Fall 8-5-2014

Towards Highly-Integrated Stereovideoscopy for *in vivo* Surgical Robots

Jay Carlson
University of Nebraska-Lincoln, jay@huskers.unl.edu

TOWARDS HIGHLY-INTEGRATED STEREOVIDEOSCOPY FOR *IN VIVO* SURGICAL ROBOTS

by

Jay Carlson

A THESIS

Presented to the Faculty of

The Graduate College at the University of Nebraska

In Partial Fulfilment of Requirements

For the Degree of Master of Science

Major: Electrical Engineering

Under the Supervision of Professor Lance C. Pérez

Lincoln, Nebraska

August, 2014

TOWARDS HIGHLY-INTEGRATED STEREOVIDEOSCOPY FOR *IN VIVO* SURGICAL ROBOTS

Jay Carlson, M.S.

University of Nebraska, 2014

Adviser: Lance C. Pérez

When compared to traditional surgery, laparoscopic procedures result in better patient outcomes: shorter recovery, reduced post-operative pain, and less trauma to incisioned tissue. Unfortunately, laparoscopic procedures require specialized training for surgeons, as these minimally-invasive procedures provide an operating environment that has limited dexterity and limited vision. Advanced surgical robotics platforms can make minimally-invasive techniques safer and easier for the surgeon to complete successfully. The most common type of surgical robotics platforms – the laparoscopic robots – accomplish this with multi-degree-of-freedom manipulators that are capable of a diversified set of movements when compared to traditional laparoscopic instruments. Also, these laparoscopic robots allow for advanced kinematic translation techniques that allow the surgeon to focus on the surgical site, while the robot calculates the best possible joint positions to complete any surgical motion. An important component of these systems is the endoscopic system used to transmit a live view of the surgical environment to the surgeon. Coupled with 3D high-definition endoscopic cameras, the entirety of the platform, in effect, eliminates the peculiarities associated with laparoscopic procedures, which allows less-skilled surgeons to complete minimally-invasive surgical procedures quickly and accurately.

A much newer approach to performing minimally-invasive surgery is the idea of using *in-vivo* surgical robots – small robots that are inserted directly into the patient

through a single, small incision; once inside, an *in-vivo* robot can perform surgery at arbitrary positions, with a much wider range of motion. While laparoscopic robots can harness traditional endoscopic video solutions, these *in-vivo* robots require a fundamentally different video solution that is as flexible as possible and free of bulky cables or fiber optics. This requires a miniaturized videoscopy system that incorporates an image sensor with a transceiver; because of severe size constraints, this system should be deeply embedded into the robotics platform.

Here, early results are presented from the integration of a miniature stereoscopic camera into an *in-vivo* surgical robotics platform. A 26mm×24mm stereo camera was designed and manufactured. The proposed device features USB connectivity and 1280×720 resolution at 30 fps. Resolution testing indicates the device performs much better than similarly-priced analog cameras. Suitability of the platform for 3D computer vision tasks – including stereo reconstruction – is examined. The platform was also tested in a living porcine model at the University of Nebraska Medical Center. Results from this experiment suggest that while the platform performs well in controlled, static environments, further work is required to obtain usable results in true surgeries.

Concluding, several ideas for improvement are presented, along with a discussion of core challenges associated with the platform.

ACKNOWLEDGMENTS

A student's adviser plays a significant role throughout the student's research career; the student's adviser influences the topic of the thesis, helps the student plan and execute the research necessary for the publication, and assists the student during writing and revising the work. Dr. Lance C. Pérez has worked far beyond those responsibilities: he has become a valued source of advice, humor, and friendship; he has helped me discover my passions and values inside and outside of academia; and were it not for Lance, I would not be presenting this work, or even be a graduate student.

I must also acknowledge Dr. Shane Farritor, who works tirelessly to help students develop a passion for solving our world's most important problems. I've had the unique opportunity to work with him and his students – Tyler, Eric, Kearney, and Tom – on a wide variety of projects over the year, resulting in successful research and deep friendship.

I must acknowledge my former lab neighbor, Sam Way, for his perspective, good conversation, and friendship – he's quite possibly the funniest person I know, and his humor saved my sanity on multiple occasions.

Dr. Eric Psota has taught me to rely on my intuition and instinct when solving problems, and to never be afraid of failure. He has made me a better researcher – and has become a close friend in the process.

My other labmates – especially Mateusz and Steven – have provided great camaraderie, wonderful insight, and plenty of laughs over the years. Work feels a lot less like work because of them.

My friends outside of work – especially Eric, Johnny, Alyssa, and Erin – have been tremendous in supporting me, and have provided useful perspective on the challenges

I encounter inside and outside of my research.

DEDICATION

To my parents, who have moved mountains for me.

Contents

List of Figures

List of Tables

Introduction

Traditional open surgery, where a single large incision is made at the surgery site and traditional tools are used, has largely been replaced in many surgical procedures with an almost-ubiquitous adoption of minimally-invasive surgery (MIS). Laparoscopic surgery – one type of MIS – allows surgeons to carry out procedures by inserting long, slender instruments into small incisions at the surgical site. Laparoscopic procedures have greatly enhanced patient outcomes when performing exploratory or curative operations, and many consider it one of the great advances in surgery in the latter 20th century [5, 15, 46]. Today, laparoscopic procedures can be performed in nearly every part of the body using a barrage of specialized tools and techniques that cover almost every sub-discipline including ear, nose and throat (ENT), general surgery, neurosurgery, gastrointestinal (GI) operations, and gynecology.[47].

Different laparoscopic procedures call for different sets of tools. These usually fall into two classes – rigid and flexible. For example, a urology procedure may use rigid tools and a rigid endoscope inserted into small incisions in the abdomen, while a GI procedure may be carried out using a flexible video endoscope, which uses a tube that carries control cables, a water channel, an air channel, a fiber bundle for transmitting light, a CCD camera, and a biopsy/suction tube that can be used to insert tools into.

A new MIS technique, called natural orifice translumenal endoscopic surgery (NOTES), is emerging as an alternative to traditional abdominal laparoscopic pro-

cedures. Instead of making a small incision into the abdomen, a NOTES procedure starts by traversing a natural orifice (urethra, mouth, anus, etc) with an endoscopic instrument, then making a small internal incision in the stomach, vagina, bladder, or colon. NOTES procedures offer faster recovery time, produce no abdominal scarring, and eliminate postoperative problems like abdominal wall pain, wound infections, and hernia formations.

While MIS procedures have had a dramatic effect on decreasing complications and bettering patient outcomes, there are some major drawbacks. Laparoscopic instruments provide the surgeon limited dexterity, limited navigation, limited instrument insertion, and limited vision. Also, laparoscopic tools have a steep learning curve, since every tool has a fixed rotational point (the incision), and therefore, operates as a lever. Depending on the length of the tool on either side of the incision, movement can either be magnified or attenuated. This requires lots of practice before a surgeon is capable enough to operate safely and efficiently.

Surgical Robots

MIS surgery has seen interest develop in partial and full automation using robotics – in hopes of making MIS surgery both easier to perform, more precise, and generally faster as well. Surgical robots have seen parallel development with MIS technologies since the late 1980s. Although surgical robots started as re-purposed industrial manufacturing robots, they have since evolved into purpose-built machines with optimal kinematics profiles suited for working in small areas.

Traditional robotic surgery systems, referred to hereafter as laparoscopic robots, are designed to mimic human-guided laparoscopic procedures. This means they use the same sorts of tools and techniques for robotic laparoscopic surgery as their human

counterparts use for non-robotic laparoscopic procedures. The advantage of using these robots over unassisted laparoscopic procedures is that they provide the surgeon with slightly better dexterity, as their tools often feature more degrees of freedom than laparoscopic tools.

Another interesting effect of using a robot is that surgeons who have no laparoscopic experience are able to operate the robot with much better control than traditional laparoscopic instruments. This is because the robot is capable of translating kinematics for the surgeon; that way, the surgeon focuses only on where she wants the tools to be while the robot performs the calculations to place the tool where it needs to be. And since the tools have more degrees of freedom, novice surgeons are not as encumbered by the dexterity issues associated with traditional laparoscopic tools.

These platforms have become quite mature, and their use in hospitals is beginning to emerge (the da Vinci system is an example of such system [13]). They are designed to mimic human movement and perform surgery using already-established MIS procedures, which decreases the learning curve for doctors and speeds up medical board approval for new procedures. This design choice, however, has a major shortcoming: by assuming the traditional laparoscopic form factor (long, slender, rigid rods with tools on the end), the robot's design is unnecessarily constrained. One of the key advantages that robots have, in any application, is that they can be designed and built in almost endless configurations – each one purpose-built for one type of job. However, the laparoscopic constraint prevents robots from being designed in the optimal way to solve a particular surgical challenge. For example, the optimal design for a robot built for gastrointestinal surgery may be a snake-like form factor. By removing the constraint that surgical robots operate in a laparoscopic manner, interesting new paradigms for robotic surgery can be created.

A promising example is the *in vivo* surgical robot. Unlike laparoscopic robots, *in vivo* robots are designed to be fully inserted into the patient and performs all work from inside. This allows an entire surgery to be completed from a single incision. These robots aim to preserve the dexterity of open surgery while being minimally-invasive.

A subclass of these robots are NOTES robots, which are designed to aid a surgeon in performing NOTES procedures. NOTES robots are even smaller than other *in vivo* robots, and must be self-anchoring; they are not physically mounted on a rod or support beam sitting external to the surgical site. NOTES robots are typically anchored to the inner-abdominal wall using magnets.

Endoscopy

Regardless of whether a surgery is performed with a surgical robot or manually, a major component of *any* MIS platform is the system that provides the surgeon a high-quality view of the surgical scene.

Endoscopy – *looking inside* – refers to a set of techniques and technologies used in industry and medicine for viewing images of internal environments that are not easily accessible. Different types of technologies have been developed over time to address requirements for the high quality transmission of an interior scene. Traditional methods involve using either a rigid tube with a series of lenses (usually called an *endoscope*), or a bundle of fiber (called a *fiberscope*), to transmit light rays directly to the observer.

Fiberscopes offer a key advantage over traditional endoscopes when performing certain types of surgery (like GI): because they are flexible, they can bend and contort, allowing them to traverse an irregular-shaped pathway (like the esophagus). However,

their flexibility is very limited – most have a maximum bend radius of several inches. This prevents them from being used effectively on *in vivo* surgical robots (especially NOTES robots), where the camera system must be precisely oriented, and must travel with the robot at all times.

With advances in miniaturization, an entire camera system can now fit in the tip of a flexible endoscopic instrument, instead of requiring lenses or fiber to be used to transmit the image to a remote camera. Because the bulky bundle of hundreds of fibers can be replaced with a small bundle of wires carrying communication and power, this sort of instrument (known as a *videoscope*) is much more flexible and maneuverable than fiberscopes.

Stereopsis

A primary limitation of endoscopic systems is that, typically, the viewer loses most depth perception while using them. This is because in addition to using context-based depth cues, humans use their binocular vision to provide direct depth perception by observing how objects appear relative to each other in both the left and right eyes' views of a scene; this is called *stereopsis*. Unfortunately, a typical endoscopic device only provides a view of the surgical scene to a single eye. Because depth perception enhances the naturalness of an image [60], and enables better judgment of depth, as well as better perception of surface curvature and material properties [25], endoscopic systems have been reworked to provide two slightly offset views of the scene – one for each eye. Any camera with this sort of configuration is *stereoscopic.* Stereoscopic endoscopes are typically built around either fiberscope technology (which have two bunches of fiber, instead of one, transmitting images from two slightly offset lenses at the tip of the endoscope), or videoscope technology (having two offset cameras wired

to a special stereoscopic display). Stereoscopic endoscopes aid medical professionals in many areas of treatment, diagnosis, pre-operative planning, surgery, and surgical training and teaching [66]. Although early tests showed mixed results [12], it is now well-established that surgeons – especially those inexperienced with performing laparoscopic surgery – perform better with a stereoscopic camera [29, 11, 59].

To provide a natural, realistic view, certain considerations must be made when designing a camera system for stereoscopic videoscopy: cameras should be mounted in a manner that mimics the arrangement of the human eyes (i.e., arranged side by side), and thought must go into deciding the optimal separation distance: most humans have a pupillary distance of 48 to 83mm, with a mean value of 63mm [17], so using a camera separation distance similar to this value will provide the same sort of depth perception as humans are used to, while deviating from this value will scale the operator's depth perception accordingly — a smaller separation value will decrease the amount of depth perception, while an increase in camera separation will increase the amount of depth perception. For *in vivo* applications, where the environment is very small, it is useful to decrease the camera separation distance to provide appropriately-scaled depth perception.

Focus of Thesis

It has been established that while traditional laparoscopic robot platforms can harness traditional endoscopic methods, like rigid endoscopes and flexible fiberscopes, *in vivo* robots, including NOTES platforms, require an imaging solution such as flexible videoscopy, which allows the robot to assume any orientation required during surgery, without having to worry about the bend radius of the cabling. It has also been established that a stereoscopic imaging solution provides key advantages over monocular

camera solutions.

This thesis covers the design and evaluation of a stereoscopic camera that can be integrated into an *in vivo* surgical robot platform, with a primary focus on a dual-sensor USB-based camera. In the following chapter, background information is presented to motivate the design of the camera. In the third chapter, the design is discussed. In the fourth chapter, results from evaluation and testing are presented.

Background

Before the proposed USB stereoscopic camera is introduced, it is important to have a firm grasp on human depth perception (which explains why the dual-camera system is especially important in a surgical environment); this topic is presented first. Afterward, camera imaging technologies are explored and briefly discussed. Finally, endoscopes are discussed thoroughly to establish the state of the art for camera systems used for medical purposes.

Human Depth Perception

Humans perceive depth using two methods: visual cues and stereopsis. Visual cues include the following:

- *Motion parallax* – when we move, the relative motion of objects around us relative to the background gives clues about the relative distances between these objects.[36]
- *Perspective* – When lines we know to be parallel converge or diverge, we can judge the relative distance between two parts of an object.
- *Relative size* – when objects that are of known physical size appear in a view at different proportions than expected, we know the objects are at different

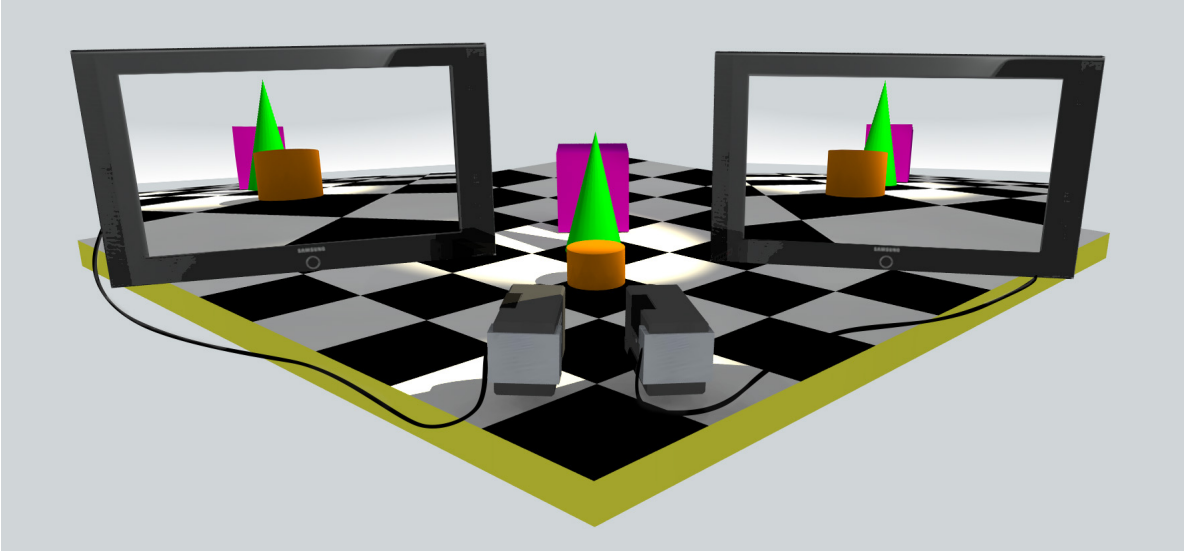

Figure 1: Demonstration of stereopsis. The left and right cameras observe the same scene from two different views. Human brains can process these two disparate views to construct depth information about the scene.

distances from us.

- *Optical effects* – the human eye's depth of field is limited, so objects at different depths from the current area of interest will be out of focus; there's also a kinesthetic sensation associated with the muscles responsible for focusing.

Separately from visual cues, human brains are also capable of perceiving depth using stereopsis; where objects observed from two different views appear at different offsets, depending on their relative distances from the observer. By recognizing the relative location of objects in the left eye, and comparing that to the locations of those same objects in the right eye, the brain can gauge distance. Figure 1 demonstrates this effect: in this illustration, two box cameras are pointed toward an orange cylinder, a green cone, and a purple box. An observer looking at the figure on paper might be able to judge the relative distance between the objects using the checkered floor pattern as a reference point; however, a more precise method is to take advantage of

Figure 2: The Half Dome at Yosemite rises more than 4,700 ft above the valley floor (left). Most people can still recognize the Half Dome's enormous size relatively quickly, even if another object of known size (an engineering student) is oriented in the view such that it is appears larger than the Half Dome (center). But when objects appear to interact with each other, our brain has trouble using visual cues to identify correct sizes of objects (right).

the parallax effect; two offset cameras shooting the same scene see the same scene from two different views (shown on the illustrated monitors). By judging how the views are different, the human brain can estimate the relative positions of objects.

This is the most precise way humans can judge short distances; while we use visual cues as our dominant source of depth information about our surrounding, these cues are only useful when interacting with familiar objects in a macroscopic environment. When objects are unfamiliar, or they are positioned in a manner that is counter-intuitive, visual cues alone can confuse our brain into creating a false depth model of the view (for an example of this, see Figure 2).

Surgeons are especially prone to misjudging depth using visual cues while operating laparoscopically for the following reasons:

- The surgical environment is usually nearly fixed; if the surgeon is unable to

move about the environment, she is confined to a single, stationary view of the scene. This prevents the surgeon from judging depth from relative motion (using motion parallax).

- The surgical environment often has low detail, complex, continuous geometry and unnatural lighting, which casts confusing shadows; this prevents judging convergent lines, which prevents the surgeon from establishing perspective.

- The relative sizes of objects cannot be used, since the distances involved in the surgical environment are highly constrictive (which diminishes the effect in the first place).

- Because the surgeon is viewing the scene indirectly, using some sort of endoscopic device, kinesthetic sensation associated with focus is largely absent or unintuitive. Furthermore, these tools typically have small lenses with limited aperture sizes; thus, these optical assemblies have extremely deep depth-of-field, which prevents any sort of selective focus from occurring; all objects in the surgical scene are always in focus all the time.

Because of these issues, requiring the surgeon to rely on visual cues alone is problematic, and requires training and experience to be able to use effectively[65]. This is why it is desirable to provide the surgeon a stereoscopic view of the surgical area taken from two different offsets; without it, the surgeon's ability to perceive depth of the scene will be greatly diminished.

This challenge can be solved with either mechanical systems (such as stereoscopic fiberscopes or stereoscopic rigid endoscopes), or with an electronic stereoscopic videoscope. Mechanical stereoscopic systems are bulky and require the viewer to press their face up to a dual-eyepiece system, while a stereoscopic videoscope enables a

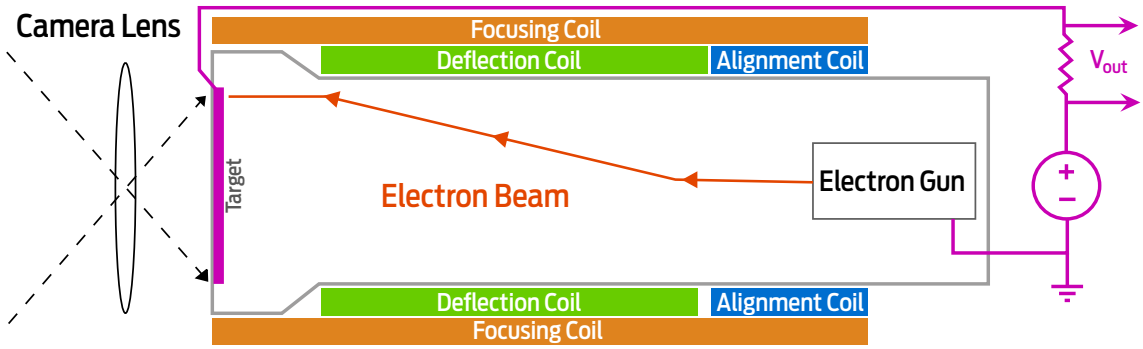

Figure 3: Vidicon tube diagram

much more natural view of the scene by using a variety of lightweight displays. This will be discussed in detail in coming sections.

Image Capture

Every videoscope system is essentially comprised of an image capture device (or in the case of stereovideoscopy, two image capture devices), so it is instructive to overview how images are captured electronically. Historically, there have been three different technologies used for capturing images: video camera tubes, charge-coupled devices, and CMOS image sensors[32].

Video Camera Tubes

Video camera tubes are the dual of the well-known cathode ray tube (CRT) display; they were widely used in image acquisition before the 1980s. While camera tubes evolved significantly – the Iconoscope, followed by the Image Orthicon, and the Vidicon – the general principle of their operation is the same: a focused electron beam scans the imaging plate (also known as the *target*), line by line. The plate has a a resistive or capacitive property that changes with respect to the amount of light present

at that spot, and the scanning electron beam activates that electrical property for the currently-scanned spot on the plate. For example, the Iconoscope's image plate has photosensitive granules that vary in capacitance with light intensity [3], while the Vidicon's target plate – made from selenium or, later, silicon diode arrays – varies in resistance. In Figure 3, this resistance is measured using a voltage-divider, producing a V_{out} that is proportional to the intensity of light at the particular location the electron beam is striking [67]. By scanning the electron beam quickly over the entire target, the light intensity at each point is read.

To display this image, the CRT monitor's electron gun's position is synchronized to the camera's; then, the output voltage is simply used to control the intensity of the electron gun's beam. In the world of discrete analog circuitry, the most challenging part of the imaging system to design was the synchronization and modulation circuitry; the fundamental operation of the camera tube and CRT display are quite simple.

CCD Sensors

While video camera tubes can provide good resolution, they are extremely bulky and expensive to produce. The CCD sensor was the first device that could be mass-produced using relatively inexpensive production techniques, which allowed the rise of consumer-oriented video cameras. The CCD sensor has a discrete number of pixel sites which gather light during an integration time period. These pixel sites function as capacitors – storing a charge that is proportional to the light intensity at that pixel site. After the image has been exposed, control circuitry causes each capacitor to transfer its contents to its neighbor, line by line. The last pixel in the array is hooked up to a charge amplifier which measures this charge. As the control circuitry

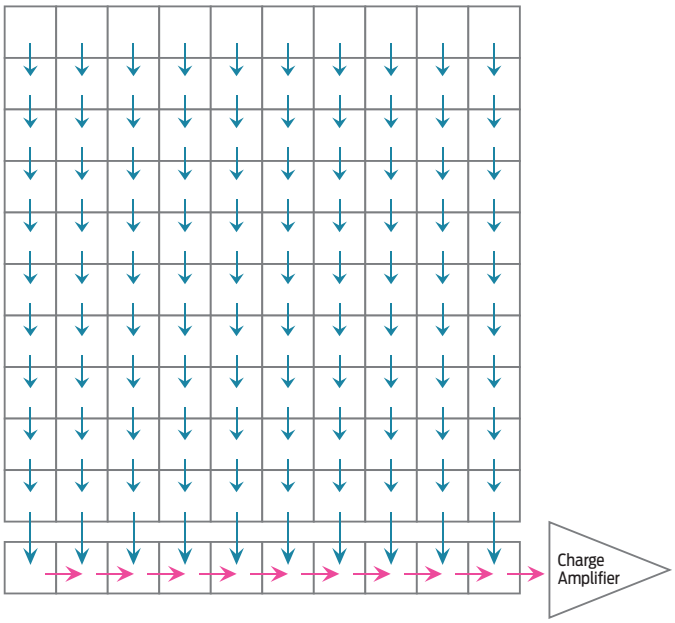

Figure 4: CCD sensor diagram

iterates over the entire array, the charge gets measured by the charge amplifier, then discarded. Eventually, every pixel site has been measured[10]. Figure 4 illustrates the shifting behavior of the pixel array – in the figure, a 10x10 CCD array is read out, pixel by pixel, by shifting the values into the charge amplifier for measurement one by one.

CMOS Sensors

The major problem with CCD sensors is that they cannot be produced on a standard CMOS process; because of this, digital circuitry cannot be integrated into the product directly. Since a camera is comprised of both the sensor and support circuitry, this makes the overall camera package larger. CMOS sensors are imagers that can be built on a silicon wafer in the same manner that any other analog or digital circuit is created. This allows them to be produced extremely cheaply, with good reliability and

Figure 5: CMOS sensor diagram

consistent image quality. CMOS sensors also make use of the photoelectric effect, but instead of shifting this charge around, each pixel site has an amplifier and a row-select transistor. Figure 5 diagrams this process. Each pixel in the 10x10 array, this time, has a photo diode (the green box labeled "PD"), an amplifier (black triangle), and a row-select transistor. At the bottom of every column is a column-select transistor. This allows control logic to route arbitrary pixel values to the output. To read out the data, the control logic simply alternates between every row/column combination. One often-used advantage this has is that small portions of the sensor can be read out, allowing the same image sensor to provide both full-quality still-frame imaging, as well as high-speed (albeit lower-resolution) video. Comparatively, the CCD sensor

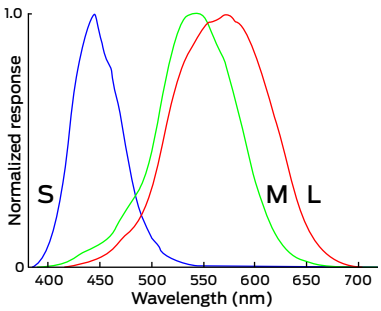

Figure 6: Approximate response curves for the short (S), medium (M), and long (L) types of cone cells in humans. These three curves roughly center on red, green, and blue – the primary additive colors mixed to represent any arbitrary visible color.

must be read out in full each time.

Because of all the ancillary circuitry required at every pixel site, the size of the photo diode is much smaller for a CMOS sensor than a CCD sensor with the same-sized pixels. This means the CMOS pixel site has more noise due to pickup than a comparable CCD pixel; it also means the CMOS sensor generally has less dynamic range (ability to capture high-contrast scenes without clipping) than the CCD sensor. However, CCD sensors have more noise introduced between pickup and sampling than CMOS sensors do, so overall noise – the addition of pickup noise and noise caused by signal travel – is similar between sensors[70]. CMOS sensors can sample faster (1000 Mpixels / second) than CCD sensors (70 Mpixels / second) [35]. CCD sensors are incompatible with standard VLSI processes, so it is not possible to integrate them into a larger image processing integrated circuit. CMOS sensors, on the other hand, can easily be integrated into an imaging system-on-chip (SoC), complete with amplifiers, analog-to-digital converters, DSP functionality, etc.[21].

Color Vision

When designing any camera system intended to relay visual information as accurately as possible, careful consideration of the colorimetry of the system is important. Poorly-calibrated colorimetry causes images to appear dull, washed out, or unnatural. Since surgeons use the appearance of tissue (including the color of the tissue) for diagnostic purposes, it is important that the colors of the scene viewed by the surgeon are as accurate as possible. Humans are trichromats [8]; they perceive color sensations using *cone cells*. There are three different types of cone cells – short (S) cone cells are receptive mostly to blue light; medium (M) cone cells are receptive mostly to green light; long (L) cone cells are receptive mostly to red light (See Figure 6). When we see a color in nature like yellow, we really perceive a proportional combination of L and M receptors. To simulate a natural color, we can add certain amounts of red and green light together; To our eyes, this appears as a single color – not as a superposition of two individual colors. This technique of mixing "primary colors" allows us to produce any color imaginable by simply varying the intensity of three primary colors – usually red, green, and blue.[1]

Color Spaces

A *color space* defines how real-world colors get mapped to numeric color values used for processing and transmission. Different color spaces are used for different purposes: some color spaces are designed to mimic how color is captured from a camera, or displayed on a monitor. Some color spaces mimic how human eyes perceive color sensations. Other color spaces are designed to represent colors in a way that makes

[1] As an interesting aside, the red, green, and blue primary set is ubiquitous today, but before we knew as much about human vision, other primary sets were used; one example, called autochrome, used orange, green, and violet primaries.[50]

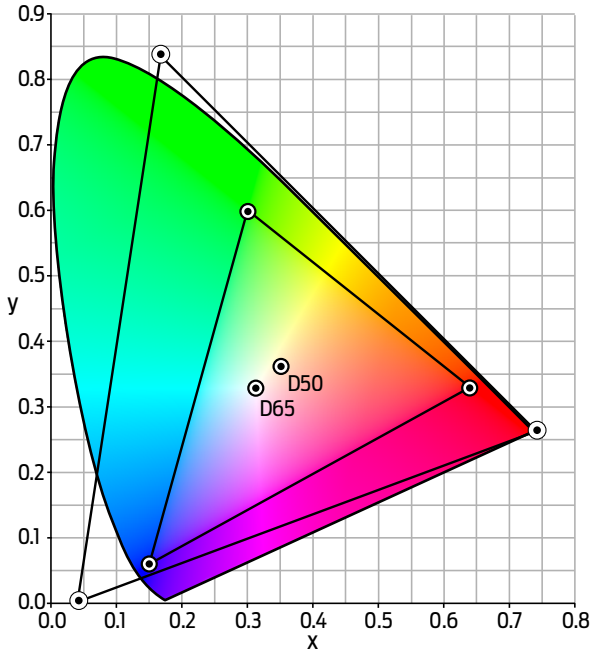

Figure 7: sRGB and ProPhoto RGB color spaces drawn inside of CIE 1931.

it intuitive for processing. Because humans are trichromats, every color space is three-dimensional; different color spaces are simply different coordinate systems representing the same information. This allows a color represented in one color space to be converted to another space. Not all color spaces are capable of representing every visible color. The set of colors a space can represent is known as the *gamut* of the space.

From the previous discussion, the most natural color space seems to be one that creates colors as a linear combination of intensities corresponding to short, medium, and long cone normalized response curves, similar to what is seen in Figure 6. This defines the LMS color spaces. The L coordinate represents the response of the long cone cells, the M coordinate represents the response of the medium cone cells, and the S coordinate represents the response of the short cone cells. Although this color space may seem like the obvious choice for working with color, for historic and tech-

nical reasons, however, this space is rarely used. Rather, for technical, physiological and psychological work on color, a more commonly used color space is CIE 1931. CIE (sometimes known as XYZ) defines a color by the luminance, Y, and the color components X and Z. Because it encompasses all color sensations that an average person can perceive, it is useful as a benchmark space (as it has an extremely wide gamut). However, because it does not encode color information in a manner consistent with how images are captured or displayed, it is typically only used for specialized applications. Instead, for most work, an RGB space is used. All RGB spaces represent colors as a linear combination of three primaries – red, green, and blue. Where RGB spaces differ from each other, however, is in the precise definitions used for the primaries, and the definition of the white point (i.e., what color of white should be displayed when all primaries are illuminated at 100%). The most commonly-used RGB color space, sRGB, defines the primaries based on the color of primaries used in CRT monitors. The white point that sRGB uses, Illuminant D65, is approximately the color of standard daylight, and has a color temperature of approximately 6500 Kelvin. ProPhoto RGB, as an example of a wide gamut color space, uses primaries that are much more saturated (the green and blue primaries used for ProPhoto are actually outside of human vision). These two color spaces are drawn on top of CIE 1931 in Figure 7. In the figure, sRGB has a gamut that covers the smaller triangle. ProPhoto RGB's gamut covers the larger triangle. The three points comprising each triangle represent the primaries used to form the color space, with green on top, red on the right, and blue on the bottom left. Using a wide-gamut profile allows storing and processing image data intended for display on media that is capable of displaying colors outside of the sRGB gamut. One such example is editing digital photography that is intended for print in magazines using a full-color CMYK process. While CMYK, as a whole, has a much smaller gamut than sRGB, it has the

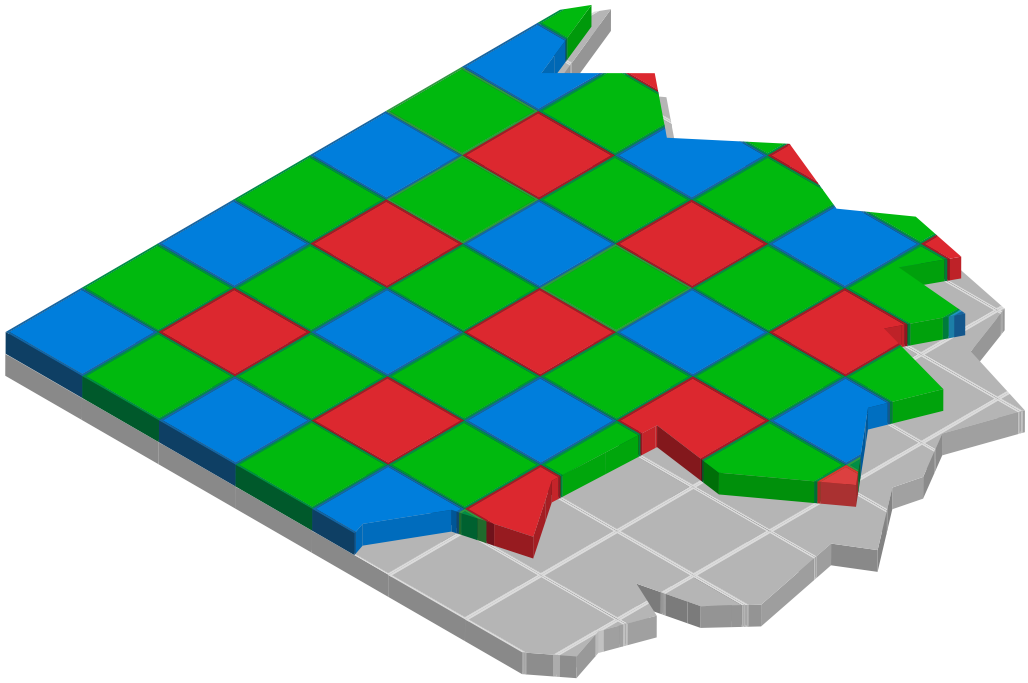

Figure 8: Tear-away view of an image sensor, with a Bayer array (colored filters) placed atop an array of pixel sites (gray boxes).

ability of producing some colors (like very saturated cyan or magenta) that sRGB is incapable of producing. ProPhoto RGB has a gamut that encompasses both sRGB and CMYK (and almost all other visible colors), so it is a useful space for working with different output media. Since almost all color work takes place on computers using fixed-point arithmetic, one disadvantage with wide-gamut spaces is that the color resolution is reduced as the color space is extended.

Since all current image sensors are luminance (not chroma) sensitive, to arrive at a color image a specially-designed system that bandpass-filters light based on wavelength must be employed. There are two methods often used to record color information – a single-sensor color filter, called a Bayer pattern, and a split-prism color filter, which splits incoming light into different colors and directs each to a separate sensor. The Bayer pattern, depicted in Figure 8, consists of an array of color

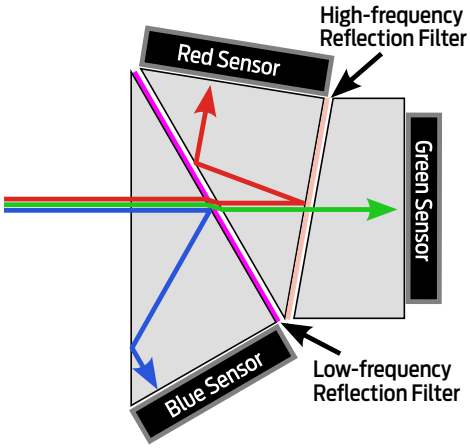

Figure 9: A dichroic prism is used to separate light into primary colors, which are directed at three different image sensors.

filters aligned to the imaging array of pixels. As can be seen in the figure, each pixel receives either red, green, or blue light. This approach is inexpensive and compact, but an interesting problem arises: what is the RGB color value of a particular pixel? Various methods exist to *demosaic* the Bayer image into a component RGB image (where each pixel has a full RGB color value assigned to it) [33]. The easiest method (and also the worst in terms of quality) is to simply perform pixel doubling. Using this method, each 2x2 square is converted into one pixel; the resulting image has one-quarter the resolution of the pixel array, and suffers from aliasing artifacts. However, this method is extremely fast to perform, and produces predictable results that are not interpolated. This may be useful for some image processing applications. A more naturally-looking method uses linear or bilinear interpolation, or a Gaussian kernel that weights neighbor colors to arrive at the correct pixel value.

When cost and size are not of concern, a better option is to use a dichroic prism (Figure 9) which splits light into red, green, and blue wavelengths, and directs them at three different image sensors. For historical reasons, this is typically referred to as a "3CCD" system. As long as the three image sensors are aligned properly, the

Figure 10: These are 400% crops of the ubiquitous SMPTE colorbars test pattern. On the left, the uncompressed test pattern. On the right, the test pattern has been chroma subsampled using YUV 4:2:2. Note the soft edge between the green and violet colors.

RGB color value at a particular pixel can simply be read at the corresponding pixel locations for each of the three sensors. No further processing or demosaicing needs to be done to arrive at the final color image. Typically, these systems are only used on broadcast video equipment and high-quality consumer/prosumer video cameras.

Compression using Chroma Subsampling

Now that the concept of color vision and color representation has been established, we can take advantage of color spaces to compress color data. Human eyes are much more sensitive to changes in luminance (brightness) than changes in color [40], so one obvious way of compressing data is by using less bandwidth to represent color information than to represent luminance. This is the idea of chroma subsampling [2]. To do this, we must first convert the image from its source profile to a colorspace that separates the representation of color from luminance. Although the previously-

[2]It is important to understand that chroma subsampling refers to sampling rates (in the spatial domain), and not sampling bitdepth (i.e., quantization of the actual color values)

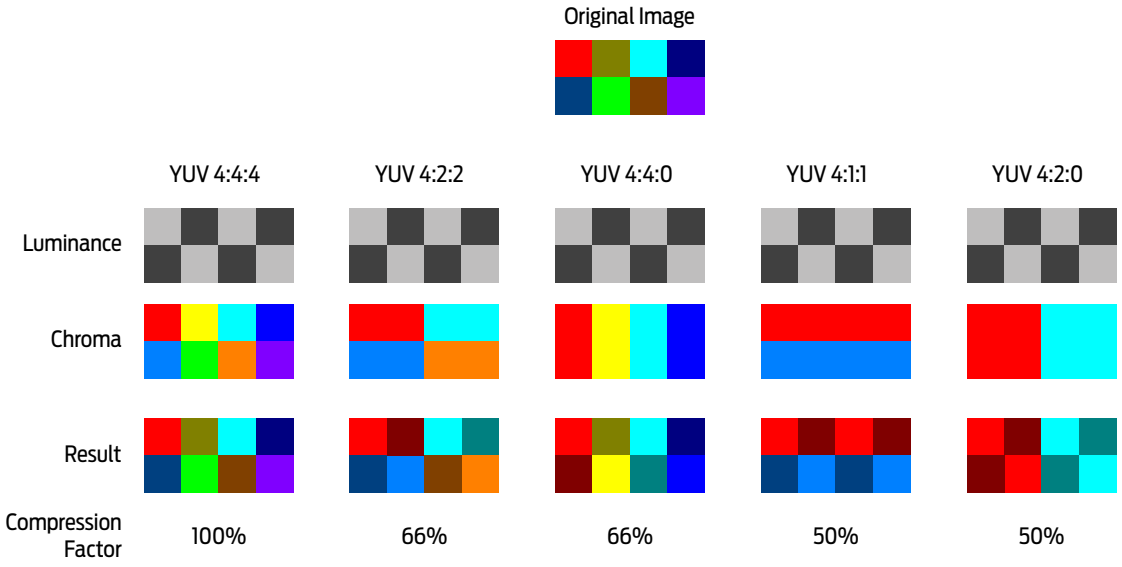

Figure 11: Common Y'CrCb (YUV) chroma subsampling ratios.

mentioned CIE space certainly does this, CIE is not practical to implement (as it takes on real values, as opposed to discrete ones). Because of this, we typically use the Y'CrCb space (which represents colors as discrete values as 8 bit, 10 bit, or 12 bit quantities). Typically, the amount of subsampling is specified as a ratio between the horizontal sampling region, the number of color samples in the first row, and the number of color samples in the second row. Popular examples with illustrations are shown in Figure 11. In this figure, the original image is decomposed into luminance and chroma values (first and second row, respectively). Each column shows a different chroma subsampling of the original image. The first column, YUV 4:4:4, shows the uncompressed image. Notice that the chroma is sampled 4 times in the first row, and 4 times in the second row. The next column shows YUV 4:2:2 compression. In this case, the chroma values are only sampled two times in the first row and two times in the second row. In-between values are copied from the left. As can be seen by comparing the original image with the YUV 4:2:2 resulting image, chroma

subsampling preserves the luminance resolution in the original image, but a pixel's colors are incorrect. However, since color usually changes very gradually across a scene (in a fairly continuous fashion), these sorts of compression artifacts are generally not visible. While YUV 4:2:2 is by far the most popular chroma subsampling rate, other rates are used in different applications. They are presented in Figure 11 for completeness.

Considerations for Stereoscopic Systems

One of the most important considerations in deciding if a camera system designed for monocular vision is suitable for stereoscopic use is to ensure that the amount of crosstalk between the cameras is minimized, since crosstalk can cause blurred vision and double-images to appear to the viewer. Other characteristics, such as optical sharpness (local contrast), resolution, frame rate, and low-light performance, are equally important for both monocular and binocular vision systems. Post-processing of stereoscopic images involves standard image processing (gamma curve adjustment, white balance, exposure, etc) and compression/decompression found in monocular systems. Although using asymmetric parameters for these tasks yields little loss in terms of depth perception, it may increase eye strain and decrease image quality.

Stereoscopic Display

To enable the viewer to perceive depth, the output display attached to the stereoscopic camera must be capable of sending the left camera's image to the left eye and the right camera's image to the right eye.

The easiest way of accomplishing this is to simply have a separate display for each eye, and to force each eye to only observe its corresponding display. For example,

using mirrors oriented at 45-degree angles, the eyes can be directed toward their respective display. This approach is bulky, but offers high-quality imaging with no optical artifacts introduced by the display.

Another implementation of the dual-display concept is wearable LCD glasses, which place a separate small display in front of each eye. Since we cannot comfortably focus our vision that closely, an optical element is positioned between the eye and the display to adjust the focal distance of the display. These glasses are much bulkier than a normal pair of glasses, and the lightweight plastic optics that are often used in these displays produce optical distortion and low resolution.

A single display can also be used, so long as it is capable of targeting each eye separately. There are two common ways of doing this, and both require the viewers to wear specialized glasses.

Passive (Polarization) Systems

Polarized systems (commonly called "passive" systems in the consumer electronics world) take advantage of polarization filters to target each eye separately. To understand how this works requires some elementary knowledge of electromagnetic waves.

Electromagnetic (EM) waves (including visible light) are comprised of a magnetic field $\mathbb{B}$ and an orthogonal electric field $\mathbb{E}$. The summation of these two components produce the resulting EM wave. So long as the EM wave is coherent (i.e., does not vary in amplitude, frequency, or phase), then depending on the phase and amplitude relationship between $\mathbb{E}$ and $\mathbb{B}$, the resulting wave can oscillate along a single axis (linear polarization), or around a circle (circular polarization). These polarizations are shown in Figure 12. The red and blue waves indicate the respective $\mathbb{B}$ and $\mathbb{E}$ fields. In the linear polarization scheme, the waves have no phase shift; in the circular

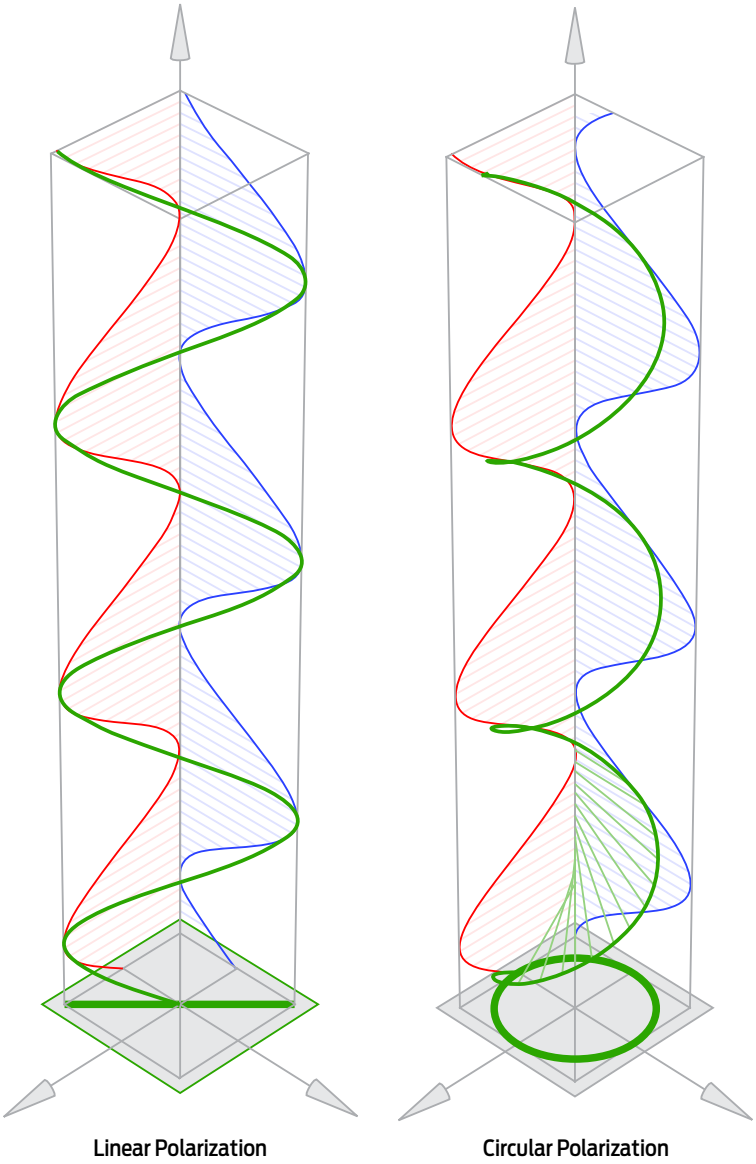

Figure 12: Polarized EM waves

polarization scheme, one field lags the other by 90°.

Non-coherent light may be linearly polarized using an optical absorptive polarizer, which only passes one polarization of light through. The most common optical absorptive polarizer is the H-sheet (originally patented by Polaroid).

Another important filter is the quarter-wave plate. When circularly-polarized light enters the quarter-wave plate, it is converted to linearly-polarized light that oscillates along one of two orthogonal axis, depending on the handedness of the circularly-polarized light. Incidentally, the filter works both ways: if linearly-polarized light enters the filter, a circularly-polarized light exits; its handedness is dependent on the angle of the linearly-polarized input.

Polarized Glasses

By placing a quarter-wave plate in front of a linear polarizer, it is now evident that we can isolate left-handed circularly-polarized light based on its handedness. Assuming our display is polarizing the image intended for the left eye using a left-handed circular polarization, and the image intended for the right eye using a right-handed circular polarization, we can build glasses that can direct light with the proper handedness into the appropriate eye. This optical filter is relatively simple, which makes these sorts of glasses very inexpensive to manufacture.

Polarized Display

This system works using the assumption the display is capable of circularly-polarizing the entire image based on which eye it is intended for. For this to work, we must essentially superimpose the left-eye and right-eye image on the same screen, and then polarize the resulting image accordingly.

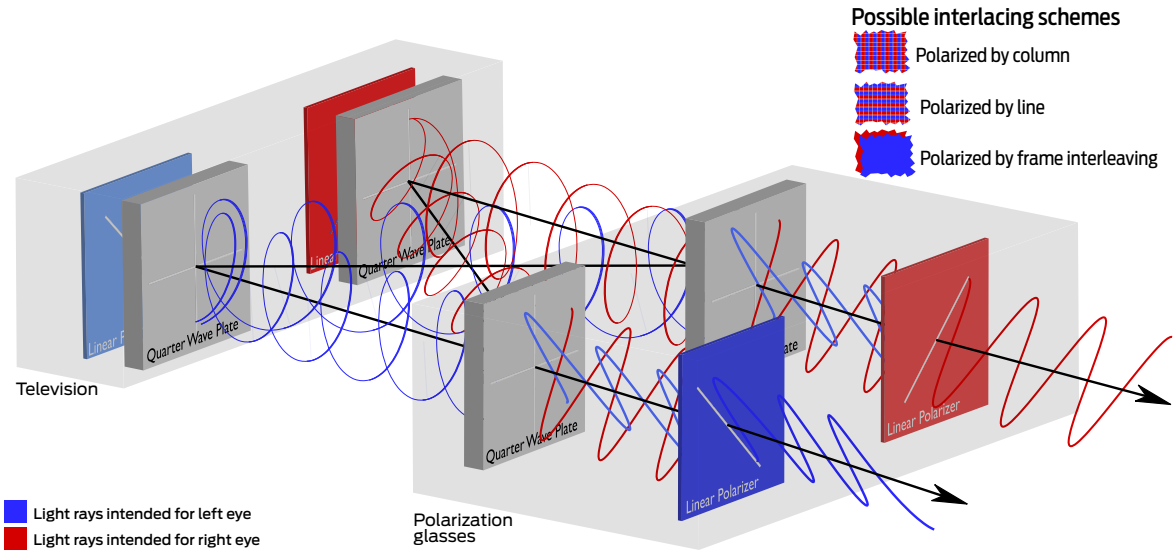

Figure 13: Polarized 3D system

One way of achieving this is by using a special projector screen that maintains polarization, along with two projectors both pointing at the same screen, each with an appropriately-handed circular polarizing filter placed in the optical path. The polarized images will bounce off the screen (maintaining their polarization), and then get filtered by the polarized glasses. [1]

For direct-view displays (CRTs, LCDs, etc), we may place an active polarizer in front of the screen which can alternatively polarize the entire screen with a left-handed or right-handed polarization scheme. The display alternates between images for the left and right eye while the polarizer synchronizes with the display to produce the appropriately-handed polarization. This technique of frame interleaving is also used slightly differently in LCD shutter systems.

If resolution degradation is acceptable, we may instead deposit optical filters on a pixel level. Common techniques assign even-numbered lines to one handedness, and odd-numbered lines to the other handedness.

An overview of a polarized system is shown in Figure 13. Here, data intended for

the left eye (seen in blue) is polarized by first traveling through a -45°linear polarizer and then a quarter-wave plate, which circularly polarizes the light right-handed. Meanwhile, data intended for the right eye (red) is linearly-polarized at 45° and then also brought through a quarter-wave plate, which polarizes it left-handed. Just as with unpolarized light emitted by the monitor, these light rays are scattered all over the room; both polarized light streams enter both lenses on the pair of polarization glasses. However, the glasses are able to separate the polarized light by reversing the process: the polarized rays enter a quarter-wave plate which converts the circular polarization to linear polarization. Then, linear polarizers oriented in $-45°$ and $45°$, respectively, filter out the unwanted polarization, which allows the left light rays to enter the left eye, and the right light rays to enter the right eye. It may seem that the quarter waveplates are an unnecessary addition to the system; it is true that if they are omitted, the system will continue to function as long as everything remains in the orientation depicted in the figure. However, if the viewer tilts her head side to side at all, the linear polarization filters will no longer align with the television's linear polarizing filters. Circularly-polarizing the light allows the image to be recovered at any orientation the viewer may be in.

Compared to other display solutions, the polarized systems may seem to be trivially simple to implement. However, the most commonly-used display technology these days – back-lit liquid crystals – only output light in certain polarizations due to how they operate on a fundamental level. Consequently, panels must be designed specifically to allow for optical polarizing filters; otherwise, these filters would interfere with the functionality of the display.

Active (Shutter) Systems

LCD shutter systems work by using the property of persistence of vision: if you put a high speed shutter in front of your eyes, you will not perceive the open and closed states; instead, you will see a slight dimming of the image. By placing a shutter over each eye, and synchronizing it with the monitor (usually wirelessly), the monitor can target each eye separately. The monitor displays the left image on the monitor, and sends a signal to the shutter glasses to close the right shutter and open the left one. Afterward, the monitor displays the right image on the monitor and sends a signal to the shutter glasses to close the left shutter and open the right one. This process repeats at twice the desired frame-rate of the video. This method and the aforementioned polarization method are the primary ways 3D consumer televisions function.

Auto-stereoscopic Displays

Auto-stereoscopic displays facilitate the display of 3D images to a single observer or multiple observers by displaying two or more views of the same scene, usually by using a curved display technology that has zero-degree viewing angle, which allows the display to isolate each eye. [7, 31]. Although convenient for the viewer (since it does not require any special glasses to be worn), auto-stereoscopic displays produce often-unmanageable crosstalk [7], though using eye tracking to create an auto-stereoscopic display may reduce this effect considerably [38].

Endoscopy Taxonomy

Endoscopy, broadly speaking, concerns technologies used to observe occluded areas located inside things – endoscopic devices are used both in commercial/industrial

applications, as well as medical applications. Endoscopic devices allow medical practitioners to observe the inside of body cavities and joints while avoiding invasive medical procedures. Although rigid endoscopes – which conform to a straight, solid, tubular form factor – are still useful for some procedures, flexible endoscopes – which can bend and contort, and can be quite long – have dramatically reduced the number of invasive exploratory procedures required.

All endoscopes have a front objective lens made of one or more optics whose properties control the field of view, depth of field, and brightness of the scene.

Rigid Endoscope

This is the oldest, simplest type of endoscopic instrument. In addition to the front objective lens, rigid endoscopes have a series of relay lenses, usually separated by air, which transmit the objective lens's light through the tube, to either the observer directly or to a full-size camera for electronic viewing or recording.

Flexible Endoscope

Fiberscope

The fiberscope uses a fiber optic image guide to transmit light through a flexible jacket that often includes additional cables and tubes used for auxiliary purposes (biopsy/suction and water spraying are common). Since individual fibers cannot transmit focused light, to achieve a focused view of the scene, the image from the objective lens is projected onto the tips of a bundle of fibers. Unlike rigid endoscopes, whose resolution is determined by the optical quality and design of the relay elements, fiberscopes have the interesting property of having a measurable, discrete resolution equal to the total number of fibers in the bundle. Figure 14 illustrates the standard

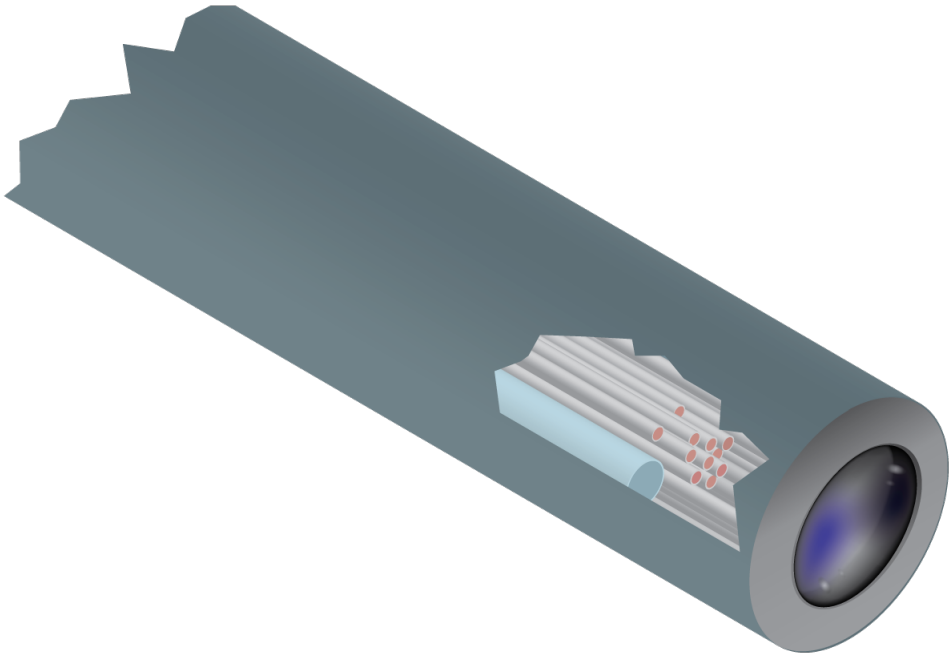

Figure 14: Fiberscope cross-sectional view. The gray tubes are the individual fibers; the blue tube is one of the control cables used to bend the fiberscope to the desired angle.

make-up of a fiberscope: the gray tubes represent individual fiber strands (providing a finite number of "pixels" to the viewer (while the blue tube represents one of the control cables used to pull the flexible fiberscope into the proper direction during navigation.

It is often desirable to view the output image electronically by attaching a camera to the eyepiece of the fiberscope. Because of the discrete resolution of the system, any resolution or alignment mismatch between the lens and the image sensor of the camera system will cause moiré. Since precise alignment of the fibers and pixels is impossible, this artifact is suppressed through an optical low-pass filter [4] or by intentionally imparting spherical aberration onto the lens [49].

Figure 15: 12mm and 8.5mm endoscopic tip from da Vinci Si surgical robot.

Videoscope

With advances in VLSI minimization, image sensors can be placed directly at the end of a flexible endoscope[30]. This allows bulky bundles of fiber to be replaced with a handful of small wires that transport the image data. Videoscopes allow the most flexibility when working endoscopically; but because of limitations of current image sensors in that form factor, videoscopes currently top out at 800×600 resolution.

Current Developments in Endoscopy

The da Vinci Si surgical robot is a popular laparoscopic robot found in hospitals already performing surgery. Since the robot has been privately developed by Intuitive

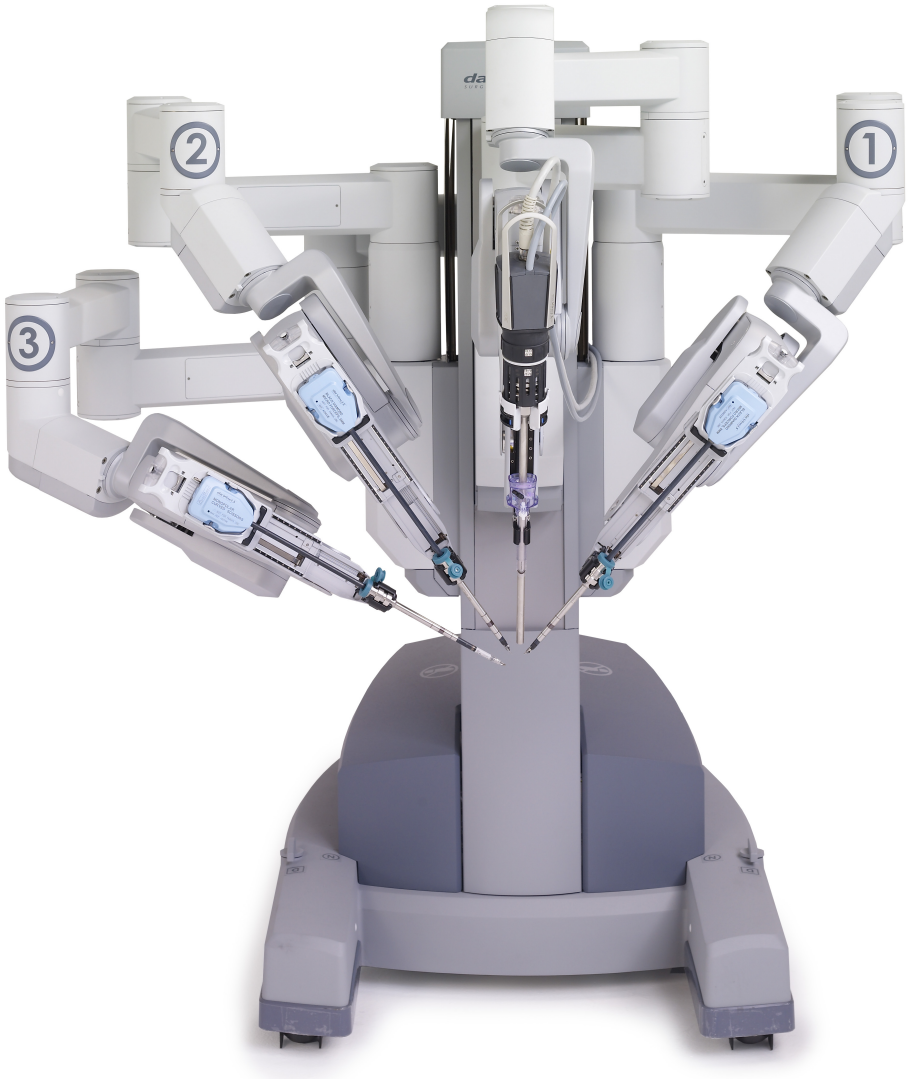

Figure 16: da Vinci Si surgical robot.

Surgical, a commercial entity, little is known about the technical details of the robot's implementation. For vision, the da Vinci uses a traditional rigid stereo endoscopic system with either 12mm and 8.5mm barrels (see Figure 15). The endoscopic rods optically feed the image back to a 3CCD HD camera, as can be seen in the center of the robot in Figure 16. This endoscopic camera system produces very high-quality images by using a large camera and quality optics – but this sort of system cannot be used for *in vivo* surgical robots, because the device is completely rigid, which means the view it can provide the surgeon is completely fixed.

As for videoscopy, the state of the art is made up, largely, of standard-definition analog cameras. Stoyanov *et al.* perform soft-tissue 3D depth recovery using an analog sensor [62], [63]. Hu and Miller developed an insertable stereoscopic device [44], tested it [27], and made refinements [28]. Again, their system uses low-resolution analog image sensors.

What does not exist yet is a high-definition stereovideoscope. Current surgical procedures only entail operating on fairly accessible areas; right now, all MIS procedures use either rigid endoscopy instruments inserted into surgical incisions or a flexible fiberscope tool that is swallowed and only requires limited malleability while traveling down the esophagus. Videoscopes are in their early infancy – especially since the tip of the smallest videoscope is still too large for many procedures. Using current technology to build a stereovideoscope (as this thesis proposes) results in a device too large to be useful in real surgical settings; however, developing this platform can provide keen insight that can be used as camera systems continue to miniaturize.

Problem Description, Work and Results

Introduction and Specifications

To best equip an *in vivo* surgical robot with stereoscopic image capture abilities, both for immediate display to the surgeon and for potential stereoscopic image processing, a custom stereo videoscopy system was designed to provide real-time video to the surgeon. The initial specifications for the camera were based on robot constraints from [68]. They are

- Dual-camera, 3D-capable – allows the surgeon to perceive the surgical environment with depth perception, as well as stereoscopic image processing.
- No larger than 0.75" in diameter – this allows the camera to fit on the then-current generation robot that was under development.

Additional specifications were established that were designed to meet image processing needs:

- At least 720p high-definition (1280×720 resolution) video running at no less than 30 frames per second (fps) – this requirement ensures captured images

have high fidelity and are delivered in real-time, which is important for the surgeon.

- Commonly-available computer interface connection (USB, FireWire, Ethernet, etc) – this enables the camera to be attached to a computer for video recording and image processing.

In terms of the demands of the application, the resolution and frame rate are relatively arbitrary; anything operating with a frame rate of at least 24 frames per second (the industry standard for film production) would be suitable for the application. Lower frame rates than 24 fps would produce unpleasantly jerky video to the surgeon. By selecting 30 fps, the video frames are fast enough to eliminate any jerkiness. The same reasoning holds for resolution; while 640×480 (standard-definition) resolution is more than adequate for most image processing applications, high-definition camera would allow the surgeon to see a clearer view of the surgical environment. And, as will be discussed, the technology required to implement any sort of device with these requirements is practically identical in terms of size, power consumption, and price. In other words, a 1080p camera would not have been any larger than a 720p camera; nor would a standard-definition camera been any smaller than a 720p or 1080p camera.

Design

In order to obtain real-time high-definition video, the system requires a high-speed interface between the camera and the computer. Without compression, a 30 fps 1280×720 video stream, encoded with no chroma subsampling and 8 bits per channel, would require $1280 \text{ pixels} \times 720 \text{ pixels} \times 30 \text{ fps} \times 3 \text{ channels per pixel} \times 8 \text{ bits per channel} = 633$ Mbps. Theoretical data rates for common video formats are presented in Table

Frame Size (pixels)	Frame Rate (fps)	Color Space	Data Rate (Mbps)
1920 × 1080	60p	RGB888	2986
1280 × 720	60p	RGB888	1327
1920 × 1080	60p	YUV422	1991
1280 × 720	60p	YUV422	885
1920 × 1080	30p/60i	RGB888	1492
1280 × 720	30p/60i	RGB888	664
1920 × 1080	30p/60i	YUV422	995
1280 × 720	30p/60i	YUV422	442

Table 1: Theoretical data rates for common formats of uncompressed video.

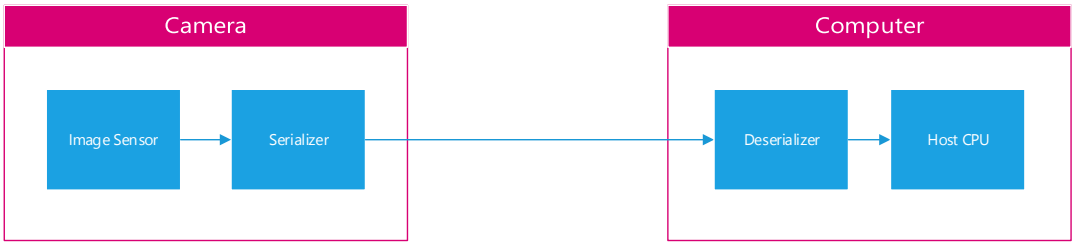

Figure 17: Block diagram illustrating the process of transmitting camera data to a computer over a serial link.

1. It is important to note that these are theoretical data rates – actual data rates are higher, because of the need to include horizontal and vertical blanking as well as other timing and ancillary data.

Interface Selection

All commercially-available chip-level CMOS sensors contain on-board processing that reads the image data from the array, demosaics the Bayer pattern, and presents the data to the host processor using a high-speed parallel interface. The challenge of designing a usable camera does not revolve around designing with or programming the sensor; in reality, these sensors are extremely easy to use. They universally use a low-speed easy-to-use serial protocol, called i^2C, to read and write configuration registers

which control the camera's parameters. For example, to change the output resolution of the sensor, the user writes a single value to a register using a single command. Even the output of the sensors is also relatively straightforward to understand; video is transmitted pixel-by-pixel, in a raster format, starting at the top-left pixel. Sync signals instruct the processor when the next line of data is being sent, or when the sensor reaches the end of the frame (the bottom-right pixel).

Instead, the challenge is figuring out how to transmit the data over relatively long distances (for the camera's 96 MHz 12-bit parallel bus, anything longer than about 6 inches is considered "long distance"), and then, how to get this data into a computer.[3]

These two challenges provide a blueprint for what any generic camera-to-computer system will look like. Figure 17 visualizes this. As the figure shows, digital cameras that interface to computers (or any digital equipment, really) use a circuit that converts the high-speed parallel data from the sensor to a serial data stream that is designed to transmit over long distances. This data is received by the computer, deserialized, and sent to the CPU. This is only a blueprint, however – different systems implement this differently. For example, a consumer USB webcam uses a microcontroller with a USB interface to serve as the "serializer" – data is read from the image sensor, packetized, and sent over the USB interface (which, itself, is obviously a serial interface). The data is received by the USB host controller on the computer's motherboard, deserialized, reassembled, and sent to the host CPU. A professional broadcast video camera uses a different implementation of this blueprint; image sensor data is serialized into an HD-SDI (high-definition serial digital interface) stream. This is sent down a 75-ohm coaxial cable, where it is received by an HD-SDI capture card

[3]Newer image sensors also have a high-speed serial bus that can be used to transmit sensor data directly. This is not so much designed for increasing the distance data can be transmitted, but more just to reduce the number of signals routed on the PCB between the image sensor and the host processor

installed in a computer, which deserializes the data and sends it to the processor.

Although the proposed camera could implement any serial protocol, it is obvious that by selecting an existing established serial protocol, existing components can be used to implement the system. There are really three possible interfaces to choose from that have widespread implementation on PCs:

- USB – completely ubiquitous on all modern desktop and laptop computer systems.
- IEEE1394 (FireWire) – less popular, but still widely available as a built-in feature on the computer motherboard. Computers without built-in FireWire support can use low cost add-in PCI / PCI-Express cards to enable FireWire access.
- HD-SDI – While not a built-in feature in many computers, desktops can support add-in HD-SDI capture cards that are readily available from BlackMagic, Matrox, NVidia, Osprey, and other companies. BlackMagic even has an HD-SDI interface that supports connecting to laptops over ThunderBolt.

We will now examine each of these interfaces in detail, paying special attention to the quality of the video supported by the interface, the latency of the interface, and how the interface is implemented on the end device (camera),

IEEE1394 (FireWire)

FireWire is a full-duplex general-purpose, bus-arbitrated serial interface originally developed by Apple, and later standardized by the IEEE P1394 Working Group. FireWire supports both isochronous (regularly-scheduled) and asynchronous (event-driven) transfers, and comes in both 400 and 800 Mbps flavors. FireWire's most

popular uses include connecting external hard drives and DV cameras, but it is also widely used in machine vision applications in industrial applications. For video applications, FireWire supports two forms of video transfer – DV and IIDC. DV is commonly used for older tape-based consumer (and entry-level professional) camcorders, and has a fixed video format – 720x480 resolution, 30 frames-per-second, YUV 4:1:1 chroma subsampling. IIDC allows the imaging device to specify its video format, which allows it to support HD resolution and different color spaces (including RGB and YUV, at different bitdepths). PCI and PCI-Express can not directly interface to the camera because they are only designed to transmit and receive data over very short distances (confined to the inside of a computer case).

The most challenging part of implementing a FireWire-capable camera is working with the FireWire interface. Although many FireWire controllers, such as the Texas Instruments TSB41BA3D, simplify interfacing with a FireWire bus, these ICs do not provide a transparent conduit for transferring data – rather, they provide a low-level interface to a FireWire bus; they must be attached to a microcontroller or FPGA which is responsible for communicating with the host to establish the device type, capabilities, parameters, and other details. FireWire transceivers also provides data encoding and decoding, plus a parallel-to-serial buffer. Once the link is established, the microcontroller or FPGA must packetize the desired data and transmit it appropriately. Because of this, an image sensor cannot interface directly with one of these FireWire controllers; instead, an FPGA or fast microcontroller must sit between it and the FireWire controller to read the streaming data from the image sensor, packetize it, and send it to the FireWire controller. Although this is a relatively simple operation, it must be done at a very high data rate (hundreds of megabits per second).

An exhaustive search yielded no results for a microcontroller with a built-in FireWire interface, so building a FireWire-enabled camera would require at least a

Active Image Format	Total Image Size	Frame Rate	Color Space	Data Rate
1920 × 1080	2200 × 1125	60p	YUV422 (10 bit)	2970 Mbps
1920 × 1080	2200 × 1125	30p / 60i	ARGB4444 (10 bit)	2970 Mbps
1920 × 1080	2200 × 1125	30p / 60i	YUV422 (12 bit)	2970 Mbps
1920 × 1080	2200 × 1125	30p / 60i	RGB444 (12 bit)	2970 Mbps
1280 × 720	1650 × 750	120p	YUV422 (10 bit)	2970 Mbps
1280 × 720	1650 × 750	60p	ARGB4444 (10 bit)	2970 Mbps
1280 × 720	3300 × 750	30p / 60i	ARGB4444 (10 bit)	2970 Mbps
1920 × 1080	2200 × 1125	30p / 60i	YUV422 (10 bit)	1485 Mbps
1280 × 720	1650 × 750	60p	YUV422 (10 bit)	1485 Mbps

Table 2: Common SMPTE formats for HD video.

three-chip solution: the image sensor, the microcontroller/FPGA, and the FireWire controller. Furthermore, because no application-specific (i.e., optimized) microcontrollers were found, a general-purpose high-speed ARM microcontroller or an FPGA would most certainly have to sit between the image sensor and the FireWire interface to glue the parallel digital video to the packet-based FireWire communication. Also, the limited speed of the interface means that lossy video compression (most likely MJPEG) would have to be used, as uncompressed video (even with chroma sub-sampling) simply has too much bandwidth to transmit over a 400 Mbps (or even an 800 Mbps) bus.

None of these FireWire controllers were designed to be particularly small; any possible FireWire design built without using custom silicon would most certainly violate the board size constraint in the design specifications. FireWire also requires double differential links, which increases cabling size. Because of all this, a FireWire interface would not be suitable for implementation of this project.

Figure 18: HD-SDI Transmitter Block Diagram

Data Stream	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
DS1 First Word	Y [9:0]									
DS1 Second Word	alternating U [9:0] or V [9:0]									

Table 3: SMPTE 20-bit data format for 10-bit YUV

Data Stream	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
DS1 First Word	G [9:0]									
DS1 Second Word	R [9:0]									
DS2 First Word	A [9:0]									
DS2 Second Word	B [9:0]									

Table 4: SMPTE 20-bit data format for 10-bit ARGB 4444

Data Stream	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
DS1 First Word	1	0	0	0	Y [11:6]					
DS1 Second Word	1	0	0	0	Y [5:0]					
DS2 First Word	1	0	0	0	alternating U [11:6] or V [11:6]					
DS2 Second Word	1	0	0	0	alternating U [5:0] or V [5:0]					

Table 5: SMPTE 20-bit data format for 12-bit YUV 422

Data Stream	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
DS1 First Word	$\overline{B8}$	R [11:9]			G [11:9]			B [11:9]		
DS1 Second Word	$\overline{B8}$	R [5:3]			G [5:3]			B [5:3]		
DS2 First Word	$\overline{B8}$	R [8:6]			G [8:6]			B [8:6]		
DS2 Second Word	$\overline{B8}$	R [2:0]			G [2:0]			B [2:0]		

Table 6: SMPTE 20-bit data format for 12-bit RGB 444

HD-SDI

Another interface that was explored was HD-SDI – the high definition serial digital interface – which is ubiquitous in broadcast video. This interface supports two data rates – 1.485 and 2.97 Gbps – and more than a dozen video formats. Some of these formats are presented in Table 2. While FireWire is a complex, general-purpose bus, HD-SDI is a simple point-to-point serial link designed specifically for high speed video. Because the protocol is so simple, it can be implemented in low-level digital circuitry (like ASICs or FPGAs); it does not need a general-purpose microprocessor.

A block diagram overview of an HD-SDI transmitter is shown in Figure 18. 20-bit video data is fed into the FIFO in the formats specified by Tables 3 to 6. The active video area (which contains the visible image) is indicated by an SAV (start active video) symbol; at the end of the line, an EAV (end active video) symbol appears. After the EAV symbol, two additional words are inserted into each video line to indicate the current line number line number. A CRC (cyclical redundancy check) code is calculated using the the generator polynomial

$$\mathrm{EDC}\left(x\right) = x^{18} + x^5 + x^4 + 1.$$

The data is then serialized, LSB first. This serialized data is channel coded using scrambled NRZI (non-return-to-zero, inverted). The serialized bit stream is scrambled using the generator polynomial

$$G\left(x\right) = \left(x^9 + x^4 + 1\right)\left(x + 1\right).$$

Scrambling is a pseudorandom operation that widens the spectral content of the serial stream and eliminates streams of 1s and 0s. At the same time, scrambling does not

introduce any additional overhead (like 8b/10b or other line codes do).

The most popular format, 10-bit YUV 422, is specified in Table 3 and only requires one data stream per pixel: notice that this format uses 422 chroma subsampling, so the two-word chroma values are split between pixels. In other words, the first data stream contains the Y (luminance) value for the first pixel, as well as the U color component for the first and second pixel; the second data stream contains the Y value for the second pixel and the V color component for the first and second pixel. This means that the serializer's data latch clock runs at the same speed as the pixel clock. The last row of Table 2 is an example of a video standard that makes use of the 10-bit YUV 422 format. We can calculate the parallel latch's input clock by first calculating the pixel rate as

$$1650 \text{ pixels/line} \times 750 \text{ lines/frame} \times 60 \text{ frames/sec} = 74.25 \text{ Million pixels/second.}$$

Since each pixel takes one clock cycle to read in, the clock rate is therefore 74.25 MHz. Notice that the preceding format, 1920×1080 30 fps 10-bit YUV422, also runs at 74.25 MHz. This is the original HD-SDI format, which has a nominal output data rate of 1.5 Gbps, and is now known as "single-link HD-SDI". By doubling this parallel clock rate from 74.25 MHz to 148.5 MHz, we can either double the frame rate of the video, or increase the quality of color reproduction. This can either be accomplished by adding a second HD-SDI lane (known as dual-link HD-SDI), or by simply doubling the clock speed of the interface, known as 3G-SDI. These formats appear first in Table 2, and all have a nominal data rate of 3 Gbps.

Three additional color formats are supported by the 3G-SDI / Dual-Link standard: 10-bit ARGB4444, 12-bit RGB, and 12-bit YUV422. The 10-bit ARGB4444 format, shown in Table 4, uses two data streams (either one after another in the case of 3G-

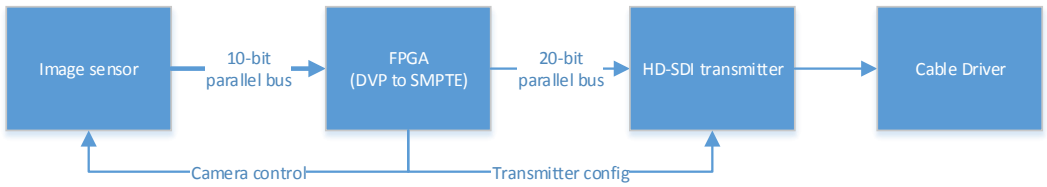

Figure 19: HD-SDI Camera Block Diagram

SDI or two simultaneously, in the case of dual-link) for transmitting color; the first data stream transmits the green and red components in the first and second word respectively, while the second data stream transmits the alpha and blue components. The alpha channel describes the opacity of the pixel, and is useful for broadcast video compositing (mixing video feeds together). For example, if white text is to be overlayed on top of a live video stream from a camera, the video mixer needs to know which pixels from the character generator should be transparent and which ones should not be. To this end, the character generator can output an ARGB signal where the background of the image has 0% alpha (totally transparent), while the foreground (the white text) has a 100% alpha.

If the alpha component is not required, the double-speed stream can also be used to transmit RGB video using 12-bit sampling depth instead of the standard 10-bit used for the other video formats discussed so far. This format splits up the 12-bit color values according to Table 6. Although 12-bit YUV 422 (Table 5) is a recognized standard, it is not used as often, since 12-bit RGB provides the same sampling depth and better color resolution.

It should be clear by now that HD-SDI is an ideal fit for transmitting video – it was designed from the ground up solely for this purpose, and supports near-zero latency, uncompressed full HD video. The principle components in an SDI camera

consist of an image sensor, an FPGA, an SDI transceiver, and a cable driver. This is presented in Figure 19. Some FPGAs (like the Altera Cyclone IV GX and other higher-end Altera FPGAs, along with several Xilinx and Lattice FPGA products) integrate a reconfigurable general-purpose SERDES chain that can implement the HD-SDI transceiver logic. Because of the extremely high switching speed of this block (2.97 Gbps), the serializer cannot be implemented in standard FPGA logic, so if the FPGA does not have a built-in SERDES peripheral, an outboard transceiver circuit must be used to serialize and scramble the 20-bit SMPTE data. An analog component – the cable driver – is responsible for driving the digital signal into a 75-ohm coax cable. It is designed to drive the signal strongly enough to meet the rise/fall timing requirements, while matching the cable impedance and minimizing transmission line reflection.

Obviously, this application is highly constrained in terms of size, so designing a system with an outboard HD-SDI transmitter is out of the question. Altera's Cyclone IV GX FPGAs are the lowest-cost devices with integrated SERDES functionality that can implement a 3G-SDI transceiver. The devices are available in different packages with various amounts of logic. The N148 package has 15,000 gates, and uses an 11×11mm 0.5mm pitch BGA. This package is extremely expensive to design with, since it requires special PCB manufacturing techniques – via-in-pads, laser-drilled microvias, etc – to be able to route signals to the package properly. The F169 package uses a 1.0mm pitch BGA, and measures 14×14. It has a similar number of I/O as the N148 package; it is also available in higher logic densities of 22,000 and 30,000 gates. Since this FPGA's package has moved up to 1.0mm pitch, traces can now be routed between pads, and standard 8mm vias can be dropped between pins. The Cyclone IV GX EP4CGX30 (with 30,000 logic cells) in the F169 package costs approximately $80 a unit in single quantities.

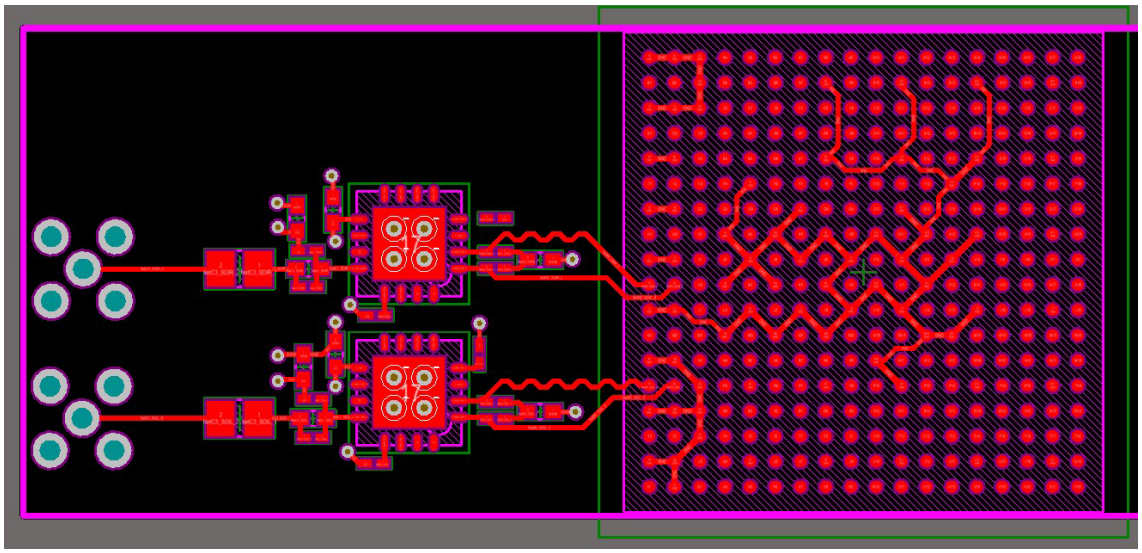

Figure 20: Example routing of a dual HD-SDI transmitter. A 324-ball 19 × 19mm FPGA is attached to two cable drivers; one for the left camera, and one for the right. The squiggly traces are designed to ensure both the positive and negative traces in the differential signal pair arrives at the transceiver at the same time.

To gauge the feasibility of implementing an HD-SDI camera, an Altera Cyclone IV development kit was purchased (along with an HD-SDI transceiver board) to aid the development of firmware. At the same time, the PCB design process was started to gauge the minimum size of the board that could be achieved (when constrained to standard-spec PCB manufacturing techniques). The initial design used the aforementioned 14 × 14mm FPGA; however, several weeks into the development process, it was discovered that this particular FPGA only supported a single 3G HD-SDI link. The device was chosen based on the fact the datasheet indicated it supported two transceivers; however, it was discovered that only one transceiver could operate in high-speed 3G mode (2.97 Gbps). Because of this, the PCB design moved to the larger 19 × 19mm BGA. The design shown in Figure 20 shows a best-guess of the minimum size required for a two-camera 3G-SDI PCB, assuming dual-sided component placement. Although it may appear to have plenty of extra room, this design does not

include image sensors or the necessary PMIC (power management integrated circuit) required to provide the analog and digital 1.2, 1.5, 1.8, 2.8 and 3.3V supplies needed for the analog and digital components of the image sensor, the FPGA's core logic, peripheral logic, and integrated transceiver, and the cable driver power supplies. The board measured 1.8×0.8 inches in size, which falls outside of the design specification.

USB

So far, this discussion seems to indicate that transmitting live video to a computer is inordinately difficult, costly, and bulky. Yet, every day, millions of people videochat with friends and relatives over the internet using inexpensive webcams that can be purchased for under $25. How do these cameras transport video to computers? Can the same technology be used in this application?

The most commonly used computer interface supporting video is the Universal Serial Bus (USB) interface; this is the ubiquitous interface used by webcam manufacturers. Version 2.0 of the USB specification provides a High-Speed mode supporting up to 480 Mbps transfer rate, which, calculations show, is capable of streaming a stereo pair of JPEG-compressed frames of 1280×720 resolution video at 30 fps using reasonably high-quality JPEG compression settings. While similar in spirit to FireWire, USB is much simpler to implement; many microcontrollers ship with some sort of USB transceiver block built-in.

USB 2.0 transceivers capable of High-Speed mode, however, usually only exist in mid-level 32-bit ARM microcontrollers. An example, the STM32F4, is available in a high-density, ultra-fine-pitch (0.4mm) BGA package measuring only 4.2×4.0 mm, but the circuit board specifications required to fan-out the pins on the package would make the PCBs cost-prohibitive to manufacture. The microcontroller is also expensive and provides more processing power than is necessary to read in a frame of video and

push it into a USB buffer. When disassembling off-the-shelf microcontrollers, it was discovered that these products use application-specific 8-bit microcontrollers coupled with features not typically found in 8-bit MCUs, like a high-speed USB 2.0 peripheral interface (as well as the necessary DMA pathways to be able to use it) and a dedicated camera interface port. It seemed like the best course of action was to design a system using one of these processors.

Other interfaces

For completeness, other interfaces that were not considered are discussed below. All of these interfaces required far too much circuitry to make them practical to implement in a system whose size is so tightly constrained. Entirely different from traditional peripheral interfaces, computers also have network access through WiFi and Ethernet connections, however, the author does not consider this a peripheral interface; although many embedded devices are starting to ship with Ethernet ports (including cameras), TCP/IP is not designed for real-time data transfer, and there is significant overhead involved in sending and receiving network traffic; these devices usually offer network interfaces simply out of convenience to the end-user (for example, security cameras with WiFi access can be easily set up around a building and configured to stream video to a centralized server for storage). Returning to the long-running theme in this project, both Ethernet and WiFi were eliminated as possible interfaces simply because of the size constraints imposed by the design specifications; either interface would require, at a bare minimum, a large FPGA/ARM combination IC, RAM and flash memory ICs, along with an Ethernet transceiver module, magnets, and a PMIC to power the system.

PCI-Express is a high-speed peripheral interface designed for accessory cards installed in a desktop computer. Thunderbolt is an external, mobile implementation

of PCI Express. Both of these interfaces would easily be able to handle the data rate of uncompressed video, however, again, they would require large FPGAs and transceiver/cable driver units, as well as several pairs of cables to provide the needed signaling. Size constraints of the device rules out the PCI-Express/ThunderBolt interface.

DVI/HDMI suffers problems similar to the HD-SDI and PCI-Express solutions; it requires a large transceiver/cable driver IC, along with a powerful FPGA. The board would have to be slightly larger than the HD-SDI transceiver board, and would require more cabling, too (as DVI/HDMI uses 4 twisted pairs to transmit video data, along with additional wires for control).

Another option investigated was using an off-the-shelf camera module. While highly-integrated single-chip camera modules exist, every chip investigated provided an analog NTSC composite video output. This is a low-quality standard-definition analog video connection that would require separate hardware for the computer to capture video. This was ruled out, since it did not provide high-definition video, and external computer hardware would be required for capturing video.

Processor selection

With the USB 2.0 interface selected, the next task is to find a USB camera controller suitable for this application. Sonix, Alcor, and OmniVision are some of the companies that make the controllers found in off-the-shelf consumer webcams; and all of these companies manufacture ICs that, for all intents and purposes, are functionally identical. The project, at this point, is less about circuit design and computer science, and more about supply chain management and finding communication channels with suppliers.

The bulk of the work (and time spent on the project) involved navigating the Asian import market (where most of these controllers are developed) and trying to find a vendor willing to partner with us on a project with such a low yield. The only vendor that was willing to supply us with development tools was EETI with their EM2780 USB camera controller.

EETI's EM2780 camera controller integrates an Intel 8051-core microcontroller with a USB peripheral and DMA bus between the parallel video interface and the USB peripheral. The controller loads its firmware off serial Flash memory. The firmware image for the camera controllers was provided by EETI, and instantiates a USB Video Class (UVC) compliant interface using control and bulk endpoints. Source code for the firmware was not provided, so the operation of the firmware cannot be discussed here.

EETI supplied a demonstration board and reference schematics for the EM2780 USB camera controller IC, as well as a firmware binary image for the demonstration board. The EETI demonstration board used OmniVision's OV9710 image sensor, which has no publicly-available datasheets. OmniVision refused to send any information on the OV9710, even under a Non-Disclosure Agreement.

No firmware source code was available, which meant that a firmware rewrite (from scratch) would be required in order to use a different image sensor. General-purpose microcontroller vendors, such as TI, Microchip, Atmel, and Freescale, usually offer a wealth of development tools, including C header files, peripheral libraries, and example code. While the EM2780 datasheet does have a list of registers, along with short descriptions of each one, it would have taken too long to build up a development environment from scratch, including writing a USB 2.0 stack that was compatible with their transceiver. So, no attempts were made to modify the provided firmware; thus, the design was locked to the OmniVision OV9710.

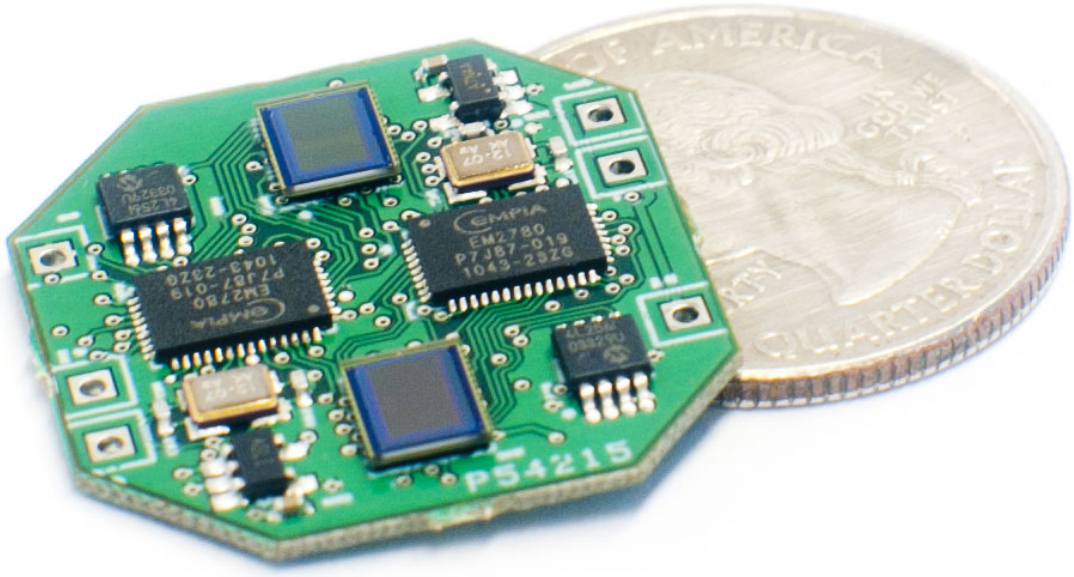

Figure 21: Completed camera, shown without lenses mounted.

The OV9710 image sensor is a 1/4" format sensor with a 1280×800 image array and can deliver video at up to 30 fps. The sensor features on-board image processing and formatting capabilities, allowing programmable resolutions, auto exposure, and auto white balance. The image sensor was programmed for 720p HD video capture. Detailed technical information on the OV9710 and its operation is unavailable, as the datasheet for the image sensor is only available to approved parties, and is protected by a Non-Disclosure Agreement.

A block diagram of the proposed system is shown in Figure 22. In the figure, it can be seen that the proposed camera system consists of two image sensors, two controllers, and a voltage regulator for supplying power. The only integrated circuits not present in this figure are the two serial flash ICs used to store program data used by the controllers.

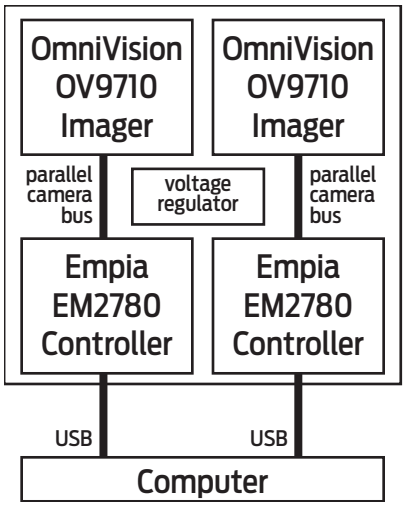

Figure 22: Block diagram of proposed camera

Circuit Design

To design the camera, two copies of the reference design from EETI were placed on the circuit board; one for the left camera, the other for the right camera.[4] Minor modifications to the reference design were made: first, a transistor allowing the EM2780 to control power to the OV9710 was removed; second, one of the voltage regulators was removed – this way, one voltage regulator powers both designs. These changes minimize the number of components on the circuit board while maintaining compliance with EETI's design.

A 4-layer 0.031" PCB was manufactured and assembled by Advanced Circuits on their standard-specification fabrication line with 5mil trace widths and 0.01" hole sizes. The total size of the design is 26mm×24mm. A picture of the completed board is shown in Figure 21, and Figure23 shows the PCB artwork for the PCB. The red and blue traces indicate the top and bottom layer; the green traces indicate the middle layers (power planes are not shown, but exist on all four layers).

[4]Actual schematics of the reference design used cannot be printed here, as it is protected by a Non-Disclosure Agreement in place with EETI.

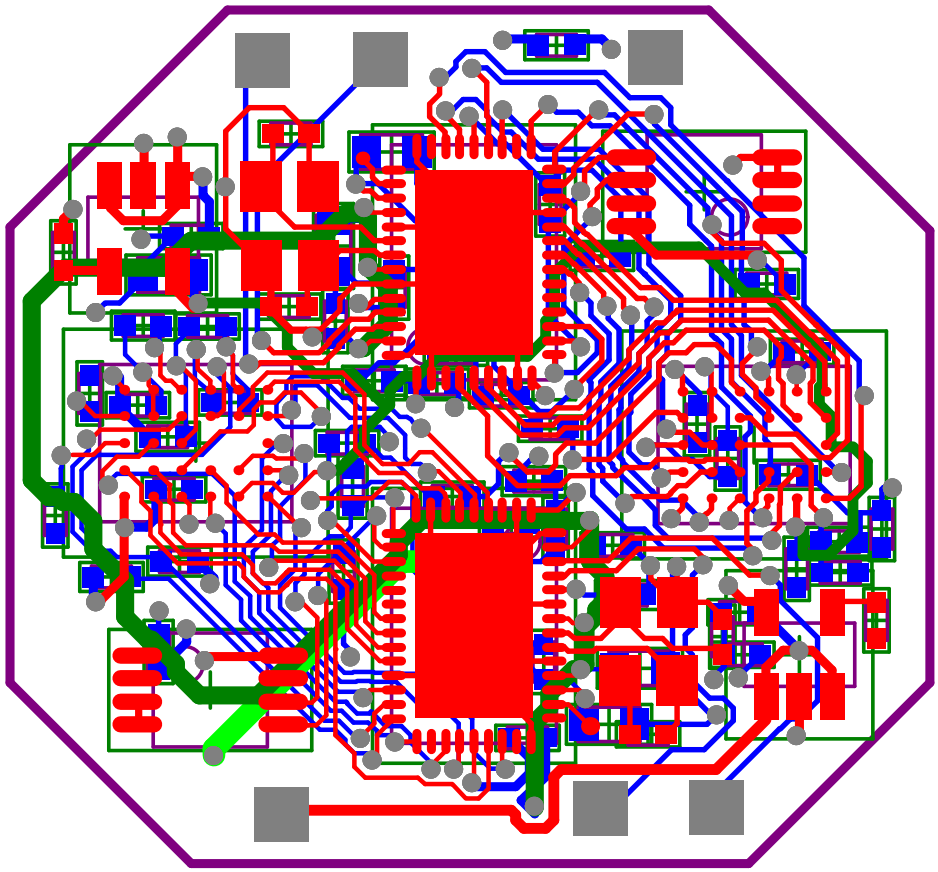

Figure 23: Composite PCB drawing.

The passives were supplied by DigiKey and Newark; the EM2780 controllers were supplied directly by EETI. It was very difficult to find a supplier willing to sell low-quantity OV9710 image sensors (at any price), so it took several months of communicating with various businesses to find a supplier. Eventually, a U.S.-based importer of electronic components was found, Odyssey Electronics, who helped us track down a shipment of the image sensors.

The first revision board came back from Advanced Circuits nonfunctional due to a package sizing problem with the image sensor. Since the datasheet for the image sensor is unavailable, the package dimensions of a similar OmniVision sensor, the OV9740, were used. Although the OV9710 and OV9740 are similar in terms

of high-level specifications (resolution, imager size, output format, price, etc), they have different package sizes and ball pitches. This was the only major setback of the project – after several emails to EETI's engineering support, a scanned page from the OV9710 datasheet with the package outline was sent over. A second revision was assembled, which functioned identically to the EETI evaluation module.

Camera Performance

The frame rate and the sharpness of the image were used as metrics to compare the performance of the camera to both a standard-definition analog camera, as well as a commercially-available USB-based Logitec C905 webcam. The C905 was selected because it has the same sensor resolution and interface as the implemented USB camera.

Resolving Ability

The resolving ability of the implemented camera was compared with a standard-definition stereoscopic analog camera. Optical resolution is a measurement (in lines per inch) of a camera system's ability to distinguish black and white lines from 50% gray The image sensor's pixel resolution, the camera's lens, and the processing algorithms performed on the platform all influence the optical resolution of the system. A consistent daylight-balanced florescent light source, measuring 2491 lux, was used to light the scene consistently throughout the experiment. Low-cost plastic thin-profile Sunex DSL756 3.8mm f/2.8 quarter-inch format lenses, which provide a wide-angle field of view of 60°, were used for both the analog and the digital cameras; in this way, the non-ideal optical properties of the lens affect both systems in the same way, which creates a better comparison between image sensors.

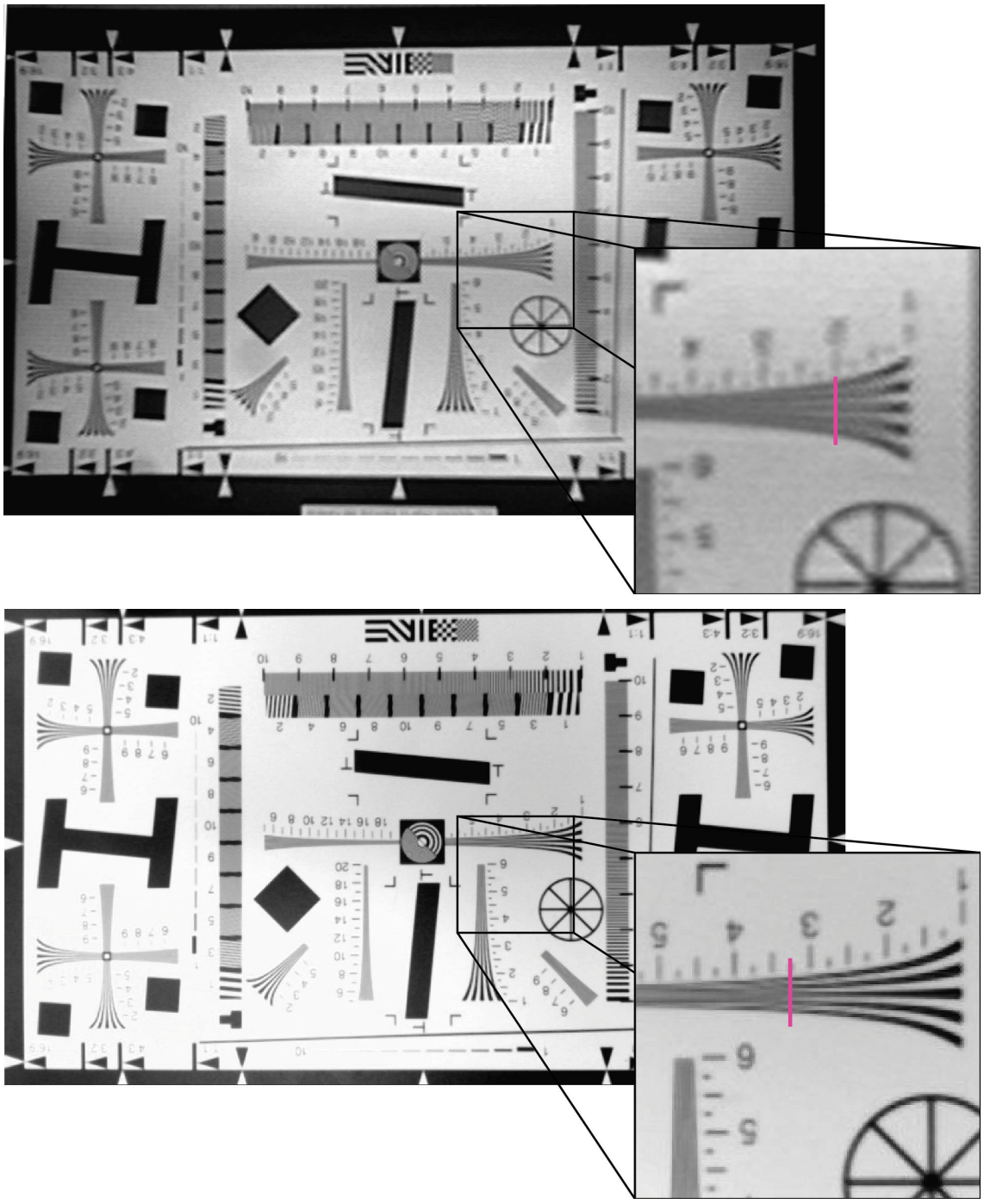

Figure 24: Resolution test comparing the analog standard definition camera (top) and the implemented USB 720p HD camera (bottom). The purple lines indicate the approximate threshold of resolving ability.

The resolution was tested using the ISO 12233 resolution chart[20]; the ISO 12233 is an industry-standard graphic for determining optical resolution; it allows different cameras to be compared directly with each other. From visual inspection, captured images from the standard definition camera and proposed high-definition camera given in Figure 24 illustrate the implemented USB camera is capable of resolving approximately 325 lines-per-inch, while the analog standard-definition camera can resolve approximately 200 lines per inch; this is consistent with the resolution disparity between 1280x720 high-definition video and 640x480 standard-definition video.

Noise Performance

An important metric used to judge the quality of a camera system is a measurement of noise. For the purposes of this paper, noise can be defined as any phenomena that distorts the ability of the system to accurately record the image that has been projected onto the image sensor. Modern digital cameras are afflicted by noise from a variety of sources:

- *Quantization* – real-valued pixels are mapped to one of a finite number of discrete levels. The resulting image has noise due to rounding errors.
- *Conversion and transmission errors* – "salt and pepper" noise is caused by bit errors and analog-to-digital conversion errors.
- *Photodiode Leakage* – leakage through the image sensor's photodiode array causes capacitors to have inaccurate charge values.
- *Thermal noise* – this is the dominant source of noise; it is caused by agitation of charge carriers in semiconductor devices.

Figure 25: The experimental set up.

- *Compression artifacts* – lossy image compression changes the per-pixel values of the image, which can be considered a form of noise (though this is usually treated as a separable phenomena).

Since most noise comes from the image array, the gain of the camera system correlates strongly with noise. When the image array is pointed at a poorly-lit scene, to obtain proper output levels, the pixel values must be greatly amplified. This multiplies the amount of noise in the image, and introduces additional noise as well (since all amplifiers introduce thermal noise and nonlinearities).

In order to gauge the noise profile of the camera system, the camera was placed in a suction-grip vice and pointed at a test pattern, which consisted of a black-to-white radial gradient (Figure 26). This pattern is thought to be the ideal test pattern to use, as it requires the camera to register every possible value for the red, green, and blue channel. Because this pattern has no sharp edges or lines, it is less critical to achieve

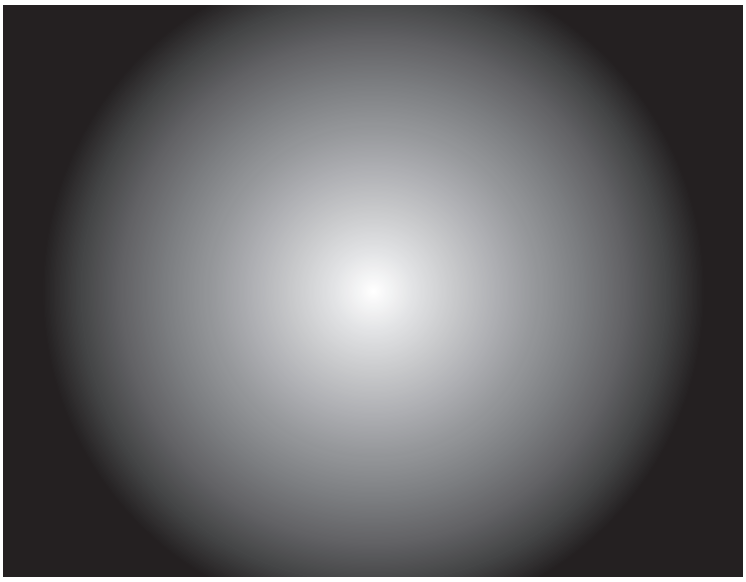

Figure 26: The test pattern used during the experiment.

perfect alignment between captured frames; this makes the variance calculation more accurate (i.e., even with minor registration issues, the captured images should have nearly zero variance except in the presence of noise). The light level was measured using an Extech HD450 to be 517 Lux. Next, $N = 12$ images $\left\{I^{(1)}, I^{(2)}, \ldots, I^{(N)}\right\}$ of a test scene were captured with a stationary camera pose. A picture of the experimental set-up is shown in Figure 25.

To calculate the variance of each pixel, $\text{var}\left[p_{x,y}\right]$, first the arithmetic mean of the pixel is calculated as

$$\bar{p}_{x,y} = \frac{1}{N}\sum_{i=1}^{N} p_{x,y}^{(i)}.$$

Then, the variance of $p_{x,y}$ is

$$\sigma_{x,y}^2 = \text{var}\left[p_{x,y}\right] = \sum_{i=1}^{N}\left(p_{x,y}^{(i)} - \bar{p}_{x,y}\right)^2.$$

This process was repeated for each $p_{x,y}$, and for each red, green, and blue channel.

First, total signal-to-noise ratio (SNR) was calculated by averaging all $\sigma^2_{x,\,y}$ variances together into σ^2 and all of the mean pixel values $\bar{p}_{x,\,y}$ into μ. Then, the $\mathrm{SNR}_{\mathrm{dB}}$ is

$$\mathrm{SNR}_{\mathrm{dB}} = 10\,\log_{10}\left(\frac{\mu^2}{\sigma^2}\right),$$

After the aforementioned calculation, it was found that the custom-designed USB camera achieved a total SNR of 39 dB; the Logitech camera had a measured SNR of 44 dB.

To further characterize the noise profile of the implemented camera, the SNRs were calculated as a function of intensity for each of the three color channels – this provides a strong visualization to show what parts of an image are most likely to be noisy. To do this, each of a channel's pixel variances were grouped based on average pixel value across all test images, i.e.,

$$S_v = \left\{\sigma^2_{x,\,y} \;:\; \bar{p}_{x,\,y} = v\right\},$$

and these groups were averaged independently as $\bar{S}_v$; for example, the set of variances of all red pixels that have an average value of 5, $S_5 = \left\{\sigma^2_{x,\,y} \;:\; \bar{p}_{x,\,y} = 5\right\}$, were averaged together into $\bar{S}_5$.

The results of this evaluation for both the Logitec web cam and the implemented camera are presented in Figure 27. Oddly, in every scenario, the red channel appears most noisy – typically, the blue channel will appear most noisy, especially in low-intensity regions of the image [23, 39, 43]. This discrepancy is probably because of the camera's poor-quality auto white balancing – i.e., the red channel's gain was significantly higher than the green and blue channels. Higher gain always contributes to more noise. Manually overriding this white balance setting would simply shift the noise to a different channel, while not actually reducing it. Noise is more prevalent

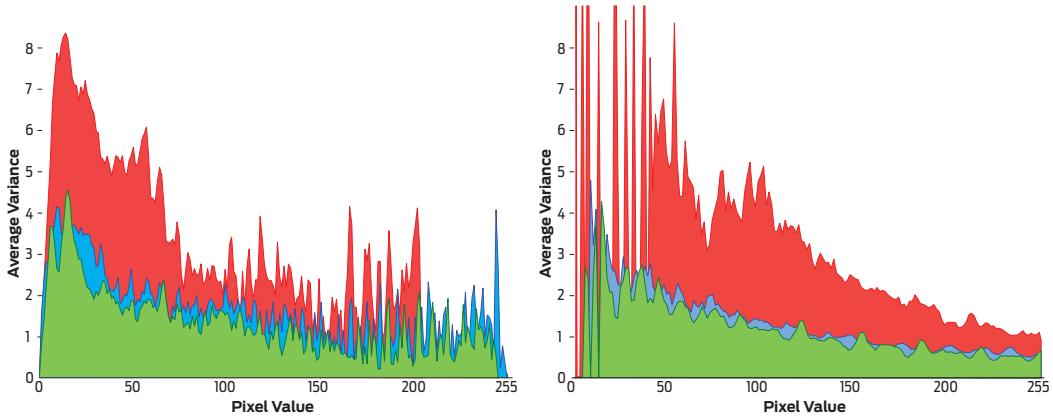

Figure 27: Noise performance in the red, green, and blue channels for the reference Logitec webcam (left) compared to the implemented camera (right).

in dark pixels than in brighter pixels overall, this is seen most dramatically in the red channel. For Bayer-pattern sensors, the green channel should be the least noisy, since there are twice as many green pixels than red or blue ones. The reason the implemented camera's low-intensity noise has so much variation (compared to the obvious trends seen in brighter values) is because there were very few pixels with an average value less than 50 in the image – i.e., there were very few dark areas in the captured images. After reviewing the captured images, each one was visibly washed out; the black outside of the test image was a muted gray color. This is caused by two phenomena: the auto-exposure control used by the camera's firmware needs to be adjusted – the camera consistently produces images that are over-exposed. Secondly, the 3D-printed enclosure the camera is mounted inside is not perfectly light-tight; despite plenty of duct-taping and other attempts at reducing light-leak, light is getting inside the camera box, reflecting around off the glossy 3D plastic interior, and hitting the image sensor, which causes a washed-out, low-contrast haze on all images captured. This is extremely problematic and cannot easily be corrected for in software.

Frame Rate

The number of distinct frames an image sensor supplies per second is an important metric when evaluating platforms. Because the EETI controller exposes a 30 frames-per-second 720p video stream, any application connecting to it will report 30 fps playback. However, this does not indicate that the camera is capable of delivering 30 fps performance – not every frame turned out to be unique; i.e., some of the frames were dropped, and the previous frame was substituted in its place.

To test the frame rate of the camera, video of a quickly-moving scene (specifically, a graphical stopwatch application running on a computer monitor) was recorded for ten seconds. The number of distinct frames was counted and compared with the total number of frames. The results of this experiment are shown in Table 7. The results show the implemented EM2780-based device is incapable of capturing real-time video. Because of the closed-source nature of the platform, no testing or analysis can be performed to explain the discrepancy between the EM2780 specifications sheet and the results obtained. This limitation was observed on the reference design provided by EETI, so the PCB layout can be ruled out as a cause. This is most likely a bug with the demonstration firmware that was loaded onto the camera.

This limitation represents a significant problem with the proposed platform – because it is only capable of streaming video at approximately 16 fps, and because it drops frames unpredictably, it does not provide a realistic view of the surgical scene for the surgeon, and it also is unsuitable for stereoscopic image processing applications, since the frame rate fluctuates, and there is no synchronization between frames in the left camera and frames in the right camera. This means the platform is only useful for stereoscopically processing static scenes that do not change over time, or change very slowly.

Camera	Total Frames	Unique Frames	Frame Rate
Implemented EM2780 device	300	158	16 fps
Logitec USB 2.0 webcam	300	300	30 fps

Table 7: Frame Rate Tests

Other Parameters Tested

Power Consumption

While not relevant to our requirements, power consumption was also measured by measuring the current consumed, in-circuit, with a multimeter, as well as the voltage appearing at power terminals of the device. After multiplying these quantities together, it was found that the implemented camera consumes approximately 1.5W of power, across both USB ports (which is equivalent to 0.75W per camera). This per-camera figure is well below the 1.5W maximum allowed by USB's 500 mA-per-port current limit. The reference Logitech C905 webcam uses approximately 0.7W per camera. The discrepancy is most likely due to different image sensors or chipset features.

Temperature

Somewhat more important in our application is the temperature profile of the board. Both the imager's BGA package, as well as the processor's exposed pad QFN package are specifically designed to transfer heat to the PCB. To effectively do this, the PCB must be carefully designed such that enough copper area is connected to the package with low thermal resistance. Most of these design techniques were ignored to reduce board size. To measure the board's temperature, a Fluke infrared thermometer was used. The board heats up to 126 °F within 1 minute of powering the board. This is similar to the Logitech's thermal profile.

Conclusion

Given the design constraints, a low-cost stereo camera was designed using off-the-shelf webcam technology. While this camera is much larger than conventional laparoscopic cameras, it provides much higher quality optical resolving ability and is suitably sized to fit on current-generation *in vivo* robotic platforms. When compared to a commercial off-the-shelf webcam, the proposed device has similar SNR performance – though firmware changes could enhance noise reduction and fix the mentioned auto-exposure gain control problem. It is also apparent that more time should have been spent designing a suitable enclosure for the device. A better enclosure that provided better sealing would prevent light leaking in – and the enclosure should be coated with a matte finish to prevent any stray light from reflecting around inside the box. Also, the two cameras should be separated with some sort of wall or curtain to prevent light leaking from the left lens to the right sensor, and vice-versa.

The biggest problem, however, is the random frame dropping. This reduces the effective frame rate of the device, making it awkward to use for live viewing. And because frames are dropped randomly and without notice, it makes it nearly impossible to use captured footage for stereo computer vision tasks that require synchronized video. It is speculated that the source of this problem is in the inner-workings of the EM2780 device, or its firmware, but further investigation cannot be completed without obtaining the source code for the device, or writing new firmware from scratch.

Applications

Once the proposed stereoscopic camera was implemented and tested, several different uses were explored. This chapter will explore some applications of the proposed stereoscopic camera.

Stereo Image Processing

In addition to allowing the surgeon to view the surgical scene using a 3D monitor, the proposed camera enables many novel computer vision algorithms that rely on a stereoscopic camera. These can augment the view of the surgical environment, enhance the control of the surgical robot, and, eventually, help automate surgical tasks. All stereo image processing algorithms generally do the same thing:

1. Establish the projection matrix for the two imagers that comprise the camera – in other words, establish the mathematical model for determining where a feature in 3D space gets projected in the left and right views.

2. Attempt to match pixels (or features) between the left and right image.

3. Using these feature correspondences, calculate the original point in 3D space that projected these features onto the two imagers.

Once features found in images are mapped back to their original 3D points found in space, many things can be done with this information. For example, if the camera is mounted to a surgical robot, the precise location of the robot's arms can be determined without requiring encoder feedback [52]. Because stereo image processing can rapidly (and precisely) measure the surgical scene, it can be used in diagnostic [69] and triage [53] scenarios. Because stereo algorithms can provide information about the surgical scene, these algorithms can also be used to align preoperative scans with a view of the surgical scene [14, 6], which can be used to augment the view of the surgical scene for the surgeon. And – applicable to general robots more broadly – since stereoscopic algorithms can be developed to produce dense 3D maps of the environment, they can be used for automated robot navigation.

All multiview image processing algorithms make use of projective geometry, so it is important to start with a formulation for the pinhole camera model. After that, epipolar geometry will be discussed, which will allow us to back-project 2D points into 3D space.

Homogenous Coordinates

Throughout this section, we will make use of homogenous coordinates. Homogenous coordinates are especially useful for projective geometry since all coordinates (including those at infinity) can be represented as finite homogenous coordinates. Also, transformations and other formulations are much more compact with homogenous coordinates than with Cartesian coordinates. All common transformations, including projection, rotation, translation and scaling, can be represented as a linear operation with a transformation matrix.

A point in a plane may be represented by a pair of coordinates $(x, y) \in \mathbb{R}^2$. We

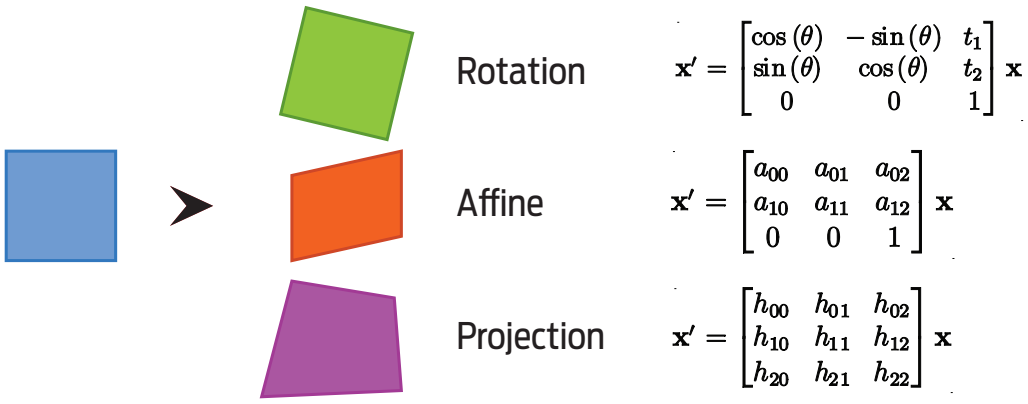

Figure 28: Matrix representations of various two-dimensional transforms

can think of the plane as being identified by $\mathbb{R}^2$, and thus, considering $\mathbb{R}^2$ as a vector space, the previous point can be thought of as a vector $[x, y]^\intercal$. Now, in Euclidean space, we can represent a line as $ax + by + c = 0$, where different choices of a, b, and c create different lines. Then, we can represent a particular line as a vector $[a, b, c]^\intercal$. Interestingly, $k\,[a, b, c]^\intercal$ represents the same line for all values of k – so two vectors related by a scaling constant are equivalent. This is the basis of homogenous vectors. A point $\mathbf{x} = [x, y]^\intercal$ lies on the line $\mathbf{l} = [a, b, c]^\intercal$ iff $ax + by + c = 0$, i.e., iff $[x, y, 1] \bullet [a, b, c]^\intercal = 0$, written in inner-product notation. Thus, all vectors $[kx, ky, k]\,, \forall k \in \mathbb{R}$, represent the same point $[x, y]$. In homogenous coordinates, $[x, y, w]^\intercal$ represent a two-dimensional point or line.

Representing coordinates this way is very useful for projective geometry. The point $\mathbf{x}$ lies on $\mathbf{l}$ iff $\mathbf{x}^\intercal\mathbf{l} = 0$. The intersection of two lines $\mathbf{l_1}$ and $\mathbf{l_2}$ is the point $\mathbf{x} = \mathbf{l_1} \times \mathbf{l_2}$. The line $\mathbf{l}$ through two points $\mathbf{x_1}$ and $\mathbf{x_2}$ is $\mathbf{l}$=$\mathbf{x_1} \times \mathbf{x_2}$.

Transformations

Points represented using homogenous coordinates can easily be transformed using transformation matrices. Transformation matrices are often classified by how many degrees-of-freedom they possess. While all two-dimensional transformation matrices that operate on homogenous coordinates are 3×3, there are different classes of transformations that preserve different amounts of geometry; the less a transformation preserves, the more degrees-of-freedom it is said to have.

The simplest transformation is a translation; i.e., moving a point in the x direction by a value of t_1 and in the y direction by a value of t_2. This can be accomplished using the following operation:

$$\mathbf{x}' = \begin{bmatrix} 1 & 0 & t_1 \\ 0 & 1 & t_2 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}$$

This transformation has two degrees of freedom. This is obvious by inspection, as there are only two parameters that affect the values the matrix takes on.

A rotation transform preserves all distances and angles between points, while shifting and rotating all points simultaneously. If you wish to rotate by θ and translate points, this can be accomplished with:

$$\mathbf{x}' = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & t_x \\ \sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}$$

This transformation has three degrees of freedom; two for the x and y translation, and one for the θ rotation.

This transformation is commonly grouped as

$$\mathbf{x}' = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{x}$$

where $\mathbf{R}$ represents the 2×2 rotation matrix and $\mathbf{t}$ is a length-2 column vector representing the translation coordinates.

An affine transformation preserves parallel lines. It can be accomplished with:

$$\mathbf{x}' = \begin{bmatrix} a_{00} & a_{01} & t_x \\ a_{10} & a_{11} & t_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{x}$$

Again, this is commonly grouped as

$$\mathbf{x}' = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{x}$$

Although this transformation looks similar to the rotation transformation, it is important to understand that $\mathbf{R}$ is constrained by θ. In other words, it *must* take on a value that can be written in the form

$$\mathbf{R} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

On the other hand, $\mathbf{A}$ can take on arbitrary values, so the affine transformation has six degrees-of-freedom.

The projective transformation preserves lines; everything else can be manipulated. This type of transformation has the fewest constraints, as it can be any arbitrary 3×3

matrix:

$$\mathbf{x}' = \begin{bmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{bmatrix} \mathbf{x}$$

This may also be expressed as

$$\mathbf{x}' = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^\intercal & v \end{bmatrix} \mathbf{x}$$

Although this matrix has 9 elements, only their ratios affect the transformation, so the transformation has 8 degrees-of-freedom. While the transformation matrix can be scaled so $v = 1$ (and this is common), it is entirely possible that v can take on a value of 0, which would prevent the transformation matrix from being normalized by v. A visual representation of these transforms can be seen in Figure 28.

Points and Transforms in 3D

Homogenous coordinates and transformations of these coordinates can be extended to 3D relatively easily. A three-dimensional point at coordinate (x, y, z) can be represented as homogenous coordinate vector $\mathbf{X} = [x/w, y/w, z/w, w]^\intercal$. Again, these coordinates are equivalent for all $w \in \mathbb{R}$ and fixed x, y, and z values. A plane is uniquely defined by the join of three points or the join of a line.

Camera Pinhole Model

Now that we have established a basis for projective geometry, we can develop the model of a camera. We ignore the complexities and irregularities involved in real-

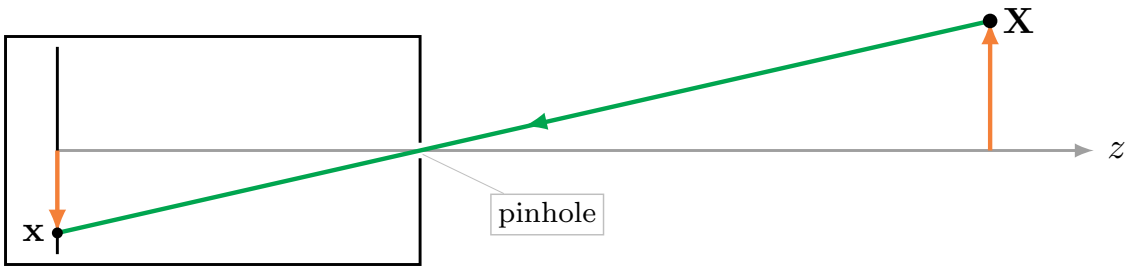

Figure 29: Pinhole camera model

world optical systems and, instead, model the camera using the *pinhole model* – where the world is observed through an infinitely small pinhole. This model forces every point in space to be transferred to the image plane through a single ray of light, which causes everything to appear in focus, with no optical deformities. This model is shown in Fig 29. Here, a point in space passes through the pinhole, and projects onto the image plane. This projection maps 3D world points into 2D points on the image plane. This mapping can be written as

$$\vec{x} = \mathbf{P}\vec{X},$$

where $\vec{X}$ is a point in 3D space (in homogenous coordinates), and $\vec{x}$ is a 2D homogenous vector appearing on the image plane. With these constraints on dimensions, $\mathbf{P}$ is a 3×4 matrix called the *camera projection matrix*.

We can reformulate the pinhole model by placing the pinhole at the origin, and placing the image plane some distance f along the z-axis. This is shown in Figure 30. Although this arrangement makes no sense, from an optical standpoint, it is functionally equivalent to the model presented in Figure 29; this model simply eases the derivation. By inspection,

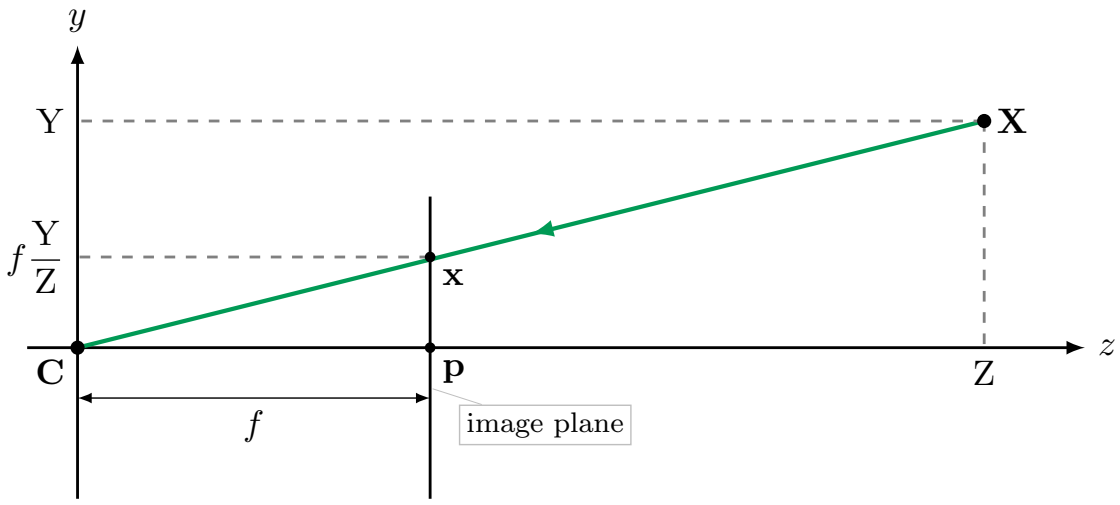

Figure 30: Central projection pinhole model

$$\mathbf{P}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} \sim \begin{bmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \\ 1 \end{bmatrix}$$

where

$$\mathbf{P} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix}.$$

Cameras, being pixel-based, are normally accessed from the top left of the image, not the center. So, we must add an offset to the pixel's values, according to

$$\left[X, Y, Z\right]^{\intercal} \mapsto \left[fX/Z + p_x, fY/Z + p_y\right]^{\intercal}.$$

This can be incorporated into the projection matrix as

$$\begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

We will extend this projection matrix next, but first, it is important to note that we often think of this projection matrix as a composition of a camera calibration matrix, $\mathbf{K}$, with other vectors and matrices. This is useful when trying to solve for the values in these matrices; the user of the system knows that as long as the camera is not altered, the values in the camera calibration matrix are constant. Thus, we write

$$\mathbf{P} = \mathbf{K} \left[\mathbf{I}_3 \,|\, \vec{0} \right].$$

At this point, we have assumed the camera is positioned at the origin looking down the z-axis; however, it is often useful to operate relative to a different – usually *world* – coordinate system, especially when multiple cameras comprise the system. This is accomplished by rotating and translating the points using a rotation matrix, $\mathbf{R}$, and translation vector, $\vec{t}$. This can be integrated into our camera projection matrix as

$$\begin{aligned} \mathbf{P} &= \mathbf{K} \left[\mathbf{R} \,|\, \vec{t} \right] \\ &= \mathbf{KR} \left[\mathbf{I} \,|\, -\vec{C} \right] \end{aligned}$$

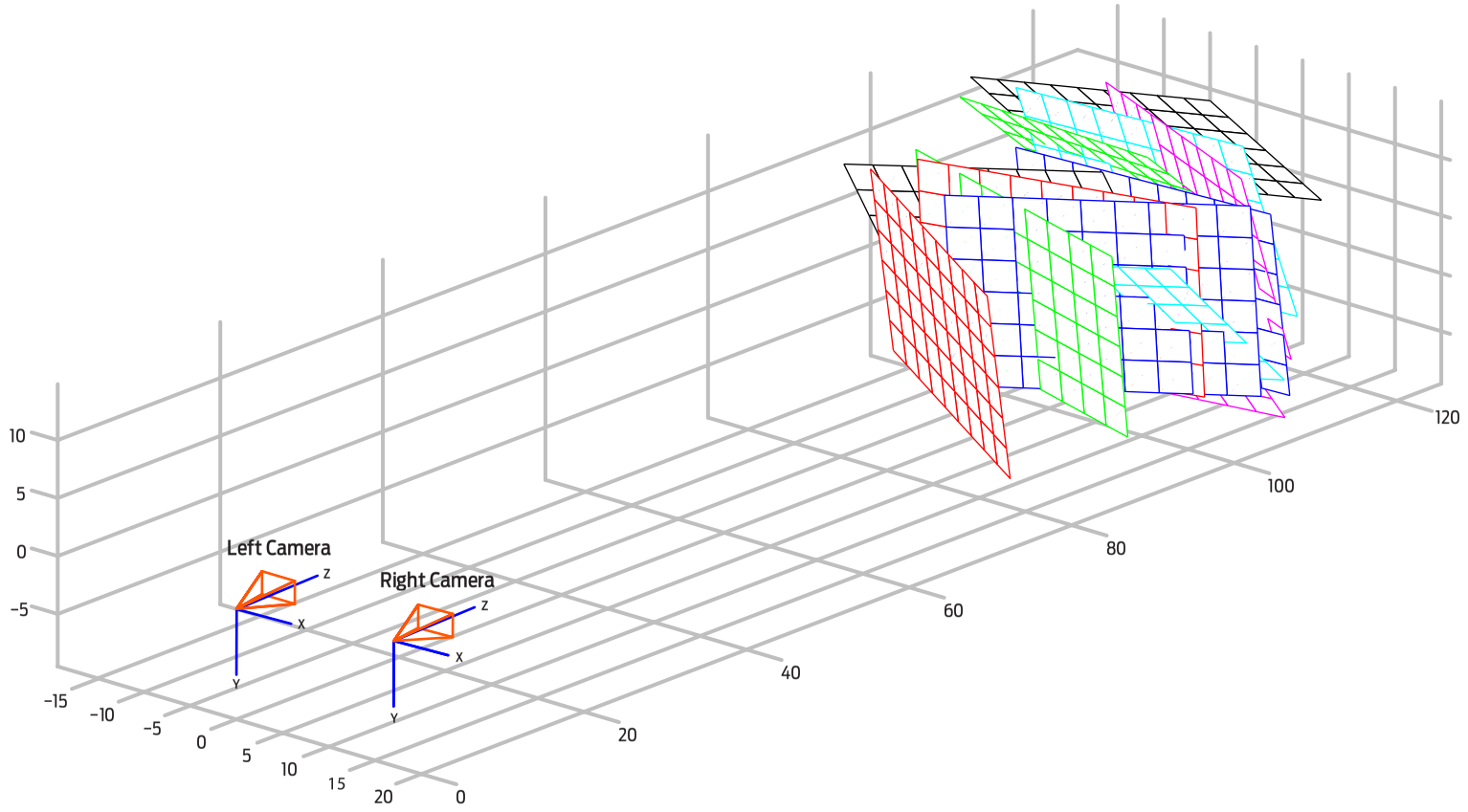

Figure 31: The CalTech Camera Calibration Toolbox was used to compute the calibration matrices for the two imagers, along with the translation vector and rotation matrix that relates the two cameras together. Once these matrices are found, a 3D recreation of the calibration environment can be created. This figure shows the calculated locations of each calibration image.

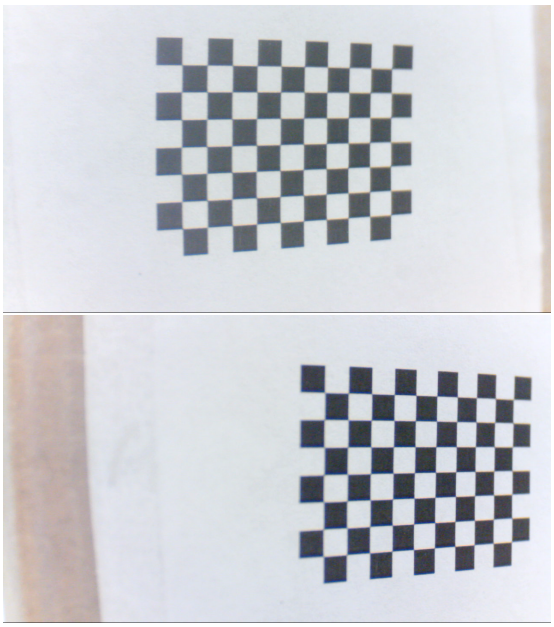

Figure 32: Example images from the left and right imagers, respectively, captured for camera calibration purposes. A total of 12 images like this were captured – all showing the checkered pattern in various orientations.

Camera Calibration

The camera projection matrix depends on **K**, the camera calibration matrix, which can be directly computed using known parameters of the cameras (focal length, pixel size, offset, image array dimensions, etc), but because of manufacturing tolerances (and resulting defects), it is more accurate to create a camera calibration matrix by calibrating the actual camera using test images. By capturing a set of images containing an object of known geometry (say, a checkered board), the camera calibration matrix can be solved for using a direct linear transform (DLT) matrix solving algorithm. Existing software, like the Camera Calibration Toolbox for MATLAB [9], are useful for this task. Figure 31 illustrates a 3D model of the scene that was derived by the Camera Calibration Toolbox by capturing a series of pictures (from both left and right image sensors in the camera) of a checkered board pattern. These sets of

images (one of them is depicted in Figure 32) are tagged as a “left” or “right” image and fed into the Camera Calibration Toolbox for processing.

The camera calibration matrices for the left and right imager are, respectively:

$$\mathbf{K}_{\text{left}} = \begin{bmatrix} 2041.69 & 0 & 792.43 \\ 0 & 2051.30 & 440.72 \\ 0 & 0 & 1 \end{bmatrix},$$

and

$$\mathbf{K}_{\text{right}} = \begin{bmatrix} 2017.84 & 0 & 692.44 \\ 0 & 2019.75 & 407.38 \\ 0 & 0 & 1 \end{bmatrix}.$$

The calibration toolbox also calculates extrinsic properties; i.e., how the cameras relate to each other. The position of the right imager with respect to the left imager is defined by the translation vector

$$\vec{t} = \begin{bmatrix} -14.97 \\ 0.10 \\ -1.67 \end{bmatrix},$$

and the rotation vector

$$\vec{r} = \begin{bmatrix} 0.00566 \\ 0.05233 \\ 0.00003 \end{bmatrix}.$$

Interpreting this, it can be seen that the two image sensors in the proposed camera have measurable differences in terms of optics, as well as flaws in alignment. The two imagers have close – but not equal – focal lengths. The left imager has a focal length

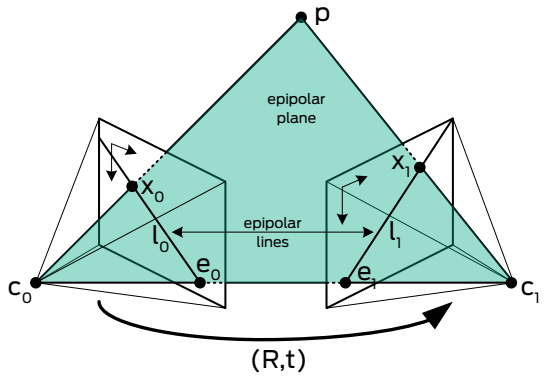

Figure 33: An epipolar arrangement is created by forming a triangle from the two camera centers and a point in space.

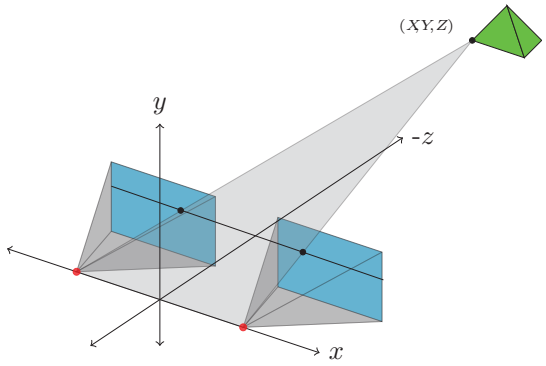

Figure 34: Ideal epipolar scenario for stereo viewing and processing.

of approximately 2046.5 pixels (obtained by averaging the f_x and f_y focal lengths together), and the right imager has a focal length of approximately 2018.8 pixels (again, obtained by averaging the x and y focal lengths together). Although the sensors on the PCB were designed to be exactly 15mm apart from each other, the camera calibration indicates the actual optical axis difference is 14.97 mm. The right imager is 0.1 mm higher than the left imager, and 1.67mm closer to the subject.

Stereo Rectification

For viewing and image processing, we would like to remove as many alignment defects as possible. A stereo camera that is not properly aligned can cause viewer fatigue and perceived blurriness. And, as we will see, there are computational advantages to working with *rectified* images – i.e., a set of images warped to mimic the ideal scene depicted in Figure 34. Rectified images have the property that every point in 3D space projects to the same vertical line number in both images; in other words, every point found in the left image appears at the same height (line number) in the right image, and line numbers remain parallel to the x-axis. The process of transforming these images to enforce this constraint is called *rectification*. The terminology used in this setup refers to the epipolar arrangement seen in Figure 33 ; in it, a point $\vec{p}$ is projected onto two cameras C_0 and C_1 – where C_1's camera center is positioned with rotation $\mathbf{R}$ and translation $\vec{t}$ with respect to C_0's. Then $\vec{p}$ (located along the ray direction vector $\hat{x}_0$ from C_0's perspective) projects onto C_0 at image coordinates $\vec{x}_0$ via $\mathbf{K_0}$– the camera calibration matrix for C_0. At the same time, $\vec{p}$ (located along the ray direction vector $\hat{x}_1$ from C_1's perspective) projects onto C_1 at image coordinates $\vec{x}_1$ via $\mathbf{K_1}$. The 3×3 cross-product between $\vec{t}$ and $\mathbf{R}$ is called the *essential matrix,* $\mathbf{E}$. There is a relationship between the local ray direction vectors $\hat{x}_0$ and $\hat{x}_1$, called the *epipolar constraint*, as

$$\hat{x}_1^\intercal \mathbf{E} \hat{x}_0 = 0,$$

which is such that $\mathbf{E}$ maps any point $\hat{x}_0$ in C_0 to a line $\vec{l_1} = \mathbf{E}\hat{x}_0$ in the other camera, C_1. All of these *epipolar lines* must pass through the epipolar point (epipole), $\vec{e_1}$. The same is true for point $\hat{x}_1$, line $\vec{l_0}$, and epipole $\vec{e}_0$.

This is advantageous for doing any sort of triangulation to recover the three-dimensional coordinate of a point in space; $\mathbf{E}$ will highly constrain the search problem

to a single line – given a point in one image, and $\mathbf{E}$, we know the line in the other image along which to search for a matching point. Keep in mind, though, that $\hat{x}_0$ and $\hat{x}_1$ are not image coordinates – they are ray direction vectors. Since we don't have direct access to ray direction vectors (the camera only has access to pixels – $\vec{x}_0$ and $\vec{x}_1$), we can obtain them from image coordinates as

$$\begin{aligned} \hat{x}_0 &= \mathbf{K_0^{-1}}\vec{x}_0 \\ \hat{x}_1 &= \mathbf{K_1^{-1}}\vec{x}_1, \end{aligned}$$

where $\mathbf{K}_0$ and $\mathbf{K_1}$ are the two camera calibration matrices for the cameras C_0 and C_1.

To find $\mathbf{E}$, typically the eight-point algorithm [41] is used, which takes eight or more $\hat{x}_0$ and $\hat{x}_1$ correspondences as inputs, formulates a homogenous linear equation, solves it, and then enforces internal constraints of the essential matrix by decomposing the solution using singular-value decomposition, adjusting $\mathbf{S}'$ so that two of its singular values are equal and nonzero and the other is zero, and rebuilding it.

Plugging these into the previous expression for the epipolar constraint, we see that

$$\vec{x}_1^{\intercal}\left(\left(\mathbf{K_1^{-1}}\right)^{\intercal}\mathbf{E}\left(\mathbf{K_0^{-1}}\right)\right)\vec{x}_0 = 0.$$

These middle terms – which are constant for specific cameras mounted in a fixed orientation relative to each other – comprise $\mathbf{F}$, the *fundamental matrix*

$$\mathbf{F} = \left(\mathbf{K_1^{-1}}\right)^{\intercal}\mathbf{E}\left(\mathbf{K_0^{-1}}\right),$$

which can be found from the epipolar constraint on $\mathbf{F}$,

$$\vec{x}_1^{\intercal}\mathbf{F}\vec{x}_0 = 0.$$

Note that we are now working with real pixel coordinates $\vec{x}_0$ and $\vec{x}_1$ in the image, and no longer with ray direction vectors. This means we can find $\mathbf{F}$ with nothing other than a set of pixel correspondences in the image – we don't need to know anything about the cameras or how they're aligned with respect to each other. While the same method to find $\mathbf{E}$ can be used to find $\mathbf{F}$, the large variation in image coordinates (on the order of thousands, with modern digital cameras) compared with the small homogenous third coordinate causes the inputs to the algorithm to be ill-conditioned. Instead, these inputs are typically normalized with a transform, processed using the eight-point algorithm into a resulting $\mathbf{F}$, which is then inversely transformed to its correct value. This approach is known as the normalized eight-point algorithm [24].

To rectify the images, a projective transform is created that warps the image so that the epipolar lines in each image are parallel and horizontal, and that corresponding epipolar lines are at the same scan line; i.e., that the cameras are arranged as seen in 34. This is done by moving the epipole to be in-line with the camera center, perpendicular to both the optical axis and the vertical direction of the image, but located at infinity. Without any additional constraints, this turns out to be an under-defined problem – there are many transforms that will do this; and in the process, the image may get warped, stretched, or skewed considerably. An entire area of research attempts to constrain the transform to make the resulting images look natural or maintain properties that make feature detection and matching reliable [42, 45, 34, 51, 22].

Figure 35: Example of a high-quality disparity map created using a laser rangefinder. The left and right views of a scene (top) are stacked on top of each other (bottom left). The horizontal distance between features increases as objects get closer to the camera. This distance is expressed in grayscale in the bottom right image, where darker pixels indicate less disparity, and brighter pixels indicate more disparity.

Stereo Matching

Once the images are rectified, dense stereo reconstruction techniques, coupled with camera calibration, can recover the actual 3D coordinates which correspond to every pixel in the images. For every pixel $p_{x,y}$ in the left image, the matching pixel $p'_{x',y'}$ in the right image is found. Because the images are rectified, the pixels are on the same scan line – $y = y'$. This reduces the search space considerably, resulting in better performance and more accurate matching. Once this p' pixel is found, the disparity,

$d = x - x'$, is returned for $p_{x,y}$. Once this process is completed, a disparity value exists for each pixel in the image. The resulting disparity information, itself, is often represented as an image as well. Figure 35 illustrates an example disparity map that corresponds to a scene. The top two images are the left and right views of a scene taken using a stereo camera. The bottom left view is of these images stacked on top of each other and colorized to show the offset of content in the scene. The final image shows the depth map of the scene.

Stereo Reconstruction

From the camera projection calculations worked out previously, the original (X, Y, Z) coordinates of the pixel $p_{x,y}$ with disparity d can be recovered as

$$\begin{aligned} X &= \frac{-(x - p_x)\,Z}{f} \\ Y &= \frac{-(y - p_y)\,Z}{f} \\ Z &= \frac{-fb}{d}, \end{aligned}$$

where (x, y) is the pixel coordinate in the left image, f is the focal length of the left camera, and p_x, p_y are the principle point offsets of the left camera. The distance between the right and left cameras is b, and it is assumed that the left camera's center is at the origin looking down the negative z axis. As can be seen, while rectification and stereo matching does not require a calibrated camera setup, recovering the 3D coordinates of pixels from their location and disparity value does. This is why obtaining accurate camera calibration data is so important.

Matching Techniques

Finding the p' correspondences to p is not a trivial task. While it may seem like p' should simply be selected so that it has the same value as p, in practice, many factors make this unreliable or impossible. First, p's value is rarely unique; pixels are not real-valued phenomena: because of digital representation, there are a finite number of values a pixel can take – therefore, there may be several p' that have the same value as p. In images with low texture, there may be hundreds – or thousands – of pixels that have exactly the same value. Second, due to noise and changes in camera parameters (exposure, white balance, focus, lossy compression, et cetera) p and p' likely do not have the same value. Because of this, window-based searching is usually used that compares a window of pixels around p with a window of pixels around a p' candidate; The p'candidate window is moved across the horizontal scan line to cover all possible candidates, and the best p' candidate is selected. Extensive research has focused on how these windows of pixels are compared (the distance metric used), and, more importantly, what makes a p' a "good" match. The combination of these two parameters is called the *cost function*. A simple cost function might be

$$C = \left(I\left(x,\, y\right) - I'\left(x',\, y\right)\right)^2,$$

where $I\left(\right)$ and $I'\left(\right)$ are functions which return the intensity of the pixel at the given coordinates in the left and right image, respectively. This cost function performs poorly since it is only trying to match to pixels with the closest intensity without any regard for location of the pixel. More practical cost functions penalize disparities that are significantly different from the disparities of neighboring pixels (smoothness constraint), or for disparities that do not meet the uniqueness constraint (i.e., if p' matches to p, does the same p match to the same $, p'$?).

A comprehensive list of benchmark data from many different stereo matching algorithms is available at the Middlebury Stereo Evaluation pages [58].

in vivo Porcine Testing

To evaluate the camera's suitability for *in vivo* insertion, the camera was tested in a living porcine model. This experiment took place as part of a larger evaluation of a surgical robot platform developed by Dr. Shane Farritor's mechanical engineering lab. A fiberoptic xenon light source was used to illuminate the interior during video capture. Following IRB approval, the experiment took place at the University of Nebraska Medical Center on December 21st, 2011, under the supervision of Dr. Dmitry Oleynikov and Dr. Share Farritor. After a surgeon created a large incision in the abdomen, the camera was inserted into the test subject along with a xenon fiberoptic light source. The camera was attached to a laptop, and approximately 5 minutes of video was recorded during the experiment. The experiment consisted of holding the camera stationary for several minutes, as well as moving it around the surgical environment. After the experiment, the video footage was temporally aligned, so that left and right frames corresponded to the same instant in time.

Video Quality

Figure 36 shows a set of left and right frames extracted from the video captured during the evaluation. As can be seen, the camera has shallow depth-of-field, and the fixed-focus lens is not set to focus the correct distance. The only way to adjust focus is to remove the camera, re-focus it manually, and re-insert. Due to time constraints, this was not done.

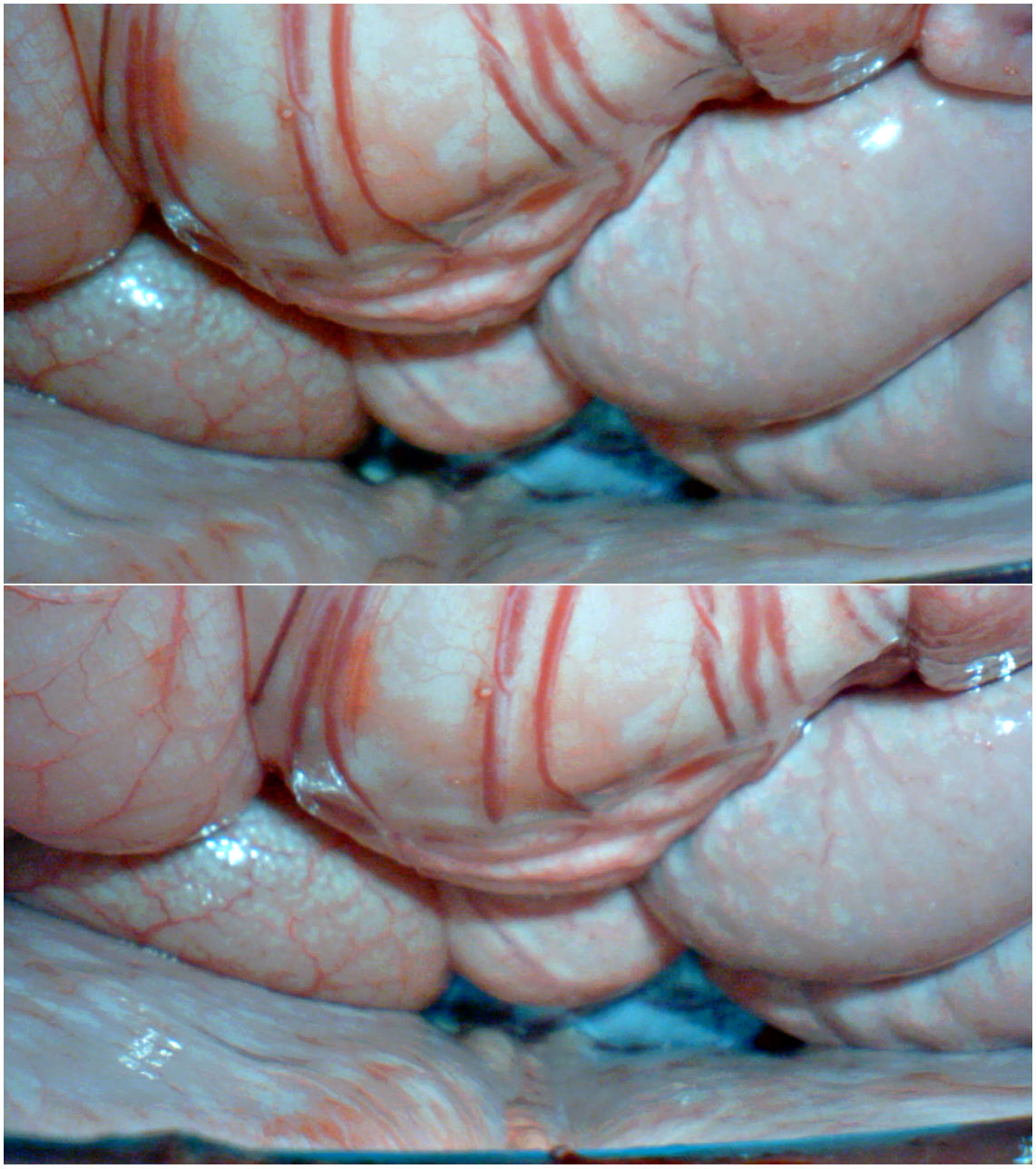

Figure 36: *in vivo* stereo images captured from inside a porcine model during an experiment at UNMC.

Stereo Matching

To evaluate the stereo performance of the camera, two frames of the video footage obtained during the porcine experiment were reconstructed into a disparity (depth) map using stereo reconstruction. As previously mentioned, dense stereo reconstruction requires images to be rectified, which is usually accomplished using a method similar to the CalTech Camera Calibration toolbox mentioned previously. Unfortunately, the camera used during this particular experiment started malfunctioning before a full camera calibration could be performed on it (the calibration results presented in the previous section are for a different unit than the one used in this experiment).

To this end, to perform 3D reconstruction on these images, MATLAB's built-in rectification functions – which work without knowing camera parameters – were used instead.

This method works by finding sets of corresponding features in a stereo pair of images and then warping the images to enforce the epipolar constraint: that a feature in the left image will appear on the same scan line as the same feature in the right image. Then, a stereo matching algorithm attempts to find the disparity between corresponding content in the left and right images. The main downside to this approach is that the resulting disparity map can only be expressed in terms of pixels – not real-world coordinates.

Feature Matching

First, two matching frames were extracted from the video sequences recorded during the experiment. The images were converted to grayscale. SURF feature detection was performed on each image, which resulted in dozens of features associated with each image (Figure 37). Once features were found, they were matched using a sum

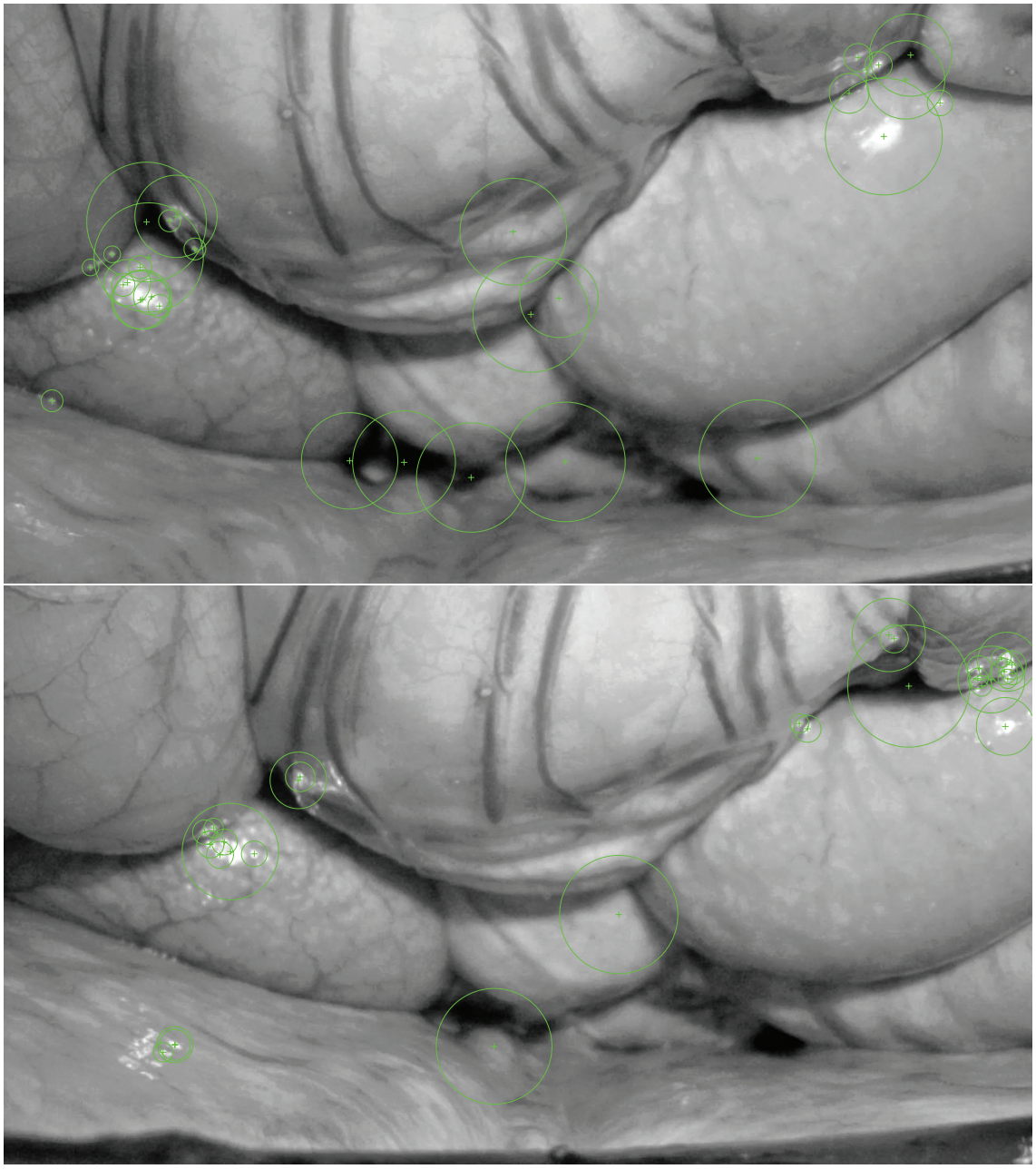

Figure 37: Top 30 SURF features are highlighted in the left and right images, respectively.

of absolute distances (SAD) metric, which maps each set of feature vectors to be compared to a distance (error) value:

$$d_{\mathrm{SAD}} : (x,\, y) \mapsto \|x - y\|_1 = \sum_i |x_i - y_i|$$

These matches are shown in Figure 38.

With any feature-matching technique, there are bound to be false matches – two uncorresponding features in the left and right image may happen to "look similar" to the SURF algorithm, and therefore, match up. Unfortunately, image warping using homographies is not robust to errors at all; one false feature-match will drastically change the homography matrix which will in turn warp the image incorrectly. Therefore, we must ensure no false matches are propagated to the next step. While increasing the SAD threshold will remove some of these false matches, it will also eliminate many correct matches; because of image noise and camera differences, even perfectly matched features can have vectors that differ.

Fundamental Matrix Estimation and RANSAC

A more robust way of removing incorrect matches is to use random sample consensus (RANSAC) which, in general, is used to select parameters of a model while rejecting outliers. It works by iteratively picking a small sample of possibly-correct parameters, creating a model from those parameters, and evaluating it against other parameters. Figure 39 illustrates an example of this process. In the left, example data has a strong trend line in addition to many outliers. Least-squares regression cannot handle outliers, and the resulting trend line (purple) is incorrect. With RANSAC, a small sample of these points is selected, least-squares regression is performed, and then all the points are evaluated against the regression. If the points that were randomly

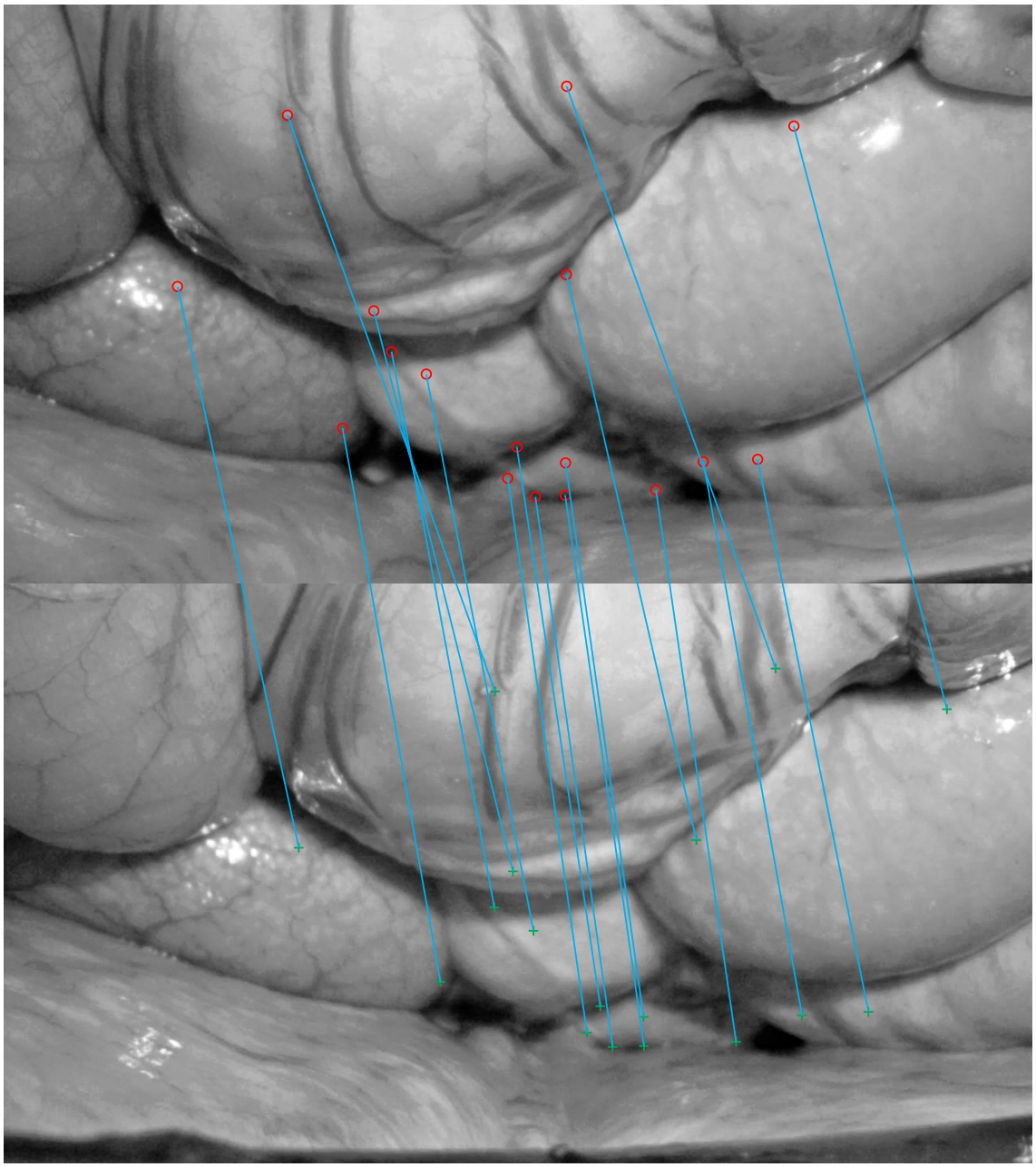

Figure 38: Matches found in the two images, overlayed on top of each other with cyan-magenta coloring.

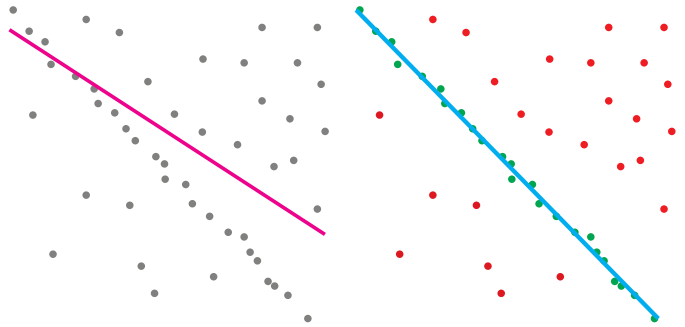

Figure 39: Example data with a strong trend line along with many outliers. Least-squares regression alone (magenta) cannot identify the correct trend line with the outliers (left), while RANSAC identifies and removes the outliers (in red) while performing least-squares regression (cyan line) on the inliers (green).

selected lie along the trend line, the number of inliers will be high, and the algorithm will keep the sample set and continue to add more points, iteratively evaluating the model at each step. If the points that were randomly selected include any outliers, the least-squares regression will not fit the data well, so when the rest of the points are checked against the model, the percentage of points that fit the line would be low; the model is discarded and another sampling of points is selected.

For this application, RANSAC can be used to remove false matches while estimating $\mathbf{F}$, the fundamental matrix. MATLAB's estimateFundamentalMatrix() function does this by randomly selecting a set of eight ($\tilde{\mathbf{x}}_\mathbf{i}$, $\tilde{\mathbf{y}}_\mathbf{i}$) corresponding features, calculating the fundamental matrix with those features using the normalized eight-point algorithm, and then evaluating how well the resulting fundamental matrix fits all of the feature matches by calculating $\tilde{\mathbf{x}}_\mathbf{i}^\intercal \mathbf{F} \tilde{\mathbf{y}}_\mathbf{i}$(which should be close to 0 for a good match) for all the feature matches in the image set.

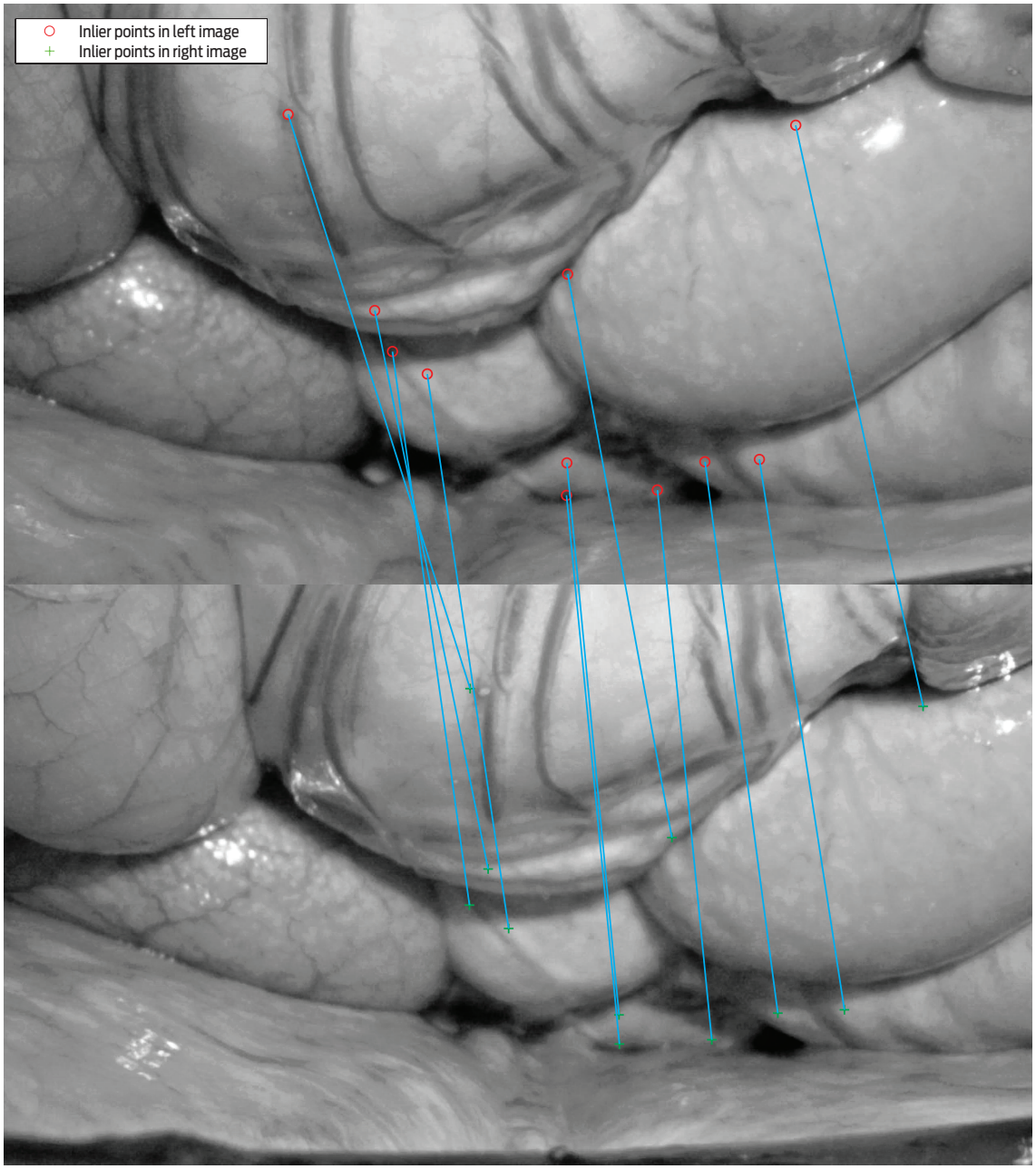

Figure 40: RANSAC

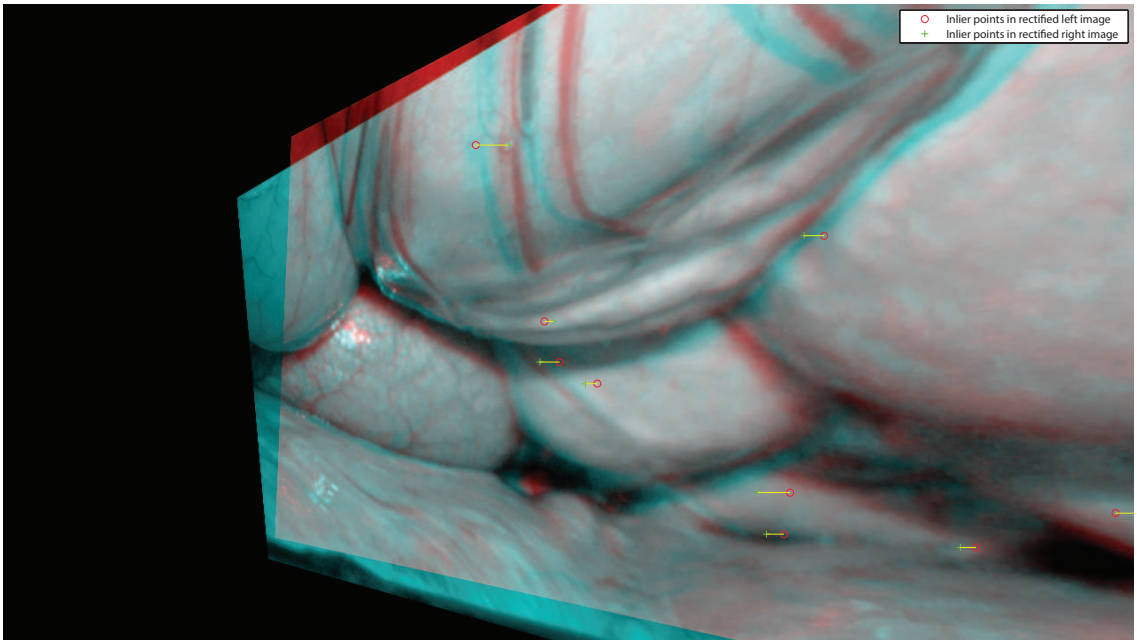

Figure 41: Image warping the left and right images so that features in the left image appear on the same scan line as the corresponding features in the right image.

Rectification Warping

Once the fundamental matrix was found, MATLAB's estimateUncalibratedRectification() function was used to generate transform matrices that warp the images in a way that matching features appear on the same scan line (i.e., lines drawn between matched features are horizontal). The two rectified images, when laid on top of each other, are shown in Figure 41. Before stereo matching, these images are cropped, as shown in Figure 42.

Stereo Matching

Once the two images were rectified, a stereo matching algorithm [2] was used to generate a disparity map, shown in Figure 43. Because the cameras were uncalibrated, true stereo reconstruction could not be performed. From visual inspection, the re-

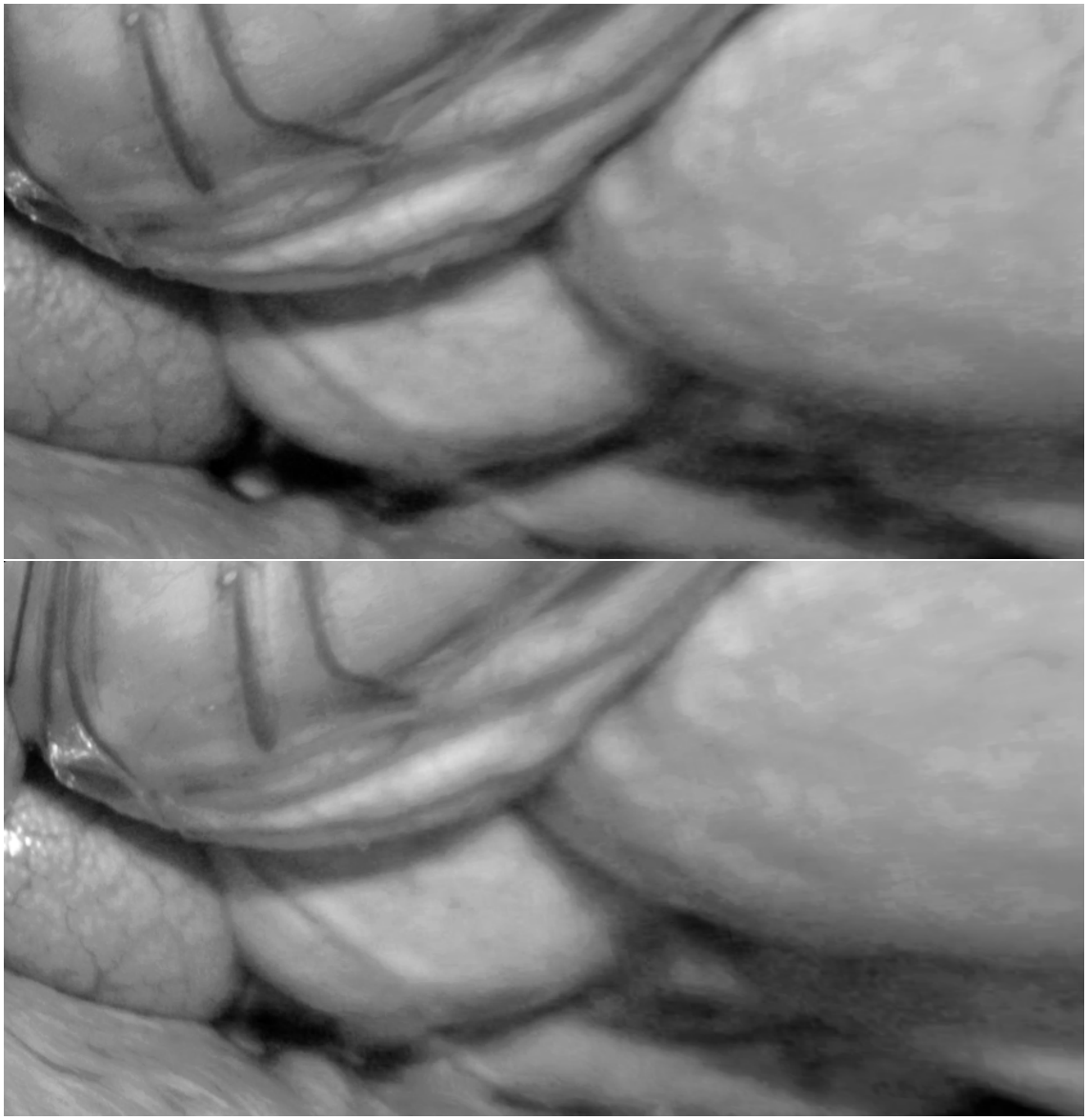

Figure 42: Warped images after cropping.

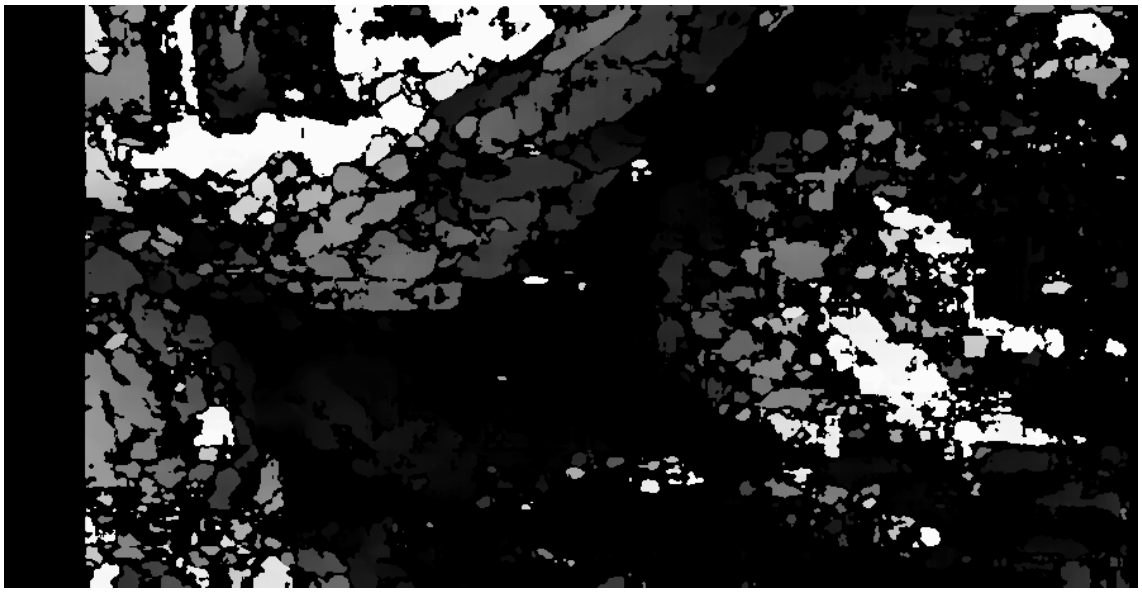

Figure 43: Resulting disparity map generated from rectified images using stereo matching.

constructed disparity map has many errors. This was largely caused by frame-level synchronization issues; i.e., even though the video footage was aligned in time overall, the frame-to-frame alignment was very poor. While this is not problematic in static scenes, the live porcine test subject exhibited a great amount of organ movement throughout the experiment. Further problems encountered include focusing issues, which caused the already limited amount of texture to blur considerably into smooth gradations. As discussed before, window-based stereo matching algorithms rely on a highly textured scene to be able to correctly identify matching windows in images.

Conclusion

While the previous chapter explored the design of the camera and evaluated its technical abilities for general-purpose imaging, this chapter focused on applying it to *in vivo* surgical robotics applications, which have a host of challenges specific to their application. There are several key challenges that need to be overcome before this

Figure 44: Stereo reconstruction of a surgical stitch trainer using the proposed stereo videoscopy camera.

platform is suitable for use in its intended function, including better frame synchronization and focus control, as well as redesigning the enclosure to provide better contrast in the scene.

There are other problems peculiar to this particular application: internal tissue is highly reflective (due to both tissue composition and moisture in the environment), which causes specular highlights (bright dots on the surfaces of objects caused by reflected light); also, many organs and other internal tissues have smooth, consistent color with no patterning or textures at all; it is impossible for current stereo-matching algorithms to perform effectively, since windows of pixels at almost any offset will look the same. Pure black areas in the disparity images indicate no displacement at all; this is usually a tell-take sign that those areas in the image have low texture, and the algorithm is simply picking the closest window (corresponding to a disparity value of 0).

When all of these mentioned issues with the proposed device are controlled for (by

using a static scene with no movement) – and the scene under test has low reflectivity and sufficient texture – the stereo imager is capable of performing reasonably accurate reconstructions. Figure 44 shows a synthetic view created by reconstructing a surgical stitch trainer, obtaining a disparity map, then re-projecting the disparity map from a different angle and assigning pixel color values obtained from the original image. There are many fewer discontinuities, and the reprojected scene mimics the original environment quite well.

Conclusion

In this thesis, a stereoscopic camera for *in vivo* surgical robotics has been presented. This device was developed to address the need for a compact, low-cost, high definition camera capable of providing depth perception in surgical situations. The device uses two 720p high definition image sensors and USB-enabled microcontrollers mounted on a single printed circuit board, which allows for plug-and-play operation.

The implemented dual-camera system has been tested using a resolution chart to allow for objective comparison to an existing standard-definition analog stereoscopic camera system. Significant improvements in both local contrast and image resolving capabilities were demonstrated using the proposed device. However, the implemented device compares poorly to existing commercial webcams, such as the Logitech C905 – a 720p HD webcam that uses a similarly-sized CMOS sensor and control logic. While the Logitech C905 is larger in size than the proposed device, it is because the Logitech's design offers several features the proposed solution lacks:

- A push button for shutter release
- On-board microphone
- Auto-focus lenses (with associated transistor drivers for the AF voice coil)
- Status LEDs

- Detachable USB cables
- ESD protection

The Logitech camera is also relatively inexpensive when compared to the proposed device because it uses lower-cost, bulkier components, and lower-quality PCBs with looser tolerances. The Logitech board is designed for low-cost assembly as well, by using low-density landing patterns that are less prone to solder problems during assembly. Because of this, it is easy to see that the Logitech C905 webcam could be miniaturized to the same size as the implemented system without loss of image quality; this would be superior in performance to the implemented system.

The proposed device, while smaller than other off-the-shelf USB webcams, is still much too large to be useful in a surgical environment. The device will not be able to replace existing endoscopic systems until it is at least as small as these systems; da Vinci uses 12mm endoscopes as a standard instrument, which is therefore a suitable rule of thumb for useful size. From this, it can be seen the proposed system, which is more than 580 mm^2, must shrink by at least a factor of four before it is similar in size to current endoscopic tools. Having said this, the proposed device fits on current-generation *in vivo* surgical robotics platforms without exceeding size requirements.

Once integrated with a surgical robot, the dual-camera device can be used for visual servoing, object identification, and vision-driven semi-autonomous surgery [68]. The device can also be used to capture 3D high definition video of the surgical environment for medical education and surgical training.

Future Work

While the presented platform achieves some preliminary goals satisfactorily, the system can be improved significantly. From a performance perspective, the camera suffers from a fixed-focus design, sporadic frame dropping, and exposure issues. Under the hood, the major problem with the current design is the highly proprietary nature of the EETI EM2780 USB camera controller IC involved. Without firmware source code, the camera controller ICs serve as a virtual "black box" – changes to the firmware to fix the dropped-frame issue, or correct the overexposure problem, or to add support for a different image sensor or an auto-focus drive coil would require a complete re-write of the firmware – which would be extremely difficult and time-consuming, given the limited documentation available for the EM2780 IC.

While additional components can be integrated into the current hardware, like an LED light source, a voice-coil-powered auto-focus lens assembly to fix the focusing issues mentioned, or a pan-tilt mechanism, these extra features would require additional, external control – since there is no way to hook into the camera controller without modifying the firmware. This would necessarily increase the size of the design.

Before any additional development can occur on this project, the platform must be re-engineered to be non-proprietary. As mentioned previously, a core challenge with this is finding a suitable controller; it must have high-speed interfacing, while also being extremely small. These controllers simply do not exist, outside of proprietary ASICs. So, there are two main ways of solving this problem:

- Continue using an applications-specific controller, such as the EETI one currently used, but re-create the firmware from scratch so that it may be modified and enhanced to support future functionality.

- Relax the size constraint, and migrate to an entirely new controller – probably either using an ARM processor or an FPGA.

The first solution proposed will require writing a USB stack from scratch, or trying to port an existing USB stack to the particular processor. Neither task is trivial. Even once this is complete, setting up the DMA transfer between the parallel bus and the USB transceiver – necessary for streaming high-definition video at full frame rates – will be complex, given then limited documentation of the processor. Also, the current platform, which uses an 8-bit microcontroller, provides a serious limit on possible resolution and frame rate combinations.

Moving to a new platform would allow the selection of a faster controller that comes with built-in libraries to handle the sorts of high-speed interfacing required.

Reducing Camera Size

Another mentioned problem with the camera is the size of the platform. While suitable for current-generation *in vivo* robots, these platforms are quickly shrinking – and for the platform to be viable, the camera will have to shrink, too. One way to shrink the camera (and relax the size constraint of the controller at the same time) is to move the controller off-board. The current design for the camera contains the USB interface controller in addition to the CMOS sensor; however, an alternative solution would mount only the CMOS sensor (with minimal support circuitry) on the *in vivo* robot. This camera would be connected to an off-board camera control module mounted outside the surgical environment.

While this design strategy was considered for the current design, the camera's parallel bus is highly constrictive; routing a high-frequency parallel bus more than a couple of centimeters requires treating the wires as a transmission line: the bus would

have to be driven with line drivers capable of controlled output impedance, the bus itself would have to use cabling with guaranteed impedance, and the system would require tuned termination on the receiver side, to control ringing and signal loss.

One other option to separate the image sensor from camera controller is to switch from a parallel to serial interface. Most modern image sensors have a MIPI interface – a high-speed differential serial interface that uses one lane for data and one lane for clocking. This interface works reliably over relatively long runs – up to 50cm in one test [37]. The MIPI bus would still have to be interfaced with the computer somehow, but this step could be performed outside of the surgical site.

Other Interface Options

For higher-quality imaging (uncompressed video, 1080p HD, etc), a different interface could be selected. This is difficult to do with the size constraints imposed for this project, however, by relaxing these constraints slightly, much better-quality imaging can be achieved.

USB 3.0

USB 3.0 offers a 5.0 Gbps communications link that is more than capable of carrying two 720p streams at 60 fps, using YUV 4:2:2 subsampling. This would offer tremendous image quality. Cypress Semiconductor currently manufactures the FX-3 USB controller, which integrates an ARM9 microprocessor with a USB 3.0 transceiver, as well as a high-speed reconfigurable parallel interface. Because the platform is designed for traditional general-purpose development, the platform is much more open and well-documented, which would eliminate the hassles encountered with the proprietary camera controllers used in this project.

HD-SDI

As mentioned previously, another interface option is the High Definition Serial Digital Interface (HD-SDI), which is the ubiquitous digital video interface used in the broadcast television and film industry. No current microcontrollers on the market implement an HD-SDI interface; thus an HD-SDI solution would have to be developed on an FPGA that has a high-speed transceiver module built-in. The Altera Cyclone IV GX is the smallest FPGA that supports this functionality, however, at 23×23mm, it is probably too large to integrate on an *in vivo* robot – though if a long-run MIPI interface was used, the FPGA circuitry could reside out of the surgical environment altogether, eliminating the stringent size requirements.

While HD-SDI video can be captured by a PC for image processing, it can also be displayed in real-time with industry-standard monitors, switches, and multiplexers. This eliminates possible fatal situations that could happen if a computer responsible for rendering live view to a monitor were to suddenly crash or freeze.

Conclusion

Videoscopy for *in vivo* surgical robotics applications has been explored. A novel, USB-based stereo video camera has been proposed, which meets aggressive cost constraints while maintaining a level of miniaturization suitable for current-generation *in vivo* surgical robotics platforms. The proposed camera was evaluated with standard benchmarking techniques, including optical resolving ability and noise analysis; because of the application of the platform to stereo computer vision tasks, the design was also evaluated in a porcine experiment to demonstrate aptitude for these sorts of image processing techniques. While the proposed device compares admirably in image quality tests when compared to existing analog imaging solutions, several key

challenges must be overcome before this design is suitable for use; major problems include the camera's fixed-focus design, low-contrast video caused by the enclosure design, and sporadic frame-dropping leading to unsynchronized video. Because of platform constraints, many of these design challenges cannot be overcome without a significant redesign.

Bibliography

[1] Digital cinema 3d, November 2012.

[2] Wim Abbeloos. Real-time stereo vision. Master's thesis, Karel de Grote-Hogeschool University College, Belgium, May 2010.

[3] A. Abramson. *The history of television, 1942 to 2000*. McFarland, 2003.

[4] R. Adachi and T. Takahashi. Eyepiece and photographing device for fiberscope, jun 1987.

[5] AL Magos AG Gordon. The development of laparoscopic surgery. *Bailliere's Clinical Obstetrics and Gynaecology*, 3:429–449, 1989.

[6] Fabienne Betting, Jacques Feldmar, Nicholas Ayache, and Frederic Devernay. A new framework for fusing stereo images with volumetric medical images. In *Computer Vision, Virtual Reality and Robotics in Medicine*, pages 30–39. Springer, 1995.

[7] A. Boev, A. Gotchev, and K. Egiazarian. Crosstalk measurement methodology for auto-stereoscopic screens. In *3DTV Conference, 2007*, pages 1 –4, may 2007.

[8] R.G. Boothe. *Perception of the visual environment*. Springer, 2001.

[9] Jean-Yves Bouguet. Camera calibration toolbox for matlab (2008). *URL http://www. vision. caltech. edu/bouguetj/calib_doc*, 2008.

[10] W.S. Boyle and G.E. Smith. Charge-coupled semiconductor devices. *Bell System Technics*, 49:587–593, 1970.

[11] John C. Byrn, Stefanie Schluender, Celia M. Divino, John Conrad, Brooke Gurland, Edward Shlasko, and Amir Szold. Three-dimensional imaging improves surgical performance for both novice and experienced operators using the da vinci robot system. *The American Journal of Surgery*, 193(4):519–522, 2007.

[12] A. C. W. Chan, S. C. S. Chung, A. P. C. Yim, J. Y. W. Lau, E. K. W. Ng, and A. K. C. Li. Comparison of two-dimensional vs three-dimensional camera systems in laparoscopic surgery. *Surgical Endoscopy*, 11(5):438–440, 1997.

[13] William H.H. Chapman, Robert J. Albrecht, Victor B. Kim, James A. Young, and W. Randolph Chitwood Jr. Computer-assisted laparoscopic splenectomy with the da vinci surgical robot. *Journal of Laparoendoscopic & Advanced Surgical Techniques*, 12(3):155–159, June 2002.

[14] Robert W Cox and Andrzej Jesmanowicz. Real-time 3d image registration for functional mri. *Magnetic resonance in medicine*, 42(6):1014–1018, 1999.

[15] A Cuschieri. Minimal access surgery and the future of interventional laparoscopy. *American Journal of Surgery*, 161:404–407, 1991.

[16] N.A. Dodgson. Autostereoscopic 3d displays. *Computer*, 38(8):31 – 36, aug. 2005.

[17] Neil A. Dodgson. Variation and extrema of human interpupillary distance. In *Proceedings of SPIE*, volume 5291, pages 36–46, January 2004.

[18] J. Dumpert, A.C. Lehman, N.A. Wood, D. Oleynikov, and S.M. Farritor. Semi-autonomous surgical tasks using a miniature in vivo surgical robot. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 266 –269, sept. 2009.

[19] Asim F. Durrani and Glenn M. Preminger. Three-dimensional video imaging for endoscopic surgery. *Computers in Biology and Medicine*, 25(2):237 – 247, 1995. Virtual Reality for Medicine.

[20] International Organization for Standardization. Iso 12233, photography – electronic still-picture ccamera – resolution measurement. ISO12233:2000(E), September 2000.

[21] E.R. Fossum. Cmos image sensors: Electronic camera-on-a-chip. *Electron Devices, IEEE Transactions on*, 44(10):1689–1698, 1997.

[22] Andrea Fusiello, Emanuele Trucco, and Alessandro Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[23] Donald E Groom, Steven E Holland, Michael E Levi, Nicholas P Palaio, Saul Perlmutter, Richard J Stover, and Mingzhi Wei. Quantum efficiency of a back-illuminated ccd imager: an optical approach. In *Electronic Imaging'99*, pages 80–90. International Society for Optics and Photonics, 1999.

[24] Richard I Hartley. In defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(6):580–593, 1997.

[25] N. S. Holliman. *3D display systems*. IOP Press, 2006.

[26] I.P. Howard and B.J. Rogers. *Binocular vision and stereopsis*. Oxford University Press, USA, 1995.

[27] Tie Hu, P.K. Allen, and D.L. Fowler. In-vivo pan/tilt endoscope with integrated light source. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 1284 –1289, 29 2007-nov. 2 2007.

[28] Tie Hu, P.K. Allen, T. Nadkarni, N.J. Hogle, and D.L. Fowler. Insertable stereoscopic 3d surgical imaging device with pan and tilt. In *Biomedical Robotics and Biomechatronics, 2008. BioRob 2008. 2nd IEEE RAS EMBS International Conference on*, pages 311 –316, oct. 2008.

[29] G. Hubens, H. Coveliers, L. Balliu, M. Ruppert, and W. Vaneerdeweg. A performance study comparing manual and robotically assisted laparoscopic surgery using the da vinci system. *Surgical Endoscopy*, 17:1595–1599, 2003. 10.1007/s00464-002-9248-1.

[30] Y. Ikuno, T. Nakamura, Y. Tojo, S. Nishigaki, H. Suzuki, H. Yabe, J. Yoshinaga, T. Yokoi, K. Ohzeki, M. Kanno, et al. Video scope system, February 13 1990. US Patent 4,901,142.

[31] Ashish Jain and Janusz Konrad. Crosstalk in automultiscopic 3-d displays: Blessing in disguise?, 2007.

[32] J. Janesick and G. Putnam. Developments and applications of high-performance ccd and cmos imaging arrays. *Annual Review of Nuclear and Particle Science*, 53(1):263–300, 2003.

[33] Rémi Jean. Demosaicing with the bayer pattern. *Red*, 33(R53):2, 2010.

[34] Yun-Suk Kang and Yo-Sung Ho. An efficient image rectification method for parallel multi-camera arrangement. *Consumer Electronics, IEEE Transactions on*, 57(3):1041–1048, 2011.

[35] S. Kleinfelder, S.H. Lim, X. Liu, and A. El Gamal. A 10000 frames/s cmos digital pixel sensor. *Solid-State Circuits, IEEE Journal of*, 36(12):2049–2059, 2001.

[36] J.J. Koenderink and VAN DooRN. Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer. *Optica acta*, 22:773–791, 1975.

[37] P. Kotiranta, I. Kelander, M. Rouvala, and J. Takaneva. Characterization of flexible interconnects in mobile devices. In *Signal Propagation on Interconnects, 2007. SPI 2007. IEEE Workshop on*, pages 113–116. IEEE, 2007.

[38] Christoph H. Krah. Three-dimensional display system. U.S. Patent No. 7,843,449, September 2006.

[39] Michael P Lesser. Improving ccd quantum efficiency. In *Proc. SPIE*, volume 2198, page 782, 1994.

[40] Margaret Livingstone and David H Hubel. *Vision and art: The biology of seeing*, volume 2. Harry N. Abrams New York, 2002.

[41] H Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Readings in Computer Vision: Issues, Problems, Principles, and Paradigms, MA Fischler and O. Firschein, eds*, pages 61–62, 1987.

[42] Charles Loop and Zhengyou Zhang. Computing rectifying homographies for stereo vision. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, volume 1. IEEE, 1999.

[43] Guy Meynants, Bart Dierickx, and Danny Scheffer. Cmos active pixel image sensor with ccd performance. In *SYBEN-Broadband European Networks and Electronic Image Capture and Publishing*, pages 68–76. International Society for Optics and Photonics, 1998.

[44] A. Miller, P. Allen, and D. Fowler. In-vivo stereoscopic imaging system with 5 degrees-of-freedom for minimal access surgery. *Studies in Health Technology and Informatics*, 98:234–240, 2004.

[45] Pascal Monasse, Jean-Michel Morel, Zhongwei Tang, et al. Three-step image rectification. In *Proceedings of the British Machine Vision Conference*, 2010.

[46] U. D. A Mueller-Richter, A. Limberger, P. Weber, K. W. Ruprecht, W. Spitzer, and M. Schilling. Possibilities and limitations of current stereo-endoscopy. *Surgical Endoscopy*, 18:942–947, 2004. 10.1007/s00464-003-9097-6.

[47] E Neugebauer, H Troidl, CK Kum, E Eypasch, M Miserez, and Paul A. The e.a.e.s. consensus development conferences on laparoscopic cholecystectomy, appendectomy, and hernia repair. consensus statements – september 1994. In *The Educational Committee of the European Association for Endoscopic Surgery*, volume 9, pages 550–563, 1995.

[48] D.P. Noonan, P. Mountney, D.S. Elson, A. Darzi, and Guang-Zhong Yang. A stereoscopic fibroscope for camera motion and 3d depth recovery during minimally invasive surgery. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pages 4463 –4468, may 2009.

[49] M. Okabe and A. Kikuchi. Television camera for endoscopes, June 1990.

[50] Walter Hines Page and Arthur Wilson Page. *The World's Work: Volume XV: A History of Our Time*. Doubleday, Page & Company, 1908.

[51] V Papadimitriou and Tim J Dennis. Epipolar line estimation and rectification for stereo image pairs. *Image Processing, IEEE Transactions on*, 5(4):672–676, 1996.

[52] E. T. Psota, K. Strabala, J. Dumpert, L. C. Pérez, S. Farritor, and D. Oleynikov. Stereo image-based arm tracking for in vivo surgical robotics. In *Studies in Technology and Informatics*, volume 163, pages 454–460, 2011.

[53] Eric T Psota. *Stereoscopic Wound Measurement Device and Algorithm*. PhD thesis, University of Nebraska–Lincoln, 2006.

[54] Eric T. Psota, Jedrzej Kowalczuk, Jay Carlson, and Lance C. Pérez. A local iterative refinement method for adaptive support-weight stereo matching. In *International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*, July 2011.

[55] Y. Reibel, M. Jung, M. Bouhifd, B. Cunin, and C. Draman. Ccd or cmos camera noise characterisation. *The European Physical Journal Applied Physics*, 21:75–80, 2003.

[56] Brian J. Rogers. *Seeing in Depth: Depth perception*, volume 2. University of Toronto Press, 2002.

[57] Jacob Rosen, Blake Hannaford, and Richard M. Satava, editors. *Surgical Robotics: Systems Applications and Visions*. Springer, 2011.

[58] Daniel Scharstein and Anna Blasiak. Middlebury stereo evaluation - version 2, July 2014.

[59] Philip Servos, Melvyn A. Goodale, and Lorna S. Jakobson. The role of binocular vision in prehension: a kinematic analysis. *Vision Research*, 32(8):1513 – 1521, 1992.

[60] Pieter J. H. Seunti, Ingrid E. J. Heynderickx, Wijnand A. Ijsselsteijn, Paul M. J. Van Den Avoort, Jelle Berentsen, Iwan J. Dalm, Marc T. M. Lambooij, and Willem Oosting. Viewing experience and naturalness of 3d images. In *Three-Dimensional TV, Video, and Display IV, Proceedings of SPIE*, volume 6016, Bellingham, WA, ETATS-UNIS, 2005. Society of Photo-Optical Instrumentation Engineers.

[61] Pieter Seuntiens, Lydia Meesters, and Wijnand Ijsselsteijn. Perceived quality of compressed stereoscopic images: Effects of symmetric and asymmetric jpeg coding and camera separation. *ACM Trans. Appl. Percept.*, 3:95–109, April 2006.

[62] Danail Stoyanov, Ara Darzi, and Guang Yang. Dense 3d depth recovery for soft tissue deformation during robotically assisted laparoscopic surgery. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004*, volume 3217 of *Lecture Notes in Computer Science*, pages 41–48. Springer Berlin / Heidelberg, 2004.

[63] Danail Stoyanov, Marco Scarzanella, Philip Pratt, and Guang-Zhong Yang. Real-time stereo reconstruction in robotically assisted minimally invasive surgery. In Tianzi Jiang, Nassir Navab, Josien Pluim, and Max Viergever, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2010*, volume 6361 of *Lecture Notes in Computer Science*, pages 275–282. Springer Berlin / Heidelberg, 2010.

[64] G. Svaetichin. Spectral response curves from single cones. *Acta Physiol Scand Suppl*, 39(134):17–46, 1956.

[65] N. Taffinder, SGT Smith, J. Huber, RCG Russell, and A. Darzi. The effect of a second-generation 3d endoscope on the laparoscopic precision of novices and experienced surgeons. *Surgical endoscopy*, 13(11):1087–1092, 1999.

[66] Maurice H. P. H. van Beurden, Gert van Hoey, Haralambos Hatzakis, and Wijnand A. Ijsselsteijn. Stereoscopic displays in medical domains: a review of perception and performance effects. In *Human Vision and Electronic Imaging XIV, Proceedings of the SPIE*, volume 7240, 2009.

[67] P.K. Weimer, S.V. FORQUE, and R.R. Goodrich. The vidicon photoconductive camera tube. *Electronics*, pages 70–73, 1950.

[68] Tyler D. Wortman, Kyle W. Strabala, Amy C. Lehman, Shane M. Farritor, and Dmitry Oleynikov. Laparoendoscopic single-site surgery using a multi-functional miniature in vivo robot. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 7(1):17–21, 2011.

[69] Tao T Wu, Jianan Y Qu, et al. Optical imaging for medical diagnosis based on active stereo vision and motion tracking. *Opt. Express*, 15(16):10421–10426, 2007.

[70] L. Zhang, Y. Jin, L. Lin, J. Li, and Y. Du. The comparison of ccd and cmos image sensors. In *International Conference of Optical Instrument and Technology*, pages 71570T–71570T. International Society for Optics and Photonics, 2008.