 cm. For a ≈ 5 cm wide (along the minor axis) large lenticular array, this means that a specific incident viewpoint angle will produce two hue appearances at the ends of the lenticular array that differ by $\approx 8\%$ of the hue wheel and the consequential gradual hue shift in-between.

Even despite this challenge in matching resolution, the physical process of printing presents other challenges in creating the intended back-plane texture. First, it is unclear what kind of re-sampling or color manipulations a printer does before physically printing a given image. Despite precise resolutions being indicated in an image file, printers may up-sample or down-sample the back-plane texture image file to match the machine's representation as a dot matrix. Second, as paper is fed through the printer, it is unclear how precisely paper is moved or held stationary. For a back-plane texture with 12 hues under a 0.34 mm lenticule, this means that discrete color strips are ≈ 28 microns wide. Even a small error in the papers intended position could introduce a shift in the order of hue strips and drastically alter the appearance of the lenticular array.

The small size of lenticules presents challenges in fabricating a back-plane texture. In the case of making small lenticular markers, these challenges are addressed by individually calibrating each marker. For the case of using a large array, the tolerances required to print the back-plane texture consistently across the whole array were beyond our ability to overcome and beyond the ability of the professional printers at Pixalen Studio, so in Section 3.7 we derive a way to solve for a calibration that varies over the size of the array.

3.6.4 Hue Measurements

The final limitation in the lenticular array arises because the measurement is an RGB color. The HRF defines a map from an angle to a color. We consider the inverse case where we are trying to infer the orientation of a lenticular array given an RGB measurement. In that case, we are mapping a color to an angle. This is an interesting mapping because 1) the set of fully saturated hues (which comprise the back-plane texture) is smaller than the set of RGB colors, and 2) the set of RGB colors in images is finite. Therefore, there is a limit on the angular accuracy that is possible from the hue of the RGB measurement. We present a theoretical exploration of these limits in Section 3.8.

3.7 Addressing an Inconsistent HRF

In the previous section, we introduced the challenge of creating a lenticular array that produces a consistent HRF across the entire lenticular array. In this section, we experimentally document this shift in the HRF in our most professionally produced prototype. We then offer an approach to a spatially varying calibration and experimental evaluation.

In this section, we demonstrate this challenge, introduce a simple remedy, and then test the remedy's performance to give a limit on angular precision of a lenticular array.

3.7.1 Inconsistent HRF Across the Lenticular Array

For a lenticular array created as described in Section 3.3, we found that the relationship between hue appearance and orientation varied across the array. We demonstrate this in

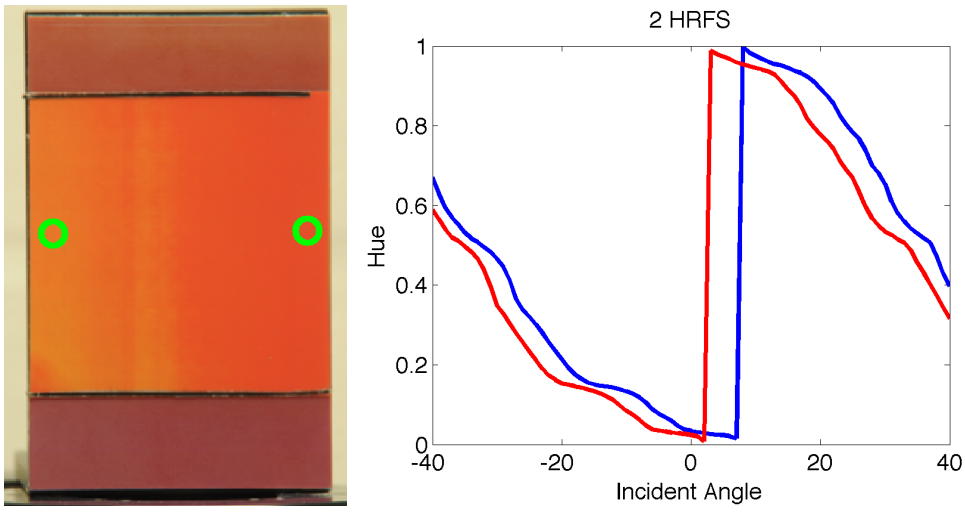


Figure 3.6: (Left) an image of the calibration object taken with a very long lens, observing all parts of the calibration array with a nearly orthographic imaging geometry. The observed color differences indicate the the hue for a given viewing direction is not consistent. (Right) The observed hue measured at the two yellow circles as the calibration object is rotated, showing a consistent bias.

Figure 3.6, which shows a picture of a lenticular array taken from 1 meter away with a 300mm zoom lens, giving a field of view in this picture of $\approx$ 1 degree. Given the lack of perspective, one would expect the hue appearance at these two locations to be very similar. However, one can visually see a different in the hue of the array at opposite sides, and when measuring the HRF at these two locations by rotating the lenticular array, we see a consistent shift. This is due to the challenges of manufacturing and printing the back-plane texture introduced in Section 3.6.3. The observed color shift in Figure 3.6 is explained by a 0.1 mm stretch over the width of the array ($\approx$ 10cm).

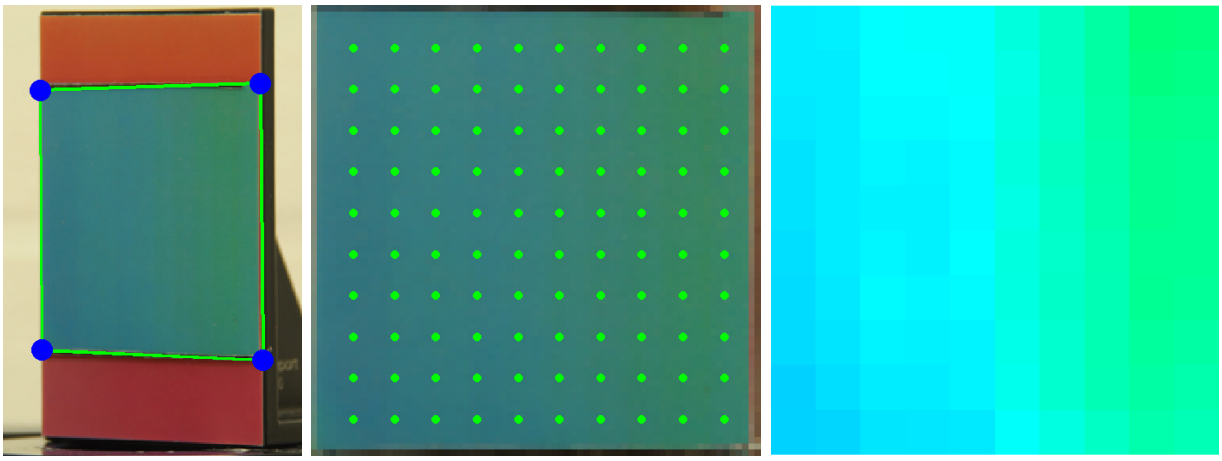


Figure 3.7: To take hue measurements for calibration and estimation, we employ the following strategy: With the original image (left), we identify anchor points, shown as blue points, at the corners of the lenticular array. These points are used to learn a homography (center) where we then take local measurements evenly across the array, shown as green dots. (Right) a simplified image of the hue recorded at each calibration point.

3.7.2 Multiple HRFs per Array

Because even professionally printed HRFs had manufacturing errors, we try a calibration approach to correct for these errors. We explicitly calibrate the HRF at regular intervals in the local reference frame of the arrays. Similar to the method above, the lenticular array is placed on a controlled rotation mount and rotated through 1 degree increments. For each calibration point, we record the angle at which that calibration point is viewed (which may vary across the calibration grid because of perspective effects), and the measured hue for that angle. The result of this is a HRF like those shown in Figure 3.6 for each of the calibration points. With this process, we have measured and pre-calibrated a grid of HRFs across the entire array. Although the HRF may be drastically different on opposite sides of the array, in the local neighborhood, HRFs may be only slightly different.

Because we create location specific HRF models, we need to know where in the array a hue value is measured, both in the aclibration phase and when used for inference. For this purpose, we use the corners of a rectangular lenticular array as anchor points, and for each image compute a homography that maps the observed lenticular appearance to a canonical coordinate system. The process is illustrated in Figure 3.7. This approach may be especially sensitive to estimating the position of the corners of the calibration object. We evaluate this by rotating the lenticular array around the vertical axis in one degree increments from -35 to 35 degrees. For each image, we follow these steps:

1. determine 4 anchor points of the lenticular array,
2. project the lenticular array into a canonical reference frame via a homography
3. sample the hue from the grid-points of the local reference frame image.

For each grid point we compute the angle at which the point was viewed to the angle predicted by the measured hue. To estimate the effect of noise in estimating the lenticular array corners, we perturb the anchor points by 8 pixels in random directions 20 times per image and follow the above procedure. We show the scale of one such perturbation in Figure 3.8. Using previously calibrated HRF look-up tables at each of the grid points, we estimate the angle θ using hue measurements at the perturbed locations caused by the perturbed anchor points. We compare this angle against the true viewpoint angle to determine the robustness of the grid of HRFs.

Figure 3.9 shows results. The top shows a box and whisker plot showing the distributions of errors in estimating the angle for each of the 100 grid points where the HRF was calculated. The box in each column shows the 25th and 75th percentiles of the distribution. This

experiment shows that modeling the HRF at each location of the lenticular array leads to nearly all angular measurements being within 0.25 degrees of the true incident angle. The lenticular array shows colors across the hue spectrum over a range of about 40°, so 0.25° is less than 1% of the range of angles that are viewed.

We also evaluate if the errors in estimating angle from hue depend on the angle at which the calibration object is observed. Figure 3.9 computes the distribution of errors across the entire array for each image angle. Again the error is consistently small, even though these statistics are computed using anchor points that are substantially perturbed.

Figure 3.8: In Section 3.7.2, we explored how the anchor point localization affected HRF prediction accuracy. In that experiment, we randomly moved the 4 anchor points (blue) of a lenticular array by 8 pixels in a random direction (green).

3.8 Orientation Encoding Limits

We choose to encode orientation with hue because hue is invariant to illumination intensity or camera exposure settings. However, the saturation and value of the appearance is not invariant. In this section, we explore how the saturation and value of the lenticular array appearance affects the theoretical angular measurement precision of predicting θ given a hue.

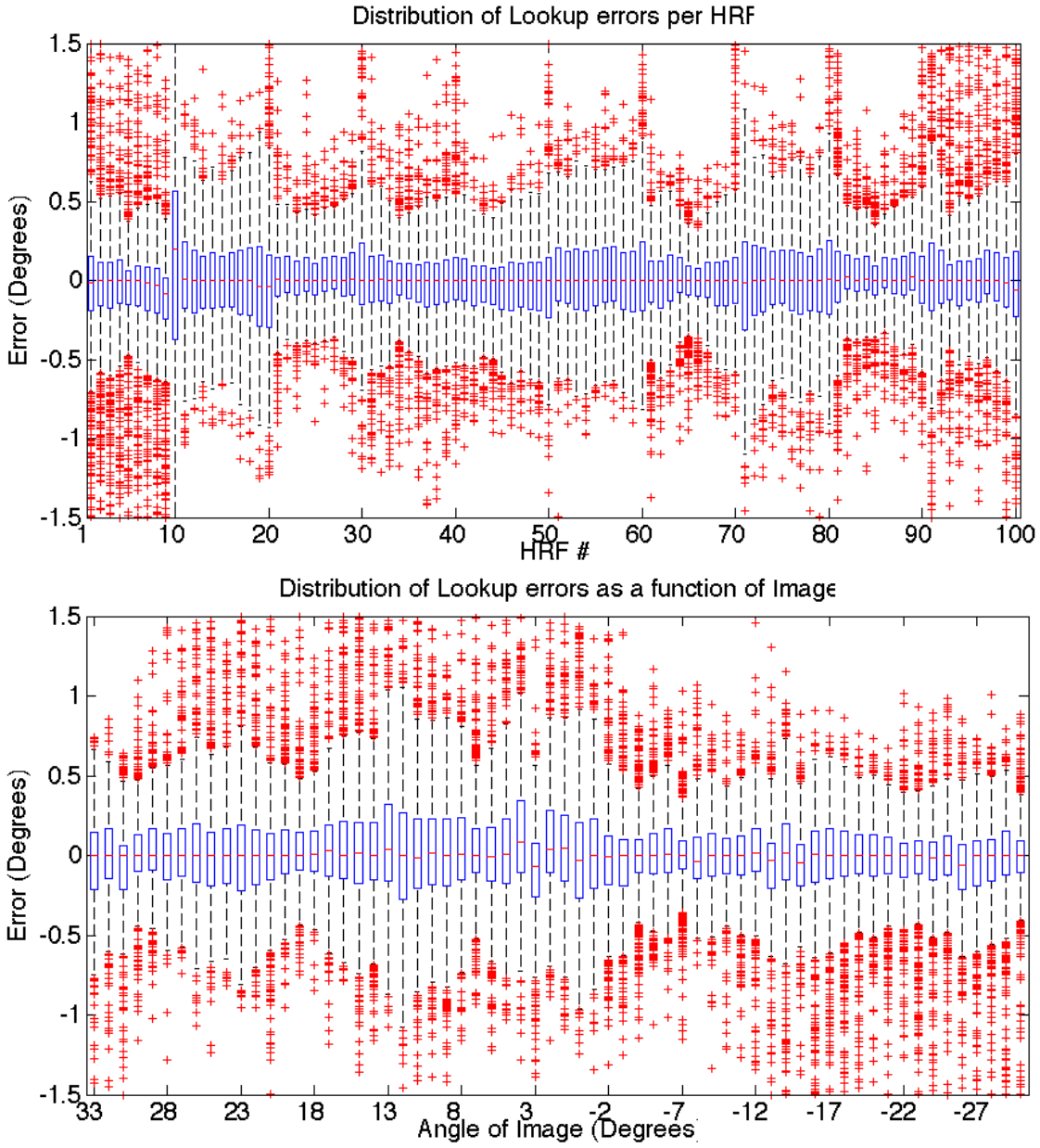


Figure 3.9: To create location specific HRF functions that map measured hues to angular constraints, we sample hues of a lenticular array at a local grid of points. The location of these points depends on localizing the anchor points at the corners of the lenticular array. We show the small prediction errors for 8 px permutations of these anchor points per HRF (top) and per image (bottom).

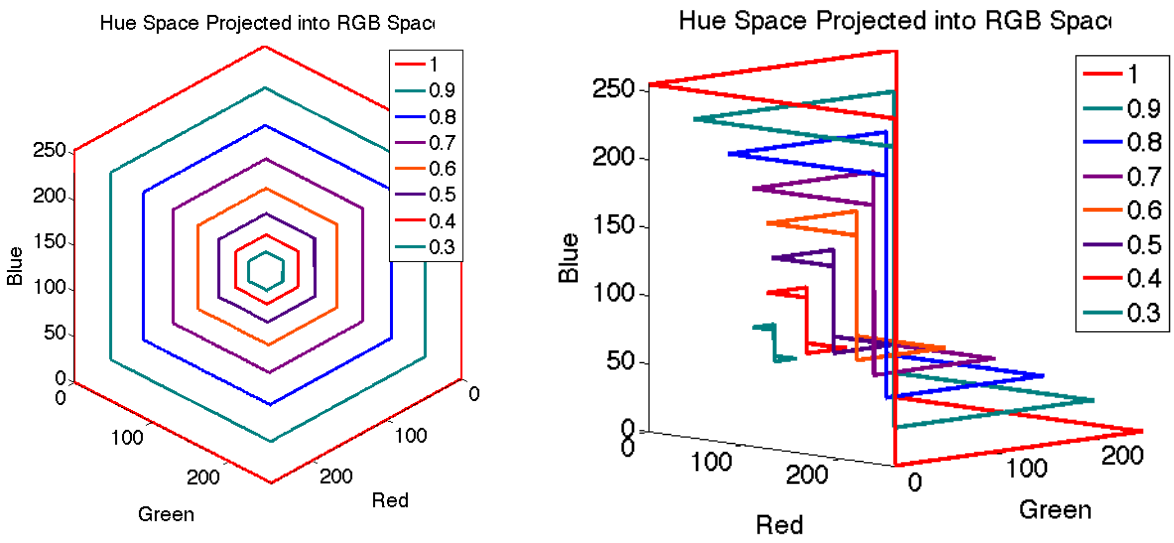


Figure 3.10: The space of hues with a constant saturation and value lies on a 2D curve in RGB space. This curve follows a few corners of a regular cube. Above we show 2 views of the same plot to show the shape of the this curve. As the amount of light reduces, or the saturation and value of HSV measurements reduces, the measurable space in RGB gets smaller as well.

At first pass, the measured precision may be limited by the number of encodable values of an 8-bit RGB image. The HRF maps a continuous set of angles to a continuous set of hues. However, the measurement of hues is limited to the discrete set of 8-bit numbers in the RGB image. Therefore, the precision of angular measurement would be governed by the ability to measure a change of color in RGB space. For example, if we tried to measure 30 degrees of angular change with just the red channel, we could theoretically measure an angular change of $\frac{30^\circ}{2^8} \approx 0.12^\circ$.

However, the set of RGB values does not map perfectly to the set of HSV values. And perhaps more importantly, since we are printing hues on the back-plane in full saturation and value, the set of appearances of the lenticular array (the hue wheel) does not map to the whole RGB space. In fact, the set of RGB values corresponding to the hue wheel for

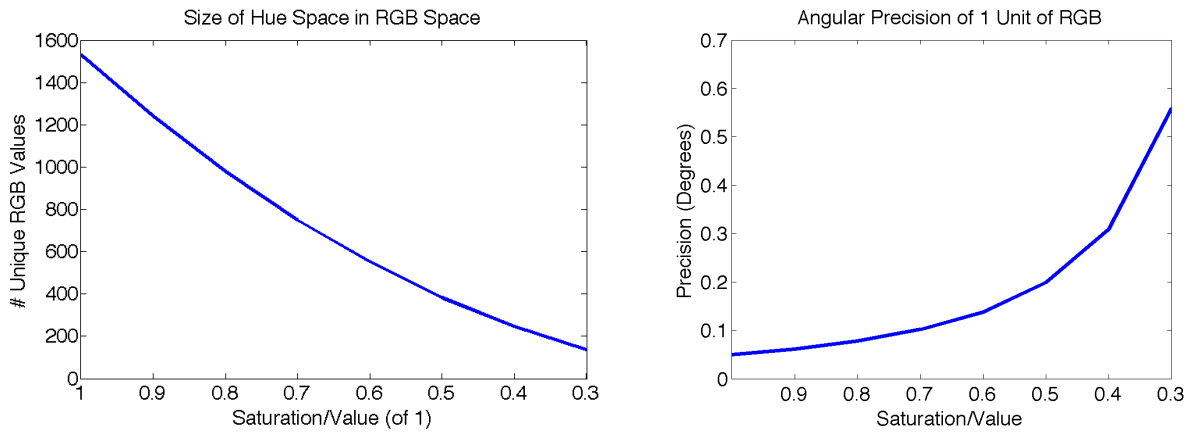


Figure 3.11: As the amount of light in a scene decreases, so too does the space of possible measurements of the color of a lenticular array. On the left, we show the number of unique RGB measurements for an 8-bit camera as a function of the saturation and value. On the right, we show how this translates to angular precision. An 8-bit camera has a maximum theoretical precision of 0.05 degrees, but would more realistically be limited to 0.1 degrees for moderately illuminated scenes.

a given saturation and value level lie on a 2D curve in RGB space. This curve represents the set of RGB measurements a camera would take of a lenticular array at any orientation. In Figure 3.10, we show two views of this RGB sub-space for various levels of saturation and value. The 2D curve is the shape of a cycle along the sides of a cube aligned with the RGB space axes. As the intensity (saturation and value) goes down, the set of RGB values corresponding to the hue wheel gets visually smaller. Since RGB measured values are discrete, this means that the number of measurable orientations is smaller as well.

The theoretical limit of angular precision afforded by the HRF is thus the number of uniquely encodable discrete hue measurements in this RGB sub-space. On the left of Figure 3.11, we show the number of unique 8-bit RGB triplets for each 2D curve, for each saturation and value level. Each RGB triplet maps to one angle in the 76 degree range of the lenticular array. So, in the best case of a linear HRF, we expect an angular precision of $\frac{76}{\#\text{ of discrete RGB triplets}}$. On the right of Figure 3.11, we show this angular precision for each level of saturation and

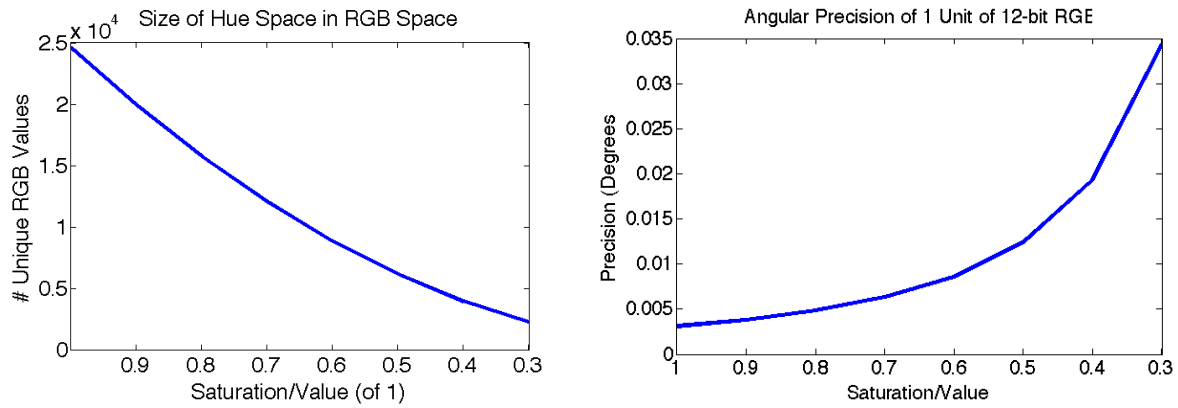


Figure 3.12: We repeated the experiment depicted in Figure 3.11, but for a camera that can capture 12-bit RGB images. As the camera can now measure color with 16 times more values, we achieve an order of magnitude more of uniquely measurable hues in RGB space (left) and therefore an order of magnitude smaller angular precision.

value. In the best possible case with maximum saturation and value, an 8-bit RGB camera is able to uniquely encode the angle of a lenticular array at a precision of 0.05 degrees. However, at a saturation and value level of 0.3, the precision drops to 0.55 degrees. In practice, we notice appearance measurements to have a saturation and value of around 0.7 or 0.8 of a maximum of 1 and so would expect a theoretical max precision of ≈ 0.1 degrees.

A natural way to increase the angular precision achievable by a camera, then, can be to move to a larger color representation. In Figure 3.12, we show the same experiment as before, but for 12-bit measurements. With 16 times more possible values for a single color channel versus an 8-bit camera, the number of unique RGB values for the color wheel and the angular precision both improve by an order of magnitude.

As mentioned in Section 3.6.1, the final appearance of the lenticular array depends on the printed color of the back-plane texture and the illuminating ambient light color. In the theoretical results presented above, we do not disambiguate these two. We also assume that there is no noise in the measurements, and we do not consider the potential to average

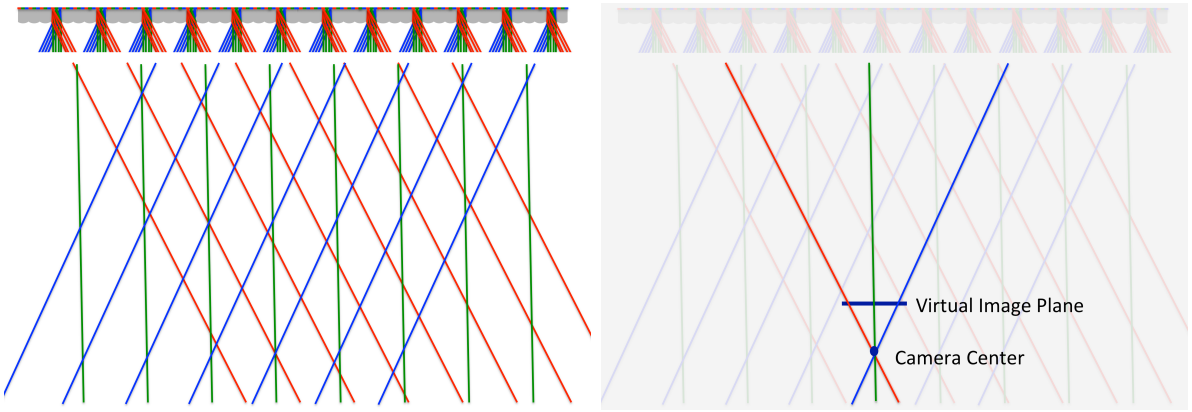
neighboring measurements to reduce that noise. We do this in order to provide an upper bound on the performance of the entire optical system that creates and images the chromo-coding lenticular arrays.

3.9 Chromo-coded Light Field

In Section 3.5, we described the ideal case where all rays of one color from a lenticular array lie in a parallel set of planes. Several examples of these planes that will result in different hue appearances are shown in Figure 3.4. Because of the organized structure of the appearances of the lenticular array, it can be useful to describe this hue/orientation relationship as a lightfield.

A lightfield is a complete representation of the light in a scene that describes the amount of light going through every direction and point in space. Typically the light field is represented by a 4D function, which parameterizes all the rays of the light in a given space by where they enter a 2D plane and pass through a second 2D plane. This enables the ability to represent not only the origin of rays, but also their direction.

We can describe the lenticular array as an object that creates a structured lightfield, where rays of the same orientation leaving the lenticular array and entering space have the same hue appearance. We call this the chromo-coded lightfield and visualize in in Figure 3.13a. The hue of a ray depends only on its orientation parameterized by the rotation around the major axis. Therefore, the chromo-coded lightfield is heavily structured and is 1 dimensional, not 4 dimensional. In effect, the chromo-coded lightfield is comprised of rays that enter 3D space with hues according to the lenticular array's HRF and specific major axis orientation.



(a) (b)

Figure 3.13: a) Each lenticule projects the same color ray at the same angle of incidence creating a chromo-coded lightfield. b) In this light field, a pin-hole camera would capture the same image for any translation, but would capture different hues for different orientations and a different variety of hues depending on the optical properties of the camera.

It is interesting to consider an infinitely large lenticular array and what type of images a pinhole camera would capture while in a chromo-coded lightfield. We visualize this in Figure 3.13b. The images of the chromo-coded light field captured by the camera have 3 interesting properties:

- The color of a ray for an observer of the chromo-coded lightfield only depends on the ray's angle incident to the lenticular array. Therefore, any translation of the camera would result in the same image of the chromo-coded lightfield.
- A pinhole camera captures a set of rays radiating from its optical center. These rays will have different angles incident to the lenticular array, and therefore, the camera will capture an image that will be a gradient of colors.
- Depending on the focal length and thus the field of view of the camera, the rays will vary more or less in their incidence angle and the resulting images will see more or less varying hues.

These observations motivated our exploration of lenticular arrays as camera calibration objects in Chapter 5.

3.10 Conclusion

In this chapter, we presented how to create lenticular arrays that encode orientation with hue. The lenticular arrays are cheap plastic sheets that can create different appearances for different orientations because of simultaneous lensing of each lenticule onto a similar local pattern. To be invariant to light intensity, we choose to create lenticular arrays with hue appearances over other potential appearances. The long cylindrical shape of the lenticules means that there is a 1-to-1 relationship, called the HRF, between the hue of the lenticular array and a limited amount of rotation around the major axis. Because of the optical properties of the lenticular array, the HRF has challenges with extreme light color or direct illumination.

Due to manufacturing challenges, we saw that localizing a hue measurement on the lenticular array is important to properly infer the true orientation angle. When our system is limited by the ability to accurately find reference points to localize a hue measurement, we saw orientation inference errors of ≈ 0.25 degrees. On the other hand, if our hue measurements are perfectly localized and the limiting challenge is color representation in our optical system, then the lenticular array will have orientation estimation errors of ≈ 0.1 degrees for a typical image. As such, the chromo-coding lenticular array can give precise, 1D orientation cues relating the orientation of a lenticular array relative to the camera.

Chapter 4

Pose Estimation using Lenticular Arrays

This chapter uses a single image of at least 2 small, chromo-coding lenticular arrays to estimate the pose of an object. Because markers are small, we consider a single HRF per marker and derive de-coupled constraints on object rotation to more quickly solve for pose estimation of an object with 2 small lenticular arrays. We show that our lenticular arrays have slightly better pose estimation performance than traditional fiducial markers over various imaging conditions. In addition, by over-constraining the pose estimation problem with many lenticular arrays, we can mitigate some lighting color challenges that were detailed in the previous chapter.

4.1 Camera Geometry and Camera Calibration

First, we introduce the camera geometry used in this and future chapters. We consider a pinhole camera imaging an object. The object, at 3D location P, is projected onto an image at 2D pixel location p according to a linear transformation matrix K:

$$\hat{p} = KP \tag{4.1}$$

where $\hat{p}$ is the homogeneous representation of p.

K is an upper triangular 3x3 transformation matrix that describes the geometric properties of the camera, including the optics and the sensor construction:

$$K = \begin{bmatrix} f_x & s & u \\ 0 & f_y & v \\ 0 & 0 & 1 \end{bmatrix} \tag{4.2}$$

where f_x and f_y are scaling factors that describe the focal length of the camera along 2 axes, s is the skewness and describes how rectilinear pixels on the imaging sensor are, and u and v are the x and y coordinates of the center of the optical axis in the image. For more general situations, having all the free parameters in K is useful. However, because of the high quality of modern manufacturing of optics and image sensors, it is common to simplify K to:

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.3}$$

where f represents a single focal length; pixels are square and so $s = 0$; and the optical center is designed to be in the center of the image and so u_0 and v_0 are the known center of the image. Therefore, the only unknown parameter in K is f.

In the reprojective geometry of Equation 4.1, the 3D point P is measured relative to the the same reference frame as the camera. However, often P is not known relative to the camera, but rather relative to some different, local coordinate system. The 2 coordinate systems are related via a 3D space rotation, R, and translation T. We can extend the projective geometry to describe how a point in a different reference frame projects into an image:

$$\hat{p} = K\left(RP + T\right) \tag{4.4}$$

where P is now in a local reference frame, for example on a 2D plane in 3D space in front of the camera. Typically, P is known from manufacturing or manual measurement.

In the above equation, there is often a distinction made between the intrinsic parameters of the camera, K, and the extrinsic parameters, R and T. The relationship between a world point P and its image p is fully defined with K,R, and T. Camera calibration aims to estimate the K matrix, but often also estimates R, and T as a side-effect. When K is known, the problem of solving for R and T is the pose estimation problem.

4.2 The Lenticular Constraint

In Chapter 3, we described how lenticular arrays can encode relative orientation by hue. The image of the lenticular array may have a different hue depending on a ray angle incident to the lenticular array. Consider the lenticular array in Figure 4.1. If the chromo-coding

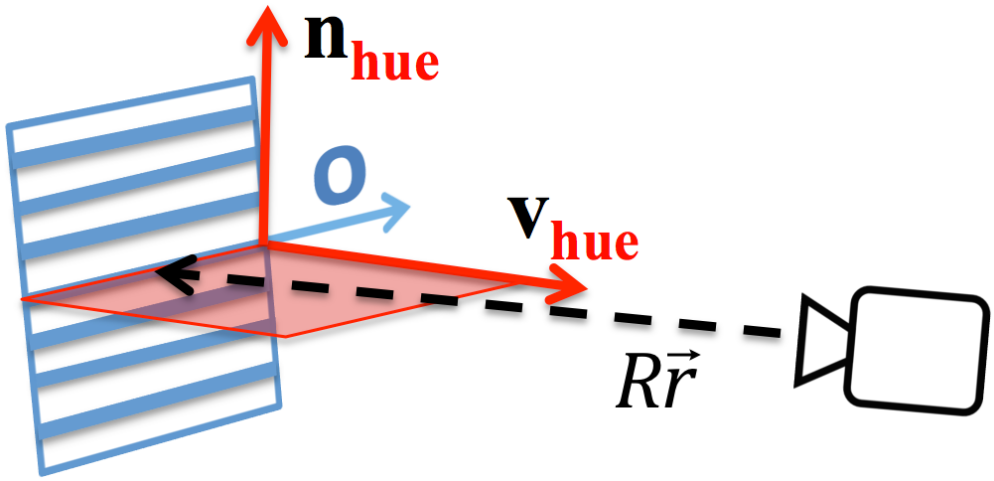


Figure 4.1: The Lenticular Constraint relates the color of the lenticular array to the orientation of the camera. In this constraint, a ray viewing the lenticular array must be perpendicular to the plane $\vec{n}_{hue}$ that corresponds to the observed hue. In this case, ray r is red, and so it must lie on the corresponding $\vec{n}_{hue}$.

lenticular array appears red, for example, the incident ray projecting from the camera must lie on the plane perpendicular to $\vec{n}_{hue}$. $\vec{n}_{hue}$ can be calculated as the cross product of any example ray of that hue and the major axis orientation, $\vec{o}$.

We will now formalize this geometric relationship. Using the standard geometric framework, we assume the origin of the camera coordinate system is centered on the camera, and the camera intrinsic properties are known and represented by a camera calibration matrix K. Then, if a pixel p is represented in homogeneous coordinates, it views an object that appears along a ray in space $\vec{r}$ that is defined as:

$$\vec{r} = K^{-1}\hat{p} \tag{4.5}$$

Therefore, if a pixel is imaging a lenticular array at pixel location p, then the lenticular array must lie along the ray $\vec{r}$.

A characteristic constraint on the rotation of a lenticular array comes from the apparent color of the lenticular array. We call this constraint, the lenticular constraint. In this constraint, the ray $\vec{r}$ from the camera that observes that lenticular array must lie in the plane $\vec{n}_{hue}$ defined by the hue that the camera observes, and therefore $\vec{r}$ must be perpendicular to $\vec{n}_{hue}$. In the coordinate system of the object, $\vec{n}_{hue}$ is defined by the 2 vectors that span it: $\vec{o} \times \vec{v}_{hue}$. In practice, $\vec{v}_{hue}$ is determined via the Hue Response Function, where a vector in the direction of the lenticular array's surface normal is rotated by θ degrees (inferred by hue) around $\vec{o}$.

This constraint can be used for pose estimation. It gives the constraint that the ray from the camera must be perpendicular to $\vec{n}_{hue}$, once $\vec{n}_{hue}$ is rotated into the camera coordinate system by R. So the lenticular constraint on the rotation is that the viewing direction and the hue surface normal are perpendicular:

$$\begin{aligned} R(\vec{o} \times \vec{v}_{hue}) \cdot K^{-1}p &= 0 \\ R\vec{n}_{hue} \cdot \vec{r} &= 0 \end{aligned} \tag{4.6}$$

This equation constrains the relative rotation between the lenticular array and the camera, using the relative position and observed hue of a lenticular array. The lenticular constraint is interesting for several reasons:

1. It is a simple mathematical relationship.
2. It does not require identifying a precise point-correspondence.
3. It is only dependent on the relative orientation R, *not* the relative position T, decoupling and simplifying the optimization.

The lenticular constraint is a foundation for this chapter where it is used as a rotation constraints for pose estimation.

4.3 Chromo-coded Markers and Pose Estimation

To support pose estimation, we create fiducial markers composed of small lenticular arrays, called chromo-coded markers. Information about the pose of an object comes from both the position and the hue of the observed chromo-coded markers. Many configurations of chromo-coded markers are possible and interesting, but we first derive the constraints for 2 markers attached to the same plane. Because the hue of a marker changes primarily due to rotations around the major axis of the lenticular array, we orient the major axes of our chromo-coded markers to be perpendicular to each other so that they provide complementary information about the surface normal. This ensures that any change in orientation will produce a change in appearance of the chromo-coded markers. The derivations of pose constraints for other configurations of chromo-coded markers is similar, the only requirement being that at least 2 chromo-coded markers have major axes that are non-parallel.

Figure 4.2 shows an example of a skinny planar object with one chromo-coded marker at location C_1, and a coplanar chromo-coded marker at location C_2 which is oriented orthogonally. A pinhole camera (on the right) observes an object (on the left), which contains 2 chromo-coded markers.

For simplicity, we assume that the coordinate system of the object is centered at point C_1 so that, in the coordinate system of the object, the first chromo-coded marker has position (0,0,0). We assume the second chromo-coded marker is a distance d away in the direction of

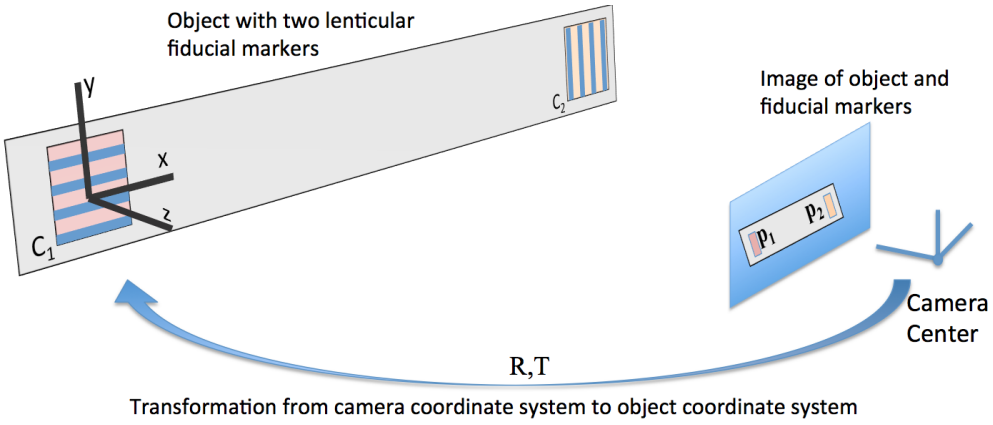


Figure 4.2: A pinhole camera (right) sees an image of an example flat object (left), which has 2 chromo-coded markers. The pose estimation problem asks to solve for the R, T describing the rotation and translation mapping the coordinate system of the camera to the coordinate system of the object.

the x-axis of the object so C_2 has object coordinates (d,0,0). These chromo-coded markers are not just points, they also have an orientation, and this orientation affects the apparent color of the patch. In Figure 4.2, the patch at C_1 has its major axis along the x-axis of the object and the patch at C_2 is oriented along the y-axis of the object.

The pose estimation problem is to solve for the rotation matrix R and translation vector T mapping a point in the object coordinate system to a point in the camera coordinate system. In our case, the chromo-coded marker at C_1 in the object coordinate system will move to $RC_1 + T$, and the marker at C_2 will move to $RC_2 + T$.

For pose estimation with chromo-coded markers, the question is to solve for the R, T that is consistent with p_1 being the image of $RC_1 + T$ and p_2 as the image of $RC_2 + T$, *and* both p_1 and p_2 have the correct hue for the angle at which they are being viewed.

We consider the specific pose estimation problem illustrated above. We show the necessary constraints from 2 chromo-coded markers for pose estimation and explain how to optimize

for R and T. Using the standard geometric framework introduced in Section 4.1, if we see chromo-coded markers at locations p_1 and p_2, then the fiducial markers must lie along rays $\vec{r}_1 = K^{-1}\hat{p}_1$ and $\vec{r}_2 = K^{-1}\hat{p}_2$. Since we assume that K is known, the remainder of this derivation is based on $\vec{r}_1$ and $\vec{r}_2$.

4.3.1 Rotational Constraints

The first constraint on R is the lenticular constraint of the first chromo-coded marker which comes from the marker's apparent color. Using the hue of this marker and its pre-calibrated HRF, we can determine the direction $\vec{v}_{hue_1}$. The orientation of the marker $\vec{o}$ is known, and since it is along the x axis, we denote it by $\vec{x}$. Therefore, the lenticular constraint indicates that:

$$\begin{aligned} R(\vec{x} \times \vec{v}_{hue_1}) \cdot \vec{r}_1 &= 0 \\ R\vec{n}_{hue_1} \cdot \vec{r}_1 &= 0 \end{aligned} \tag{4.7}$$

A similar constraint applies when the camera observes the second chromo-coded marker, except that the viewing direction is r_2, the marker is aligned along the y-axis of the object, and the observed hue and HRF may be different. This leads to a second lenticular constraint:

$$\begin{aligned} R(\vec{y} \times \vec{v}_{hue_2}) \cdot \vec{r}_2 &= 0 \\ R\vec{n}_{hue_2} \cdot \vec{r}_1 &= 0 \end{aligned} \tag{4.8}$$

A third constraint relies solely on the locations of the observed chromo-coded markers. Specifically, three rays are co-planar: the direction from the camera to the first marker $\vec{r}_1$,

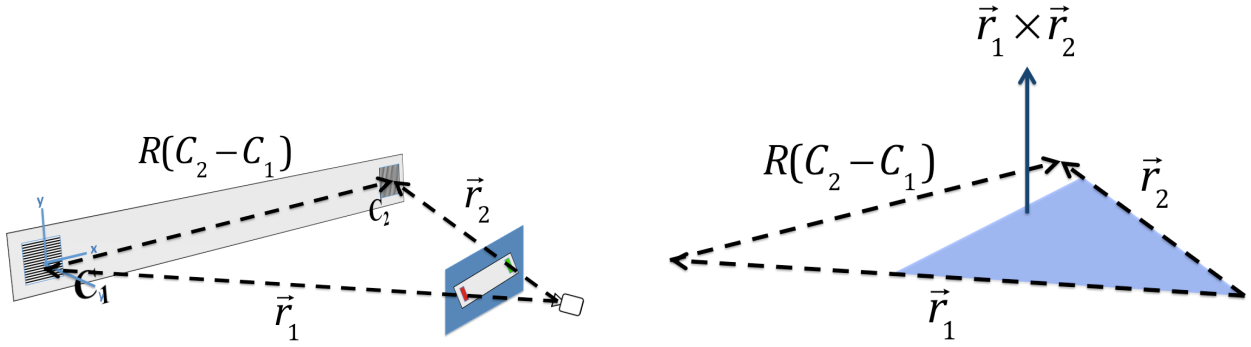


Figure 4.3: The 3rd and final rotation constraint says that the displacement between the two chromo-coded markers, must be perpendicular to the plane created by the directions of the markers in the reference frame of the camera. The left shows the relative directions of vectors used as they pertain to the image scene and the right visualizes the simplified math of the constraint.

the direction from the camera to the second marker $\vec{r}_2$, and the displacement vector between the first and second patch $(RC_2 + T) - (RC_1 + T) = R(C_2 - C_1)$. This leads to a geometric constraint on rotation:

$$R(C_2 - C_1) \cdot (\vec{r}_1 \times \vec{r}_2) = 0, \tag{4.9}$$

If the coordinate system of the object is such that C_1 is the origin, this simplifies to:

$$(RC_2) \cdot (\vec{r}_1 \times \vec{r}_2) = 0, \tag{4.10}$$

We visualize this constraint in Figure 4.3.

Thus, 2 chromo-coded markers provide three constraints on the rotation matrix R that are independent of the estimation of the translation vector T.

4.3.2 Translational Constraints

Once the rotation is known, we can derive three linear constraints on the translation. The first constraint is that the translation must be consistent with the observed location of the first chromo-coded marker. Since any point C in the object coordinate system is mapped to a location $RC + T$, the ray $\vec{r}$ viewing that point must be parallel to the vector from the origin to $RC + T$. Because we define the first chromo-coded marker at location C_1 to have object coordinates (0,0,0), $RC_1 + T$ simplifies to just T, and we can use the fact that the cross-product of 2 parallel vectors is zero to express the constraint as:

$$T \times \vec{r}_1 = \vec{0}. \tag{4.11}$$

The final constraint needed to estimate the translation is similar to 4.11, but uses the projection of the second chromo-coded marker:

$$\big(RC_2 + T\big) \times \vec{r}_2 = \vec{0} \tag{4.12}$$

These 2 constraints on the translation are both vector equations. With the 3 rotation constraints, collectively these 5 constraints let us solve the pose estimation problem visualized in Figure 4.2.

4.3.3 Optimization

There are several possibilities to solve for R and T using this set of constraints. In our experiments we first obtained the rotation matrix on its own by optimizing a non-linear error function over the Rodrigues vectors $\vec{\rho}$ that define a rotation matrix $R\rho$. The error

function is the sum of the squared arithmetic error for Equations 4.7, 4.8, and 4.10. We use $fminunc$ in matlab and initialize with a rotation that has the lenticular arrays facing the camera.

After we have the rotation matrix R, equations 4.11 and 4.12 define a linear system of equations for T. Both error functions return solutions with zero error because they are optimizing a minimal set of constraints, as we are measuring 6 numbers (2 coordinates and 1 hue for each of the 2 markers) to solve the 6 DOF pose estimation problem. We can extend this optimization routine for $n \geq 2$ chromo-coded markers to improve pose estimation results and add robustness to varying ambient light conditions.

4.4 Pose Estimation from $n \geq 2$ Chromo-coded Markers

Our pose estimation method minimally constrains the 6 degrees of freedom of the pose estimation problem with the 2 hue and 4 position measurements from 2 chromo-coded markers. In this section, we now build on our algorithm to use more than 2 chromo-coded markers to over-constrain the pose estimation problem and improve estimation results. First, we generalize the existing rotation and translation constraints to optimize with $n \geq 2$ markers. Second, we formulate a second optimization that minimizes the hue and position reprojection error of the chromo-coded markers to improve initial pose estimations. In the framework of this reprojection optimization, we later introduce 2 additional color scaling variables to add robustness of our method to different lighting environments.

4.4.1 Generalized Pose Optimization

A common approach to improve fiducial marker pose estimation results is to increase the number of fiducial markers and optimize the over-constrained system of equations. As an example, only 4 corners of a square are needed to solve for pose, however, optimizing over many corners of a checkerboard drastically improves pose estimation accuracy.

Inspired by this, we show how to optimize over the arithmetic constraints presented in Section 4.3 for $n \geq 2$ chromo-coded markers. We generalize these constraints so that each ith marker has:

- a position C_i in the local reference frame
- an orientation $\vec{o}_i$ in the local reference frame
- a hue direction $\vec{v}^i_{hue}$ in the local reference frame
- an observed direction $\vec{r}_i$ from the camera reference frame

In the generalized form, each marker has 2 constraints on rotation, 1 using hue and 1 using the relative position to an origin marker, and 1 constraint per marker on translation. The rotation constraints do not depend on translation, so, similar to with 2 chromo-coded markers, we can first optimize for the rotation over all markers with the following objective function:

$$\underset{\rho}{\operatorname{argmin}} \sum_{i}^{n} \left(\left(R_\rho(\vec{o}_i \times \vec{v}^i_{hue}) \cdot \vec{r}_i \right)^2 + \left(R_\rho(C_i - C_{i-1}) \cdot (\vec{r}_{i-1} \times \vec{r}_i) \right)^2 \right) \tag{4.13}$$

where ρ is the rodrigues parameterization for rotation and R_ρ is the 3x3 rotation matrix corresponding to ρ. Then, we can optimize for the translation:

$$\underset{T}{\operatorname{argmin}} \sum_{i}^{n} \left((R_\rho C_i + T) \times \vec{r_i} \right)^2 \tag{4.14}$$

The objective functions presented above use the arithmetic result of dot and vector products as the error terms. Therefore, we refer to using these objective functions in optimization as arithmetic optimization.

4.4.2 Reprojection Refinement

After using the arithmetic error to optimize for $n \geq 2$ chromo-coded markers, we could optimize over the reprojection error of all markers to refine pose estimation. This type of reprojective refinement is common among traditional fiducial markers, especially the checkerboard. However, while traditional markers only minimize the reprojective error of their point-correspondences, we also minimize the reprojective error of hue. Here we show the objective function to simultaneously optimize for R and T over the reprojection error of hue appearance and image position:

$$\underset{\rho,T}{\operatorname{argmin}} \sum_{i}^{n} \left(\left(h_i(\rho, T, P_i) - hue_i \right)^2 + \kappa \| g(\rho, T, P_i) - p_i \|_2^2 \right) \tag{4.15}$$

where $g(\rho, T, P_i)$ projects the position of the i_{th} marker into the image according to the translation and rotation parameters T and ρ and the marker local reference frame position

P_i. $h_i(\rho, T, P_i)$ projects a hue according to the pre-calibrated HRF given for the i_{th} chromo-coded maker at the projected image location $g(\rho, T, P_i)$ and according to the rotation ρ. The normalizing value κ is found empirically to balance the hue and position reprojection costs.

4.4.3 Color Calibration

As discussed in Section 3.6.1, the appearance of chromo-coded markers can be affected by environmental lighting factors. Ambient light color may affect the hue appearance of the chromo-coded marker at any orientation, while glare may only change the hue appearance for certain orientations. As a result, the pre-calibrated hue/viewpoint relationship of the HRF which encodes $\vec{n}_{hue}$ may not represent the observed images and orientations in a different lighting environment. The extra information given by $n > 2$ chromo-coded markers, however, makes it possible to correct for these uncalibrated lighting environments.

Inspired by white balancing methods, we therefore introduce 2 extra color scaling variables to the objective function of Equation 4.15. These 2 variables, s_r and s_b, scale the red and blue channels of the RGB measurement of chromo-coded markers to better fit the model of how chromo-coded markers appear given a relative orientation. The purpose of s_r and s_b is to color correct chromo-coded marker observations to fit the pre-calibrated HRF of a different lighting environment. The optimization in Equation 4.15 can thus be updated to include the 2 new color correcting variables by scaling the observed RGB color of each chromo-coded marker:

$$\underset{\rho,T,s_r,s_b}{\operatorname{argmin}} \sum_{i}^{n} \left(\left(h_i(\rho, T, P_i) - hue(M * rgb_i) \right)^2 + \kappa \| g(\rho, T, P_i) - p_i \|_2^2 \right) \tag{4.16}$$

Here, rgb_i is the vector representation of the RGB color measurement of a chromo-coded marker,

$$M = \begin{bmatrix} s_r & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s_b \end{bmatrix}$$

and $hue(..)$ is a function that coverts an RGB measurement to HSV space and returns only the hue.

Minimizing this objective function simultaneously solves for the R,T and color correction consistent with all markers in the scene.

4.5 Experimental Results

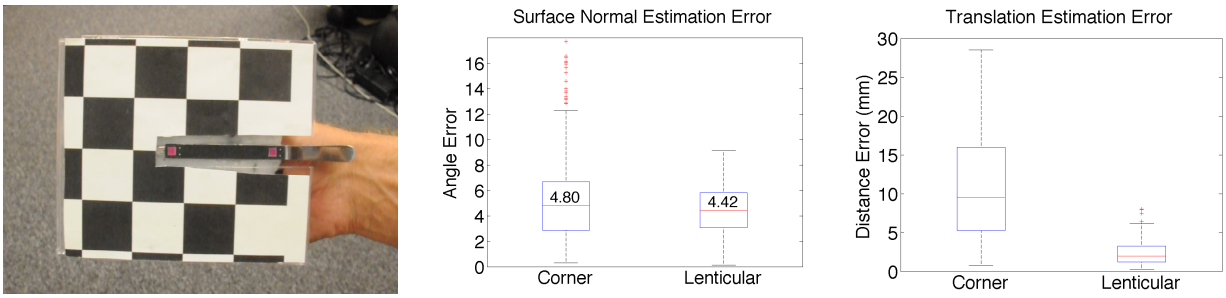
In this section, we experiment with our chromo-coded markers and their ability to estimate pose. First, we test the minimally constrained situation of using 2 chromo-coded markers against the traditional fiducial marker methods. We notice that blurring deteriorates the pose estimation performance for the traditional methods more than ours, so we then experiment to determine the extend of noises effect in simulation. Next, we experiment with using more than 2 chromo-coded markers to explore the performance in over-constrained systems. Finally, we test the secondary reprojection optimization to see how much it improves pose estimation results with and without the additional color calibration.

4.5.1 Pose Estimation from 2 Chromo-coded Markers

In this subsection, we first compare end-to-end performance of a pose estimation procedure, comparing traditional fiducial markers (4 small points at a known location on the object) and our new chromo-coded markers. We build a physical prototype of the chromo-coded marker to estimate the pose of an object throughout a video. For the experiment, we compare accuracy for an object that is long and skinny.

We track a small pair of tweezers by placing chromo-coded markers 55mm apart. These chromo-coded markers are made from the lenticular arrays discussed in Chapter 3. Each fiducial marker is a 4mm square, comprising about 12 lenticules. For comparison, there are also 4 small fiducial markers in a rectangular pattern that are also 55mm apart in one direction, and 4 mm apart in the orthogonal direction. We determine the pose of these corner markers following the ARToolkit algorithm [28]. To estimate the pose of the object using the 2 chromo-coded markers, we use the arithmetic optimizations detailed in Section 4.3. The tweezers are mounted to a large checkerboard so that we can use a standard toolbox [7], to give ground truth pose estimates per frame. Figure 4.4a shows this setup in a frame of the experimental video.

We calibrate our chromo-coded markers and compute the HRF — and the inverse HRF that maps a measured hue to a θ rotated around the major axis of a chromo-coded marker — through a lookup-table from the experimental measurements in a calibration video using the same setup. For each frame, we hand label the location of the chromo-coded marker and corners. All frames that had substantial motion blur were discarded because fiducial dots were difficult to label. For the chromo-coded marker, we click each corner of the small array, and use the centroid as its location. The color of the entire area of the small chromo-coded



(a) (b) (c)

Figure 4.4: We compared chromo-coded markers with the state of the art corner method on a video (of which an example frame is shown in a). For the length of the video, the pose estimation using chromo-coded markers produces a better normal direction estimation (b) and position estimation (c) median with a much more compact distribution. Each box in b) contains the median value in text.

marker is averaged to get the hue of each marker. To get a sub-pixel accuracy estimate for the standard fiducial dots, we fit a Gaussian distribution over the small dot and use the mean as the dot location.

Figure 4.4 shows the pose estimation results for the rotation error, defined as the angular difference between the ground truth and estimated Z-axes, and translation error, defined as the Euclidean distance of true position and estimated position. We show the results as boxplots. The top and bottom blue lines of the box indicate 1st and 3rd quartiles, while the red line in the center of the box indicates the median. For Figure 4.4b, the median error is written in each fiducial marker box. The median rotation (Figures 4.4b) and translation (Figures 4.4c) errors of pose estimation using the chromo-coded markers are less than that of the state of the art corner method. In addition, the distribution of errors is much tighter and indicates a more stable pose estimation. We hypothesize that this is due to the fact that our system is more robust to fiducial marker position noise. In the next section, we test this hypothesis in a simulated environment.

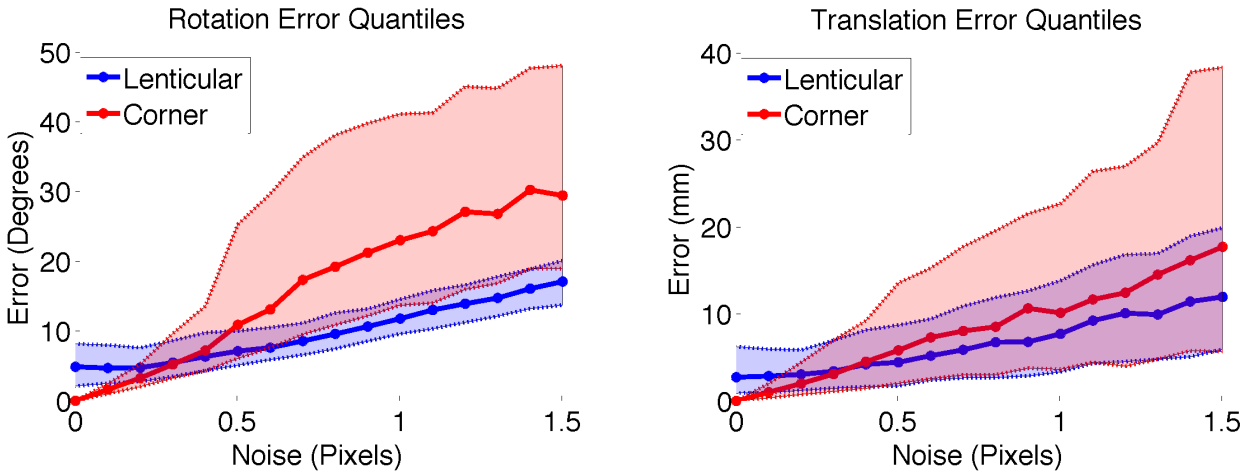
4.5.2 Pose Estimation with Noise

To test the impact of noise on pose estimation using chromo-coded markers, we simulated the physical implementation of the skinny forceps in the previous Section. We randomly generate locations for the simulated object by defining translations uniformly distributed in a box between 0.5 and 2 meters in front of the camera. Rotations are generated randomly with the constraint that the angle between the surface normal of the object and the z-axis of the camera is less than 35 degrees.

For each object location we project the locations of the standard fiducial markers and our chromo-coded markers to get image positions of simulated markers. To simulate the color of the chromo-coded marker, we assumed that the printing process created no artifacts and implemented a simple ray tracer to model the optical effects at each lenticule (including the failure of the elliptical lenticular lens to perfectly focus parallel rays) in order to compute the hue.

With this setup, we model noise to hue measurements by adding noise in the range $[-0.01, 0.01]$ to capture un-modeled effects that might come from, for example, glare off the chromo-coded marker. We model location error as a 2D Gaussian whose standard deviation is a multiple of 0.1 pixels. Figure 4.5 shows the relative error in pose estimation as this noise increases. Rotation and translation errors are defined as in the previous section. Pose was estimated using the optimization detailed in Section 4.3.

Figure 4.5a show the rotation results for different noise levels. At zero added noise, the standard approach based on tracking corners is perfect (because there is no error in the simulated point locations) while our system has hue noise. However, for all but the smallest



(a) (b)

Figure 4.5: By using chromo-coded markers instead of the typical 4 corner markers, we improve rotation (a) and translation (b) estimations. In both figures, we increase the amount of noise simulated along the x-axis, and show the errors by their median value for each simulation surrounded by the 1st and 3rd quantiles.

noise levels, the chromo-coded marker has lower median error and more consistent error magnitudes for both translation and rotation. For translation estimates, shown in Figure 4.5b, the median error is comparable among both methods. However, the distribution of errors is starkly larger using the traditional fiducial marker approach. This is likely due to the sensitivity to fronto-parallel views the traditional point-correspondence methods have; any noise introduced in the point locations on the images will result in very large errors in rotation and translation estimations.

4.5.3 Pose Estimation from $n \geq 2$ Chromo-coded Markers

While 2 chromo-coded markers are sufficient to determine pose, in this subsection we explore using more than 2 chromo-coded markers to over-constrain the pose estimation problem.

For this and subsequent experiments, we create an object shown in Figure 4.6. Four coplanar markers are arranged in a rectangle, oriented so diagonal pairs have the same orientation, but orthogonal to the other diagonal. We surround the chromo-coded markers with a radial hue pattern inspired by [50] to facilitate automated identification. We use concentric hues as a cue to refine centroid positioning of each chromo-coded marker. On the same plane as the markers, we include a checkerboard. Video was taken by orbiting and rotating the camera around the markers at about 50cm away. Using the camera calibration toolbox in Matlab 2014a [39], the checkerboard serves to ground truth the pose of the chromo-coded markers. As before, we use this ground truth to also calibrate the HRF for each chromo-coded marker with a calibration video.

In this experiment, we estimate the pose of frames from a video 3 times, each iteration adding the information from one additional chromo-coded marker. We estimate pose using the arithmetic optimization of Section 4.4.1. We estimate pose for the same video frames used to calibrate each marker's HRF in order to avoid any deleterious effects from different light environments. The video has 700 frames and captures the minimum and maximum relative orientations possible for the chromo-coded markers. As before with the physical prototype with 2 chromo-coded markers, we show the summary rotation and translation errors as boxplots.

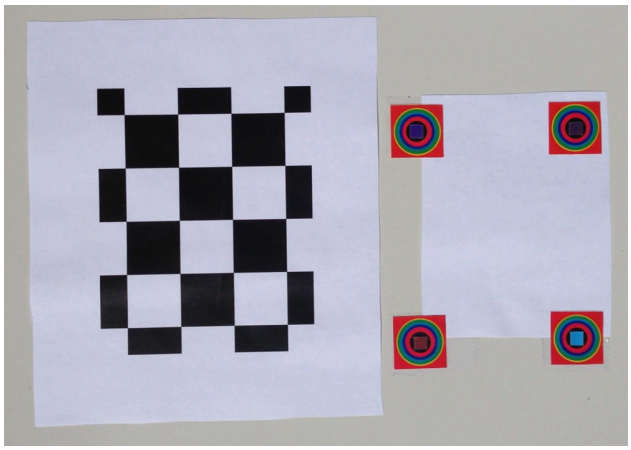


Figure 4.6: With a setup that mounts 4 chromo-coded markers on the same plane as a checkerboard, we can ground truth experiments that explore pose estimation results for more than 2 markers and color calibration routines.

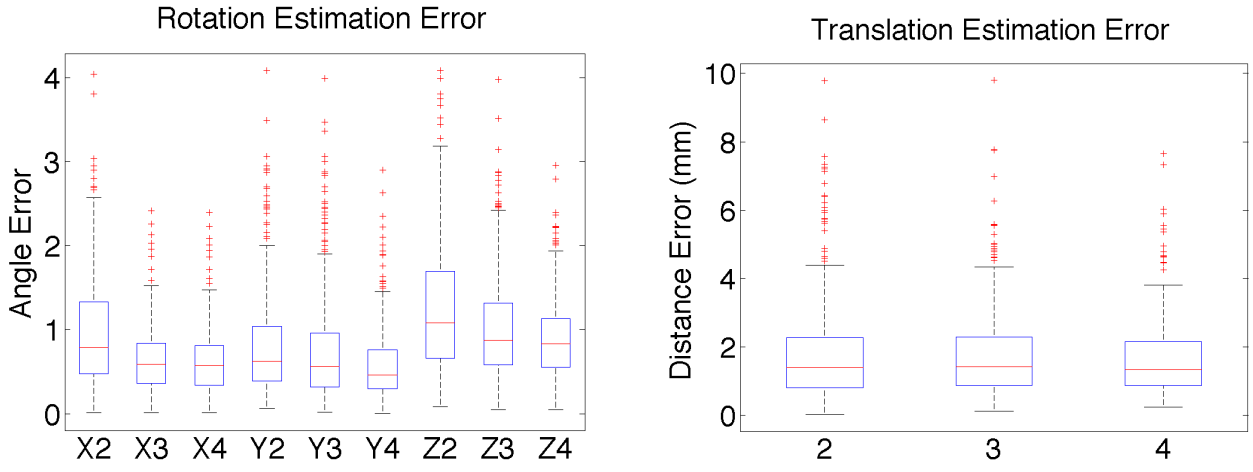
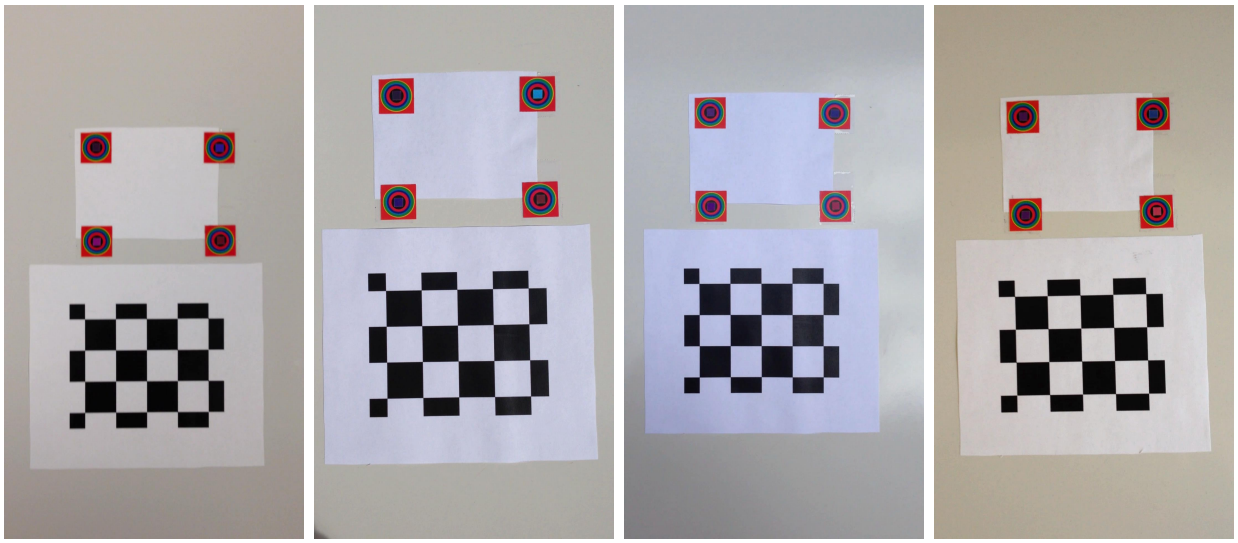


Figure 4.7: Although 2 chromo-coded markers are sufficient to constrain the pose estimation problem, using more chromo-coded markers improves rotation performance. Here we show the performance effects of increasing the number of chromo-coded markers in a) rotation estimation and b) translation estimation.

Figure 4.7 shows pose estimation results. For both the rotation and translation results, we group by trials using 2, 3, and 4 markers. Figure 4.7a shows the angular error of the individual local axes of the plane. For example, X2 indicates the error of the x-axis estimation when using 2 markers. Results show that additional chromo-coded markers reduces rotation error for each axis. The accuracy for the z-axis or the surface normal is especially improved. In addition, the inlier extremes and quantiles tend to become tighter, indicating a higher precision in pose estimations. The improvement in performance is due to the extra angular and positional information gained from adding markers across the plane.

Figure 4.7b shows the error in the translation estimates. We see slight gains in having 4 markers versus 2 or 3. The translation performance is already under 2 mm of error, so there is little room for improvement.



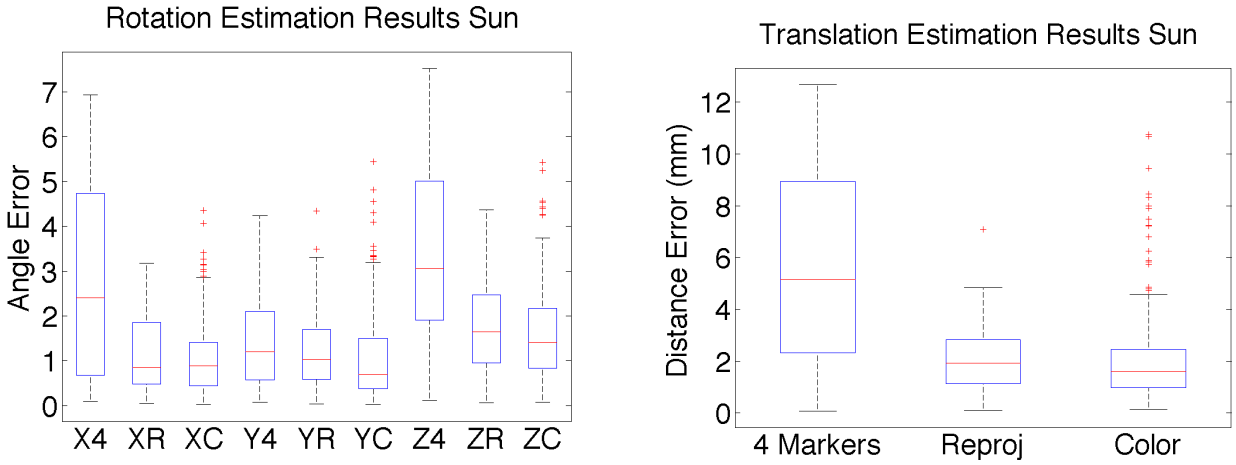
(a) (b) (c) (d)

Figure 4.8: From left to right, we show sample frames from the calibration, sunny, shade, and office videos used to test robustness of pose estimation to different lighting conditions in Section 4.5.4.

4.5.4 Reprojection and Color Calibration Refinement

Next, we experiment with pose estimation in various lighting conditions. We use 3 videos, each containing around 300 frames, taken in lighting conditions different from the calibration video used to determine the HRF. The calibration video was taken under fluorescent lights away from any window. Two test videos were taken outside: one in full direct sun (sunny) and one on a completely cloudy day (cloudy). The third video (office) was taken indoors without lights turned on, but next to a window. We show a frame from each video in Figure 4.8 and compare it to the video used in calibration. The indoor videos have similar lighting, but the outdoor scenes in full sun and full shade have a noticeable blue tint. We compare the pose estimation results using 4 chromo-coded markers 3 times:

- using the arithmetic optimization of Section 4.4.1,

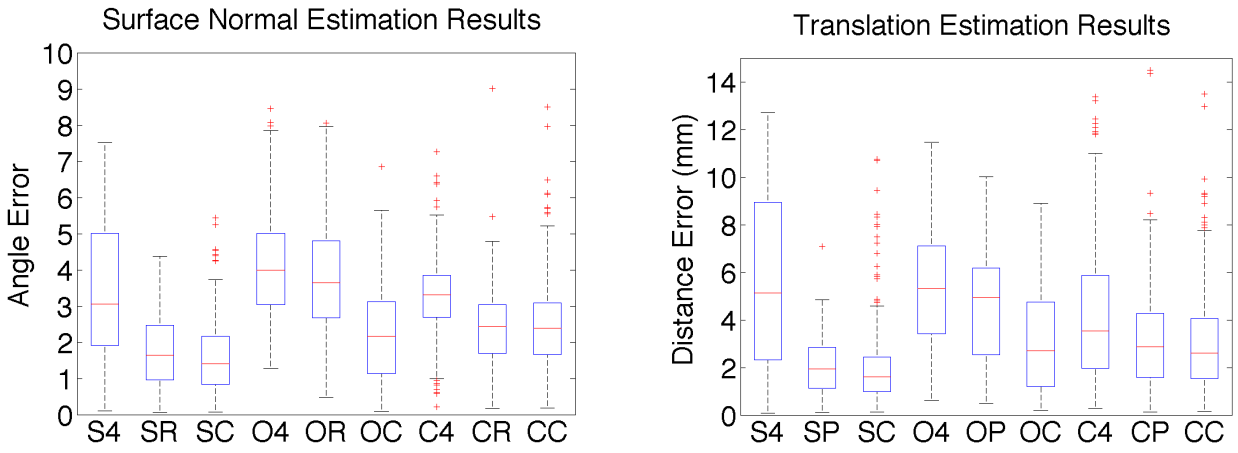


(a) (b)

Figure 4.9: Optimizing for reprojection error and color calibration variables mitigates the negative effects of a lighting environment very different from calibration, such as full sun shown here. The a) rotation error reduces for each axis and b) the translation error reduces with the reprojection color calibration routine.

- initializing from this pose estimation and doing the reprojection optimization of Equation 4.15,
- and using the extended reprojection optimization that solves for 2 additional color scaling variables shown in Equation 4.16.

To start, we show results for the test data in full sun, shown in Figure 4.9. Similar to before, we show the rotation and translation error grouped by each local axes and vary the optimization method. As examples, X4, XR, and XC label the rotation error for the arithmetic optimization, the reprojection optimization, and the color calibration optimization for the x-axis, respectively. In Figure 4.9a we analyze the rotation error. We can see that for each axis, reprojection substantially improves precision and accuracy; the median error reduces, and the distribution of errors also gets narrower. However, by adding color calibration, we ameliorate the effect of the varied lighting: rotation estimation error drops



(a) (b)

Figure 4.10: Despite these varying lighting conditions, reprojection optimization with color calibration produces results with very high accuracy and precision in a) surface normal estimation and b) translation estimation. Sunny, Office, and Cloudy datasets are denoted by S, O, and C respectively and 4, P, and C correspond to the three optimization routines discussed in Section 4.4.

down to about 1-1.5 degrees of median error, with most errors being less than 2 degrees. The same improvements can be seen in translation estimation results shown in Figure 4.9b. Using reprojection refinement improves translation precision and accuracy substantially, with additional (smaller) improvements by color calibrating as well. The median translation error for color calibration is 1.6 mm versus 1.9 mm with only reprojection optimization.

Figure 4.10 shows pose estimation results for the cloudy and office videos as well. Because the results are representative of other axes, we only present the z-axis error for rotation results. We denote the different datasets by S for sunny, O for Office, and C for Cloudy and the 3 optimization methods as 4,R, and C. For both rotation and translation results across all datasets, we see an improvement over arithmetic optimization with reprojection refinement, where the highest accuracy and precision is achieved by optimizing for additional color scaling variables. Across a variety of light environments using our color calibration optimization,

we see the median rotation error is around 2 degrees of error and median translation error less than 3 mm.

4.6 Applications

Pose estimation can enable a variety of computer vision applications including object tracking, robotics, metrology, or augmented reality. In this section, we show 2 applications of using chromo-coded markers to estimate pose for frames in a video: tracking hand writing and augmented reality.

4.6.1 Recording Hand Writing

Because of the advantages of chromo-code markers, we can accuractly measure the pose of small hand tools like a marker pen. In addition, we can place the chromo-coded markers on the end of a marker pen, so that the natural hand position is not compromised and so that the markers can be seen by a camera from an unburdensome position.

As a potential use case, we imagine a user wearing a head-mounted display such as Google Glass or Oculus Rift writing in real time. Cameras on these systems would be blocked from seeing what is written by a writer's hand, but would be able to see the end of the pen. The head-mounted display could track writing, decipher the images to text, and perhaps offer grammar correction or spelling checks in real-time.

In Figure 4.11, we show how the chromo-coded markers could enable this use case. A point of view camera records a user writing with a marker pen labeled with 2 chromo-coded markers.

(a) (b)

Figure 4.11: a) We show an application of chromo-coded markers, where we can track hand writing in a video. b) For each frame we can estimate the pose of the end of the marker, and with knowledge of the length of the marker, we can track what is being written. Results of the hand writing tracking is readable and matches the input video.

Figure 4.11a shows an example frame, where the hand has blocked some writing, but the colorful chromo-coded markers are visible at the end of the pen. At each frame, the pose of the end of the pen is determined, and with knowledge of the length of the pen, the position of the writing point of the pen is understood. By tracking the locations of the writing end of the pen, an algorithm can rebuild what is being written.

The rest of Figure 4.11 shows the pose of the pen for all frames. In Figure 4.11b, we see the position of the end of the pen in blue, the pen as a black line, and the position of the writing end of the pen in red. A video with frame by frame results is available on YouTube from the linked text. The writing is indecipherable by looking at just the position of the end of the pen in blue. However, because of accurate and precise rotation estimations, the writing end of the pen can be inferred to reveal what was written in the video. Even by just visualizing the individual positions of the pen's writing end for each frame, without

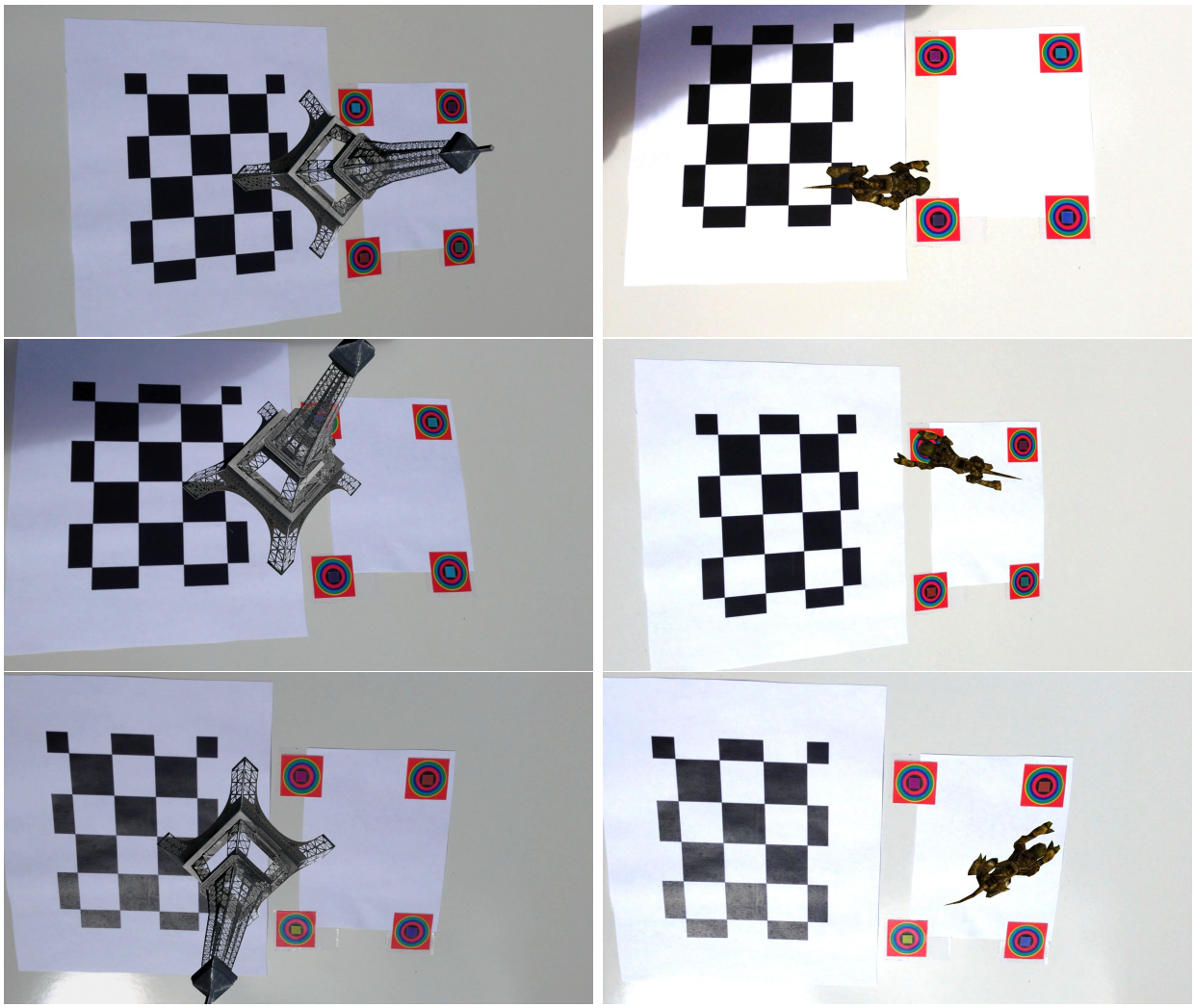
smoothing or motion, it is clear that "CVPR 2016" was written. Determining the position of the writing end is difficult because it relies on a strong rotation estimation at the end of the pen; small rotation errors are exacerbated by the long length of the pen.

4.6.2 Augmented Reality

Because they give an orientation constraint as well as a position constraint, chromo-coded markers provide a higher density of geometric constraints. Even using a few small markers provides better constraints than a large checkerboard. As a result, the chromo-coded markers can give more reliable orientation estimations than traditional fiducial markers, even for fronto-parallel views. These 2 properties are important for any Augmented Reality (AR) application.

AR can be useful for recreation, work, or educational purposes. Here we show an example, where a user would like to see 3D models of objects as if they were actually in front of them, resting on a desk, through a digital window provided by a cell phone or tablet. The tablet or cell phone could capture images of the desk which is labeled with 4 chromo-coded markers. For each frame, the pose of the desk plane is estimated. Then, a 3D model of the Eiffel Tower or a moving creature is rendered over the input frame using projective geometry. This rendered frame is output via the cellphone or tablet display screen in real-time. As a result, these 3D models seem to be on the desk in front of a user. A user can tilt and move the cellphone or table while viewing the desk through the screen in order to see different sides of the Eiffel Tower or moving creature.

We show output frames of the display in this hypothetical interaction in Figure 4.12, where the original scene, with 4 chromo-coded markers, is seen with the additional 3D model. The



(a) Eiffel Tower (b) Creature

Figure 4.12: With the chromo-coded markers, we can understand the relative pose of the plane in view for every frame in the video. Therefore, we can position a 3D model on the plane and render it as if it was actually in the scene imaged by the camera. As examples, we render a model of the Eiffel Tower (left) and a moving 3D monster (right) into the scene captured by the camera.

full AR videos of the eiffel tower and the running monster can be found on YouTube from the linked text. This AR application enables a natural and intuitive way to investigate a 3D model.

4.7 Conclusion

In this chapter, we have introduced how to use small chromo-coding lenticular arrays for pose estimation. We derived the lenticular constraint that relates the rotation of the lenticular array relative to the camera. This orientation cue is the core rotation constraint for pose estimation and is less sensitive to noise than point-correspondence constraints. As a result, our method results in better pose estimates than the traditional point-correspondence based approaches. When we use more than 2 markers, we can solve for 2 additional white balancing parameters, improving initial pose estimates in the case of lighting variation. In the end, our marker can achieve less than 2 degrees of orientation error and less than 1% position error across a variety of lighting environments.

Chromo-coded markers have two advantages over traditional point-correspondence based markers. First, our markers can continue to give consistent rotation constraints on the orientation of an object. Whereas traditional fiducial markers give unpredictable orientation estimations for fronto-parallel views, our markers can give precise orientation cues because the hue to angle relationship of the HRF is 1-to-1 for a large range of orientations. This is useful for most pose estimation applications where a camera tries to determine pose of an object that is fronto-parallel, for example, a robot estimating its position in a room. Second, our chromo-coded markers give consistent constraints even when the markers are relatively close to each other in an image. Traditional point-correspondence markers, such

as the ARTag [14], have the strongest constraints when the distance between points in an image is large. In this case, even small changes in pose will result in significant changes in the pixel locations of the points. In contrast, chromo-coded markers give reliable rotation constraints no matter the relative proximity or size in an image because the markers directly constrain rotation from the hue appearance. This has two corollaries: 1) our method thus gives more reliable orientation estimation when the object is far from the camera, and 2) our method can be deployed in use cases where the marker configuration is limited because of an object's shape, movement, or usability, for example hand tools. These two advantages make our chromo-coded markers a reliable alternative to other traditional point-correspondence based fiducial markers.

Chapter 5

Camera Calibration with Lenticular Arrays

Chromo-coding lenticular arrays offer substantial advantages for single image camera calibration. In Figure 5.1, we show how a chromo-coded lightfield created by a large lenticular array might look in images taken by a pinhole camera at two different focal lengths. For a camera with a long focal length (left), the incident angle of rays viewing the lightfield are almost parallel, so an image would have similar hues across its width and height. However, for a camera with a short focal length (right), the opposite is true: rays radiating out of the camera along the width or height will have different incident angles to the lightfield and therefore the image will be a gradient of many different hues.

This chapter explores making a larger lenticular array and deriving constraints on camera calibration from a single image. One interesting feature of this approach is that every pixel that views the calibration object measures a hue, so every pixel adds new constraints – offering thousands of measurements rather than just a few per image.

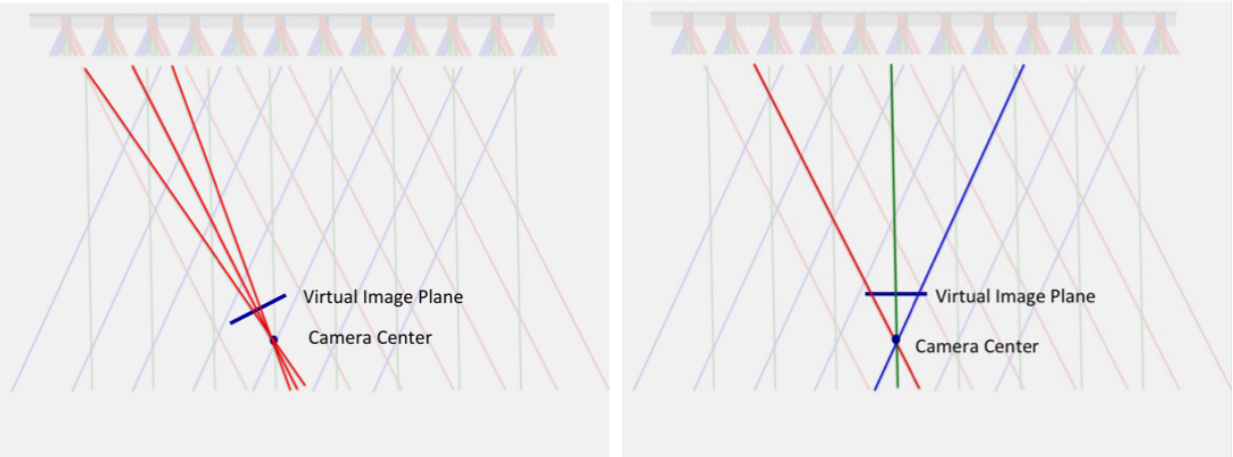


Figure 5.1: Consider a camera viewing a chromo-coded lightfield. At the scale of the whole lenticular array, this means that the perspective of a camera with a narrow field of view will only see a few hues (left), while a camera with a wider field of view will see many different hues (right).

5.1 Chromo-coded Calibration Object

We use the same chromo-coding lenticular arrays introduced and used in Chapter 3 to create chromo-coded markers, but now make an object with three lenticular arrays mounted perpendicular to each other on a plane. The 2 flanking arrays have the same orientation, but orthogonal to the middle array. These arrays are orthogonal so that any rotation of the calibration object creates a change; when the object is oriented as shown on the left of Figure 5.2, rotation around the horizontal axis causes the central part to change color, while rotating around the vertical axis causes the two edge arrays to change color. Small black strips are added to make it easier to distinguish the 3 arrays when they are oriented so that their colors are similar.

Figure 5.2 visually validates the predictions on views of the chromo-coded lightfield produced by a lenticular array for different focal lengths. For an image captured with a 300mm focal length lens (left), the lenticular array looks to have the same hue appearance across the

Figure 5.2: A calibration object made from 3 lenticular arrays. Each lenticular array has an observed color that changes depending on its viewing angle. (Left) When viewed from reasonably far away, the arrays have relatively consistent colors because they are being viewed from approximately the same angle. (Right) A wide angle view from much closer has significant color variation because the direction from the camera to different parts of the object varies substantially. This paper uses this color variation to derive strong geometric constraints for simultaneous, single-image pose estimation and camera calibration.

lenticular arrays. However, for an image captured at a 4.1mm focal length (right), the lenticular array changes hue drastically across the array. These two different appearances are due to largely different incident rays viewing the lenticular arrays. Therefore, the chromo-coded calibration object will be able to give cues about focal length from the difference in appearance across the large lenticular arrays.

5.2 Camera Calibration

In this section, we show how to use an image of the calibration object for camera calibration. We seek to estimate the camera intrinsic properties, K, and extrinsic properties, R and T, that relate an object's location relative to the camera to its projected location in an image. We do this by deriving many lenticular constraints from the hues sampled along a grid across

each lenticular array that comprises the calibration object. We optimize for the R and K using these cues, and then add point-correspondences from the corners of the lenticular arrays to estimate T. Similar to pose estimation with chromo-coded markers, we can then minimize a reprojection error to refine estimations.

5.2.1 Rotation Constraints

Similar to pose estimation with chromo-coded markers, we derive constraints on the calibration object rotation relative to a camera and the camera focal length from the hue of lenticular arrays. We follow the presentation of the lenticular constraint in Section 4.2 and pose estimation in Section 4.3. Consider one pixel at p that captures light traveling from the calibration object along a ray $\vec{r}$, according to some unknown intrinsic properties of the camera, K:

$$\vec{r} = K^{-1}\hat{p} \tag{5.1}$$

Any ray $\vec{r}$ imaging a lenticular array of the calibration object must satisfy the lenticular constraint:

$$R\vec{n}_{hue} \cdot \vec{r} = 0 \tag{5.2}$$

$$R\vec{n}_{hue} \cdot K^{-1}\hat{p} = 0 \tag{5.3}$$

By transposing one of the terms in the dot product, this constraint can be calculated by matrix multiplication:

$$(R\vec{n}_{hue})^\top K^{-1}\hat{p} = 0 \tag{5.4}$$

which is equivalent to:

$$\vec{n}_{hue}^\top R^\top K^{-1}\hat{p} = 0 \tag{5.5}$$

Given a pixel location p and a $\vec{n}_{hue}$, this is a linear constraint on the 3x3 matrix $R^\top K^{-1}$. The QR decomposition can be used to extract K and R, although practically this constraint is used within a non-linear optimization. In the ideal case, every pixel imaging the chromo-coded calibration object would give a constraint. However, as we analyzed in in Chapter 3, the chromo-coding lenticular arrays are difficult to create with a perfectly consistent HRF across the entire array. This can be mitigated by calibrating multiple HRFs across the array. Therefore, the number of constraints we can accurately derive is limited by the number of calibrated HRFs for a single lenticular array. In the future Section 5.3.2, we explore how many sample points/lenticular constraints is necessary to ensure good camera calibration.

5.2.2 Camera Calibration Algorithm

For each frame, our algorithm follows the following steps to get an initial estimate of the calibration object pose and camera focal length:

1. Find the four corners of the chromo-coded calibration object.

2. Solve for the homography to map image coordinates onto object coordinates.

3. Measure the hues along a grid on the homography, and use these hue measurements and positions to solve for an initial estimate of the rotation and the focal length.

4. Given that initial rotation and focal length, use a nonlinear optimization using the four corners of the lenticular calibration object to get an estimate of the object translation.

5. Given the initial rotation, translation, and focal length, use a second nonlinear reprojection optimization to improve pose and focal length estimations.

In the following 2 sections, we describe the two optimizations used to estimate the camera parameters.

5.2.3 Initial Rotation and Focal Length Estimation

For the initial rotation and focal length estimation, we use an optimization similar to the one used to estimate rotation with chromo-coded markers in Section 4.3, but do not assume we know K. We use an updated objective function which parameterizes K by a focal length f and R by the Rodrigues parameters ρ:

$$\underset{\rho,f}{\operatorname{argmin}} \sum_i \left({\vec{n}^i_{hue}}^\top R_\rho^\top K(f)^{-1} \hat{p}_i \right)^2 \tag{5.6}$$

where R_ρ is the 3x3 rotation matrix for the Rodrigues parameters ρ and $\vec{n}^i_{hue}$ and $\hat{p}_i$ are particular to the i_{th} sample point on the lenticular arrays. K is parameterized by f according

to the same simplification as introduced in Section 4.1:

$$K(f) = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5.7}$$

This simplified intrinsic matrix takes the common assumptions that pixels are square, that there is a single proportional focal length, and that u_0 and y_0 are the center of the image.

5.2.4 Initial Translation Estimation

With an initial R and K estimate, we now estimate T. Consider the j_{th} corner of the chromo-coded calibration object at the 3D location Q_j and its measured location in an image at q_j. To determine an initial translation T, we use the initial estimate of K and R to minimize the reprojection error of all the corner points:

$$\underset{T}{\operatorname{argmin}} \sum_{j} \| q_j - K\left(RQ_j + T\right) \|_2^2 \tag{5.8}$$

We emphasize that in this equation we use the corners of the lenticular arrays, not the sample points across the lenticular array.

5.2.5 Refined Intrinsic and Extrinsic Property Estimation

We refine our initial estimations of K, R, and T by minimizing the hue and point-correspondence reprojection error. The initial estimate is refined by minimizing the following cost function:

$$\underset{\rho,T,f}{\operatorname{argmin}} \sum_i \left(h_i(\rho, T, f, P_i) - hue_i\right)^2 + \mu \sum_j \|g(R(\rho), T, f, Q_j) - q_j\|_2^2 \tag{5.9}$$

where the first term penalizes the difference between the measured hue, hue_i, and the projected hue, $h_i(\rho, T, f, P_i)$, at the i_{th} grid point across all lenticular arrays of the chromo-coded calibration object. This projected hue is a function of the intrinsic and extrinsic properties, ρ, T, f, the pre-calibrated HRF at grid point i, and the direction of the 3D local locations P_i. The second term measures the spatial reprojection error between the observed image location q_j and the projected location, $g(R, T, F, Q_j)$, for the corners of each lenticular array. This projected location depends on the intrinsic and extrinsic properties, ρ, T, f and the 3D location Q_j. The function g is identical to the reprojection done in Equation 5.8. A relative weighting function μ was found empirically to balance hue and position error which are measured in very different coordinate systems and with very different number of samples. In all experiments we show, μ was set to 1/4000.

5.3 Simulated Camera Calibration Experimental Results

In this section, we explore the sensitivity of the lenticular constraint and the calibration object design with a simulated model of our system. First, we explore the effect of noise on rotation and focal length estimations. Second, we test the effect on camera calibration performance when varying the number of sample points used for geometric inference. Finally, we test various designs of the chromo-coded calibration object.

To explore and characterize these various aspects of camera calibration with a chromo-coded calibration object, we create a simulator that allows us to change parameters that control:

- the amount of noise in the measured hue of a simulated lenticular array
- the number of measurements taken from a simulated calibration object
- the orientation and relative positioning of various lenticular arrays that make up a simulated calibration object

This simulator randomly generates a known position and orientation for the virtual calibration object that is modeled to be 110 mm tall and 70 mm wide. This object can appear anywhere in the field of view of a virtual camera from 150 to 850 mm away. In addition, the virtual calibration object cannot be rotated more than 30° from fronto-parallel. Unless experimenting on the design of the calibration object, we use the design detailed in Section 5.1. With a randomly generated position and orientation, the simulator projects the object from a camera's 3D space onto an image plane. We model the pinhole geometry of an iPhone sensor (a 1/3.2" format image sensor at a default 4.1 mm focal length). This image is used to optimize the arithmetic error presented in Equation 5.6 to get a rotation and focal length estimation. These estimations are compared against the true simulated conditions to gauge the performance of the derived geometric constraints.

5.3.1 Sensitivity to Noise

First, we start with exploring the effect of noise on camera calibration. Most of our measurements for geometric inference are the hues observed from the chromo-coding lenticular

arrays at a given pixel location. Therefore, a large potential source of error could be from measuring an incorrect hue. In terms of the lenticular constraints in Section 4.2, this error manifests as an improper direction of $\vec{v}_{hue}$, and thus $\vec{n}_{hue}$. Therefore, to simulate the geometric effects of measurement noise, we introduce normally distributed aberrations to the direction of $\vec{v}_{hue}$. These aberrations are created by randomly choosing a 3D vector from a Gaussian distribution with a given standard deviation, adding that vector to $\vec{v}_{hue}$, and re-normalizing to again get a unit vector.

We start with zero noise as a baseline and add a maximum of 0.2 standard deviation of noise to the unit vector. Figure 5.3 gives insight into the practical effects of adding noise to $\vec{v}_{hue}$ and then computing $\vec{n}_{hue}$, by showing the angular error in the geometric constraint (the computed direction of $\vec{n}_{hue}$) as a function of the standard deviation of the added noise. Considering that our chromo-coding lenticular arrays have an effective range of viewpoints of $\approx$ 70 degrees, 15 degrees of angular error is very significant. Figure 5.4 shows the sensitivity to noise in estimation rotation as a function of the amount of noise.

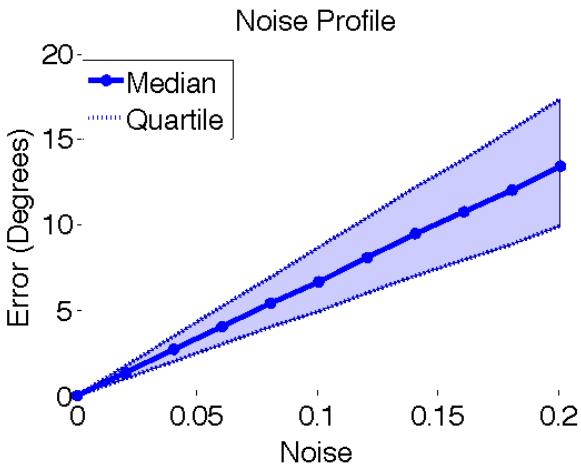
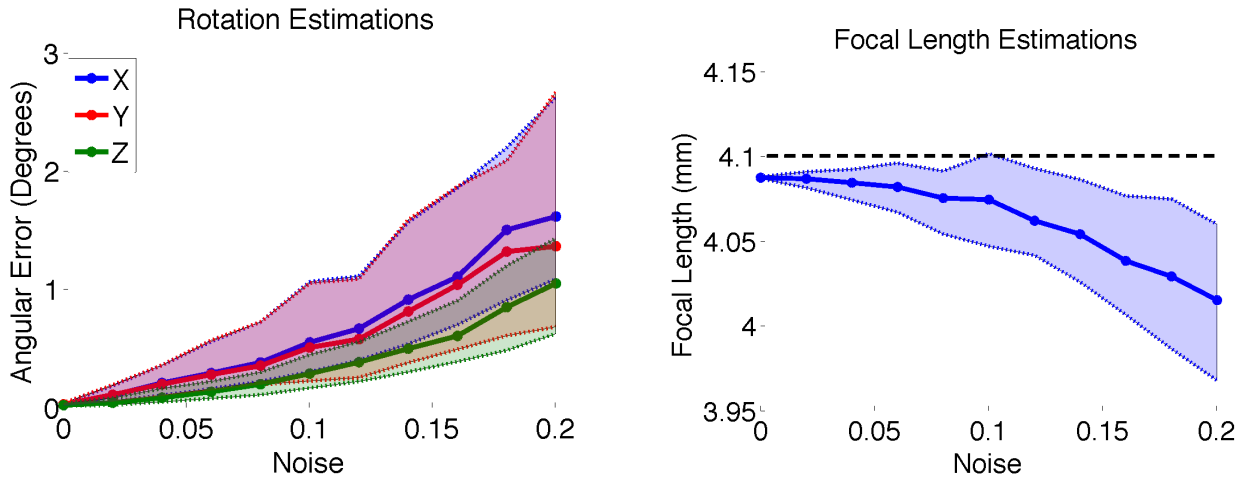


Figure 5.3: This plot shows the quartiles of angular error introduced to $\vec{n}_{hue}$ for a given standard deviation of noise.

In Figure 5.4a, we show the 1st, 2nd (median), 3rd quartiles of the errors in rotation estimation. We display the angular error for each axis of a rotated local reference frame of the calibration object. The angular error for each axis is measured as the difference (in degrees) of a coplanar ground truth projected axis and the estimated projected axis. For all three

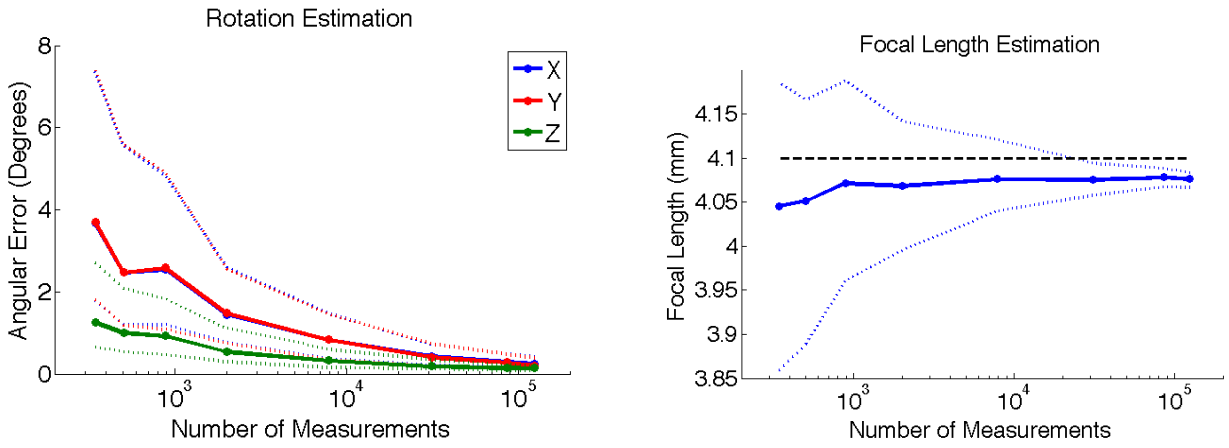


(a) (b)

Figure 5.4: As more noise is introduced to $\vec{n}_{hue}$, the a) rotation error for all axes and b) focal length error increases and is less consistent.

axes, the trend has higher median amounts of rotational error with wider distributions for increasing amount of noise. The x and y axes have slightly more error than the z axis. This is due to the fact that our lenticular arrays are directly measuring rotation *around* the x and y axes and not around the z axis. Thus, error in rotation around the z axis manifests as error in the angular error of the x and y axes. Even at very high noise levels, the median estimated rotation has less than 2 degrees of error.

In Figure 5.4b, we show the 1st, 2nd (median), 3rd quartiles of the errors in estimating the focal length. With more noise, the variance of focal length error gets larger. Perhaps more striking, however, is that with more noise, the focal estimate becomes smaller. We typically see less than 3 degrees of noise in our $\vec{n}_{hue}$ measurements, which corresponds to a noise standard deviation of 0.05 in this figure. At that noise level, our experiment shows focal length errors of about less than 1% and a rotation error of much less than 1 degree.



(a) (b)

Figure 5.5: Using more measurements, and thus geometric constraints, in the optimization results in a lower and more consistent a) rotation error, and b) focal length error.

5.3.2 Sensitivity to Number of Measurements

Because our calibration approach does not need point correspondences for rotation and focal length estimation, it is easy to use a large number of measurements to provide redundancy. Thus, we analyze how the number of measurements of the calibration object increases calibration performance.

We ran 300 trials of randomly generated calibration object poses with 0.08 standard deviation in noise and used an increasing amount of measurements sampled evenly across the calibration object for optimization. Results are shown in Figure 5.5.

In Figure 5.5a, we show the 1st, 2nd (median), 3rd quartiles of the errors in rotation estimation. As more measurements are used, rotation error reduces and becomes more narrowly distributed. The number of measurements does not seem to to affect the median results for focal estimation, shown in Figure 5.5b. However, the estimations become more consistent as more measurements are used in the optimization.

Practically, a measurement represents the hue at one pixel of the image. One can get more measurements of the calibration object by having the calibration object fill more of the image — bringing it closer or by using a higher resolution camera. For subsequent simulation experiments we use about 30,000 measurements (200 x 150 pixels), which are feasible to capture with commodity cameras at reasonable distances.

5.3.3 Sensitivity to Orientation and Relative Position of Lenticular Arrays

The constraints created by observing a single lenticular array are not sufficient to solve for the camera rotation. To get a system of equations for Equation 5.6 that is not rank deficient, we need to include observations of chromo-coded lenticular arrays of at least 2 different orientations. Thus, our chromo-coded light field object must have 2 lenticular arrays, which have major axis in different directions. Beyond this, there is also the design consideration of relative positioning of the differently oriented lenticular arrays. We explore different design choices by simulating various designs with the same simulation system as the previous experiments.

We assess how the relative orientation and placement of lenticular arrays affect the estimation accuracy by creating a large set of designs. Each design is depicted in Figure 5.6(a-g). For each design, we show sample points as blue circles and the direction of the major axis at that sample point is shown by a green arrow. For clarity in the design figures, we reduce the number of sample points. For each design, we run 400 simulations with varied position and rotation, adding 0.2 standard deviation in appearance noise (the maximum tested in Section 5.3.1) to 30,000 measurements. We measure the rotation estimation error for each axis.

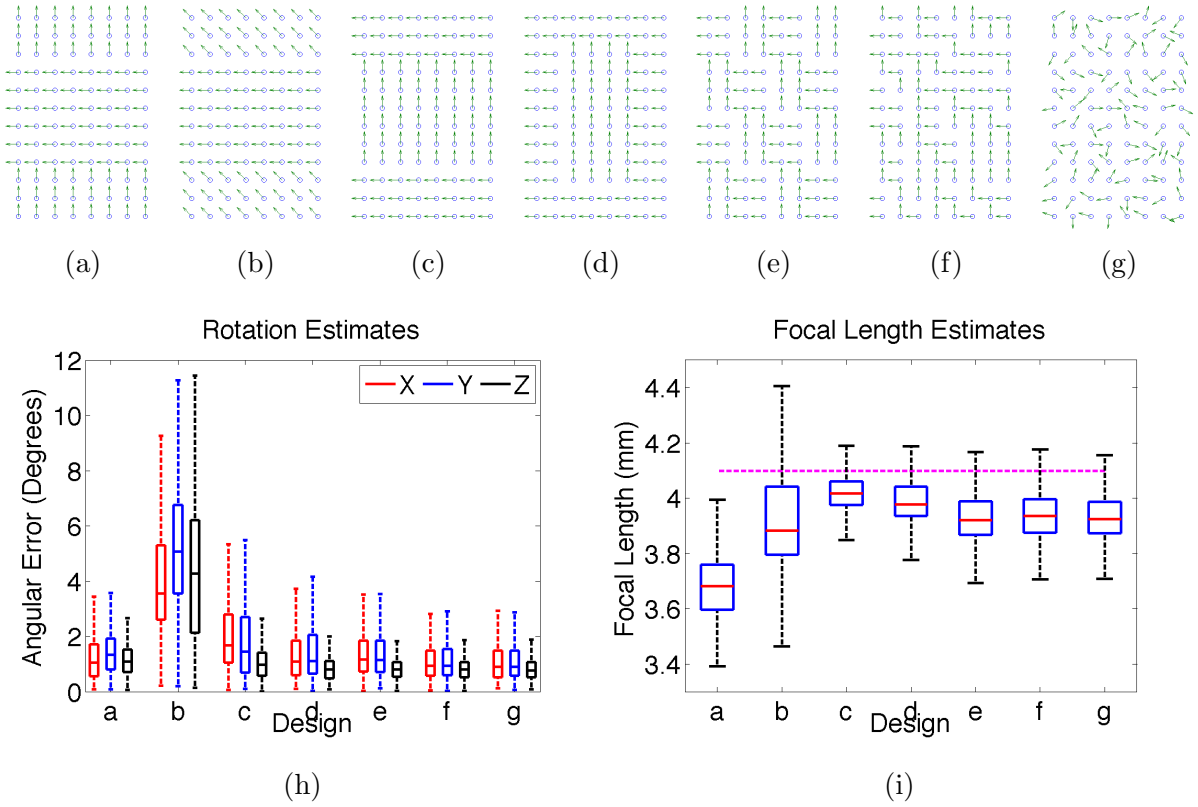


Figure 5.6: Various designs (a-g) change the orientation and relative positions of lenticular arrays to make a calibration object. Estimations achieved for design c) has the lowest combination of h) rotation error and i) focal length error.

We see that the design in Figure 5.6b is the worst in estimating rotation, because the non-orthogonal lenticular orientations give less complementary cues about rotation angle. Otherwise, most of the patterns are similar in estimating the rotation angle, and the design shown in Figure 5.6c gives the best estimate of the focal length. We believe this is because it has parallel lenticular arrays farthest apart, maximizing the angular difference at which parallel lenticular arrays are viewed and therefore maximizing the hue difference at different parts of the array. This is important because this hue difference at different parts of the array is the primary cue for estimating focal length.

5.3.4 Sensitivity to Noise as a Function of Focal Length

To derive the lenticular constraints, we use the hue to lookup the direction of $\vec{n}_{hue}$. If there is any noise in the hue measurement, then the direction of $\vec{n}_{hue}$ would have error. We explored this for various noise levels in Section 5.3.1 for a short focal length (4.1mm).

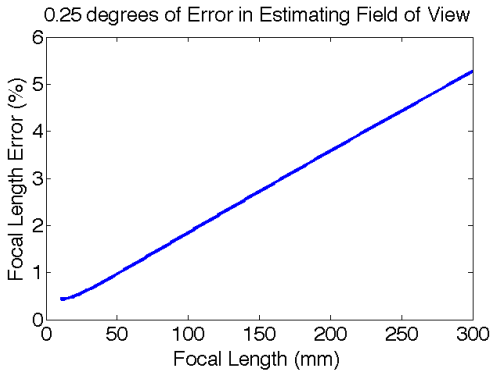


Figure 5.7: The focal length error that arises from mis-estimating the field of view by 0.25° changes as the field of view gets smaller (and, correspondingly, the focal length gets longer).

However, the effects of this type of noise are likely a function of the focal length of the image. Very long focal lengths correspond to imaging geometries with a smaller field of view, whereas a short focal length has a large field of view. Therefore, the same angular error is proportionally much larger for a small field of view than the wide field of view. Therefore, we would expect that long focal lengths would be more affected by noise. To ground this, we show the impact of mis-estimating the field of

view by 0.25° degrees on the estimate of the focal length in Figure 5.7. Although an error of 0.25° degrees is minute, the effect on focal length estimation error is as large as 5% for very long focal lengths (300mm).

5.4 Empirical Camera Calibration Experimental Results

In this section, we experiment with the prototype chromo-coded calibration object described in Section 5.1 for single image camera calibration. In the first group of experiments, we assess the performance of rotation, translation, and focal length estimation across different viewpoints in a laboratory setting. We conclude with an experiment where we use the chromo-coded calibration object for an augmented reality application with a variable zoom.

5.4.1 Rotation Estimation

In our first set experiments, we test in a laboratory environment. On a motorized stage we rotate the calibration object in increments of 5 degrees from -25 to 25 degrees around the vertical axis and take images at each increment. We calibrate the ground truth camera focal length with the MATLAB 2014a implementation of Zhang's method [81].

Figure 5.8 shows the rotation estimation performance per image in the left column as well as in summary in the right column. We show rotation error for each local axis as the angular difference of our estimate to the true rotation. The estimates from our initialization algorithm are shown in the top row and show errors at the scale of a few degrees for all axis. Similar to previous simulated experiments with the chromo-coded calibration object (Sections 5.3.1 and

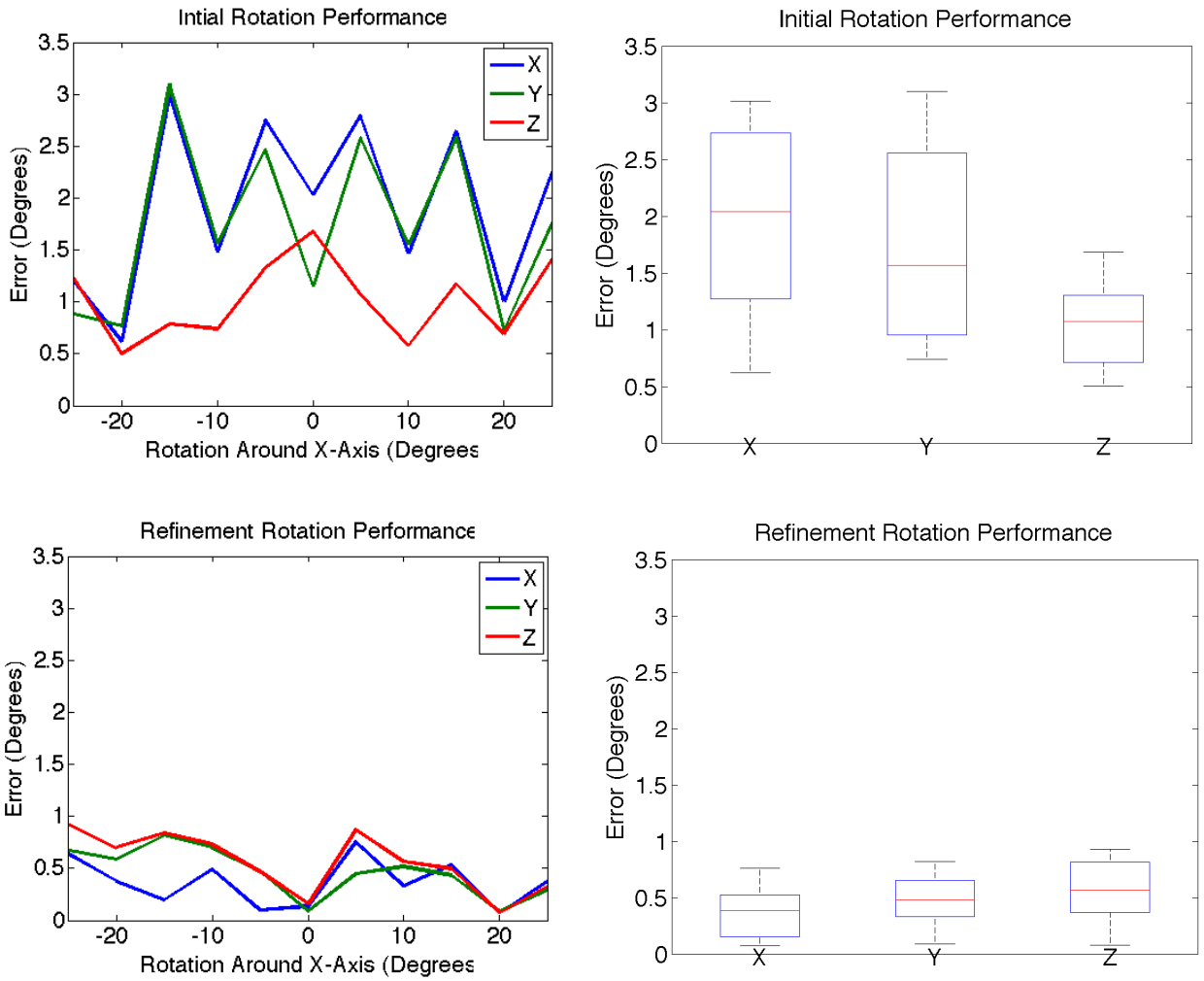


Figure 5.8: We compare the rotation estimations our method gets initially and after refinement. In the left column, we see the rotation error per local axes for each image as the calibration object is rotated, while in the right column we see the summary statistics. Although we start with good rotation estimates (top row), the refinement process still gives improvement (bottom row).

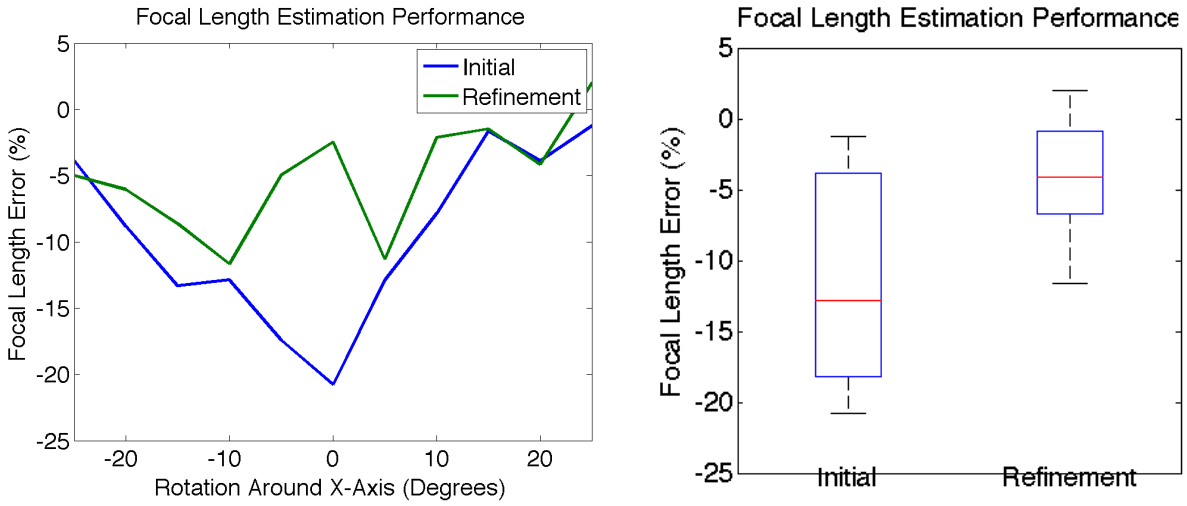


Figure 5.9: We report the focal length estimations for different orientations of the calibration object per image (on the left) and in summary (on the right). The initial estimates (top) start with considerable error in focal length estimation. After reprojection refinement, however, the results are improved significantly achieving a median of less than 5% error.

5.3.3), we generally see a bit more error for the x and y axes. This is because the lenticular arrays give cues about the rotation *around* the local x and y axes, leading to a very good z axis estimate. What is lacking are direct cues about rotation around the local z axis, so any error in estimating the rotation *around* the local z axis shows up as error in the x and y axes in the data. The bottom of this plot shows results after minimizing the reprojection error as defined in Equation 5.9. With this refinement, we see a significant improvement over initial estimations, with a final median error of ≈ 0.5 degrees for all axes. With this refinement, we see the performance discrepancy between z and x,y axes disappear. This is because in the refinement, we employ point-correspondence constraints from the corners of the lenticular arrays which directly constrain the rotation around the z axis.

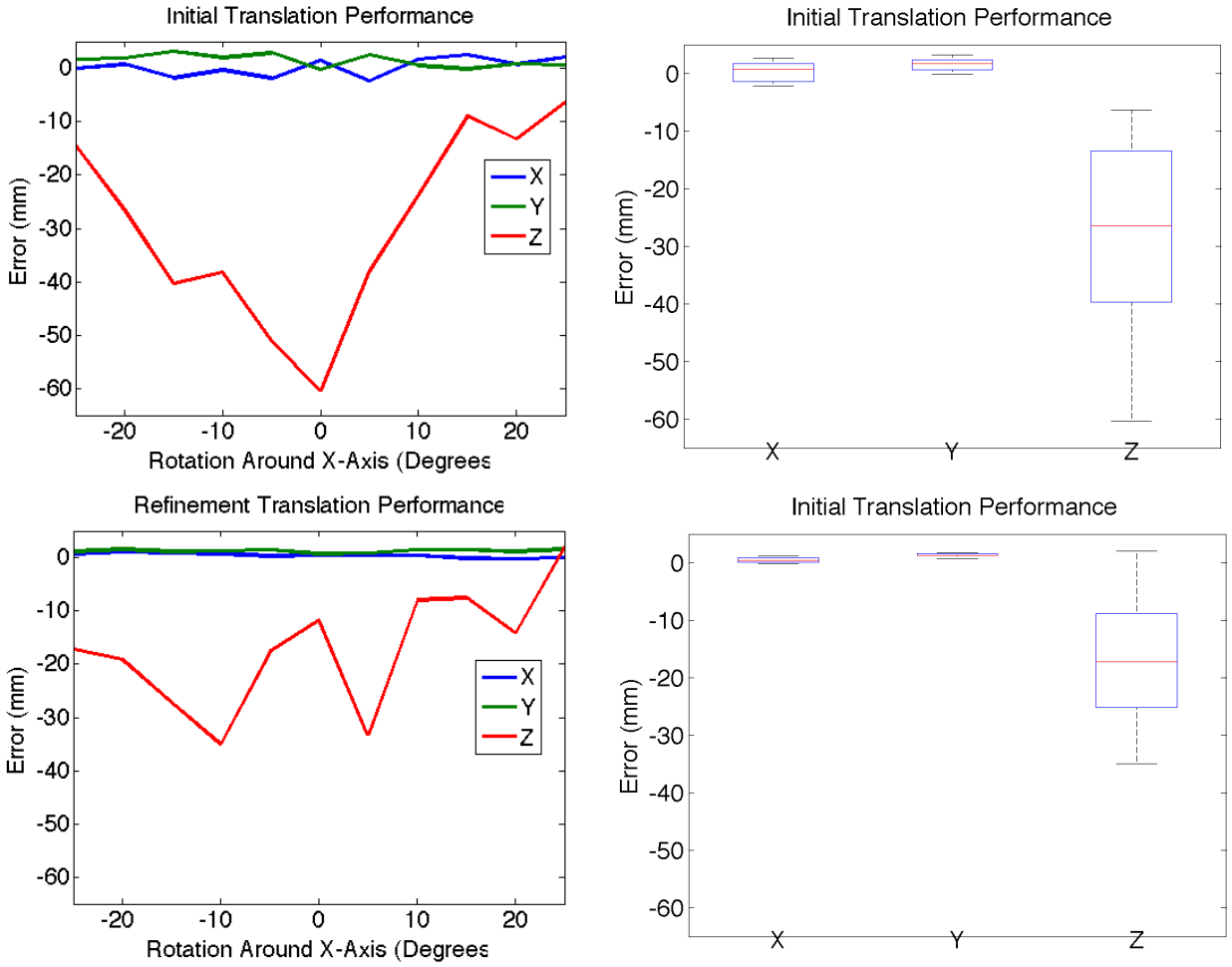


Figure 5.10: We present the per image (left column) and summary performance statistics (right column) for initial translation estimation (top) and refined translation estimation (bottom). In these plots, we show the distance error for each axis. The overwhelming majority of error is in the Z-axis, which is the depth of the camera. The z-axis translation errors reflect errors in estimating focal length.

5.4.2 Focal Length and Translation Estimation

For the same set of rotated views used above, Figure 5.9 quantifies error in the focal length estimation, and Figure 5.10 quantifies error in the translation estimation. Both the initialization and refinement results shown strong correlations between the focal length error and the translation error per frame. The refinement step reduces the error of focal length estimation to a median error of about 4%. The translation estimates show the most substantial error in determining depth. For the images used in this experiment, the camera was placed $\approx$ 1m away from the chromo-coded calibration object. So the refined translation median error of $\approx$ 18mm represents an error of $< 2\%$. The correlation in error between the focal length and the translation arises from the ambiguity that an object can appear bigger either by moving closer to the camera or by the camera changing its focal length. In the Augmented Reality experiments shown in the next experiment, we see that a 4% error does not appear to lead to a perceptually noticeable error in rendering the correct perspective of the object.

To give a sense for the general applicability of camera calibration with the chromo-coded calibration object, we shows quantitative results in Figure 5.11 for focal length and rotation estimation from single images of the calibration object taken at different orientations, different focal lengths, and with different cameras. For each image, we show the results that visualize rotation by rendering the local coordinate system on top of the original image. The image title shows the ground truth focal length, our estimated focal length, and the percent error. We include images from cell phone cameras, as well as a DSLR camera. The first image is from an iPhone 5 and the next two are from a Galaxy S6. These two cameras have focal lengths of 4.4mm and 5.8 mm, which differ slightly from specification numbers reported in other sources because changing focus changes the effective focal length. The images following those are from a Nikon D90 at focal lengths of 18, 49, 90, 115, and 185 mm.

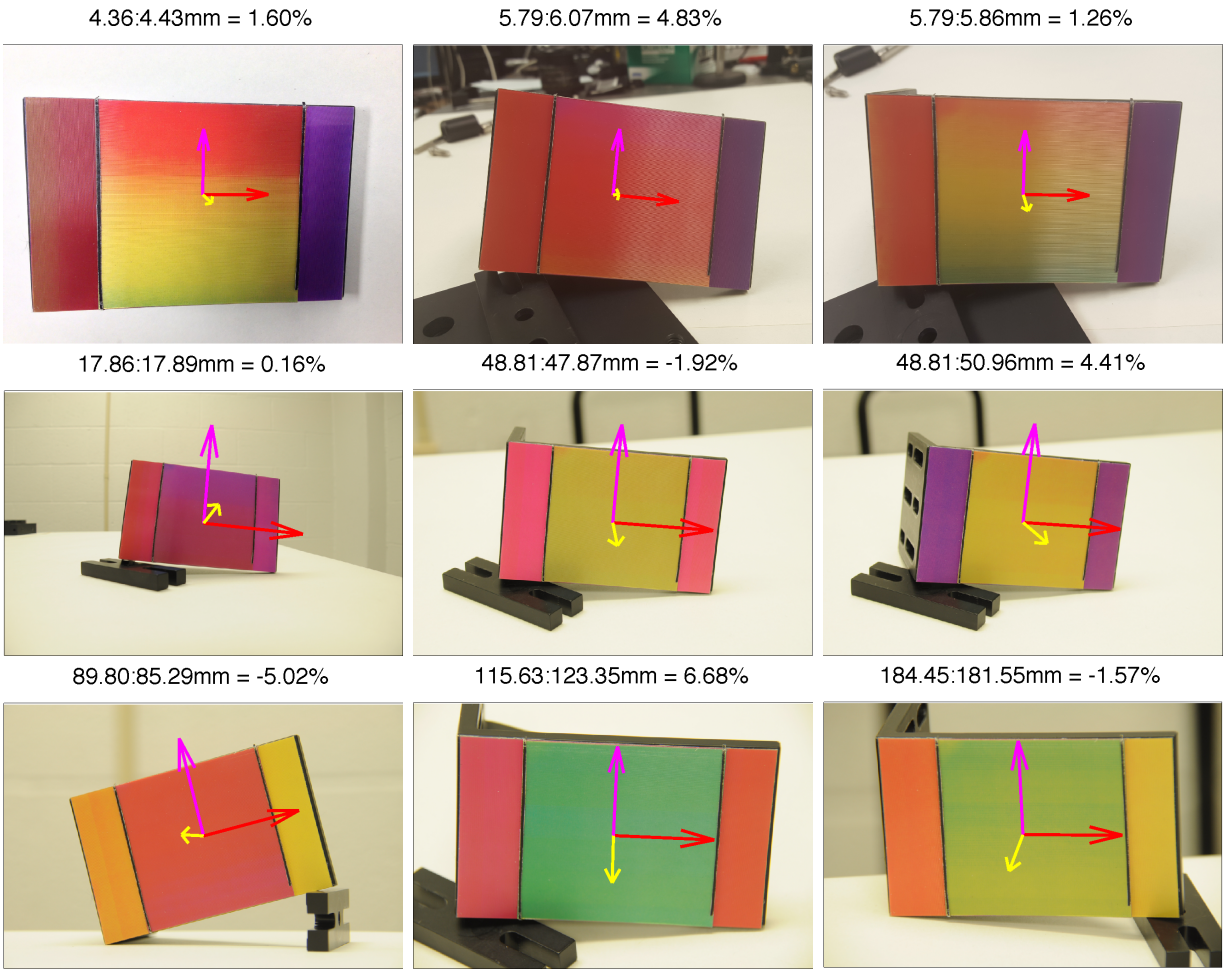


Figure 5.11: Nine examples of single frame estimation of the focal length and object rotation estimates. From the top-left, the first image is taken from an iPhone; the next two, from a Galaxy S6; and the remaining, from a Nikon DSLR camera at different zoom settings.

5.4.3 Augmented Reality Experiments

In this next section, we experiment with the ability to apply camera calibration to an Augmented Reality application. Specifically, we are interested in seeing how the single image focal length calibration is able to give realistic perspective effects of a superimposed 3D object in a video with changing zoom.

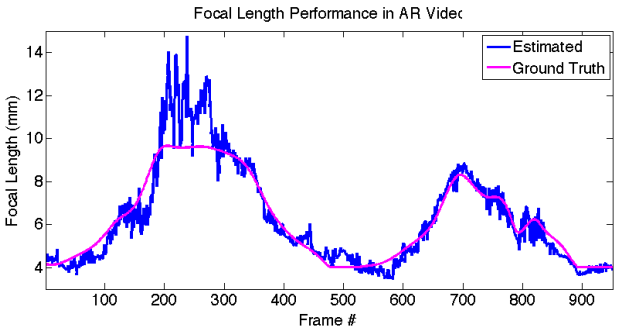


Figure 5.12: We show the focal length estimates used to render a box into a video in our AR experiment. We show how our estimates follow the focal length changes from zooming.

In a desktop scene, we record video of the calibration object while moving the camera in a freehand trajectory. When the camera is moved farther away from the scene and the calibration object, we digitally zoom to keep the calibration object as large as possible in the image. For each frame we estimate the camera focal length, rotation, and translation using the same calibration object as the previous experiment. In Figure 5.12, we compare our estimated focal length with the ground truth focal length (which we know because this is a digital zoom) per frame. We can see that the focal length estimations follow the zooming trajectory well. We emphasize that our algorithm does not have access to this digital zoom information.

As a comparison, we consider an AR algorithm that does not have access to the digital zoom and does not have the ability to estimate it from image data. When such an algorithm uses a pre-calibrated focal length which becomes wrong in part of the video sequence, virtual objects are rendered with incorrect perspective. Figure 5.13 shows 3 frames from the video in each column. We render a virtual wire-frame box to highlight perspective effects. The top

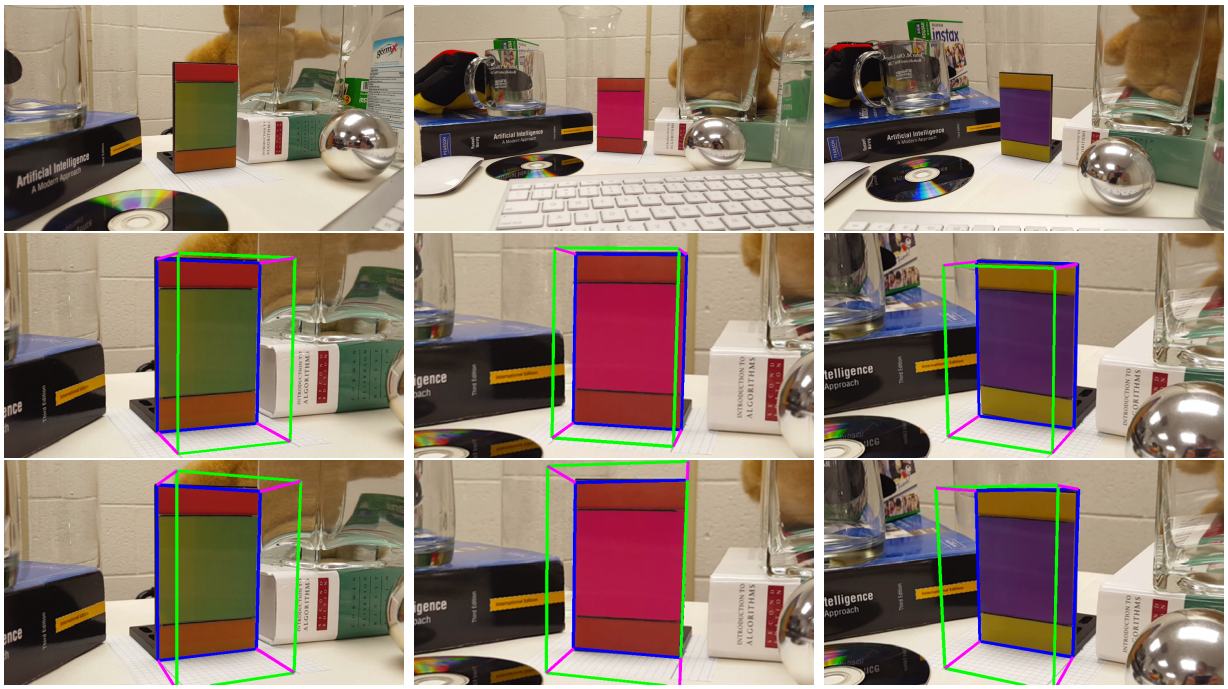


Figure 5.13: We show focal length estimation results in 3 frames of a video where we render a box over the calibration object. The original image (top row) is digitally zoomed to maximize the size of the calibration object in the image. By estimating the focal length dynamically in each image (middle row) versus estimating a single static focal length (bottom row), we achieve a much more natural rendering that is the correct relative size and has the right amount of perspective.

row shows the original images, the center row shows the box rendering given the estimates made with a dynamic focal length (our method), and the bottom row shows the box rendering given the estimates made with a static focal length. The digital box has a base that is the size of the calibration object and is 45mm deep.

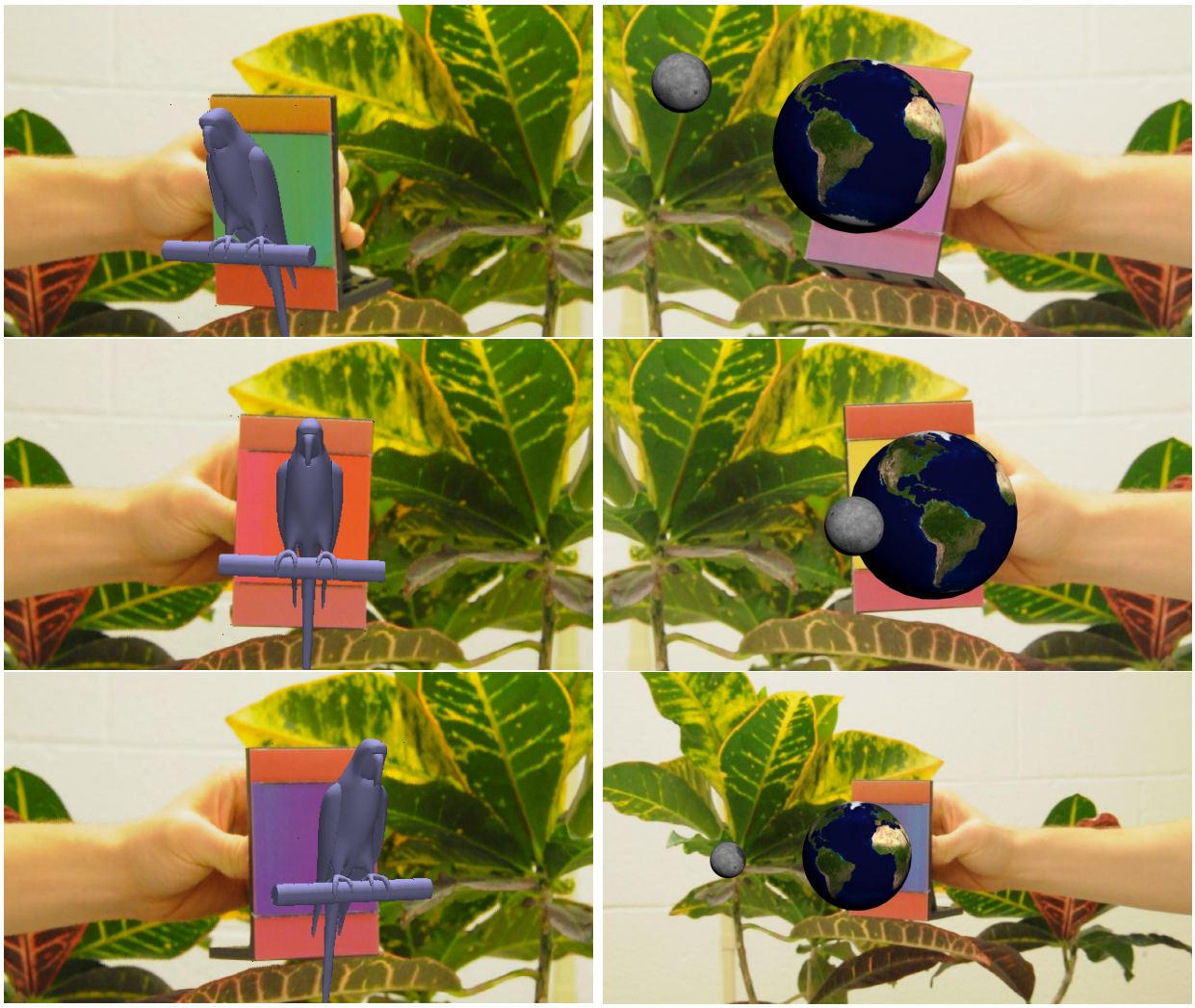
Our scene contains graph paper that is aligned to show a cartesian coordinate to help the viewer assess perspective effects. The wire-frame box should appear aligned just short (10mm or 2 boxes) of the end of the paper grid and should fall parallel to lines on the paper grid. In comparing our method of estimating a dynamic focal length against estimating a static focal length, we see that the rendered box looks unnaturally too large and with too much perspective in the case of a static focal length. This holds true in general for all frames. The full experimental video can be found on Youtube from the linked text.

5.5 Camera Calibration Augmented Reality Application

The camera calibration method described in this chapter is interesting because it can estimation the focal length of a camera using a single image of a relatively simple fiducial marker. Zhang's established method [81] requires 5 to 10 images of a checkerboard at non-parallel positions or two checkerboards on different planes in a single image for reliable camera calibration. This theme holds for any traditional point-correspondence based calibration method. Since our method can do single image calibration with a relatively small fiducial marker, it can be readily used in difficult applications like Augmented Reality (AR) with a variable zoom throughout the input video.

In this section, we show an AR application where users would see 3D objects projected over the calibration object in real time on a portable device with a camera and a screen, such as a tablet. This type of application would be useful for outreach or education. For example, a student learning about the solar system may be able to visually inspect the astronomical relationship between the earth and moon while another student moves the calibration object. Another example may be to expose young children to the variety of animals at the zoo and around the world. Using the calibration object and an AR viewing system, a child could be able to visually see any animal without having to physically go to a zoo, or without being limited by the types of animals at the zoo. Similarly, some moving animal could be displayed and inspected visually at different angles. The child could zoom in on parts of the animal that may be unique or visually interesting. These 2 AR applications may be more intuitive than using some 3D software. Indeed, this type of interaction is more akin to that done with a physical object and so may be more natural and stimulating for a child.

In Figure 5.14, we show frames captured from implementations of these 2 hypothetical examples. We take a video of our prototype camera calibration object rotating in space. While recording the video, we change the zoom by hand. Without an encoder on the lens, it would be very difficult to know the changing focal length per frame in order to correctly project a digital 3D object into the image. With our method, we calibrate the extrinsic and extrinsic properties of the camera per frame in order to realistically project a globe or parrot. In Figure 5.14, we show different frames and different zoom levels with the calibration object at different rotations per row. The full AR videos of the parrot and the earth and moon can be found on YouTube from the linked text. These frames demonstrate the potential to use our camera calibration method for difficult AR input video.



(a) Parrot (b) Earth and its Moon

Figure 5.14: With a chromo-coded calibration object made of lenticular arrays, we can estimate the focal length, rotation, and translation for each frame of a video. This enables Augmented Reality applications even for video with changing focal length. For the same input video, we show two Augmented Reality outputs with different 3D models. On the left, we show different angles of a parrot by rotating the calibration object. On the right, we show the earth and its moon at different angles and zoom levels.

5.6 Conclusion

In this chapter, we described how 3 large lenticular arrays can be combined to make a camera calibration object. This fiducial object enables estimation of the focal length and pose based on the observed hues of the calibration object in a single image. This enables interesting applications, for example Augmented Reality for video input that has a variable zoom.

This camera calibration method is enabled by many lenticular constraints that are derived from hue measurements all across the lenticular arrays, giving 100's of constraints per image. The rotation and focal length is optimized independently from the translation, similar to chromo-coded markers. In empirical experiments, we saw the the calibration method worked with a variety of cameras at a variety of focal lengths, and for a medium focal length we saw median rotation, focal length, and translation estimation errors of ≈ 0.5 degrees, 4% of true focal length, and 2% of the distance to the object.

The calibration object used in this chapter is similar to the chromo-coded markers of the last section. These two fiducials use the same lenticular arrays and derive the same lenticular constraints for geometric inference. As a result, the camera calibration method in this chapter shares the same strengths and challenges as chromo-coded markers relative to traditional fiducial markers: our camera calibration method will estimate accurate orientations regardless of distance to camera or orientation but will have challenges with color measurement and depth estimation when the calibration object is far away from the camera. In this chapter, we did not directly address lighting challenges, but it would be natural to extend the camera calibration reprojection optimization with extra white balancing parameters in a similar way as the chromo-coded markers.

As we saw in Section 5.3.4, our method will suffer from poorer focal length estimations for long focal lengths because a small error in estimating the angle of $\vec{n}_{hue}$ at a sample point is relatively large compared to the small field of view when the calibration object is far away. Because the estimated focal length is correlated with estimated depth, depth estimates are also less accurate for situations where the object is far from the camera.

Chapter 6

Pose Estimation with a Lenslet Array

This chapter considers angularly sensitive patterns that are black and white. This avoids challenges that arise in measuring color because the measured color depends on the light intensity, the surface reflectance, the physical measurement process used by the camera, and post processing often done before the image data is available. This is especially pertinent for lenticular arrays, where hue measurements are used for geometric inference. By using more chromo-coded markers and employing additional optimizations, we were able to mitigate some effects on measuring the correct hue, for example changing lighting environments. However, this added complexity may not be appropriate for different pose estimation settings, for example small hand tools that may not have space for more than 2 markers or real-time mobile pose estimations which have minimal computational resources. Even grayscale measurements require knowledge of lighting and camera response functions, so here we explore the extreme case of thresholding grayscale values into a few discrete categories.

In this chapter, we explore using black and white cues for geometric inference. We explore how to estimate viewpoint with a lenslet array, the 2D analog to the lenticular array, by discretely encoding orientation with a pattern of black or white appearances. Whereas every lenticule in our chromo-coding lenticular arrays changed colors for an orientation change, we

design lenslet arrays where each lenslet has potentially different black or white appearances which change quickly and independently for small orientation changes. Therefore, a range of viewpoints of the lenslet array can be uniquely identified by measuring the discrete black or white appearance of all lenslets in an array. By using additional point-correspondences, we can recover the full pose of the lenslet array.

6.1 Lenslet Array

A lenslet array is a planar plastic sheets that is comprised of many small lenses called lenslets, that are arranged in a regular grid. They are similar to lenticular arrays, except that the individual elements are spherical instead of cylindrical. Typically they are used to diffuse or collimate light, however, they have been used to capture and create view-dependent appearances. In 1908, Gabriel Lippman used lenslet arrays for "Integral Photography"[36]. Here, integral is used in the sense of complete, where lenslet arrays were used to both capture and display the full lightfield in a scene. These lenticular arrays displayed perspective effects as a viewer moved. More recently, lenslet arrays have been used to create lightfield probes to infer about the refraction of light rays through transparent mediums [74, 26, 75]. In these methods, the lenslet arrays showed hue appearances that changed for different orientations.

When the lenslets have a focal length equal to the thickness of the sheet, lenslet arrays can create orientation specific appearances similar to lenticular arrays. A lenslet focuses parallel rays of light from a given orientation onto a specific point on the back-plane of the lenslet array. We show this in Figure 6.2. Adhered to the back of the array is a pattern called the

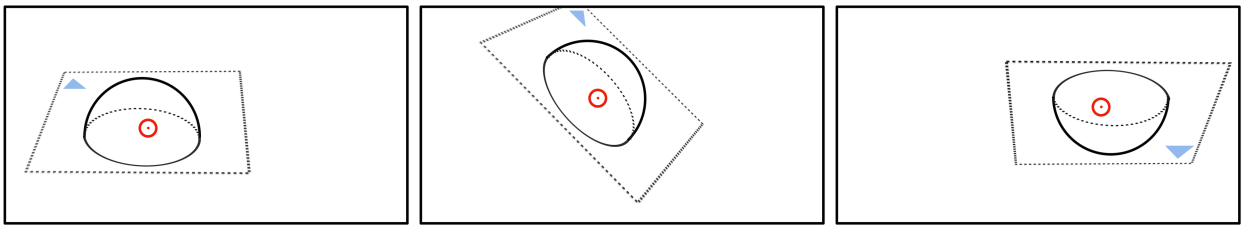


Figure 6.1: We show three hypothetical images of a lenslet. We show the direction from the origin of the lenslet to the camera by its intersecting point on the surface of the lenslet shown as a red dot inside a red circle. From the appearance of the lenslet we may be able to infer the 2D spherical direction to the camera from the lenslet. However, this viewpoint direction does not encode the full rotation that relates the local reference frame of the lenslet to that of the camera. As a result, rotating by any amount (in this case 60 and 180 degrees) around the axis pointing in the direction of the camera will result in the same lenslet appearance for different 3D rotations.

back-plane texture. As a result of the lensing, the lenslet appears to be a magnified version of the back-plane texture underneath the lenslet at the focus point.

Since the lenslet is spherical, the set of orientations which focus onto a different point can be expressed in spherical coordinates; appearances can vary for orientations in 2 dimensions: azimuth and polar angle. This encodes the direction of the viewpoint from a camera to a lenslet array. However, it would not encode the full 3 degrees of freedom of rotation for pose estimation. It cannot because the back-plane texture is only 2 dimensions. There is an ambiguity from rotating around the ray from the viewer to the lenslet array. We show this ambiguity in Figure 6.1. As terminology, to differentiate against the full 3D rotation, we refer to a 2D orientation inferred from a lenslet array as a viewpoint.

6.2 Discretely Encoding Viewpoint with Lenslet Arrays

There are a few lenslet array methods that encode viewpoint direction with color. For example, in the lightfield probe used for Schlieren Photography, the back-plane texture consisted of a circle of different colors which varied conically by hue. If measuring viewpoints by their direction in spherical coordinates relative to the lenticular array, this back-plane texture would encode the azimuth angle. Adding a saturation gradient along the radius of the circle pattern of the back-plane texture would facilitate encoding the polar angle of viewpoints as well. However, as discussed at the beginning of this chapter, these types of color cues would be challenging to measure accurately. We would like to design black and white appearances that change abruptly for small orientation changes.

One way to do this is to employ binary coding on the back-plane texture of a microlens array. In Section 3.2, we proposed a potential design for lenticular arrays which distributes different frequencies of binary patterns across the array. In this case, sections of lenticules would have different back-plane textures to produce appearances which change more or less quickly. In the ideal case, each lenticule would have a different frequency of black and white bars underneath it. However, the lenticules are very small (0.3mm wide) and so would be difficult to measure separately in an image.

Instead, we propose a back-plane texture of many small, randomly placed black blocks. With this back-plane texture, the lenslet array would have an appearance of random flickering as each lenslet changes appearance independently of the others. Figure 6.2 shows two example viewpoints of a lenslet array with this random back-plane texture. Because this texture is

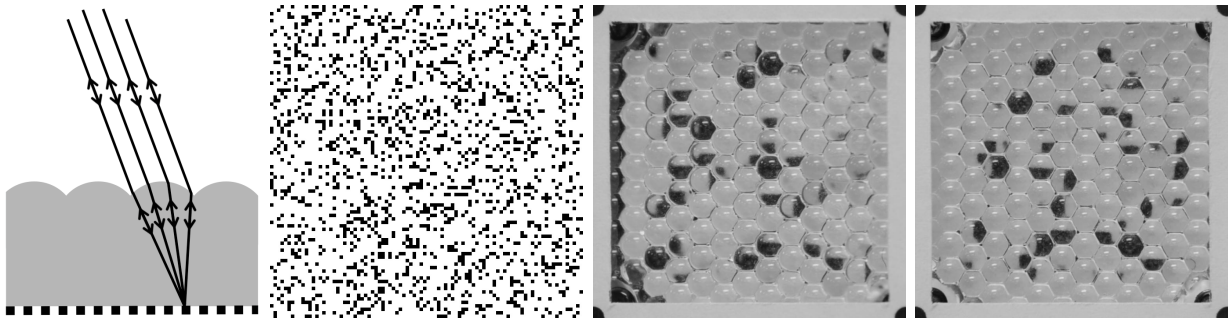


Figure 6.2: A lenslet array placed on top of a pattern creates an image where each lenslet magnifies a piece of the pattern below it. (Left) The lenslet array is designed so that parallel rays focus at a point on the back of the array, so that point is magnified. (Left center) We show an example of a pattern with randomly placed black squares. For different viewpoints, each lenslet will focus at a different location on the pattern and thus produce the different sets of black and white appearances. (Right center and Right) We show the appearances for two different viewpoints after being transformed with a homography to make the images more easily comparable. The appearance of the array changes dramatically, and the discrete measurement of which lenslets are dark and light encodes its orientation.

high contrast, the resulting lenslet appearance changes abruptly for very small viewpoint changes.

By thresholding the grayscale appearance of each lenslet, one can limit a lenslet to 2 states to carry a bit of information. Considered together for a single viewpoint, all lenslets on the lenslet array express a bit string that varies based on viewpoint. In an ideal case, a lenslet array with n independent lenslets, could encode 2^n unique bit strings, and thus 2^n uniquely encoded appearances. With 12 lenslets giving binary measurements, this could suffice to encode viewpoints in the viewsphere up to 30 degrees from fronto-parallel to within 1 degree.

While this intuition of a discrete encoding of pose inspired us, random patterns may not give optimal encodings, and there is value in a non-binary classification of lenslet appearance that has more than 2 states. In the next section we consider practical approaches to choosing

patterns that are most useful and consider what is the best discretization of the apparent lenslet brightness.

6.3 Discretization and Entropy of Single Lenslet Measurements

A single lenslet will share the same appearance for a set of viewpoints because it is magnifying the black and white pattern directly beneath it. This magnified view constrains the orientation at which the lenslet is being viewed. To model this constraint, we consider a measurement of the intensity at the center of each lenslet, and experimentally measure the response across a set of viewpoints to create a response map. We explore thresholding the measured intensity at k intervals. The response map characterizes the apparent intensity of each lenslet when viewed from each orientation, and the state-map is the discretization of that response map in k states. Section 6.4 describes our measurement setup and Figure 6.6 shows examples of the measured response map and discrete state maps.

We characterize the value of a lenslet based on its entropy. The entropy H for a given lenslet and its map from viewpoint to state is:

$$H = -\sum_{i}^{k} p_i * log_2(p_i) \tag{6.1}$$

where p_i is the frequency in the range $(0, 1)$ of a lenslet being in state i for all viewpoints.

A lenslet which is in all states equiprobably has maximum entropy, and it should be easy to design a pattern to put underneath the lenslet array which has this property. However, challenges in accurate printing, aligning a pattern to a lens array, and imperfectly manufactured

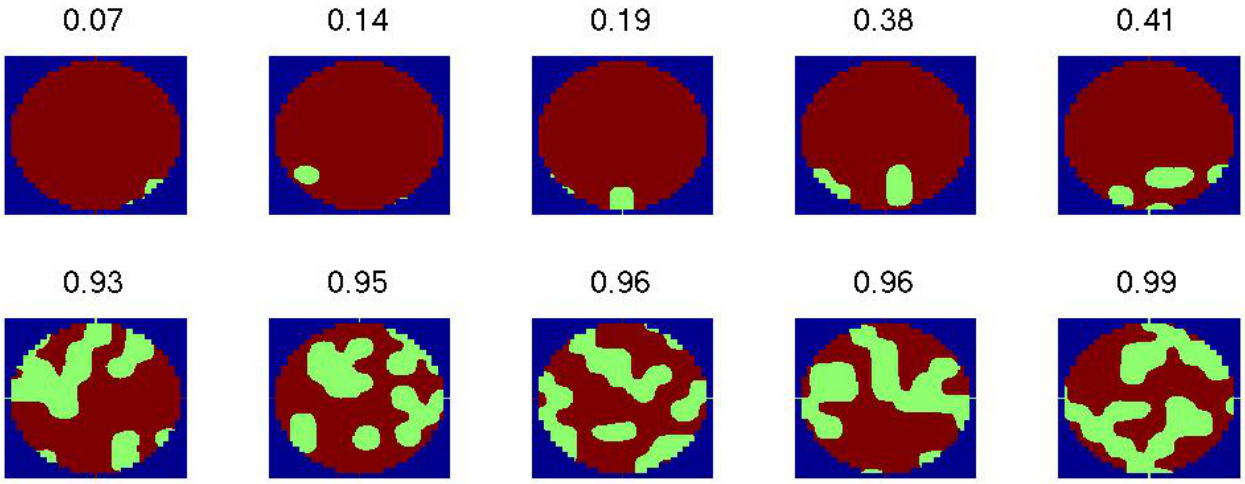
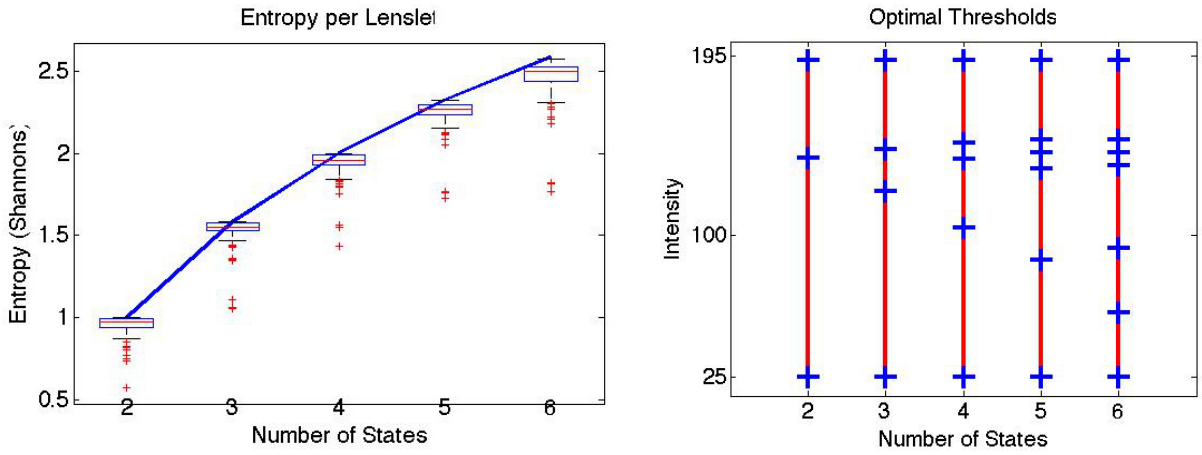


Figure 6.3: Here we show the state maps which show the discrete state maps of lenslets for a viewing dome. Each subplot is a different lenslet, and the color of the x,y coordinate in each subplot is the discrete state of the lenslet for a viewpoint direction that is rotated around that respective axis from -30 to 30 degrees from fronto parallel. Green represents the views where the lenslet is in state 0; red, state 1; and blue for viewpoints not observed. The five lenslets whose discrete state map has the lowest (top row) and highest (bottom row) entropy. The bottom lenslets are much more likely to be useful in estimating orientation; the median entropy of 90 lenslets in an array covering random dots is 0.8.

lenslets that may be out of focus led us to use a random pattern. So, our first question is, what is the entropy of a lenslet array mounted on top of a random dot pattern?

Using the prototype array and texture patten shown in Figure 6.2 we calculate the entropy of each lenslet. Figure 6.3 shows the binary response maps and entropy measures for lenslets that are thresholded at a intensity of 100 (out of 255). We show the 5 lenslets with the smallest entropy in the first row, and the 5 lenslets with the highest entropy in the 2nd row. The median entropy for all lenslets is about 0.8 highlighting that using random textures is a reasonable choice.

In Figure 6.4a, we show the distribution of entropies for all 90 lenslets for optimal discretizations of 2,3,4,5, and 6 states. In Figure 6.4b, we show these optimal thresholds. This shows



(a) (b)

Figure 6.4: We experimentally measure the entropy in discrete measurements from a lenslet array covering a random pattern of small squares. Optimizing to choose discretization thresholds over all lenslets gives a distribution of the measured entropy that is close to the maximum entropy. On the right we show the optimized thresholds. This motivates the use of a random pattern as having nearly as much information as an optimal pattern.

two interesting features. First, measurements of the lenslet intensity are not binary and discrete states can be selected to give a discrete encoding that maximizes the (per lenslet) entropy. Second, when more than 2 discrete states are used, the thresholds often clump, highlighting the value of noticing when a lenslet is changing between black and white.

6.4 Viewpoint Estimation

The state map can be used to characterize the discrete measurement for a lenslet viewed from a given viewpoint. We parameterize the viewpoint with the two-vector Θ which captures the direction of the camera rotated around the local x and y axes, and we define a_i to be the measured appearance of the lenslet. If a lenslet can exist in k states, then the lenslet appearance, a_i, is one of the first k positive integers.

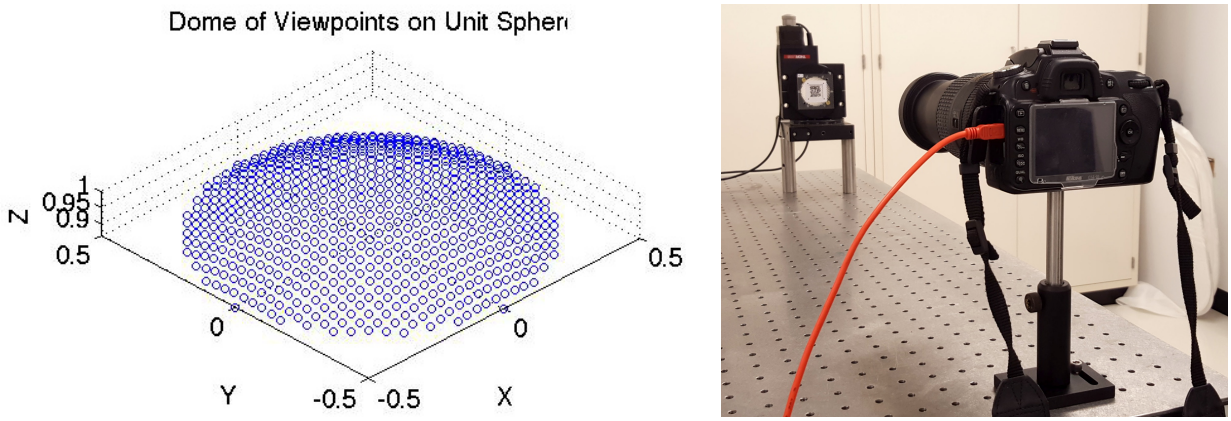


Figure 6.5: The left shows the set of viewpoints from which we image the lenticular array, represented as points on a sphere. The right is a depiction of our image setup which includes a DSLR camera viewing the lenslet array on a 2-axis motorized mount.

We empirically determine the measurement a_i for each rotation Θ by imaging the lenslet array for a grid of viewpoints with a DLSR camera. The lenslet array is rotated with two programatically controlled motors which can change the orientation of the lenslet array to any rotation around the x and y axis. We scanned over the dome of viewpoints that are $(30, -30)$ degrees from fronto parallel in 2 degree increments, yielding 677 images. We visualize the sampled viewing dome and show the motor setup in Figure 6.5.

For each image, four reference points on the corners of the lenslet array are tracked and a homography is used to warp the image to a common coordinate system. Once warped, the center of each lenslet is sampled to create response maps for each lenslet. We show the response map and the responses thresholded into a trinary state map in Figure 6.6.

We up-sample the response maps to have approximate measurements at every 0.5 degrees, using linear interpolation, then threshold to create the state map. The experimental section explores the performance gains for different amounts of up-sampling.

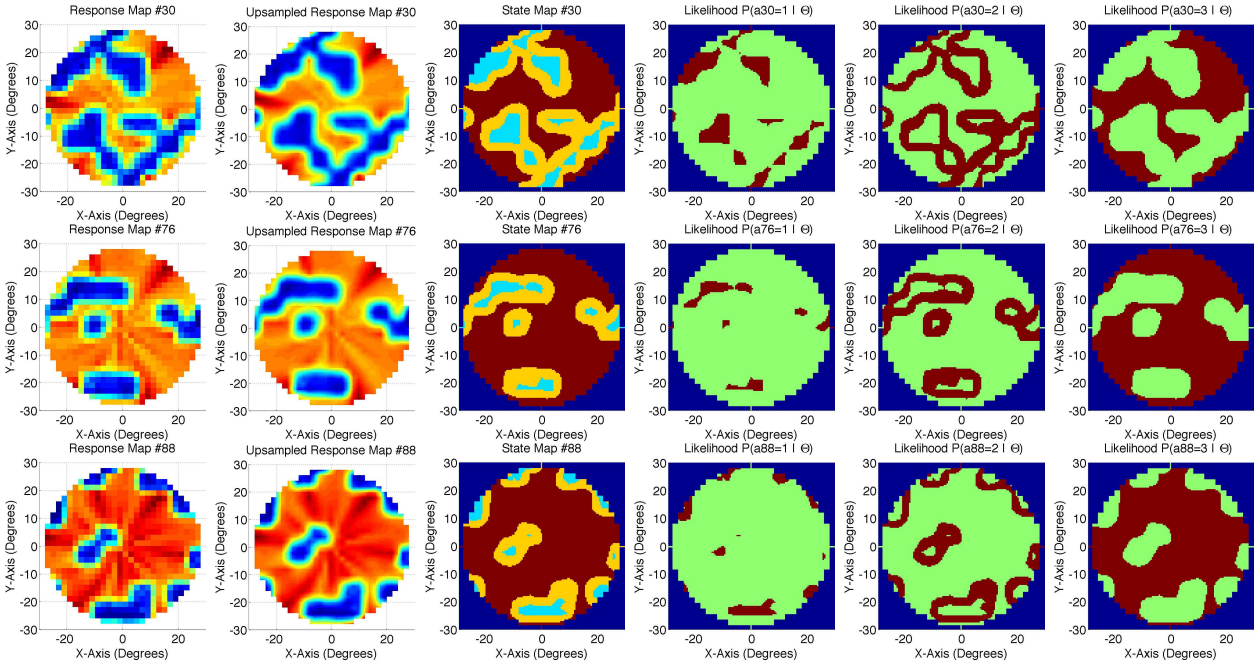


Figure 6.6: For each lenslet, our calibration process characterizes the response as a function of viewpoint. Each row shows a different lenslet. From the left, the figure depicts first the raw, then the up-sampled response map. Third is the discrete state map where the response map is thresholded into one of three categories. The last three maps show the 0-1 likelihood function of the orientation as a function of each of the three possible discrete measurements.

6.4.1 Inference

Using the lenslet array, our task is to determine the 2D viewpoint orientation of camera in the reference of the lenslet array. In the ideal case, the different appearances of each lenslet would uniquely encode the relative viewpoint. With this 1-to-1 relationship, we could employ a simple look-up table to determine a viewpoint. However, this look-up table would be limited to a domain of discrete measurements for a discrete set of viewpoints. Therefore, it would not be robust against discrete measurements from a continuous set of viewpoints. In addition, any noise or error in measuring a particular lenslet could lead to unreliable viewpoint predictions.

Instead, we seek to find the most probable viewpoint given the observed discrete appearance of all lenslets in the array. We employ a simple approach where black and white appearances at each lenslet vote for the most likely viewpoint.

Consider a lenslet at inference time with a state b_i. For all the Θs in that lenslet's statemap with state a_i that match b_i, we give one vote. Each lenslet will have a different statemap and will vote for a different set of Θs.

To estimate the viewpoint using the entire lenslet array, we choose the Θ with the maximum number of votes. In the case of a tie, which sometimes happens for similar Θs, we take the average. This enables inference for continuous viewpoints which lie between the discrete range we have encoded in the statemaps of each lenslet.

In Section 6.7, we explore the ability for a lenslet array to encode viewpoints. We test the design and environmental factors that may impact the ability to infer a viewpoint from a lenslet array. These include the response map granularity, the maximum number of lenslets,

and the light environment, the number of states, and the number of maximum entropy lenslets lenslets. The next section describes how to combine this voting inference method with measurements of the corner points of the lenslet array to determine the full pose of the lenslet array.

6.5 Pose Estimation

The pose estimation problem seeks to estimate the R and T needed to transform the camera reference frame into the lenslet array reference frame. We consider the pose estimation problem for images whose geometry is defined by a pinhole camera model. Using the standard geometric framework, we assume the origin of the camera coordinate system is centered at the pinhole, and the camera calibration K is known. According to this model, a point in the camera reference from P is thus projected to the image location p, via K:

$$\hat{p} = KP \tag{6.2}$$

where $\hat{p}$ is the homogeneous representation of p.

Therefore, a reference point Q in the reference frame of the lenslet array is projected at the pixel location q on the image according to the following linear projection:

$$\hat{q} = K\left(RQ + T\right) \tag{6.3}$$

where $\hat{q}$ is the homogeneous representation of q.

Because the viewpoint estimate derived in the former section is in spherical coordinates, it only gives 2 of the 3 rotation parameters (because the appearance of a lenslet is invariant to rotation around the line from it to the camera). Our approach is to use the θ and ϕ estimate of the rotation of the lenslet around the x and y axis from before. We construct R parameterized by the unknown rotation ψ around the z-axis as:

$$R = R_z(\psi)R_x(\phi)R_z(\theta) \tag{6.4}$$

where $R_{x,z}(w)$ are the rotation matrices to rotate around the respective cartesian axes. We denote this transformation to determine R from θ,ϕ, and ψ as $R(\theta, \phi, \psi)$.

We solve for the rotation parameter ψ and the translation vector T by using the known pixel and local locations of the 4 reference points used to rectify images of the lenslet array and θ and ϕ estimated previously. Using a non-linear optimization, we solve for these four parameters via reprojection error:

$$\min_{\psi,T} \sum_{i}^{4} \left\| q_i - K\big(R(\theta, \phi, \psi)Q_i + T\big) \right\|_2^2 \tag{6.5}$$

where q_i is the measured pixel location of a reference point, θ and ϕ are known and held constant, Q_i is the location of the reference point in the local reference frame, and $\|.\|_2$ denotes the euclidean norm. After optimizing for ψ, we can recover the full R matrix using $R(\theta, \phi, \psi)$.

The estimated R and T capture the orientation and position of the lenslet array relative to the camera. In Section 6.8 we show experimental results with a lenslet array for full pose

estimation for a variety of orientations. We put these results in context by comparing against other fiducial markers which use microlens arrays, including our chromo-coded markers.

6.6 Simulation Experiments on Back-plane Textures

In this section, we explore different back-plane textures. Using a simulator, we generate the discrete lenslet appearances at each viewpoint to gauge the texture's performance. A good back-plane texture will have certain properties, such as the ability to uniquely encode many different viewpoints with as few lenslet as possible. In this section, we first explore a random texture where each pixel has a random and uniform chance of being black. Then, we compare this type of pattern against a more structured binary pattern using Discrete Cosine Transform bases. Finally, we explore a random texture where we enforce a local criteria to prevent a lenslet from creating the same appearance for similar viewpoints.

6.6.1 Properties of an Ideal Back-plane Texture

An ideal back-plane texture will create a set of appearances which have certain properties. Let us consider a lenslet array with binary discrete appearances. Then, we can describe a lenslet discrete appearance with a bit, and all lenslet appearances for a given viewpoint by a bit string, which we call a state string. To describe good back-plane texture properties, we

consider a hypothetical example with 6 lenslets and the state strings for 4 viewpoints:

$$a = \{000100\}$$

$$b = \{000001\}$$

$$c = \{110101\}$$

$$h = \{001110\}$$

With state strings, there are two ways to compare viewpoints. If two viewpoints have similar incident angles to the lenslet array, we say that they are angle-wise close, and if two viewpoints have similar state strings, we say that they are bit-wise close. In this example, the viewpoints a,b,and c are angle-wise close but angle-wise far from h. In addition, viewpoint pairs a b and c h are both bit-wise close.

The first ideal property is that the state strings for all viewpoints are unique and therefore the lenslet array uniquely encodes every viewpoint. In our example above, all viewpoints have a unique combination of bits in their state string to encode the viewpoint. Viewpoint uniqueness, or the percentage of uniquely encoded viewpoints for a dome of viewpoints, is a proxy for how well the lenslet array will perform in pose estimation.

A second ideal property is viewpoint uniqueness with as few bits as necessary. Since using fewer bits would be a more compact representation of all viewpoints, we call this property compactness. A back-plane texture with high compactness makes the system robust to noise, as a bit not in the compact representation is not necessary to differentiate between viewpoints. In our example above, the 3rd and 4th bit are the minimum necessary to distinguish between all 4 state strings. If when measuring the viewpoint a, the first bit flipped to give the measured state string $\{100100\}$, it is still possible to infer that this

state string belongs to viewpoint a. This robustness would translate into pose estimation robustness for varying lighting conditions or camera perspective effects.

A third ideal property is that the most similar state strings for all viewpoints have a maximal Hamming distance [22] between them. We refer to this property as dissimilarity. Typically, viewpoints that are in angle-wise local neighborhoods will have similar state strings. We refer to the dissimilarity in this local neighborhood as local dissimilarity. For example, in the local neighborhood that includes viewpoints a, b, and c, the local dissimilarity is 2 since a and b have a Hamming distance of 2. It can be useful to understand how large of a neighborhood of viewpoints must be excluded before the two bit-wise nearest viewpoints are no longer angle-wise near. For example, by excluding the viewpoints in the neighborhood around a, the closest bit-wise viewpoint is h. We refer to the dissimilarity in this condition as global dissimilarity. By maximizing local and global dissimilarity, we minimize local and global viewpoint inference error. For pose estimation, reducing local error will increase precision, while reducing global error will increase accuracy.

In the experiments in this section, we measure these properties to analyze and compare different back-plane textures with the aim to find the best back-plane texture for viewpoint estimation, and thus pose estimation.

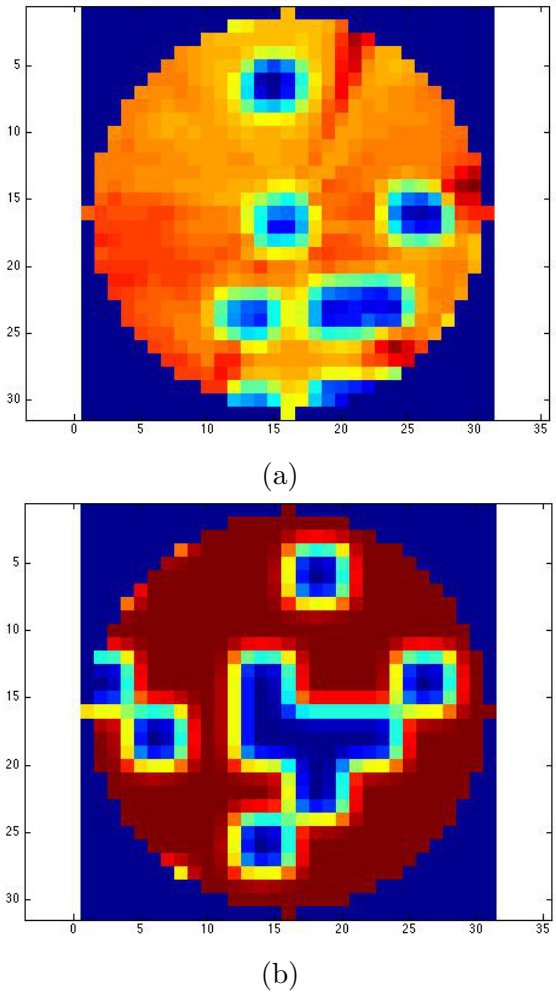
6.6.2 Lenslet Array Simulator

In order to quickly explore various back-plane textures, we create a simulator that allows us to create the back-plane texture and compute the appearance of a virtual lenslet array for a variety of viewpoints. We use a simplified model of the optics to create appearances. This models the lenslet array appearance as the result of integrating the product of a Gaussian

filter placed at a particular location on the back-plane texture. The parameters of the Gaussian were adjusted to give visually similar response maps as shown in Figure 6.7. The position of the Gaussian moves linearly across the back-plane for different angular rotations around the x and y axes, which is consistent with informal laboratory observations.

The output of the simulator is a response map for each lenslet which can be thresholded to create a statemap. For the upcoming experiments, we translate the statemaps for each lenslet into state strings for each viewpoint. State strings are the complement to state maps; the state strings describes the discrete appearance for each lenslet for a particular viewpoint, while a state map describes the set of discrete appearances of a dome of viewpoints for a single lenslet.

In the remainder of this Section, we use the simulator to produce statemaps for various types of back-plane textures. We evaluate different textures and design parameters based on viewpoint uniqueness, compactness, and dissimilarity.



(a)

(b)

Figure 6.7: We use a measured response map (a) to guide the parameters of our simulator to create virtual response maps (b).

6.6.3 Random Texture

In the first experiment, we investigate a back-plane texture that has pixels that are randomly made black according to some probability.

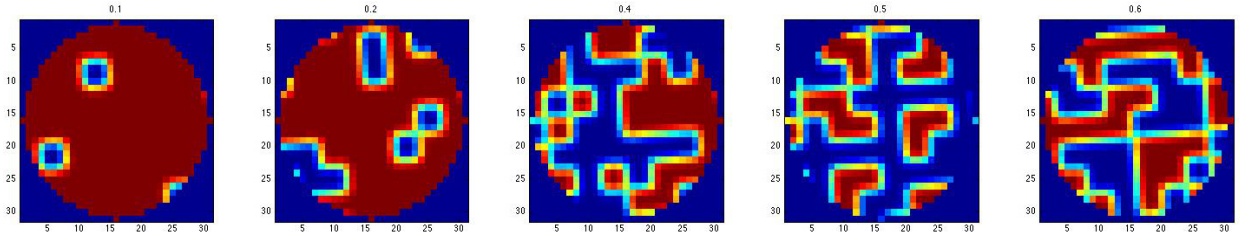


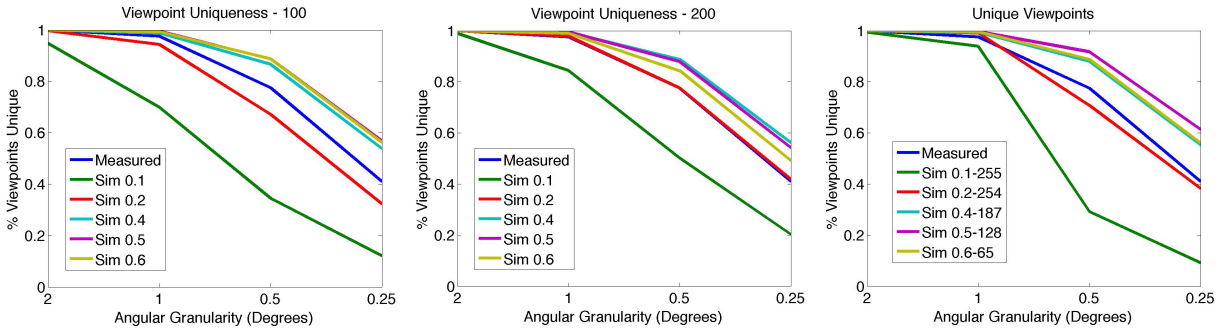
Figure 6.8: Example simulated response maps produced from 5 back-plane textures. From left to right, the respective random back-plane textures are 10,20,40,50, and 60 % black.

The texture has a 10x10 grid of pixels underneath each lenslet. The manufacturing process of mounting a back-plane texture to the lenslet array is challenging, particularly aligning the texture precisely under each lenslet. To model this, we up-sample the generated texture by 10 and slightly rotate the texture. We experiment with different versions of this random back-plane texture where a pixel has a 10,20,40,50, and 60% chance to be black. In Figure 6.8, we show example lenslet response maps output from our simulator for these 5 different back-plane textures.

Using simulated state strings, we evaluate the viewpoint uniqueness of different random textures as a function of the granularity of viewpoints. We test a viewing dome at 2, 1, $\frac{1}{2}$, and $\frac{1}{4}$ degree increments. We examine uniqueness with 3 different thresholding strategies where:

- all textures use a threshold of 100 (out of 255)
- all textures use a threshold of 200
- each texture has a different threshold which maximizes entropy across all lenslets

In Figure 6.9, we show viewpoint uniqueness results for the 3 different thresholding strategies. From these plots, we observe 3 results. First, the viewpoint uniqueness of a measured



(a) (b) (c)

Figure 6.9: A good quality for a back-plane texture is that it produces appearances which are unique for different viewpoints. We show the effect of the state threshold on viewpoint uniqueness for 5 randomly generate back-plane textures. From left to right, we show thresholds for each texture that are: 1) 100 (out of 255) for all textures, 200 for all textures, and individual thresholds which maximize entropy across all lenslets for each texture. The threshold is important for a texture to perform well, however, the best threshold is one that maximizes entropy. The best performing texture with this type of threshold is one that is 50% black, however, it still performs well with non-optimal thresholds.

prototype with 20% black random pattern matches the performance of the 20% simulated textures. This shows that our simulator creates lenslet appearances that give plausible performance. Second, the choice of threshold impacts the viewpoint uniqueness of a back-plane texture. For some textures, a higher or lower threshold improves uniqueness, but ultimately, a threshold that maximizes entropy also maximizes viewpoint uniqueness. In Figure 6.9c, we indicate the optimal threshold in the legend for each texture. Third, a random back-plane texture that is 50% black has the highest viewpoint uniqueness. This texture's performance is maximized with a threshold that maximizes entropy, but also performs well compared to other textures with non-optimal thresholds.

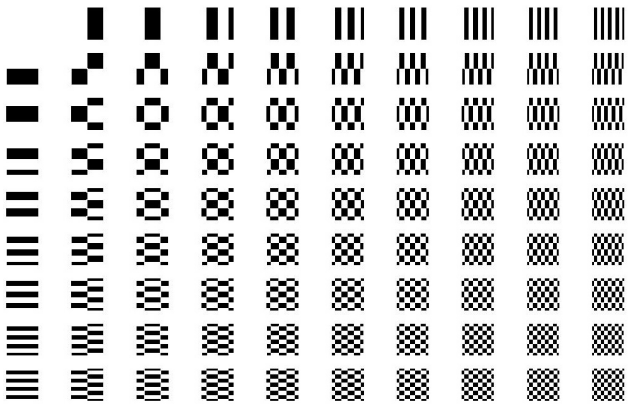


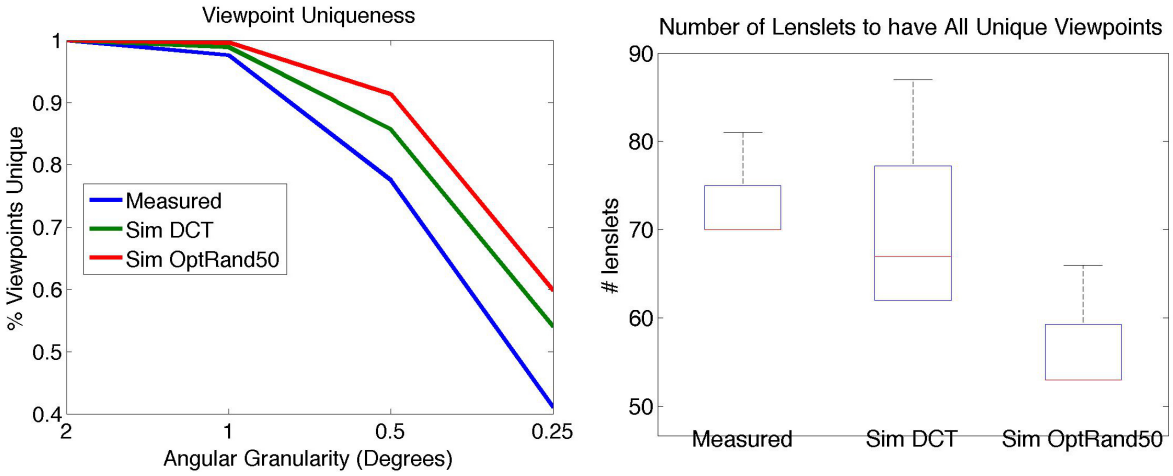
Figure 6.10: The 2D components of the Discrete Cosine Transform that were used in a back-plane texture.

6.6.4 Discrete Cosine Transform Texture

In this section, we evaluate a second type of back-plane texture which uses the Discrete Cosine Transform (DCT). DCT is a way to express a signal as a linear combination of various cosine signals at different frequencies and is most popularly known for its use in lossy image compression standard JPEG [73]. Because the DCT basis has orthogonal components, it seems natural that using each component as a back-plane texture for a lenslet gives an encoding unique to each viewpoint.

We create a back-plane texture by concatenating together a 9x10 grid of 2D DCT components at π radian increments. With this texture, each lenslet has a different spatial and frequency cosine pattern. In Figure 6.10 we show the 90 DCT sinusoidal components used for each lenslet in the back-plane texture. Because we rotate the texture underneath the virtual lenslet array, the texture for each lenslet is not precisely the DCT component.

Using our simulator, we perform a similar experiment to above to determine the viewpoint uniqueness achieved by using a DCT component for the back-plane texture of each lenslet.



(a) (b)

Figure 6.11: A good quality for a back-plane texture is that it produces appearances which are unique for different viewpoints and can do so compactly with as few lenslets as possible. We show the viewpoint uniqueness (left) and compactness (right) for a back-plane texture made from Discrete Cosine Transform bases. This texture has less viewpoint uniqueness and compactness compared to a 50% random texture with an optimal state threshold.

We compare this DCT back-plane texture against the empirically measured back-plane texture use above, and a simulated random back-plane texture with 50% black pixels. For all 3 back-plane textures, we create state strings using discretization thresholds that maximize the average entropy across all lenslets of a given back-plane texture. We show the viewpoint uniqueness results for these 3 textures in Figure 6.11a. The random texture that is 50% black uniquely encodes the most viewpoints.

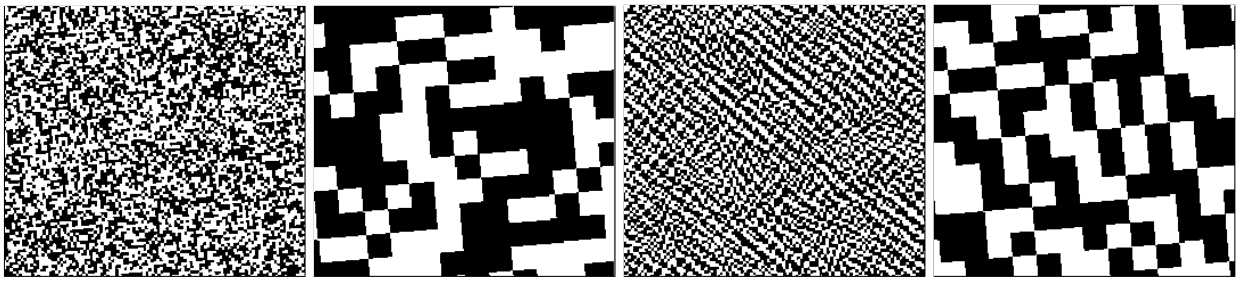
In a second experiment, we evaluate the compactness of these 3 textures. In a Monte Carlo style experiment, we remove a random permutation of bits from the state strings 1 by 1, until all viewpoints are no longer uniquely encoded. This is repeated 3000 times and the number of lenslets needed is recorded each time. We test for viewpoints at 2 degree increments in a dome that is 30 degrees from fronto-parallel. Better textures will encode the same number of unique viewpoints with fewer lenslets.

We show results of this experiment in Figure 6.11b. These results show that the simulated random texture can encode all unique viewpoints with many fewer lenslets and so has a higher compactness. At a minimum, the random texture would need 53 lenslets to uniquely encode all viewpoints. In contrast, the DCT texture would need 62 and the random texture of 20% black would require 70. For this experiment, the distribution of values indicates the best and worst case scenario for expressing all viewpoints with fewer lenslets. The max value represents the adversarial case, where we remove the lenslets that give the most information. The minimum value represents the best case, where we remove lenslets that give redundant information. Both the 50 and 20% textures have the same distribution of lenslets because they have the same type of pattern. The DCT texture, however, has a much larger distribution. This is because the DCT texture has many low frequency components, making it more sensitive to removal of lenslets. The lenslets with back-plane textures of these low frequency components must remain to uniquely encode viewpoints.

From these experiments, we have shown that a back-plane texture made from a random pattern that is 50% black has higher viewpoint uniqueness and compactness, making it the better texture.

6.6.5 Locally Constrained Texture

In this section, we explore a different type of random texture that constrains a local pattern at the scale of a lenslet. With the previous random texture, each pixel was colored independently. As an alternative, we create a texture where we enforce that a small neighborhood of pixels does not have too many or too few black pixels. With this texture, we hope to create

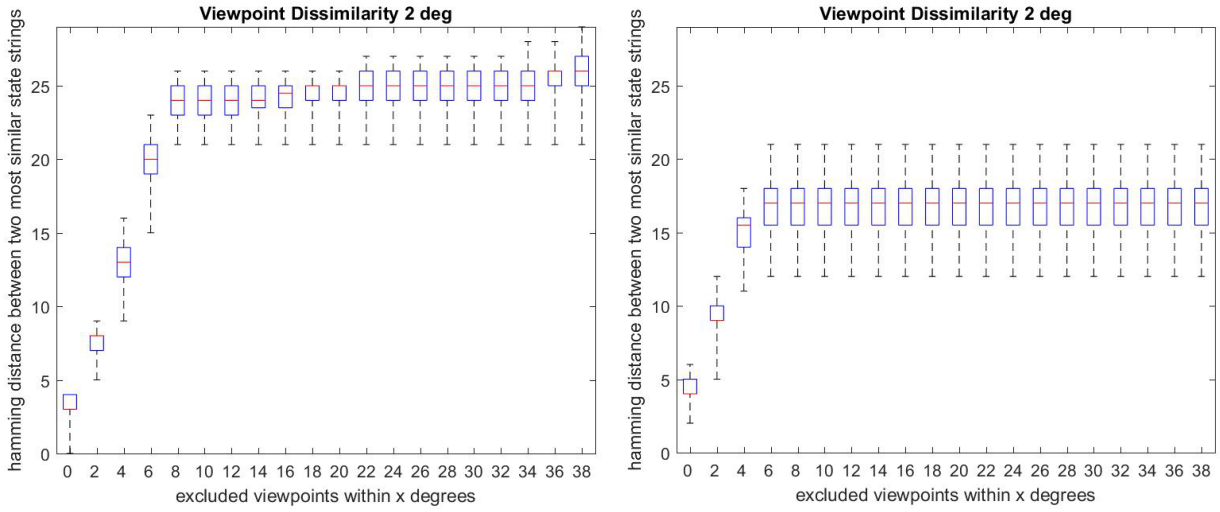


(a) Unconstrained (b) Unconstrained Detail (c) Constrained (d) Constrained Detail

Figure 6.12: We experiment with 2 types of random back-plane textures. In a unconstrained random texture (a), each pixel has a random and independent chance of being black and so big areas of the same color may result (b). In contrast, a locally constrained texture (c), enforces a local criteria that there not be too much black (d) and so creates a more diverse set of appearances for similar viewpoints.

lenslet response maps which transition from black to white as often as possible to encourage local dissimilarity.

We call this new type of texture locally constrained random to contrast with the random texture used above. For this locally constrained random back-plane texture, we enforce a local constraint in a 5x5 pixel neighborhood that each row and column should have 2 or 3 black pixels. To create the texture, we use a linear program to seed a 5x5 neighborhood and then slide a window by 1 pixel and randomly assign pixel color. If the pixel breaks the local constraint, we change the pixel's color. In Figure 6.12 we show examples of a locally constrained random and unconstrained random back-plane texture. When viewing the whole texture in Figure 6.12c, the locally constrained texture has some global structures. However, when zoomed-in in Figure 6.12d, this is not longer visible. This locally constrained texture avoids situations where many pixels in a neighborhood are the same color and thus create the same appearance for many similar viewpoints, as can happen with a unconstrained random texture as shown in Figure 6.12b.



(a) Global Texture

(b) Local Texture

Figure 6.13: A good quality of a back-plane texture is that it produces appearances which are very different for similar and different viewpoints, called dissimilarity. In this plot, we compare this property of two different random textures: (left) 1 unconstrained random texture which has all randomly colored pixels and (right) 1 locally constrained texture which has a local constraint to ensure color diversity. The locally constrained texture has higher local dissimilarity, but lower global dissimilarity compared to the globally random texture.

We perform experiments with these two random textures to measure the local and global dissimilarity of each texture. The simulator creates state strings for these 2 back-plane textures for a virtual array with a 9x10 grid of lenslets. We generate state strings for a dome of viewpoints that differ by 2 degrees. For each texture instance, we measure the Hamming distance between the bit-wise closest viewpoints that are no closer than x degrees in angle-wise distance. We do this for 100 different texture instances, in order to generalize results for a type of back-plane texture.

In Figure 6.13, we show experimental results for the unconstrained (left) and locally constrained (right) random back-plane textures. Along the x axis, we vary the angle-wise distance of viewpoints that are considered. For each size of excluded neighborhoods, we show the distribution of Hamming distances between the two bit-wise nearest viewpoints for all

the back-plane texture instances. We use the median Hamming distance for each excluded neighborhood to analyze the performance of the locally constrained and unconstrained random back-plane textures.

A way to quantify global dissimilarity is by how many bits distinguish the two bit-wise closest viewpoints that are angle-wise far. This depends on the size of an angle-wise neighborhood of viewpoints outside of which is considered far. Figure 6.13 shows the bit-wise distance between the two most similar state stings excluding some angle-wise neighborhood of viewpoints. What is striking in these plots, is that at a certain size neighborhood, the Hamming distance between bit-wise closest viewpoints does not grow. This is the point at which the bit-wise closest viewpoint is *not* the angle-wise closest viewpoint. Therefore, this elbow in the plots defines the size of the neighborhood at which point viewpoints are far enough apart that they no longer are likely to have related state strings and the bit-wise distance at this neighborhood defines the global dissimilarity.

In Figure 6.13, we can see that for the locally constrained texture, this elbow is at 6 degrees, while for the globally random texture, 8 degrees. This difference is a result of constraining the back-plane texture in a small pixel neighborhood. At these neighborhood sizes, the global dissimilarity for the locally and globally random textures are 17 and 24 bits. Therefore, the globally random texture has $\approx 30\%$ more global dissimilarity. This difference in global similarity is a function of the local constraint. There are a finite number of possible 5x5 patterns with our constraint and so lenslet appearances are more likely to be redundant for angle-wise far viewpoints.

A way to quantify local dissimilarity is by calculating how many bits distinguish the two bit-wise closest viewpoints in a local angle-wise neighborhood. The closest angle-wise viewpoints depends on the size of this defined local neighborhood and cannot be smaller than the

granularity of discrete viewpoints. In this experiment, we test with viewpoints every 2 degrees, so we analyze neighborhoods from 2 degrees up to the elbow.

In Figure 6.13, we can see that for the locally constrained texture, the bit difference in the angle-wise neighborhoods of 2,4, and 6 degrees is 4, 9, and 15; while for the globally random texture, it is 3, 8, and 13. This higher local dissimilarity is because of the local constraints that ensure the back-plane texture does not have clumps of the same color and therefore produce angle-wise similar viewpoints that have the same appearance.

This experiment shows that the locally constrained texture has better local dissimilarity but worse global dissimilarity. The dissimilarity allows robustness against noise in measuring the state string. Therefore, the locally random texture will tolerate noise better for small changes in viewpoint, but worse for large changes. However, the global dissimilarity is 17 out of 90 bits, so there is still a large amount of robustness to noise for large viewpoint changes. Because of its dissimilarity properties, we would expect the local random texture to have higher precision and equal accuracy in practice.

6.7 Viewpoint Experiments with a Physical Prototype

In this section, we explore the performance of using lenslet arrays for viewpoint estimation. First, we explore how different design and environmental factors affect viewpoint estimation. Second, we explore how the number of lenslet states and the number of lenslets affect viewpoint estimation.

6.7.1 Experimental Setup

For the experiments evaluating viewpoint estimation, we test on 88 images randomly sampled from the dome of viewpoints 30 degrees from fronto parallel. These images are taken with the lenslet array on the programmable motorized stage. The Θ values are randomly selected, and not at the same location as calibration images, but known to make comparisons to ground-truth possible. We use a lenslet array with a 9x10 regular grid of lenslets, and unless otherwise stated, we use all constraints from all 90 lenslets for viewpoint estimation.

In assessing the accuracy of our viewpoint estimations, we calculate the angular difference between a vector in the direction of the estimated viewpoint direction and a vector in the direction of the true viewpoint. We show summary statistics for all 88 random views as a boxplot. Each boxplot shows the median error in red, a box that shows the range from the 25 percentile to 75 percentile. Red crosses depict outliers, and have values more than 2.7 σ away from the mean, where σ is computed assuming that the data are normally distributed.

6.7.2 Design and Environmental Factors

We first characterize the effect of design and environmental factors on viewpoint estimation. These factors affect estimation regardless of the number of states the lenslets can occupy. Therefore, to simplify this first set of experiments, we employ binary (2-state) measurements and threshold at an intensity value of 100. This threshold reflects general observations that that near white lenslets have intensities above 120 and near black lenslets had intensities below 50.

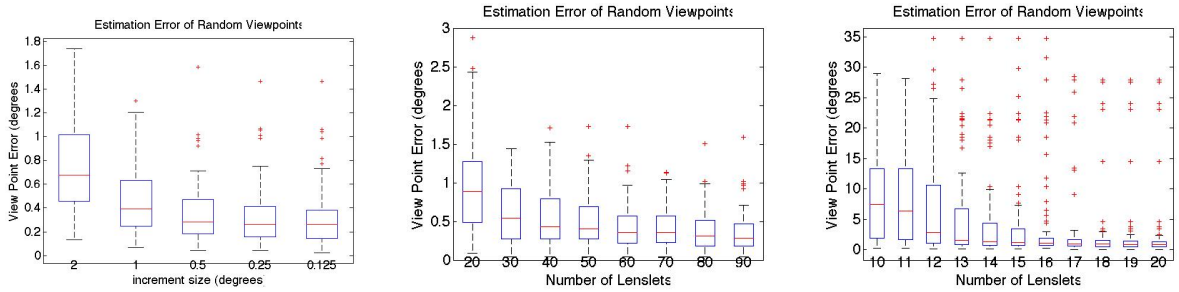


Figure 6.14: (Left) Orientation estimation accuracy, as a function of the angular spacing of the response maps. The original measurements have an angular spacing of 2 degrees; all other data is based on up-sampling and interpolating this response map. (Center and Right) As fewer lenslet provide orientation cues, estimation accuracy goes down.

We first look at performance gains from up-sampling the response map. Second, we explore the effect of the number of lenslets on viewpoint estimation. Finally, we test the lenslet arrays in varying lighting environments.

Response Map Precision

We build response maps for each lenslet by sampling rotations of the lenslet array in 2 degree increments. How much can we improve our rotation estimates by up-sampling these response maps? We create increasingly more up-sampled response maps and assess their orientation estimation performance. Figure 6.14 shows that at the initial resolution, the lenslet array constrains the orientation to a median error of 0.7 degrees. Increasing the precision of the response maps to 0.5 degrees (upsampling and linearly interpolating the response maps by a factor of 4) gives a substantial improvement, and further up-sampling the response maps has little additional benefit. We use this precision for the rest of the paper.

Number of Lenslets

Section 6.2 discusses the potential of combinatorial encodings of orientation, with the claim that in an ideal case 14 binary lenslets is sufficient to uniquely encode $\frac{1}{2}$ degree increments in a viewing dome of 30 degrees. This section gives an experimental evaluation of the correlation between estimation performance and number of lenslets. In this experiment, we randomly select k lenslets, and then perform orientation estimation. Figure 6.14 shows results using 20-90 lenslets in increments of 10 in the center, as well as a finer grain analysis with 10-20 lenslets in increments of 1 on the right. With 20 of the total 90 lenslets, we achieve orientation estimation accuracy with a median error below 1 degree. With fewer lenslets, the performance degrades. With less than 14 lenslets, the maximum error surpasses 10 degrees of error, anecdotally we see that sometimes in these cases there are far apart viewpoints that have very similar discrete encodings. To achieve a median viewpoint estimation error of 0.5 degrees, about 30 randomly chosen lenslets are necessary.

In simulated experiments in Section 6.6.4, we saw that a 50% unconstrained random texture could uniquely encode viewpoints at a granularity of 2 degrees with 53 lenslets. This means that in the worst case, we wouldn't expect more than 2 degrees of error with 53 lenslets. In this experiment, we saw that typically, we were able to infer the viewpoint with 20 lenslets with an error below 1 degree. Although the median error is quite low, on the right of Figure 6.14 we see large outlier errors of 30 degrees. This shows, that 20 lenslets can be used for viewpoint estimation but may suffer from viewpoint estimation error of 30 degrees in the worst case. To limit this worst case error, ≈ 50 lenslets would be necessary.

Light Environments

Related work uses hue to encode orientation with lenticular arrays [54], and one motivation of this work is use discrete measurements of black and white patterns to avoid problems that arise with varying lighting environments. We test the sensitivity of binary lenslet arrays to different lighting environments and exposure settings by exploring 3 different lighting conditions. The first lighting environment is inside under overhead lights. This is the lighting environment used to generate the response maps for all lenslets and is used for all other of the experiments in this paper. The second lighting environment is similar to the first, but with an additional, strong white directional light. The third lighting environment has the scene lit entirely by 2 blue directional lights. In Figure 6.15, we show the estimation results of these 3 different lighting environments. The common office environment with overhead lights achieves the lowest error with a median error of 0.3 degrees. However, even with the very extreme lighting environment of only blue directional light, the lenslet array is able to estimation orientation with a median error of 0.6 degrees. This experiment suggests that even extreme lighting environments have a minimal effect on the binary encoding of orientation given by the lenslet arrays.

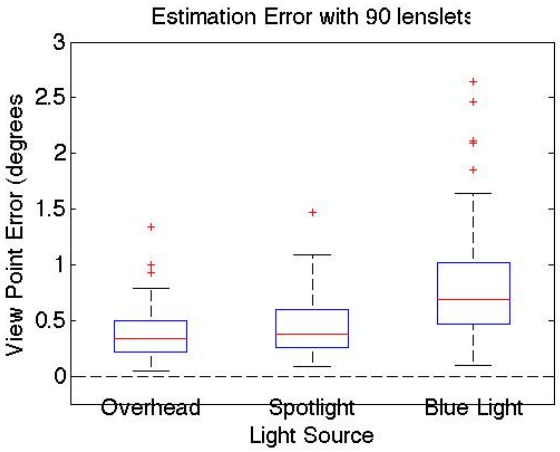


Figure 6.15: Viewpoint estimation error as a function of lighting environment. Overhead lights match the calibration environment, and adding a spotlight light source has minimal impact. Lighting the scene with strong blue lights increases the median estimation error by about half a degree.

6.7.3 Measurement Discretization and Lenslet Selection

Here we explore choices driven by the entropy in the discrete state space for each lenslet. First we experiment with the number of discretized states using all lenslets, and then we determine the effect of using the most informative lenslets.

Number of States

In Section 6.3, we showed that individual lenslets with more states had a larger maximum entropy, and that by optimizing the threshold value allows a random texture to create appearances that have entropies approaching the maximum. In this section, we validate whether there is a corresponding improvement in viewpoint estimation by optimizing for high entropy. From the same response maps, we create state maps with 2,3,4,5, and 6 states. We use the same optimal thresholds for these state maps as described in Figure 6.4. We also create a second binary statemap, but using a threshold of 100, as used in section 6.7.2. To differentiate between the two binary conditions, we label one 2_opt to indicate use of a threshold that maximizes entropy. With each choice of our discretization we get different state maps, and we use these to estimate the viewpoint. We report error versus the known true viewpoint.

The results of using all 90 lenslets are shown on the left in Figure 6.16. In all cases, the median error is less than 0.5 degrees. There is no discernible trend, except that a binary threshold at the entropy maximizing cutoff is worse than a default threshold of 100, perhaps because 100 was hand-chosen to be as far as possible from the appearance of completely white and completely black lenslets.

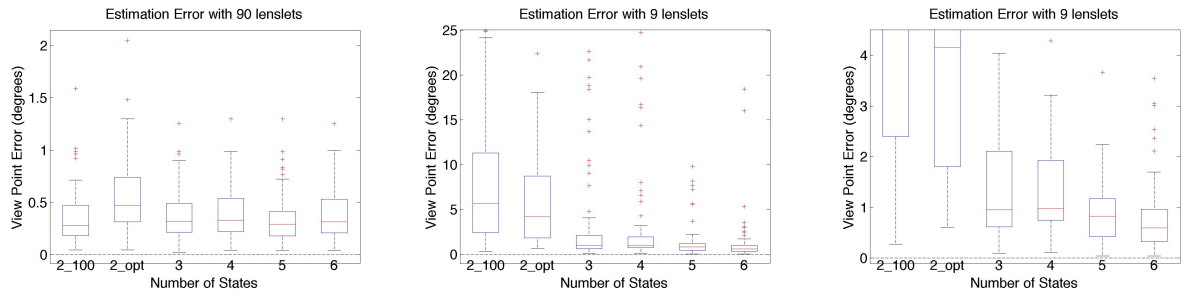


Figure 6.16: The effect of the number and choice of lenslets on orientation estimation. Left, using all 90 lenslets, the median orientation error is less then 0.5 degrees across all discretization choices. The middle and right show the same plot at two scales. They highlight that using the 9 lenslets with highest entropy gives plausible results, and when using fewer lenslets it is especially important to go beyond a binary classification.

Using Minimal Lenslets

In order to test the limits of a the number of lenslets, we tested the viewpoint estimation with the same conditions of the previous section, but only using 9 lenslets. The 9 lenslets used were the lenslets with the highest entropy state maps. The middle of Figure 6.16 shows these results. With fewer lenslets, and less information, the trend suggests that more states results in better estimates. In addition, using an optimal threshold with 2 states shows advantages with few lenslets. The right of Figure 6.16 magnifies the y axis to better analyze estimation results for state maps with more than 2 states. Having more than 2 states results in median errors less than 1 degree. A statemap with 6 states results in the best performance, with 0.6 degrees of median error. These empirical results corroborate that, when a limited number of lenslets can be used, using more states results in better viewpoint estimation performance. However compared to a binary threshold, this is less likely to be robust in high noise environments because more discrete measurements will be wrong.

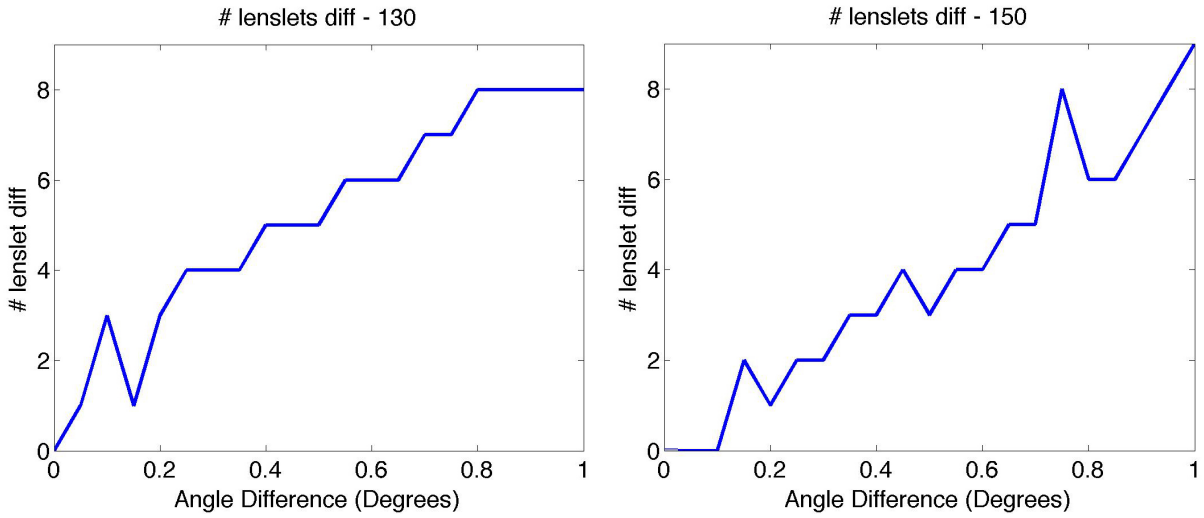


Figure 6.17: We show how many lenslet change states for rotations of 0.05 degrees. On the left, we show results using a state threshold of 130 (of 255) and on the right we show a threshold of 150. Lenslets change state for rotations of around 0.1 degrees, which suggest a precision limit of this lenslet array prototype.

6.7.4 Precision Limit of Lenslet Array

In the previous experiments, the lowest median rotation error was ≈ 0.25 degrees. This error is limited by the sampling granularity of the state map which had viewpoint encodings that were measured for viewpoints every 2 degrees and up-sampled to 0.5 degrees. To encode a high granularity of viewpoints, the lenslets of the array must change states for very small rotations.

Here, we perform an experiment to understand the minimal angle change encodable by the lenslet array. On a similar motor as used above, we started a lenslet array at fronto-parallel and rotated by 0.05 degrees around the vertical axis for a total of 1 degree. We compared the states of each lenslet for each orientation against the fronto-parallel orientation.

In Figure 6.17, we show the number of lenslets with different states for each orientation. We show the results for 2 different choices of state thresholding. For both plots, we see that for rotations of ≈ 0.1 degrees, there is a lenslet that changes state. Due to measurement noise, some lenslets seem to flip states for these very small orientation changes. These results suggest that the minimal angular change that can be encodable is 0.1 degrees. At this granularity, the calibration process to record the statemaps of each lenslet would take considerable longer, however. We estimate that it would take ≈ 30 hours to calibrate viewpoints within 30 degrees from fronto-parallel at 0.1 degree increments. Therefore, we continue to use statemaps that have a granularity that is up-sampled from 2 to 0.5 degrees.

6.8 Pose Estimation Experiments with a Physical Prototype

The unique appearance of a lenslet array results in low error viewpoint estimations. It is therefore useful to use lenslet arrays for pose estimation as well. In the next section, we determine the poses of a new set of images. In these images, we place the lenslet array ≈ 1 m away and rotate it around the vertical axis from 30 to -30 degrees in 1 degree increments. We compare pose estimation to the standard fiducial marker, ARToolkit, by using the same anchor points used by the lenslet array. We also use these results to compare against other published results of a similar experiment.

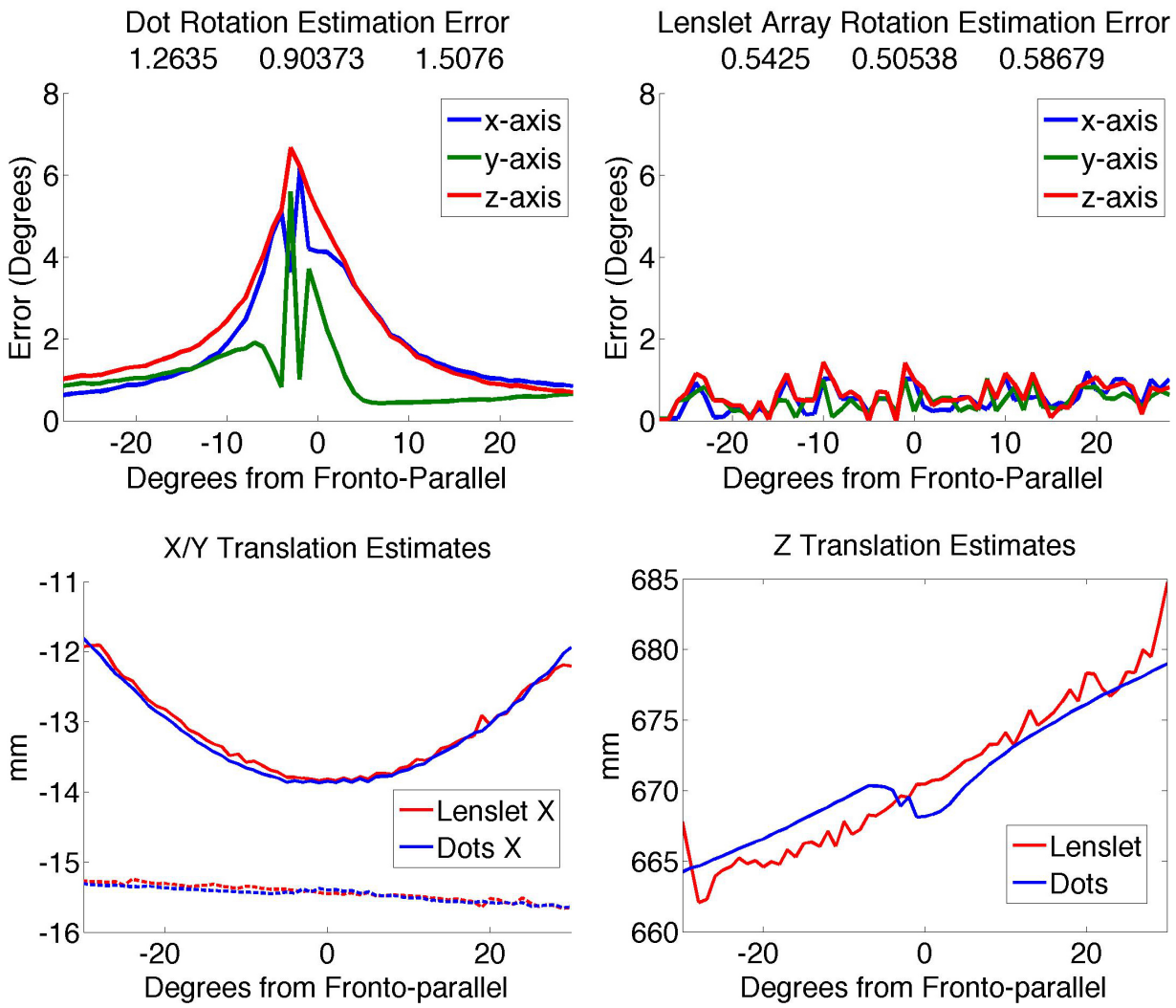


Figure 6.18: We compare the rotation and translation estimation accuracy of our result with the method employed by the popular ARToolkit. In contrast to most position based fiducial markers like those used with the ARToolkit, our fiducial marker does not suffer from the well-known ambiguities of points that lie on a plane near fronto-normal

6.8.1 Direct Comparison to ARToolkit

In Figure 6.18, we show pose estimation results of using the lenslet array. As a comparison, we also include results of using just the 4 reference points as used by ARToolkit [28] to estimate pose.

In the top 2 plots of Figure 6.18, we show the rotation estimation errors per axis using just 4 reference points (left) and using the lenslet array (right). The rotation error is defined as the angular difference between the unit axes using the true and estimated rotations. For each plot, the title shows the median errors over all frames for the axes in x,y,z order. For all views, the lenslet array is able to determine rotations accurately. In contrast, the standard 4 corner method suffers from the well understood ambiguity of points on a fronto-parallel plane [70, 1, 48]. Since the lenslet array gives orientation cues directly from the viewpoint estimation, our method does not suffer from this ambiguity.

In Figure 6.19, we show an example input image with the local reference frame visualized by the unit axis directions from the origin. The X,Y, and Z axis are represented by the red, magenta, and yellow arrows. For the rotation movement around the vertical axis from 30 to -30 degrees in this experiment, we would expect that for translation trajectory, 1) the X component will be parabolic, first decreasing and then increasing, 2) the Y component will be static, 3) the Z component will linearly increase.

In the bottom 2 plots of Figure 6.18, we show translation estimation results of the two types of fiducials. On the left, we show the X and Y components of translation, where the Y component is a dotted line. For both lenslet array and ARToolkit markers, we see the expected translation trends. However, the Y component slowly decreases. This is because of a slight rotation of the camera around the Z axis. On the right, we show the Z component

of translation. The lenslet array produces depth estimates that linearly increase, albeit with some noise. The ARToolkit has a increasing trend, but we see a large deviation near fronto-parallel orientation. These unexpected depth estimates are a symptom of the same ambiguity mentioned above.

6.8.2 Indirect Comparison Against Microlens Array Fiducial Markers

In the final experiment, we share a comparison of our rotation estimation results from above with the results in Section 4.5.3 using 2 chromo-coded markers and the results reported in work that uses the Lentimark [64] and the Arraymark [62]. The Lentimark and Arraymark papers perform similar rotation estimation experiments similar to above, where the fiducial marker is rotated around a single axis. In addition, both papers also report the rotation error for the x,y, and z local axes. Table 6.1 summarizes results of the chromo-coded markers and those reported for the Lentimark, and the lenslet array. The 4 microlens array based fiducial markers have comparably low rotation errors. However, the lenslet array has slightly superior rotation estimation error for the z axes.

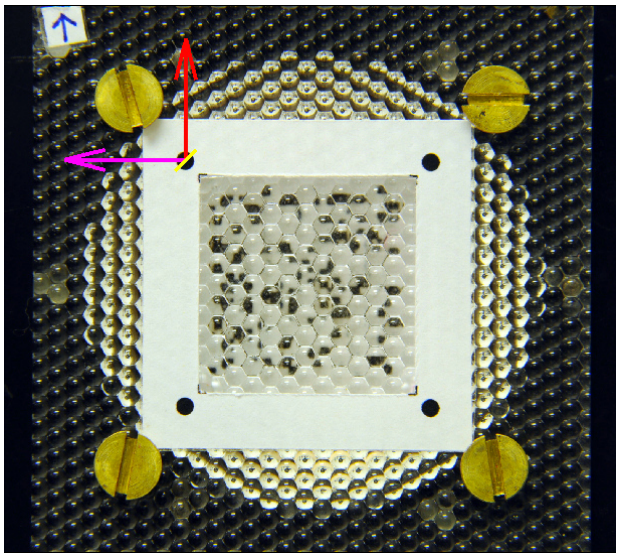


Figure 6.19: Here we show the lenslet array fiducial marker used for pose estimation. Using the lenslet array we can estimate a viewpoint and use that to initialize pose estimation using 4 point-correspondences. In this image, we visualize the local reference frame by the unit axis directions from the origin.

	LentiMark	Chromo-coded Marker	ArrayMark	Lenslet Array
x	0.372	0.910	0.63	0.488
y	1.359	0.703	0.47	0.469
z	0.324	1.222	0.61	0.269

Table 6.1: The lenslet array has similar or better accuracy than other work based on fiducial markers whose appearance is viewpoint dependent.

6.9 Conclusion

In this chapter, we introduce a new type of fiducial marker whose appearance is designed to give a combinatorial encoding of its orientation. This marker was created using a lenslet array with a random texture of black dots. By considering the discrete measurements of all individual lenslets together, a simple voting method allows inference of the viewpoint direction. With black and white cues, this method is robust to extreme lighting conditions such as direct light and monochromatic light. The full pose of the marker is recovered by combining with standard point-correspondence based fiducials.

Despite using point-correspondence based fiducials to infer the full pose, the lenslet array markers do not adopt the same orientation ambiguity for fronto-parallel orientations. We saw this same advantage for chromo-coded markers, however the lenslet array is able to do so in a manner robust to extreme lighting conditions, without the additional complexity of additional markers and optimizations. In our experiments we used an array of lenslets, however, the method would still work for different form factors where lenslets are disjointly organized across the surface.

As we saw in Chapter 5, some of the same rotation constraints derived from lenticular arrays for pose estimation could be used for focal length estimation. Similarly, we believe that the lenslet array could be used for camera calibration. One way could be to use the viewpoint estimations from two lenslet arrays in a single image. Then, the angle between the two viewpoint estimations would depend on the field of view of the camera and could be used to constrain the focal length. This single-image camera calibration method would inherit the same advantages of pose estimation with a lenslet array, such as robustness to fronto-parallel orientations and to extreme lighting environments.

Chapter 7

Conclusion

In this dissertation, we introduced new fiducial markers created from microlens arrays that are designed to create difference appearances for different orientations. Our microlens fiducial markers can be used for pose estimation and camera calibration. These fiducial markers overcome some of the challenges of traditional fiducial markers because they give visual cues on orientation.

To support one type of fiducial marker, we designed lenticular arrays that encode orientation by hue. To do this, we created a back-plane texture comprised of repeating stripes of the hue wheel. We showed how to fabricate and manufacture this chromo-coding lenticular arrays using readily available printers and blank lenticular arrays. The chromo-coding lenticular arrays create a 1-to-1 relationship between orientation and hue in order to constrain the relative orientation of the lenticular array to the camera, independent on the relative position of the lenticular array.

We used small versions of these chromo-coding lenticular arrays to create chromo-coded markers for pose estimation. We derived de-coupled constraints on object rotation to more quickly optimize for pose. Each marker gives position and orientation constraints, so only 2

markers are necessary to solve for pose. With more than 2 markers, we can solve for additional white balancing parameters to make our technique more robust to changing lighting environments. The small size and orientation cues of chromo-coded markers make it useful for pose estimation applications with size and configuration limitations, such as hand tools.

In a similar fiducial marker, we used large chromo-coding lenticular arrays to create a calibration object for camera calibration. Due to the size of each lenticular array, we can derive many constrains from the observed hue. The variation in hue across the chromo-coding lenticular arrays are related to the focal length of the camera, because rays observing the calibration object have different incident angles due to perspective. As a result, the focal length of the camera can be estimated from a single image. This enables applications in Augmented Reality for difficult input videos which have variable zooms.

We also create a different fiducial marker using the 2D analog of lenticular arrays, lenslet arrays. These lenslet arrays are design to produce discrete black and white appearance and address the challenge of measuring color accurately faced by chromo-coding lenticular arrays. These lenslet arrays encode 2D viewpoint direction and, with additional traditional point-correspondences, can be used for pose estimation. Because the orientation cues are from discrete black and white appearances, this pose estimation method is robust against extreme lighting environments.

Exploring Other Visual Cues

In this dissertation, we explored using microlens arrays to encode orientation by an appearance. There may be other materials worth exploring that may also give visual cues for pose. Glitter and holograms have different appearances for changes in light and camera directions. Iridescent materials are also sensitive to changes in light and camera direction. They add but add a dependency on the wavelength of the illuminating light and the relative position of the light and object. There may be still other materials whose color appearance depend on pose and thus could be used for geometric inference, as well.

Beyond pose, there are other physical states such as force and movement that may be interesting to visually measure in a generalized fiducial marker. Photonic gels change color based on mechanical pressure [78] and so might be able to give visual cues to infer about the atmospheric or mechanical pressure on an object. Gold nanoparticle chains change color to record the peak mechanical stress on a polymer[23]. Still yet, materials made from photonic crystals change color from the high pressure of an explosive blast [13]. Further work with these 3 materials could result in a material that can dynamically create color appearances for various forces. Another previous method has used the polarization state of light to measure the stress on tendons [80] and a material designed on this principle may enable dynamic measurement of mechanical stress on an object. Visual cues from single images could enable measurement at single time instances for movement rates such as velocity or acceleration. A material made from arrays of macroscopic pillars changes optical properties when the pillars bend sideways [34, 77] and could be used to measure acceleration. In this dissertation, we created a new type of fiducial markers which gave visual cues for the orientation of an object. These various new materials offer the opportunity to capture even more useful information

about the physical state of an object and could be used in a generalized fiducial marker to better understand the world.

Bibliography

[1] Daniel F Abawi, Joachim Bienwald, and Ralf Dorner. Accuracy in optical tracking with fiducial markers: An accuracy function for artoolkit. In *Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pages 260–261. IEEE Computer Society, 2004.

[2] X. Armangué, J. Salvi, and J. Batlle. A comparative review of camera calibrating methods with accuracy evaluation. *Pattern Recognition*, 35:1617–1635, 2002.

[3] Bradley Atcheson, Felix Heide, and Wolfgang Heidrich. Caltag: High precision fiducial markers for camera calibration. In *International Workshop on Vision, Modeling and Visualization*, volume 10, pages 41–48. Citeseer, 2010.

[4] Patrick Baker and Yiannis Aloimonos. Structure from motion of parallel lines. In *Computer Vision-ECCV 2004*, pages 229–240. Springer, 2004.

[5] Patrick T Baker and Yiannis Aloimonos. Calibration of a multicamera network. In *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*, volume 7, pages 72–72. IEEE, 2003.

[6] Mayank Bansal and Kostas Daniilidis. Geometric urban geo-localization. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3978–3985. IEEE, 2014.

[7] Jean-Yves Bouget. Camera calibration toolbox for matlab. `http://vision.caltech.edu/bouguetj/calib_doc/index.html`. Accessed: 2013-10-09.

[8] G. Bradski. Open source computer vision. *Dr. Dobb's Journal of Software Tools*, 2000.

[9] Alfred M Bruckstein, Robert J Holt, Thomas S Huang, and Arun N Netravali. New devices for 3d pose estimation: Mantis eyes, agam paintings, sundials, and other space fiducials. *International Journal of Computer Vision*, 39(2):131–139, 2000.

[10] Alfred M Bruckstein, Robert J Holt, Thomas S Huang, and Arun Narayan Netravali. Process for pose estimation of a camera viewing an image scene, 1999. US Patent 5,995,214.

[11] Qian Chen, Haiyuan Wu, and Toshikazu Wada. Camera calibration with two arbitrary coplanar circles. In *Proc. European Conference on Computer Vision*, pages 521–532, 2004.

[12] Yisong Chen, Horace Ip, Zhangjin Huang, and Guoping Wang. Full camera calibration from a single view of planar scene. *Proceedings of the International Symposium on Advances in Visual Computing*, pages 815–824, 2008.

[13] Kacy Cullen, Yongan Xu, Dexter Reneer, Kevin Browne, James Geddes, Shu Yang, and Douglas Smith. Color changing photonic crystals detect blast exposure. *Neuroimage*, 2011.

[14] M. Fiala. Artag, a fiducial marker system using digital techniques. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[15] M. Fiala. Designing highly reliable fiducial markers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1317–1324, July 2010.

[16] S Finsterwalder and W Scheufele. Das ruckwartseinschneiden im raum, 1937.

[17] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[18] Clive S. Fraser. *Photogrammetric Camera Component Calibration: A Review of Analytical Techniques*, pages 95–121. Springer Berlin Heidelberg, Berlin, Heidelberg, 2001.

[19] S. Garrido-Jurado, R. Muñoz Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, June 2014.

[20] Todor G Georgiev. Plenoptic camera, November 17 2009. US Patent 7,620,309.

[21] J. A. Grunert. Das pothenotische problem in erweiterter gestalt nebst uber seine anwendungen in geodasie. *Grunerts Archiv fur Mathematik und Physik*, 1841.

[22] R. W. Hamming. Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160, April 1950.

[23] Xiaogang Han, Yiding Liu, and Yadong Yin. Colorimetric stress memory sensor based on disassembly of gold nanoparticle chains. *Nano Letters*, 14(5):2466–2470, 2014. PMID: 24712540.

[24] Adam Herout, Michal Zacharias, Markéta Dubská, and Jiri Havel. Fractal marker fields: no more scale limitations for fiduciary markers. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 285–286. IEEE, 2012.

[25] Matthew Hirsch, Gordon Wetzstein, and Ramesh Raskar. A compressive light field projection system. *ACM Transactions on Graphics (TOG)*, 33(4):58, 2014.

[26] Yu Ji, Kinwei Ye, and Jingyi Yu. Reconstructing gas flows using light-path approximation. In *International Conference on Computational Photography (ICCP)*, 2013.

[27] R. Barry Johnson and Gary A Jacobsen. Advances in lenticular lens arrays for visual display. In *International Society for Optics and Photonics*, 2005.

[28] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IEEE and ACM International Workshop on Augmented Reality*, 1999.

[29] Bong Keun Kim, Hideyuki Tanaka, and Yasushi Sumi. Robotic wheelchair using a high accuracy visual marker lentibar and its application to door crossing navigation. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4478–4483. IEEE, 2015.

[30] Bong Keun Kim, Hideyuki Tanaka, and Yasushi Sumi. Uml-based design of a robotic wheelchair system for indoor navigation using a visual marker. In *Control, Automation and Systems (ICCAS), 2015 15th International Conference on*, pages 562–564. IEEE, 2015.

[31] Timothy J Knight, Yi-Ren Ng, Colvin Pitts, and Alex Fishman. Light field camera image, file and configuration data, and methods of using, storing and communicating same, February 10 2010. US Patent App. 12/703,367.

[32] Zuzana Kukelova, Martin Bujnak, and Tomas Pajdla. Closed-form solutions to minimal absolute pose problems with known vertical direction. In *Computer vision–ACCV 2010*, pages 216–229. Springer, 2011.

[33] Eric J. Larsen. Tracking head position and orientation. US Patent 8922644, 2014.

[34] Elaine Lee and Shu Yang. Bio-inspired responsive polymer pillar array. *MRS Communications*, 2015.

[35] Marc Levoy, Ren Ng, Andrew Adams, Matthew Footer, and Mark Horowitz. Light field microscopy. *ACM Transactions on Graphics (TOG)*, 25(3):924–934, 2006.

[36] G. Lippmann. La photographie integrale. *Comptes-Rendus*, 1908.

[37] Ameesh Makadia, Christopher Geyer, and Kostas Daniilidis. Correspondence-free structure from motion. *International Journal of Computer Vision*, 75(3):311–327, 2007.

[38] Ameesh Makadia, Christopher Geyer, Shankar Sastry, and Kostas Daniilidis. Radon-based structure from motion without correspondences. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 796–803. IEEE, 2005.

[39] MATLAB. Matlab and single camera calibration app release 2014a, 2014.

[40] Wojciech Matusik and Hanspeter Pfister. 3d tv: a scalable system for real-time acquisition, transmission, and autostereoscopic display of dynamic scenes. In *ACM Transactions on Graphics (TOG)*, volume 23 (3), pages 814–824. ACM, 2004.

[41] R Melo, M Antunes, JP Barreto, G Falcao, and N Gonçalves. Unsupervised intrinsic calibration from a single frame using a" plumb-line" approach. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 537–544. IEEE, 2013.

[42] Isao Miyagawa, Hiroyuki Arai, and Hideki Koike. Simple camera claibration from a single image using five points on two orthogonal 1-d objects. *IEEE Transactions on Image Processing*, 19(6), 2010.

[43] Ankit Mohan, Grace Woo, Shinsaku Hiura, Quinn Smithwick, and Ramesh Raskar. Bokode: imperceptible visual tags for camera based interaction from a distance. *ACM Transactions on Graphics (TOG)*, 28(3):98, 2009.

[44] Ren Ng, Marc Levoy, Mathieu Bredif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. Technical report, Stanford University Computer Science, 2005.

[45] Takanori Okoshi. *Three-dimensional imaging techniques*. Elsevier, 1976.

[46] Tomas Pajdla. Localization using svavisca panoramic images of agam fiducials- limits of performance. Technical Report CTU-CMP-2001-11, Czech Technical University in Prague, 2002.

[47] Tomas Pajdla. Simulating svavisca panoramic images of agam fiducials. Technical Report CTU-CMP-2001-01, Czech Technical University in Prague, 2002.

[48] Katharina Pentenrieder, Peter Meier, Gudrun Klinker, et al. Analysis of tracking accuracy for single-camera square-marker-based tracking. In *Proc. Dritter Workshop Virtuelle und Erweiterte Realitt der GIFachgruppe VR/AR, Koblenz, Germany*, 2006.

[49] Christian Perwass and Lennart Wietzke. Single lens 3d-camera with extended depth-of-field. In *IS&T/SPIE Electronic Imaging*, pages 829108–829108. International Society for Optics and Photonics, 2012.

[50] Meghshyam Prasad, Bharat Chandran, and Michael Brown. A motion blur resilient fiducial for quadcopter imaging. In *Proc. IEEE Workshop on Applications of Computer Vision (WACV)*, 2015.

[51] Wang Qi, Fu Li, and Liu Zhenzhong. Review on camera calibration. In *2010 Chinese Control and Decision Conference*, pages 3354–3358, May 2010.

[52] Fabio Remondino and Clive Fraser. Digital camera calibration methods: considerations and comparisons. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5):266–272, 2006.

[53] Olivier Saurer, Friedrich Fraundorfer, and Marc Pollefeys. Homography based visual odometry with known vertical direction and weak manhattan world assumption. In *Vicomor Workshop at IROS*, volume 2012, 2012.

[54] Ian Schillebeeckx, Joshua Little, Brendan Kelly, and Robert Pless. The geometry of colorful, lenticular fiducial markers. In *International Conference on 3D Vision*, 2015.

[55] Ian Schillebeeckx and Robert Pless. Structured light field design for correspondence free rotation estimation. In *International Conference on Computational Photography (ICCP)*, 2015.

[56] Ian Schillebeeckx and Robert Pless. Pose hashing with microlens arrays. In *Proc. European Conference on Computer Vision*, 2016.

[57] Ian Schillebeeckx and Robert Pless. Single image camera calibration with lenticular arrays for augmented reality. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[58] Ian Schillebeeckx and Robert Pless. Using chromo-coded light fields for augmented reality. In *IEEE International Conference on Virtual Reality*, 2016.

[59] P. Sturm and S. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1999.

[60] H. Tanaka, I. Kajitani, K. Homma, Y. Wakita, and Y. Matsumoto. A motion tracker using high-accuracy ar markers for on-site motion analysis. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, pages 1427–1432, Oct 2015.

[61] Hideyuki Tanaka, Yasushi Sumi, and Yoshio Matsumoto. Application of moiré patterns to ar markers for high-accuracy pose estimation. 2012.

[62] Hideyuki Tanaka, Yasushi Sumi, and Yoshio Matsumoto. A high-accuracy visual marker based on a microlens array. In *IEEE/RSJ International Conference on Intelligent Robotics and Systems*, 2012.

[63] Hideyuki Tanaka, Yasushi Sumi, and Yoshio Matsumoto. A novel ar marker for high-accuracy stable image overlay. In *The 1st IEEE Global Conference on Consumer Electronics 2012*, pages 217–218. IEEE, 2012.

[64] Hideyuki Tanaka, Yasushi Sumi, and Yoshio Matsumoto. A visual marker for precise pose estimation based on lenticular lenses. In *IEEE International Conference on Robotics and Automation*, 2012.

[65] Hideyuki Tanaka, Yasushi Sumi, and Yoshio Matsumoto. Further stabilization of a microlens-array-based fiducial marker. In *IEEE International Symposium on Mixed and Augmented Reality*, 2013.

[66] Hideyuki Tanaka, Yasushi Sumi, and Yoshio Matsumoto. Method and system for detecting pose of marker, June 14 2013. US Patent App. 13/918,605.

[67] Hideyuki Tanaka, Yasushi Sumi, and Yoshio Matsumoto. A solution to pose ambiguity of visual markers using moire patterns. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2014.

[68] Hideyuki Tanaka, Yasushi Sumi, and Yoshio Matsumoto. A portable 6-dof motion tracker using high-accuracy ar markers—first report on the feasibility. In *Machine Vision Applications (MVA), 2015 14th IAPR International Conference on*, pages 563–566. IEEE, 2015.

[69] Keisuke Tateno, Itaru Kitahara, and Yuichi Ohta. A nested marker for augmented reality. In *Virtual Reality Conference, 2007. VR'07. IEEE*, pages 259–262. IEEE, 2007.

[70] Yuko Uematsu and Hideo Saito. Improvement of accuracy for 2d marker-based tracking using particle filter. In *Artificial Reality and Telexistence, 17th International Conference on*, pages 183–189. IEEE, 2007.

[71] Atsushi Ueta, Keisuke Watanabe, Yuto Takei, Satoshi Suzuki, and Hideyuki Tanaka. A high-accuracy 2d visual marker for dexterous manipulation robot in space. 2014.

[72] Daniel Wagner and Dieter Schmalstieg. Artoolkitplus for pose tracking on mobile devices. In *Computer Vision Winter Workshop*, 2007.

[73] Gregory K. Wallace. The jpeg still picture compression standard. *Commun. ACM*, 34(4):30–44, April 1991.

[74] Gordon Wetzstein, Ramesh Raskar, and Wolfgang Heidrich. Hand-Held Schlieren Photography with Light Field Probes. In *International Conference on Computational Photography (ICCP)*, 2011.

[75] Gordon Wetzstein, David Roodnick, Wolfgang Heidrich, and Raskar Ramesh. Refractive shape from light field distortion. In *Proc. IEEE International Conference on Computer Vision*, 2011.

[76] Grace Woo, Andrew Lippman, and Ramesh Raskar. Vrcodes: Unobtrusive and active visual codes for interaction by exploiting rolling shutter. In *Mixed and Augmented Reality (ISMAR), 2012 IEEE International Symposium on*, pages 59–64. IEEE, 2012.

[77] ZL Wu, ZJ Wang, P Keller, and Q Zheng. Light responsive microstructured surfaces of liquid crystalline network with shape memory and tunable wetting behaviors. *Macromolecular Rapid Communications*, 2016.

[78] Dongpeng Yang, Siyun Ye, and Jianping Ge. From metastable colloidal crystalline arrays to fast responsive mechanochromic photonic gels: An organic gel for deformation-based display panels. *Advanced Functional Materials*, 24(21):3197–3205, 2014.

[79] J. Ye, Y. Ji, F. Li, and J. Yu. Angular domain reconstruction of dynamic 3d fluid surfaces. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 310–317, June 2012.

[80] Timothy York, Lindsey Kahan, Spencer Lake, and Viktor Gruev. Real-time high-resolution measurement of collagen alignment in dynamically loaded soft tissue. *Journal of Biomedical Optics*, 2014.

[81] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEE Transactions on*, 22(11):1330–1334, 2000.

[82] Zhengdong Zhang, Yasuyuki Matsushita, and Yi Ma. Camera calibration with lens distortion from low-rank textures. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 2321–2328. IEEE, 2011.

[83] Zhengyou Zhang. Camera calibration. In Gerard Medioni and Sing Bing Kang, editors, *Emerging Topics in Computer Vision*, chapter 2, pages 5–44. Prentice Hall Professional Technical Reference, 2004.