Large Core Polymer Optical Backplanes for Fluorescence Detection

by

Kevin Shao-Kwan Lee

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

Master of Science in Electrical Engineering and Computer Science
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
June 2006

Author...
Department of Electrical Engineering and Computer Science
May 19, 2006

Certified by..
Rajeev J. Ram
Associate Professor
Thesis Supervisor

Accepted by...
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Large Core Polymer Optical Backplanes for Fluorescence Detection

by

Kevin Shao-Kwan Lee

Submitted to the Department of Electrical Engineering and Computer Science
On May 19, 2006, in partial fulfillment of the
Requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

Fluorescence based sensors are used for determining environmental parameters such as dissolved oxygen or pH in biological systems without disturbing a biological system's equilibrium. Recently, there has been a drive to provide biological analysis tools in a compact form, resulting in arrays of miniature devices which can perform multiple functions in parallel such as bacteria cultures or DNA analysis. As these new types of chips become more integrated and parallel, the amount of sensors required for them increases. As more sensors are added, off chip components such as photodetectors, LEDs, and fibers also begin to scale linearly. In an effort to simplify and integrate the detection side of these systems, a platform is being developed which utilizes the same polymer materials used for biochips to create optical components. By combining elements such as waveguides and mirror couplers, arrays of small devices capable of out-of-plane detection are possible, decoupling the biochip design from the transducer design while still maintaining compact integrated functionality.

Thesis Supervisor: Rajeev J. Ram
Title: Associate Professor

Acknowledgements

There are many people to thank for helping in the creation of this thesis. First and foremost, I would like to thank my professor, Rajeev Ram, for always providing insight, explanation, and support towards the many problems I faced in both research and other endeavors. Without his help, I would have never even been introduced to this topic or even finish with such an optimistic outlook towards research and my future. I would also like to thank Harry Lee, for not only developing most of the tools and fabrication process I used, but also in providing help with many of the fundamental and experimental challenges encountered. He is always willing to help discuss whatever problems I might be facing over a thrown football. While already graduated, I would also like to give thanks to Gustavo Gil, for his always cheerful cynicism towards everything. Without his humor and friendship, I would have never survived my first year. To Peter Mayer and Tom Liptay, thanks for always lending a hand with equipment and explanations and being nice enough to stop me from diving head first into pointless experiments or tasks that were not well thought through. It was nice of both of you to tolerate my slight lack of chemical hygiene. To my DDR buddy, Tauhid Zaman, thanks for the helpful discussions on waveguides and waveguide simulations. I would also like to mention the newest of the lab members, Reja Amatya and Jason Orcutt, for being very helpful in letting me borrow their equipment, helping me take measurements, and not getting upset when I leave their setups in a complete mess. Also thanks to thank Xiaoyun Guo for her helpful insight into fabrication and measurement techniques. Last but not least, thanks to Cathy Bourgeois for both the wonderful conversations and the vital help with getting things, considering my lack of organization and deficiency in reading directions.

Outside of the lab, I would have never learned how to do deposition or metrology without the aid of Kurt Broderick. His cheerful presence made the clean room much more bearable. I would have never made it to MIT without the help of Matt Espiau, who was not only an excellent friend, but also taught me a lot about engineering and life in general. Without his help, I would have never learned about the research and working environment during undergrad. To my girlfriend, thanks for always being supportive and

optimistic, even when I was incredibly irritable, and of course to my family for always having high expectations but also allowing me to pursue any and all of my interests no matter what they may be with support in place of judgment. Since I am sure that anyone who reads this and knows me has helped me along the way, I would just like to say thank you.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Fluorescence based sensors are used for determining environmental parameters such as dissolved oxygen or pH in biological systems. Such optical sensing techniques can provide real time monitoring capabilities without disturbing a biological system's equilibrium. Recently, there has been a drive to provide biological analysis tools in a compact form, usually resulting in arrays of miniature devices which can perform multiple functions or operations in parallel such as bacteria cultures, cell counting, or DNA analysis [1, 2, 3]. As these new types of chips become more integrated and parallel, it becomes increasingly important to develop sensors and sensing methods which are cheap and scalable.

An approach to realizing this goal is to provide optical detection through fluorescence intensity measurements. This method of detection is ideal for applications involving cell sorting or DNA since different species can be labeled with different wavelength dyes and intensity can directly translate into concentration. The advantage to a stationary approach such as measuring fluorescence intensity is the ability to use low bandwidth detectors such as CCD arrays. Dynamics on time scales of 100 ms or larger are easily captured by CCD arrays, and scalability is limited only by the resolution of the camera used. While intensity based methods can measure concentrations of molecules in solution, intensity fluctuations due to noise sources such as external lighting, alignment error, or dye photobleaching can lead to errors large enough to inhibit accurate detection. In order for such methods to work consistently, sensors would need calibration on a routine basis to ensure that intensity measurements accurately reflect concentration estimates.

The alternative approach to a concentration measurement is a lifetime based measurement. For this type of measurement, a fluorophore which is sensitive to a specific molecule or chemical must be used. The concentration of the molecule in contact with the fluorophore changes the fluorescence lifetime of the fluorophore. By modulating an excitation signal and comparing the detected phase against the initial phase, the lifetime and concentration can be determined. While this approach virtually eliminates intensity based errors, the detection bandwidth required for lifetime measurements depends specifically on the fluorophore. For example, for an oxygen sensitive dye such as Platinum Octaethylporphine Ketone (PtOEPK), the phosphorescence lifetime is 61.4 μs at zero oxygen [29] which implies that a detector would need a bandwidth larger than 4 kHz if a maximum phase shift of 90 degrees is allowed. Since the bandwidth required is large, using large array CCD cameras, which generally take 30-60 frames a second, is not possible. Since CCD cameras are too slow for detection, a reasonable alternative is to use photodiode arrays (PDA). To uniquely capture the entire chip's fluorescence sensors, a two dimensional PDA would be required, with a detection scheme similar to detection with a CCD camera. While arrays as large as 16x16 are available commercially with bandwidth up to 100 MHz, the cost for a 16x16 PDA from Hamamatsu is over $7000, not to mention that imaging a large density of fluorescing sensors from an arbitrarily designed biochip to a PDA with only 256 square pixels can be difficult. Therefore, in order to perform detection at reasonable expense, bulk photo detectors and collection optics must be supplied to each sensor or multiplexing methods must be use to reduce the amount of detectors and sources.

One established method for performing fluorescence detection without imaging optics is through the use of large core fiber bundles. Since step index multimode fibers offer bandwidths up to 5 MHz-km for cores as large as 1 mm diameter, fiber bundles are ideal for routing fluorescence signals from fluorophores to detectors. To reduce signal leakage from the excitation source into the collection fiber, collection can be performed at a right angle to the excitation as shown in Figure 1.1.

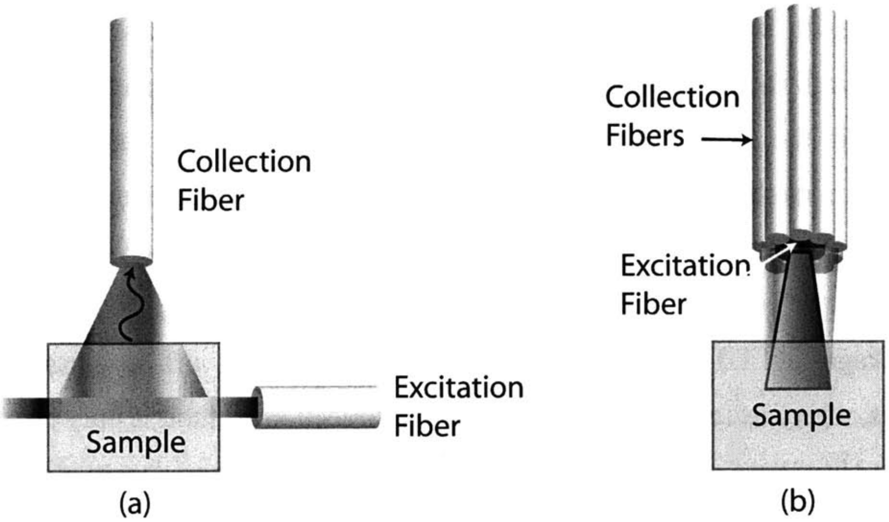

Figure 1.1.Methods of fluorescence detection with fibers. Right angle collection (a) and back collection (b) methods are shown.

While this method is very effective, it also requires space for both the excitation and the collection fibers. In miniaturized systems, it is either impractical or impossible to create sample areas open enough for this method without impeding the system design. An example can be seen with the miniature bioreactor array shown in Figure 1.2.

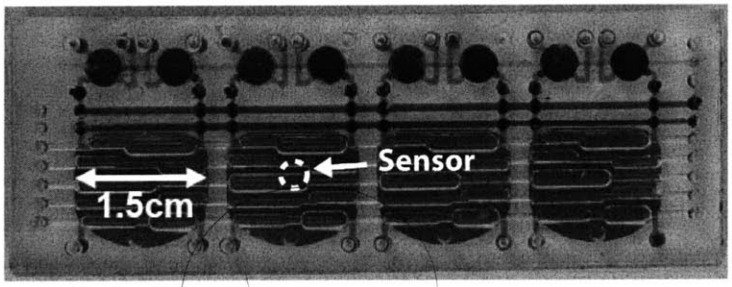

Figure 1.2. Illustration of a miniature bioreactor array device from [28], showing where an oxygen sensor is typically located. With sensors situated in the center of the device, it is difficult to perform right angle fluorescence detection.

Due to the compact parallel design, bioreactor wells situated in the middle of the device, surrounded by neighboring bioreactor wells, have no space for fibers to perform right angle sensing on sensors located in the center of the wells.

For devices with such large fluidic component density, fluorescence sensing can be achieved through the use of a fiber optic probe, where the excitation and collection fibers are oriented in the same direction, in some cases the same fiber can even be used for both. With this method, there is more leakage due to surface reflections of the excitation beam. For example, a typical 1 mm diameter center excitation fiber with 9 surrounding 500 μm diameter collection fibers incident 1 mm away from a glass surface result in a signal leakage of 1.3 percent due to surface reflections off a glass interface in simulations, where as leakage from a right angle collection method is only proportional to the amount of scatters within the sample. Nevertheless, the probe design is more compact than a right angle design, requiring only one interface between the fluorescing system and the probe. In this way, the probe is much more compatible with sensors located on planar devices. An illustration of the miniature bioreactor array device with fiber bundles performing probe type measurements is shown in Figure 1.3.

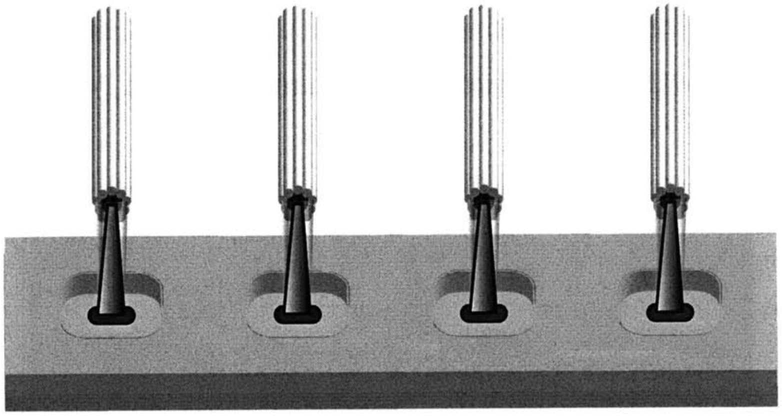

Figure 1.3. Illustration of the miniature bioreactor array utilizing out of plane detection using fiber bundles.

Even though fiber based systems provide a useful method for interrogating fluorescence sensors, they are limited to out-of-plane detection unless systems are specifically designed with fiber input ports [5, 8]. In addition, each individual sensor requires a separate fiber bundle, LED, and photodetector if a non-translating measurement system is desired. Systems requiring so many components per sensor are not scalable cheaply and also cost hours in the assembly and interface of fiber bundles since bundles can not be batch processed. Waveguides offer a solution to the scalability problem of fiber bundles. They can be integrated with the chip to provide compact methods of delivering fluorescence signals to photodetectors. With a proper waveguide design, the optical detection system existing off-chip can be reduced by adding optical functionality on-chip. Other issues such as alignment related measurement errors are also reduced by the integrated nature of the system.

1.2 Previous work

Waveguide integration with fluidic chips has been explored by a variety of groups, and many different methods of fabrication are possible. It has been demonstrated that waveguides can be fabricated with Polydimethylsiloxane (PDMS) and are capable of monitoring glucose concentration through fluorescence detection [4]. While this method consisted of biochip compatible materials, the achieved index contrast was low, leading to issues with signal strength. In order to increase the signal to noise ratio of the system, a right angle fluorescence detection method was implemented, resulting in a complicated waveguide dependent microfluidic design very different from the possible simple flow channel as shown in Figure 1.4.

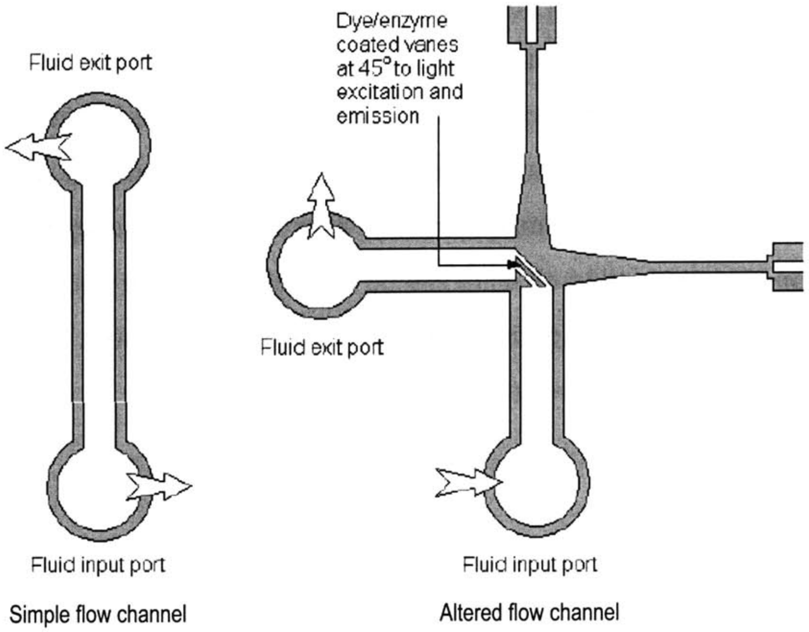

Figure 1.4. Illustration of a simple fluid channel in comparison to the modified channel in [4]. Channel modification due to waveguide design restrictions results in a bulky structure that is difficult to design around.

Other experiments have also demonstrated absorption measurements with integrated microfluidic structures and waveguides [6]. Waveguides in this device were coupled to fluidic channels containing solutions to be analyzed. Again, since waveguide integration with the biochip required that waveguides were in plane with fluidics, the microfluidic structure was made more complicated than necessary to accommodate waveguide elements as shown in Figure 1.5. For waveguides to function properly, both the shape and the path length of the microfluidic channel within the optical sensing region had to be designed.

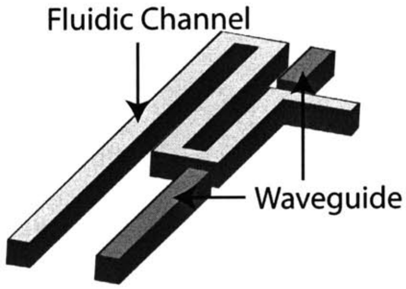

Figure 1.5. Illustration of the device for concentration measurements presented in [6]. The fluidic channel must be designed with respect to the sensing capabilities of the waveguides.

There have also been experiments in which polymer waveguides were fabricated for the purpose of providing optical interconnections between different electrical systems by utilizing UV laser direct writing of spin coated polymer layers [34]. In this application, the desire is to create a backplane consisting of optical waveguides and out of plane couplers to route signals between different electrical components. While the application is different, the problem is similar to that of fluidic chips. Just as optical signals are routed underneath the electrical layers to provide a means of contact between different electrical systems, optical signals can be routed underneath fluidic chips to provide an interface to the fluidic chip without interfering with fluidic operations. By applying a backplane approach to optical detection in fluidic systems, conflicts between waveguides and fluidic channels can be avoided.

As demonstrated by previous experiments, polymer waveguides have been fabricated by a variety of methods and straight waveguide structures are possible. While polymer waveguides are cheap and can be mass produced, two difficulties arise which limit performance, one from mold creation which can limit the physical size of waveguide cross-sections and one from material selection which can limit the numerical aperture of the waveguide. The numerical aperture (NA) is set by the maximum input or output angle which is guided by the waveguide as shown in Figure 1.6, and is directly dependent on the critical angle of the waveguide system.

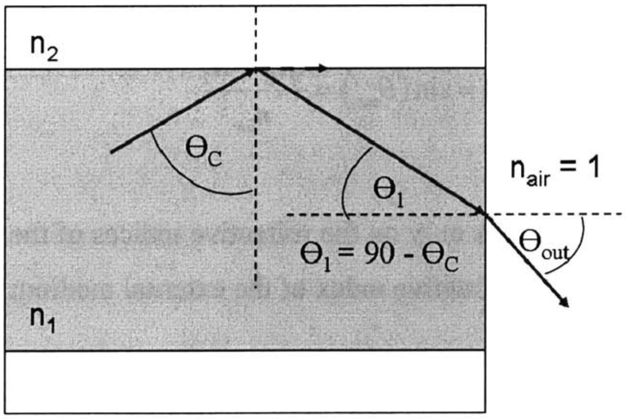

Figure 1.6. Illustration of refraction into the cladding and reflection into the waveguide leading to a waveguide output.

The critical angle of the system is directly found through Snell's Law,

$$n_1 \sin(\theta_1) = n_2 \sin(\theta_2) \tag{1.1}$$

which relates angle of the incident light, θ_1, and the angle of the transmitted light, θ_2, to the refractive index n_1 of the incident light medium and the refractive index n_2 of the transmitted light medium. Since the critical angle θ_C is the cut-off angle between light transmission into the cladding and light total internal reflection, the condition on Equation 1.1 is that θ_2 must be 90 degrees.

$$\theta_C = \sin^{-1}\left(\frac{n_2}{n_1}\right) \tag{1.2}$$

As shown in Figure 1.6, the output angle which occurs when light is guided in the waveguide at the critical angle is the maximum output angle.

$$\theta_{out} = \sin^{-1}\left(\frac{n_1}{n_{air}}\sin(90-\theta_C)\right) \tag{1.3}$$

After performing some trigonometry, we arrive at an equation for the numerical aperture.

$$NA = \sin(\theta_{out}) = \frac{\sqrt{n_1^2 - n_2^2}}{n_{air}} \tag{1.4}$$

We can see that the NA depends only on the refractive indices of the core and cladding of the waveguide as well as the refractive index of the external medium.

While waveguides fabricated by lithographic molding procedures are very smooth, they are small in dimension and have small waveguide cross-sections, which are restricted by the typical spin-coating layer thicknesses of 100 μm, limiting their ability to collect fluorescence from large area fluorescence emitters. Even though the numerical apertures may be large, the process of spin coating photoresist layers limits the actual dimensions of the physical aperture of the waveguide. The other problem faced by methods which rely on highly compatible materials, or in the case of [4] on the same material for the core and cladding, is a small refractive index contrast (in [4], an index contrast of 0.02), leading to a small solid angle where light can be collected. The dependence of the collection efficiency on a system limited by numerical aperture and on a system limited by physical aperture is shown in Figure 1.7.

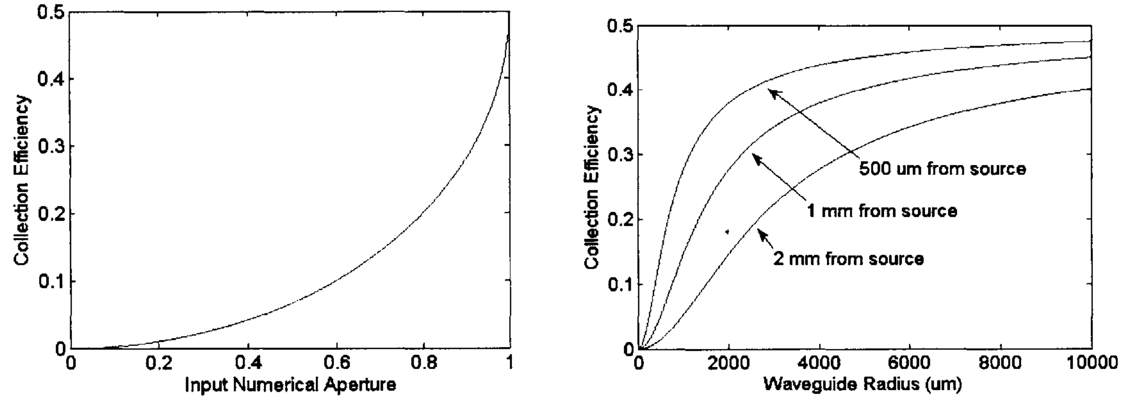

Figure 1.7. Collection efficiency versus numerical aperture (left) and versus physical aperture (right) for a spherically uniform point source emission. Reflection effects are not considered since they are dependent on the actual values of the refractive index in the core and the cladding of the waveguide.

Many designs suffer from low signal collection efficiency due to their physical size or index contrast as mentioned above. While systems such as the optical interconnect system [34] quote input collection efficiencies of 90%, they are designed for collecting the emission of light from a collimated laser beam rather than from a uniformly emitting point source.

Another concern that affects lifetime based measurements which depends on the waveguide index contrast is the modulation frequency which can be supported by the waveguide. This limitation results from the physically different path lengths of rays propagating in the structure and is called multimode distortion. Even though it seems that arbitrarily increasing the numerical aperture and physical aperture will increase the collection efficiency, this modulation bandwidth, which is dependent on the waveguide index contrast, can become the limiting factor. Fortunately, step index waveguides with core diameters of 1 mm as discussed in Section 1.1 have bandwidths in the range of 5 MHz-km, which are more than capable of handling fluorophores such as PtOEPK which require frequencies in the kHz range. Therefore, it is more beneficial to create waveguides with extremely large cores to improve LED input coupling efficiency and high index contrasts in order to collect as much fluorescence as possible from as large of an active fluorescence area as possible. This task is difficult for batch fabrication procedures suited to the development of microfluidics.

A more practical and cheap approach to fabricating larger structures is to use conventional direct writing processes. It has been demonstrated that milling can achieve submicron roughness in polymer structures [7]. While these waveguides are capable of propagating light, they exhibit marginal loss characteristics, typically greater than 1 dB/cm due to the surface roughness characteristics of milling. The fabrication procedure used also requires direct milling for each device, making it hard to scale for large volume production.

Whether large core waveguides are fabricated with lithography, etching, or direct writing processes, devices with features tens of microns and larger are often left with roughness that results in waveguides with non-ideal propagation characteristics. In addition to the importance of numerical and physical aperture in the waveguide's ability to collect fluorescence, these non-ideal propagation characteristics resulting from waveguide roughness also affect the waveguide's ability to route that collected fluorescence to the desired location or photodetector, impacting the overall efficiency of the sensing system.

Experiments to measure the effects of fabrication generated roughness on the characteristics of large polymer waveguides have been explored. Researchers at IBM [17] conducted measurements on waveguide samples with dimensions of 70 μm x 55 μm x 70 mm fabricated using UV-curable epoxies. Results from their measurements suggest that waveguide roughness actually decreases the numerical aperture of the waveguide and attempts to use the entire theoretical aperture lead to higher loss in the waveguide during propagation and eventual reduction of numerical aperture at the output. From a ray optics picture, this loss results since higher order modes, or rays traveling at sharper angles within the waveguide, hit the rough interfaces more frequently than rays traveling at shallow angles. Therefore rays which contribute to the large numerical aperture are more easily lost. In their measurements specifically, the numerical aperture reduced from 0.65 to 0.29 and attempts to increase the input angle from 4° to 17° increased the loss by 0.28 dB/cm [17]. These loss and numerical aperture characteristics resulting from roughness can play an important role in fluorescence collection efficiency of large core waveguides.

Attempts to model the roughness characteristics of fabricated multimode waveguides have been explored in an effort to provide simulation tools which can properly capture the roughness effects resulting from fabrication. Since roughness can not be determined exactly, the focus is set on determining statistical models which can be integrated into simulation programs such as ray tracers or mode solvers. Incorporating roughness into simulations has been explored for both ray tracers and mode solvers and in general an exponential model is assumed for the distribution of the roughness [19, 30]. However, by assuming a statistical exponential model for the roughness, the models can only determine how power is distributed within the waveguide.

In the mode model, a description of loss due to roughness is modeled as point sources resulting from variations in the index boundary of the waveguide. This description captures the mode dependent losses for straight waveguides which have roughness only on the side walls. Unfortunately, the loss characteristics must be calculated for each mode in the system, which can be very time consuming for heavily multimode systems with mode quantities larger than 1000. In addition, this method does not consider how the radiation reenters the waveguide system resulting in the generation of more guided modes. The alternative approach taken by [19] is to apply the statistical roughness to a ray tracer algorithm through the use of probabilistic perturbations determined by the roughness. This approximate method works well in not only capturing the loss characteristics, but also the mode coupling characteristics of the roughness. In addition, the method is easily applied to arbitrary structures. This ray tracing model, however, is limited to situations where the roughness is small in comparison to the wavelength, in order to make a first order approximation, leading to roughness effects which act diffuse just like [30].

1.3 Proposed Research

In an effort to simplify and integrate the optical detection side of fluidic systems without hindering fluidic system design, a fabrication platform is being developed which utilizes the same polymer materials used for fluidic chips to create integrable optical components. With this new fabrication method, waveguides, power splitters, combiners, and bends can be introduced and integrated cheaply into existing biochips. By combining these elements, arrays of small devices capable of out-of-plane detection are possible, allowing for the optics to rest on a backplane, decoupling the biological design from the transducer design.

The first step to achieving an optical backplane is to demonstrate a fabrication process capable of fabricating waveguides. In Chapter 2, a few materials usable as waveguide cores will be explored and a new fabrication process will be demonstrated. Since it has been demonstrated that waveguides can be prototyped quickly with conventional milling, waveguide molds will be created using conventional milling processes. To further improve the waveguide propagation loss characteristics resulting from conventional milling, a process called vapor polishing will be introduced. PDMS rapid prototyping will then be used to create waveguide structures suitable for integration with existing PDMS devices. Measurements will then be performed similar to [17] to determine the effects of milled and vapor polished surface roughness on waveguide performance.

By measuring waveguide parameters such as loss and numerical aperture, a model can be built to describe any non-ideal factors contributed by the fabrication process. The goal of Chapter 3 is to develop a ray tracer which can model the roughness effects measured. By modeling these factors into a ray tracing program, an accurate description of the waveguide can be generated allowing for design and optimization of more complicated structures with simulations. By utilizing the simulator in combination with the fabrication process, a compact waveguide approach to fluorescence detection can be realized, allowing for cheap and miniaturized integrated detection systems.

After demonstrating that waveguides are possible within a polymer platform, Chapter 4 will focus on the creation of a variety of other devices which can aid integration. By moving to a polymer platform, functionality of nanoparticles can be explored since many nanoparticles can be integrated into or between polymer chains including dyes and quantum dots. With proper selection of dopants, devices such as wavelength filters and light sources can be integrated within waveguides, furthering the benefits of on-chip waveguide integration. The advantage of milling in the generation of arbitrary structure shapes will also be explored. Arbitrarily angled structures can be fabricated, enabling the creation of out-of-plane components which can direct optical signals in arbitrary directions. With out-of-plane bends, the optical detection system remains compact and chip-sized, while the deficiencies of planar detection discussed in Section 1.2 are reduced.

After developing a variety of polymer optical components, Chapter 5 will focus on the integration of components into a working optical backplane. A waveguide based system will be proposed, and the ray tracer developed in Chapter 3 will be utilized to optimize the fluorescence detection system. The fabricated device will be designed as an addition to the bioreactor array system developed in [28], and will demonstrate simultaneous on-chip detection of oxygen concentration for four separate oxygen sensors without the need for any fiber bundles. The signal to noise ratio of the waveguide system will be compared to the fiber bundle system in order to determine the effectiveness of the waveguide design. After demonstrating the effectiveness and versatility of the polymer optical backplane, Chapter 6 will summarize the results presented and suggest future work which can be performed to further enhance the benefit of a polymer optical backplane system.

Chapter 2

Large Core Waveguide Fabrication

2.1 Introduction

The first step to realizing waveguides in a biochip platform is to fabricate and characterize waveguides. Measurements on straight waveguides will help determine factors contributing to loss, scattering, and changes in numerical aperture. For large waveguides, these are mainly caused by material properties and waveguide imperfections. While problems relating to the material properties can be reduced by selecting proper polymers with ideal attributes, problems relating to waveguide imperfections mostly result from fabrication processes, which are much harder to characterize.

Before developing a fabrication process, proper materials must be chosen. In Section 2.2, a biochip material platform will be chosen and a variety of materials will be explored and tested for compatibility with the chosen polymer platform. A fabrication process will then be developed in Section 2.3 utilizing the chosen materials and straight waveguide fabrication will be demonstrated. Measurements performed in Section 2.4 on the fabricated straight waveguides will show that even though their numerical aperture and loss properties are acceptable for the application, they are significantly less than the theoretical values. Surface roughness on the waveguide is demonstrated to be the probable cause.

2.2 Materials

Since most cheap or disposable biochips are made from polymer materials, the cladding should also be chosen from these materials. A common prototype material used for these applications is PDMS (Sylgard 184) which has a refractive index of 1.43. This choice of cladding limits the cores of the waveguide to polymers that are both compatible with PDMS and also of higher refractive index.

2.2.1 PolyDiMethylSiloxane (PDMS)

The most compatible material to add to PDMS would be another silicone based polymer with an increased index. While it has been shown that PDMS baked at higher temperatures leads to increases in refractive index [8], index contrasts using this technique are small, with a demonstrated maximum increase of 0.02, leading to small numerical apertures, theoretically NA=0.24, and a resulting small collection efficiency of 0.014. Since PDMS itself is not an option, three other types of silicon based polymers were explored as possible core materials, RTV 656, LS-3357, and LS-6257.

The first compound, RTV 656 made by GE silicones, is much like Sylgard 184. While it is compatible enough to cure in contact with Sylgard 184, a waveguide created from RTV656 does not exhibit waveguiding, indicating that the index contrast between the two materials is either reversed or small. The other two materials, LS-3357 and LS-6257 from NuSil Technology, both have a refractive index of 1.57 and are silicone based materials. LS-3357 cures to a gel, so it is not structurally stable and can not be used as a polymer material for the system. LS-6257, however, is a thermally cured polymer like Sylgard 184. The plot of the optical absorption properties for LS-6257, given in Figure 2.1, shows excellent transmission in the visible wavelengths.

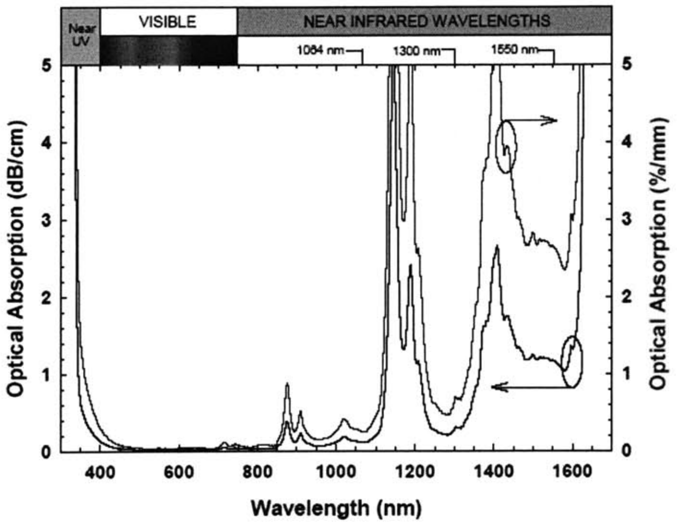

Figure 2.1. Plot of the absorption of LS-6257. http://www.nusil.com/PDF/PP/LS-6257P.pdf

Waveguides made with LS-6257 cores exhibit guiding behavior, but material compatibility problems arise. As a silicone based material, achieving an index of 1.57 requires the addition of high index chemical groups, in this case, phenyl groups. Since these groups are not native to the polymer, LS-6257 requires temperatures in excess of 100 °C to cure. During thermal curing, these groups also precipitate out and diffuse into the Sylgard 184. While the core remains clear, this leads to the interface having a milky white color due to diffusion of high index particles into the PDMS as shown in Figure 2.2.

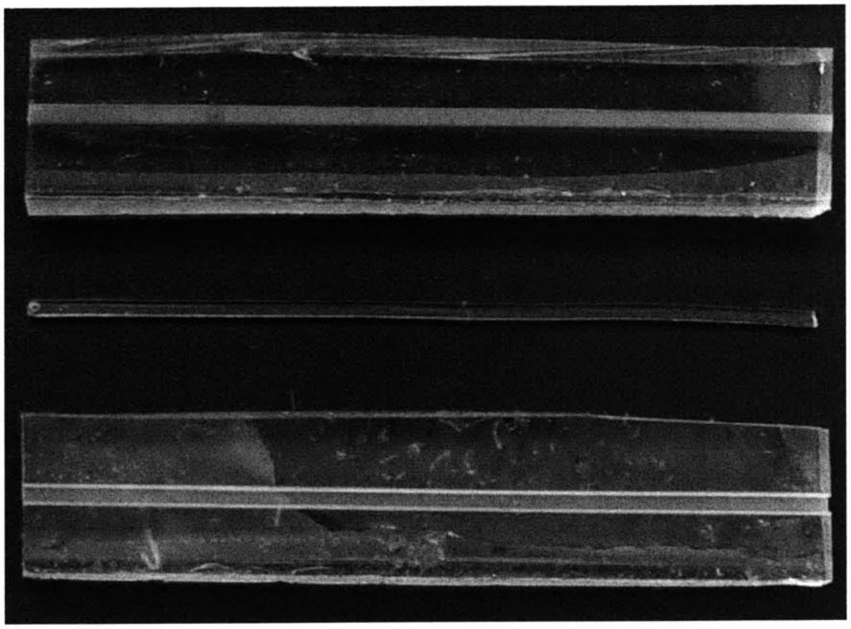

Figure 2.2. A picture of a PDMS channel with LS-6257 cured in the middle and then removed. A milky white color on the cladding results from the diffusion of high index molecules through the interface. The core, however, remains clear.

In addition to loss, LS-6257 is only 98% solids, and upon curing, there is considerable shrinkage, leading to problems at the input and output faces of the waveguide. All of the resulting problems result in all three silicon based polymers not being suitable for use as waveguides in a PDMS environment.

2.2.2 Polyurethane

An alternative material to a silicon based polymer is a polyurethane polymer. Products supplied by Norland Products are examples of thiol based polyurethane polymers. Norland Optical Adhesives (NOA) cure under ultraviolet light into flexible polymer materials which can be bonded to most plastics, metals, and glass.

Optical Properties

Since most of the fluorescence markers used in biological experiments range from the edge of the UV to the NIR regions, a waveguide material chosen needs excellent optical clarity throughout the visible spectrum. The spectrum shown in Figure 2.3 is taken for NOA71 from Norland Products [31] and shows the estimated absorption coefficient versus wavelength for the polymer normalized relative to air. Since there is no reference loss to compare with, the absorption plot is shifted to zero absorption at 850 nm as an estimate. This leads to an absorption coefficient of 0.02 dB/cm at 470 nm, which is reasonable in comparison to the absorption coefficient of an NOA71 waveguide as measured in Section 2.4.

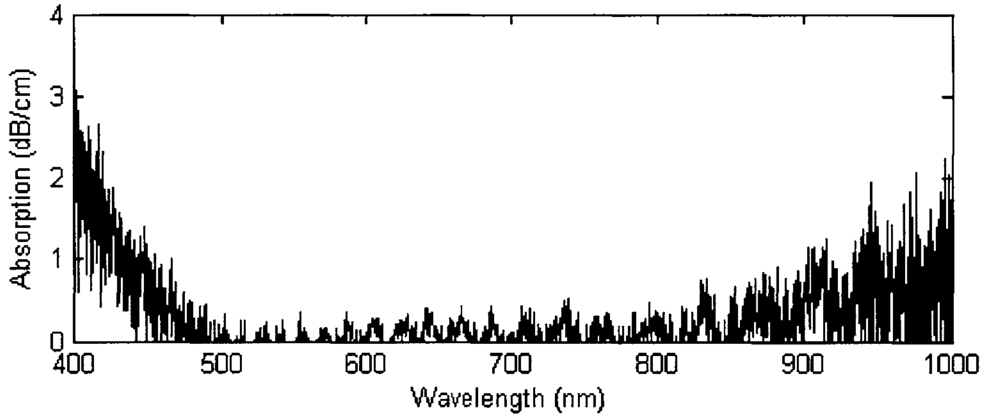

Figure 2.3. Estimated absorption coefficient of NOA71 relative to air. NOA71 exhibits usable transmission characteristics for short length waveguides.

From the plot of transmission versus wavelength given in Figure 2.3, NOA71 is seen to exhibit excellent optical clarity above 400 nm and near uniform transmission above 500 nm and is usable as a core material for sensors operating with excitation wavelengths in the visible spectrum.

In addition to excellent optical clarity, the waveguide core must have a higher index of refraction than the surrounding material in order to contain light within. The refractive index of NOA71 is 1.56 [31] while the refractive index of PDMS lies near 1.43 [4]. This results in a theoretically calculated numerical aperture of 0.62 or a solid angle of 72 degrees.

Physical Properties

While NOA 71 has excellent optical properties for visible optical sensors, its physical properties determine its compatibility with current micro total analysis systems. As with most materials in contact with fully cured PDMS, NOA 71 does not spread readily and wet to PDMS since PDMS in its cured state has a low surface energy of 19.8 mJ/m^2 [32]. The water contact angle for PDMS is much higher than for polyurethane, also indicating that the surface energy of PDMS is lower than the surface energy of polyurethane [9]. It should be noted that in [9], the water contact angle is defined as the angle between the liquid-gas interface and the solid-gas interface, where as the water contact angle is defined here as the angle between the liquid-gas interface and the liquid-solid interface. It is hard for a liquid with high surface energy to wet to a substrate with low surface energy since it is easier for the liquid to be attracted to itself at the surface as will be discussed in Section 2.3.2. If the liquid does not wet and is not attracted to the substrate, the strength of the bond at the interface is weak.

In order to allow the NOA prepolymer to wet to the PDMS, the same process which is used to bond PDMS to glass is performed. The reverse process also occurs, with air plasma treated polyurethane having a more hydrophilic state [9]. By subjecting the PDMS to air plasma at 800 mTorr for 30 seconds, a thin active glass-like layer is formed, decreasing the contact angle between the polyurethane prepolymer with the cured and treated PDMS. The NOA can be cured in this wetted state, resulting in a strong interface bond between PDMS and cured NOA. To measure the contact angle, 100 μl of NOA71 was placed on a hydrophilic and hydrophobic PDMS substrate. After waiting for the droplets to fully relax on the PDMS surface, the NOA71 was cured under UV illumination. Figure 2.4 shows the estimated reduction of 34 degrees in the contact angle between cured NOA with air plasma treated PDMS and hydrophobic PDMS

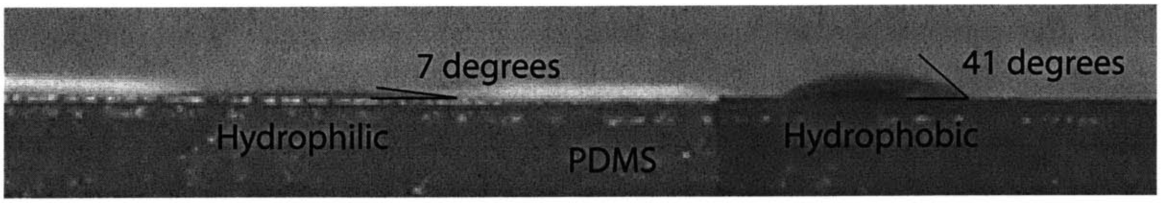

Figure 2.4. (Left) Cured NOA71 on a PDMS surface treated with air plasma. (Right) Cured NOA71 on a normal PDMS surface. A decrease of approximately 34 degrees is observed.

Also, since NOA71 contains adhesion promoters for glass surfaces, UV curing of the polyurethane on the glass-like PDMS surface creates an interface bond so strong that efforts to separate the two materials results in tearing of the PDMS.

In comparison with LS-6257, NOA71 does not require high temperatures to cure, and has no diffusion of internal components which lead to loss. While it is much harder and is also 100% solids, cured NOA71 still experiences 6% shrinkage as shown in Figure 2.5.

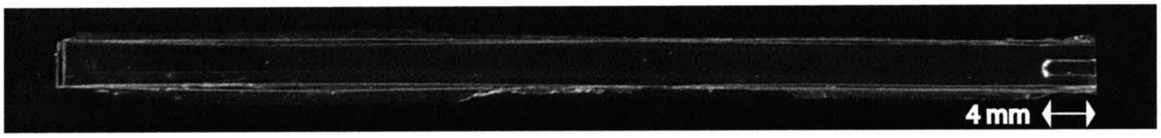

Figure 2.5. Picture of the shrinkage induced from curing. An originally filled channel with NOA71 prepolymer shrinks by 6% after curing.

Unlike LS-6257 which is thermally cured, NOA curing occurs under UV illumination, which can cause photobleaching for some sensors in which the chemical structure breaks down quickly under UV illumination. A more ideal polymer would be a high index polyurethane polymer which could be thermally cured at the same curing temperature as PDMS.

2.3 Fabrication

Fabrication of silicone based elastomers requires the creation of a mold, from which the elastomer will take its shape. Since the interest of the waveguides is in fluorescence signal collection, the dimensions used are on the order of millimeters which are quite large. Consequently, the SU-8 based lithographic fabrication methods conventionally used when patterning PDMS are not convenient. Instead, a different approach is taken, utilizing mechanical methods.

2.3.1 Surface Roughness

In order for waveguides to be machined, roughness acquired by the cutting process must be minimized. This can be accomplished through the proper adjustment of feed rates and the proper choice in a cutting tool. The ideal average roughness for milling a slot is given as

$$R_a = \frac{f_t^2}{32\left(R \pm f_t n_t / \pi\right)} \qquad 2.1$$

Where f_t is the feed rate per blade of the end mill, R is the radius of the cutter, and n_t is the number of teeth on the end mill [10].

R_a is defined as the average surface roughness given to be

$$R_a = \frac{1}{L}\sum_{n=1}^{L} |F[n]| \qquad 2.2$$

Which is the sum of the absolute value of all points in the roughness graph (F[n]) divided by the number of points L. From Equation 2.1, we see that the ideal roughness from milling is determined by the cutting tool and the cutting speed. For a 1 mm diameter cutting tool with 2 teeth spinning at 2800 rpm and feeding at 1 in/min, the R_a is calculated as 1.3 nm.

Although Equation 2.1 seems to imply that simply milling a slot can produce optical clarity, other factors tend to dominate the roughness. Vibrations from the motor can lead to increased roughness, material specific heating issues arise, and misaligned machine effects such as run-out can occur [10]. For example, polycarbonate starts to melt at high rpms and low feeding speeds. A major concern when milling smaller sized features is the effect caused by run-out, which results in a wavy surface due to the spindle not being held tightly by the rotating motor as illustrated in Figure 2.6.

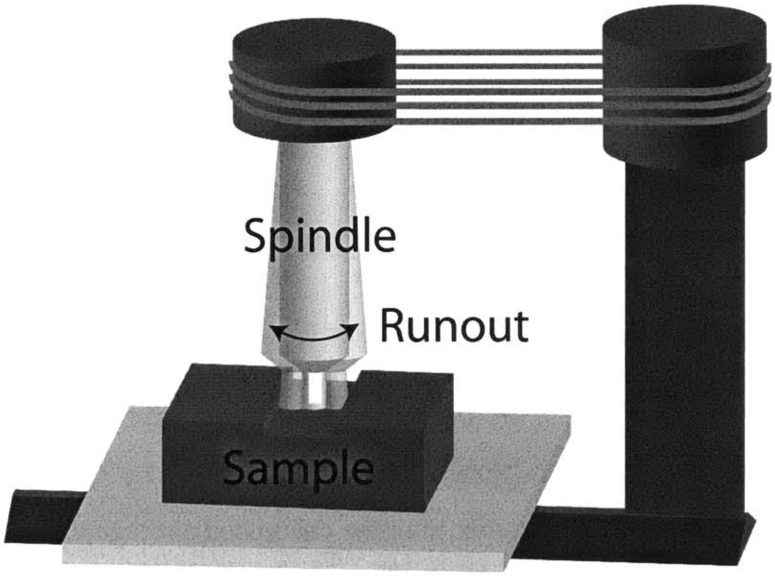

Figure 2.6. Illustration of a sample being milled with a machine suffering from runout. The milled channel becomes larger and rougher due to runout.

This effect is not dependent on the end mill diameter, so the roughness added by run-out becomes proportionally larger if smaller feature sizes are desired. For the motors used in the Sherline CNC mill, which is the machine used for devices fabricated in this thesis, the manufacturer specifies the run-out to be typically 5 μm with a maximum of 13 μm, which definitely causes problems for devices intended for use as optical components. While the effect can be reduced by using faster spindle speeds or by tightening the bearings holding the spindle in place, there is always a trade off with heat generation. In practice, roughness due to these non-ideal factors, including vibrations and run-out, can be over 100 times greater, as shown in Figure 2.7.

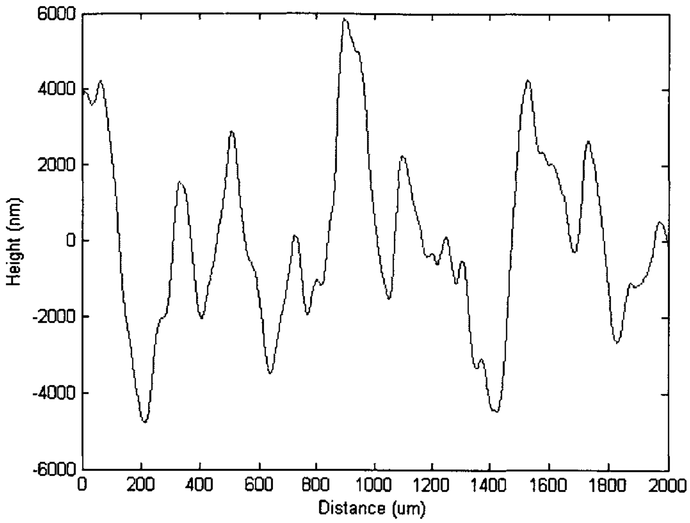

Figure 2.7. Surface profile trace using a Dektak surface profilometer for a 1 mm diameter 2 tooth cutting tool at 2800 rpm and a feed speed of 30 mm/min. The measured R_a is 1.9 μm while the calculated R_a is 1.8 nm.

As seen from Figure 2.7, the roughness average measured by the Dektak surface profilometer is 1.9 μm and is much larger than the calculated average value of 1.8 nm by Equation 2.1. The maximum roughness from the graph is most likely caused by a mixture of run-out and motor vibrations. By placing a dial indicator against the end mill and manually spinning the spindle, a maximum planar displacement of 100 μm is visible, indicating that the end mill is not centered in the spindle. A displacement of nearly 20 μm is also measurable by simply shaking the motor attached to the spindle. In comparison, the maximum planar displacement measured by the dial indicator incident on the spindle is less than 10 μm, which is within the manufacturer run-out specification. This suggests that the increased run-out is resulting from the collet and drawbar rather than the motor and spindle.

It should also be noted that an off-center collet and end mill does not necessarily generate greater roughness. If the dominant cutting blade cuts the surface at a displacement of exactly 100 μm for every cut, then it would appear as if the machine was perfectly on axis and was using a single bladed end mill. It is actually the variance in the displacement of the end mill that generates the majority of roughness. For this reason, the sidewall height for milling cuts is expected to be much less than the absolute displacement. At low feed speeds, it is difficult to see the effect of the off axis end mill since the variance is multiplied by the higher frequency of cuts per distance. For faster feeding speeds, the roughness is still dominated by the variance in run-out, but the effect of the single blade cut, which still occurs at the rotational frequency [10], becomes visible, with one blade of the end mill achieving maximum displacement and dominating the roughness.

$$d_{runout} = n_t f_t \qquad 2.3$$

An example of a faster feed speed profile trace is given in Figure 2.8. For an end mill spinning at 2800 RPM and moving at 200 mm/min, the calculated distance between variations in height due to dominant blade cuts is 71 μm. This periodic variance is visible in the measurement.

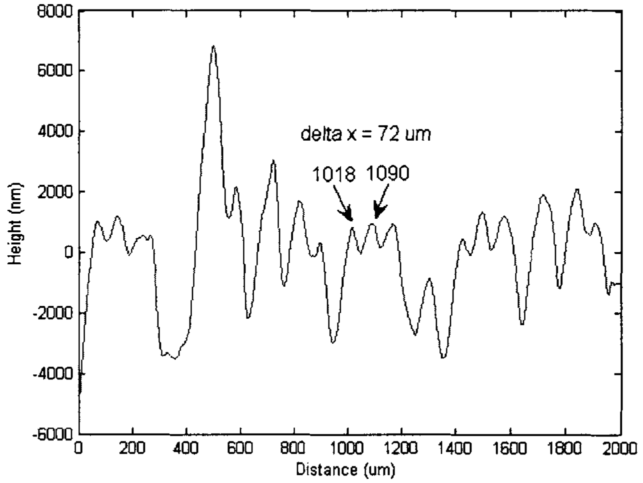

Figure 2.8. Surface profilometer trace of a milled surface with a 1 mm diameter 2 tooth end mill spinning at 2800 RPM and feeding at 200 mm/min. The spacing between little peaks in the trace shows a non-ideal artifact resulting from run-out.

While the little peaks are spaced at the end mill cutting frequency, it should be noted that this is only visible at faster feeding speeds because the number of cuts per distance is decreased, making single cuts more noticeable. The other factors mentioned above, such as displacement variance most likely due to motor vibrations, result in even larger end mill displacements. In comparing the slow and fast feed speed cuts, the roughness resulting from the slower spinning speed in Figure 2.7 does not seem to indicate that roughness results from a run-out dominant single blade, since run-out would occur with a period of 10μm.

2.3.2 Vapor Polishing

Since it is nearly impossible to control all of the factors contributing to surface roughness in machining parts, an effort must be made to improve surface roughness through post processing. The most common method utilized is mechanical polishing, which can achieve optical quality finishes. Unfortunately, this only works well for planar surfaces or large feature sizes. Non-planar topographies require local polishing which is both difficult and time-consuming for complicated structures.

There are also a variety of polishing methods that are material specific. For metals, there is electropolishing [33], which can remove roughness on a metallic surface through reverse electroplating. In this process, the rougher features on the surface experience large electric fields forcing metallic ions off of the surface. This process is much more capable of polishing non planar surfaces, but it is limited to conducting materials. Unfortunately, metallic materials are harder to machine than other materials such as plastics and a variety of chemicals are required in large supply to create a suitable bath for electropolishing.

For thermoplastics, machining is easier and there also exist methods of polishing that take advantage of thermoplastic reflowability. This type of "reflow polishing" melts the surface of the polymer so that the surface can reform in a planar fashion before rehardening. Two established polishing methods exist for thermoplastics, flame polishing, which takes advantage of the low melting point of thermoplastics, and vapor polishing, which takes advantage of the solubility of thermoplastics with certain solvents. Flame polishing can cause reflow in non-planar structures, but precision over the local substrate temperature is necessary to selectively reflow rough surfaces without damaging the mold, especially for thin molds. On the other hand, vapor polishing relies on diffusion of a solvent vapor into the substrate to cause reflow. This is a surface based process and can be controlled more easily than flame polishing. In both methods of reflow polishing, it should be noted that additional polymer material is never added or removed from the surface, therefore it is sometimes impossible to remove very gradual surface variations over long distances simply because it is impossible for flowable material to be transported long distances just by surface tension forces.

The vapor polishing process is widely used in industry for enhancing aesthetics of thermoplastic parts such as old telephone cases, or removing scratches in the lenses of glasses and hard paints on cars. For the purpose of making master molds, the polymer chosen is polycarbonate because through trials, it is affected by vapor polishing more than plastics such as PMMA. In order to vapor polish polycarbonate, a solvent must be chosen that is able to wet to the substrate and dissolve in between the polymer chains. The process for dissolving a solvent into a polymer is governed by the free energy equation in [11].

$$\Delta G = \Delta H - T\Delta S \tag{2.4}$$

For dissolving to occur, the process must occur spontaneously, which is equivalent to saying that the dissolving process gives off energy or $\Delta G < 0$. As stated in [11], dissolving increases the total entropy (ΔS) since the polymer is being changed from solid phase into a flowable liquid phase, so in order for the dissolving process to occur spontaneously, the total heat energy (ΔH) must be small or negative.

Since the heat energy for a dissolving process results from the mixing between the two components, ΔH can be written as

$$\Delta H = V_m \left(\sqrt{\frac{\Delta E_1}{V_1}} - \sqrt{\frac{\Delta E_2}{V_2}} \right)^2 \phi_1 \phi_2 \tag{2.5}$$

As given in [11] where V_m is the total volume of the mixture, ΔE is the energy of vaporization for each component, V is the molar volume of each component, and ϕ is the volume fraction of each component in the mixture. It should also be noted that $\sqrt{\Delta E / V}$ is a defined quantity called the solubility parameter.

Since the heat of mixing results in a positive quantity, the best way to ensure a spontaneous reaction is to minimize ΔH, which leads to the conclusion that the difference between the solubility parameters of polycarbonate and the solvent should be minimized. Solubility parameters for various plastics and solvents have been published [12], and from these, polycarbonate and dichloromethane are seen to have very similar solubility parameters of 9-10 and 9.7 $(\mathrm{cal/cm^3})^{1/2}$ respectively.

In addition to showing that a substrate can be dissolved by a solvent, the polishing process can be influenced by the wetting of the vapor to the substrate. In order for the vapor to wet to the substrate, the surface energy, or the strength of bonding between molecules on the surface, of the solvent vapor should be less than the surface energy of the substrate. With a larger substrate surface energy, solvent molecules at the surface of any solvent droplets in contact with the substrate will feel a greater attraction to the substrate molecules than to other solvent surface molecules, promoting spreading and wetting of the solvent to the substrate. Since the surface energies of polycarbonate and dichloromethane are 42 dynes/cm and 26.52 dynes/cm respectively [12], dichloromethane easily wets to polycarbonate.

The process explored for vapor polishing polycarbonate consists of exposing polycarbonate substrates to fixed vapor pressures of solvent vapors for a given exposure time. In order to attempt vapor polishing with methylene chloride (dichloromethane) gas, the chamber used must be able to maintain a constant vapor pressure either below or above atmospheric pressure and must also be made with materials which are chemically resistant to methylene chloride. There are handbooks both published and on-line which give tables that rates the chemical resistance for different materials qualitatively, generally with three rankings, either stable under exposure, unstable under certain conditions, or unstable. Appendix A gives a list of the chemical resistance of a variety of metals and plastics to polycarbonate specific solvents, taken from Burkert Fluid Control Systems.

Since we are aiming to create a chamber which can house methylene chloride vapor for short periods of time, it should be suitable to use materials which are stable as well as materials which are conditionally stable. This will make material selection much easier. In addition to chemical stability, the chamber must also be thermally conductive to prevent condensation of the vapor onto cold regions of the chamber. This limits the chamber to a variety of metals instead of chemically resistant plastics for good thermal conductivity coefficients. In comparison, brass has a thermal conductivity of 1.22 W/cm-K where as Teflon has a thermal conductivity of 0.002 W/cm-K. As stated in the MSDS for methylene chloride, methylene chloride is corrosive at high temperature and in the presence of water for iron, certain stainless steels, copper, and aluminum. In addition to chemical incompatibility, iron, stainless steel, and copper are all materials which are not easy to machine because of their hardness. Instead, the material we will use to create a pressure maintaining oven chamber is brass. From the chemical resistance guide, brass is seen as a conditionally stable material for methylene chloride, and in experimental tests, brass is seen to resist corrosion upon exposure to methylene chloride. In addition, other vacuum components such as fittings and valves are available in brass, allowing for compatibility between the chamber and the tubing interface components.

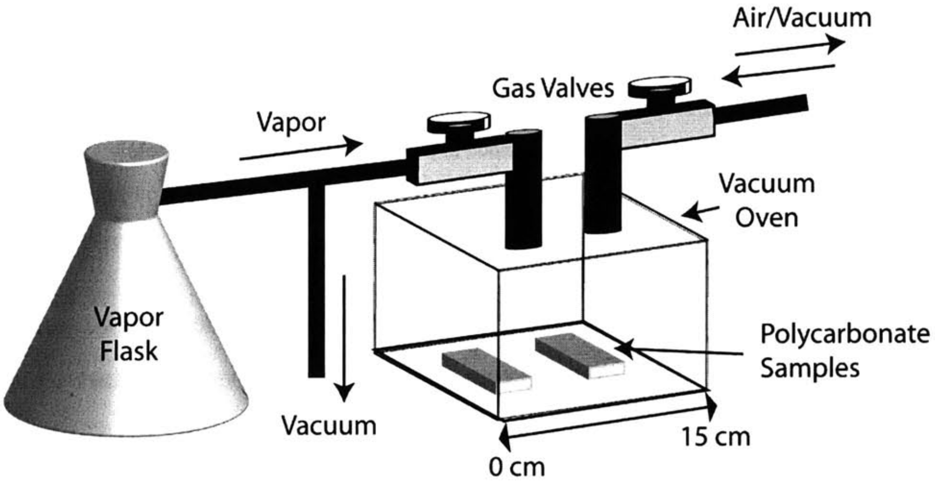

Figure 2.9. Schematic of the polishing chamber setup. The flask containing solvent vapor is heated separately from the polishing chamber. Both vapor pressure and substrate temperature affect the polishing quality of the sample.

To find optimal polishing conditions, polycarbonate samples are sanded with 400 grit sandpaper and placed into a polishing chamber shown in Figure 2.9 [13]. 400 grit sandpaper is chosen since it provides a roughness average of approximately 1 μm, which is similar to roughness generated by the milling process. Samples in the chamber are heated to 36° C and the chamber is evacuated to below atmospheric pressure by at least the desired pressure of solvent vapor. This allows for easy backfilling of solvent vapor from the flask into the chamber. Solvent vapor from a heated flask is then backfilled to the desired pressure, and the remaining difference between chamber pressure and atmospheric pressure is filled by dry air or nitrogen. Through experimentation, nitrogen is preferred, since ambient air often results in clouded surfaces, possibly due to humidity. Samples are exposed to this mixture for 3 minutes before air is flowed through the chamber to vent out the solvent vapor.

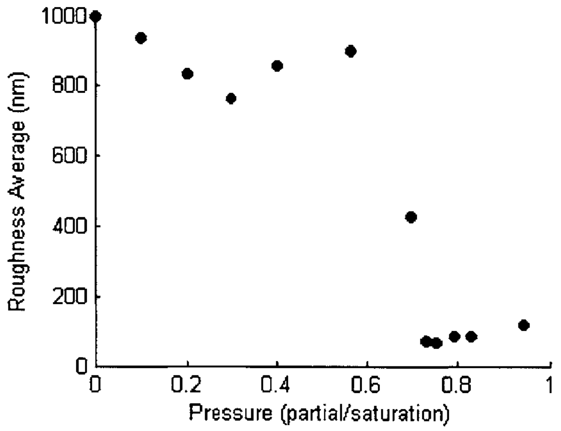

Figure 2.10. A plot of the surface roughness for 400 grit dry sanded polycarbonate samples exposed to methylene chloride vapor at different pressures for 3 minutes. Variance in the roughness for pressures below 50% is due to uneven sanding and not polishing. A minimum occurs in the surface roughness near 75% saturation pressure for a substrate temperature of 36° C and a 3 minute exposure.

The plot given in Figure 2.10 shows an ideal pressure range where maximum polishing occurs for the processing conditions given above. Vapor does not start to affect the substrate until after 60% of the saturation pressure for a 36° C substrate.

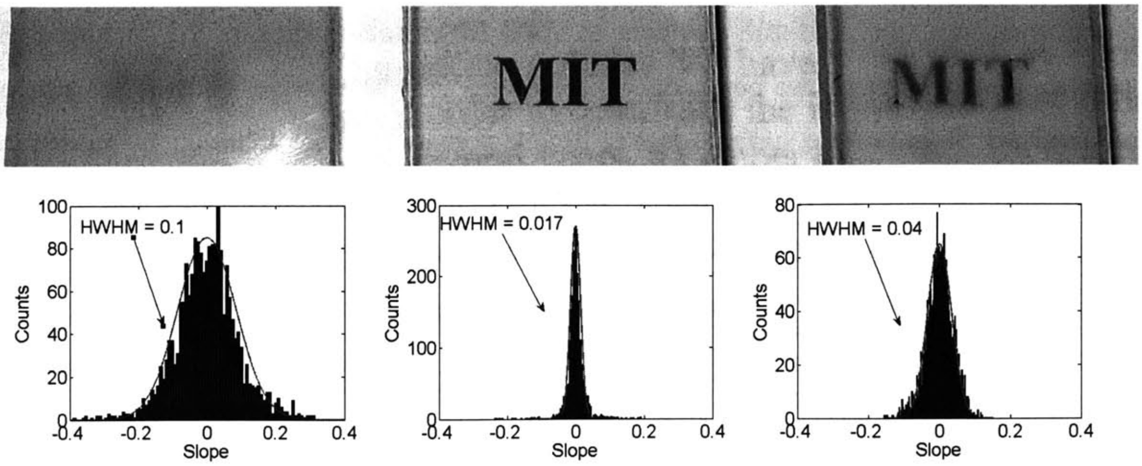

Figure 2.11. Polished polycarbonate samples under different polishing conditions and the associated histograms of the slope distribution seen at the surface. Left) 10% saturation pressure results in no polishing. Middle) 75% saturation pressure results in excellent polishing. Right) 95% saturation pressure results in over polishing and is seen as a developing haze. Samples are still very smooth when over polished but optical quality decreases.

While haziness starts to affect the optical quality of the polycarbonate at higher partial pressures, the effect can be reduced by exposing for shorter time, which is equivalent to reducing the amount of polishing. Figure 2.11 shows examples of polycarbonate samples polished under low, ideal, and high pressure.

2.3.3 Chip Fabrication

Since the focus is on fabricating optical devices in a PDMS platform, negative molds are chosen to be fabricated out of polycarbonate. By cutting negative molds out of polycarbonate, we can use the optically polished side of the polycarbonate as the base of the waveguide. While waveguides can still be fabricated using positive molds and cutting channels, loss can be reduced by taking advantage of the fact that polycarbonate sheets are pre-polished to optical quality. Both fabrication methods will still be used to compare the differences between waveguides fabricated using positive and negative molding processes.

Waveguides are cut out of polycarbonate using a CNC mill from Sherline Products with a motor step size of 3.175 μm in x, y, and z directions. Polycarbonate is cut with a 1 mm diameter square steel end-mill spinning at 2800 RPM and feeding at 1.2 inch/min. In general, a second pass can be made with the end mill to remove 100 to 200 μm of material and improve the surface quality before polishing. After fabrication, molds are placed into the vapor polishing chamber described in Section 2.3.2 to further improve the surface finish of the milled surfaces. Molds are polished at 36° C for 150 seconds at 493 torr Methylene Chloride vapor and 267 torr dry Nitrogen gas. After vapor polishing, the molds are baked in an oven at 70° C until the solvent vapor is removed from the surface of the polycarbonate.

After molds are created in polycarbonate, a well established PDMS procedure is utilized to create either positive or negative PDMS replicas of the molds [14]. PDMS is poured into the mold to replicate the reverse of the mold. The mold with PDMS is then placed into an oven at 70° C for 4 hours to harden the PDMS. For PDMS molds that are made from negative polycarbonate molds, the cured PDMS is the desired chip. For PDMS molds that are made from positive polycarbonate molds, a non-stick surface treatment must be applied to the PDMS mold. This is done by placing the PDMS mold in a vacuum chamber with a few drops of (tridecafluoro-1,1,2,2-tetrahydrooctyl)-1-trichlorosilane for 4 hours [16]. After the treatment, PDMS is again poured into the negative PDMS mold and cured at 70° C for 4 hours to make the desired chip. In general this process is done for two layers of the device and then air plasma bonding or partial cure bonding is utilized to bind the two halves into a single chip. Plasma bonding results in hydrophilic surfaces while partial cure bonding results in hydrophobic surfaces. An overview of the process for fabricating an all PDMS open ended channel using this method is shown in Figure 2.12.

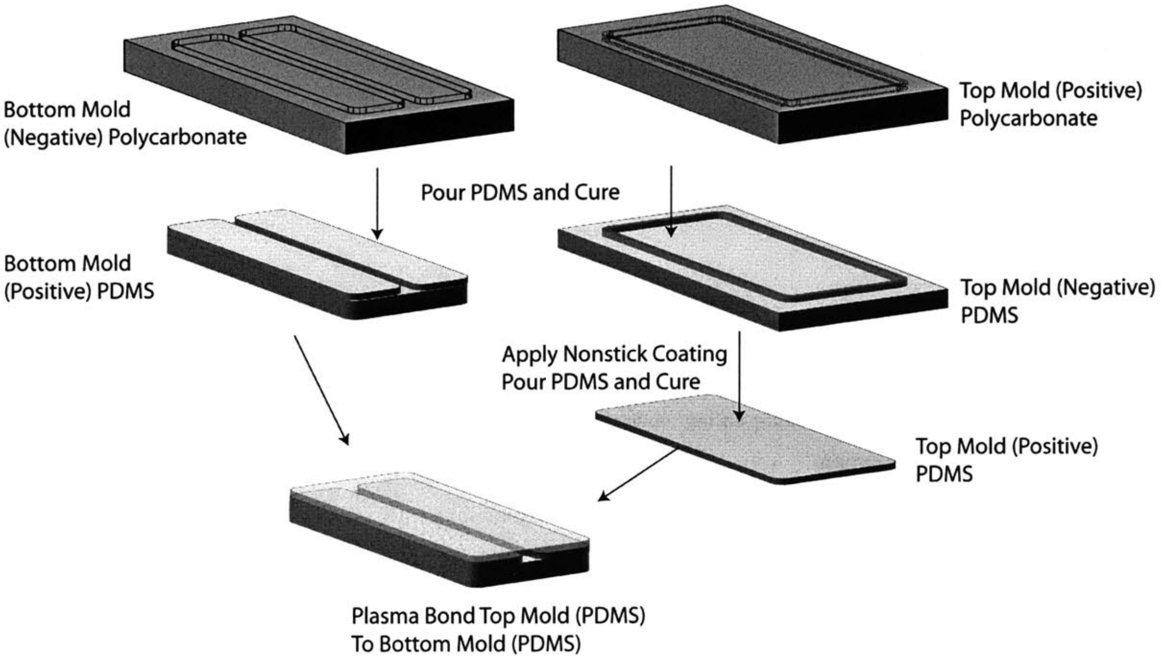

Figure 2.12. Fabrication process overview for creating a PDMS channel from a polycarbonate master mold. The process on the left is for negative polycarbonate molds and the process on the right is for positive polycarbonate molds.

After channel fabrication, a core material NOA71 is injected through the channel and cured under a 380 nm UV light for 4 hours or longer to reach the desired hardness required. This can be accomplished by direct injection with a syringe, or by placing the mold into a bath of NOA71 and pulling vacuum, similar to the MIMIC process [15]. In addition, NOA71 can be heated to reduce its viscosity and decrease filling time. If the channel is not treated with oxygen plasma before curing the core, the core will delaminate from the cladding under mild strain during cure shrinkage as shown in Figure 2.13. Therefore, it is recommended to bond the two halves on the channel with air plasma rather than with partial cure bonding.

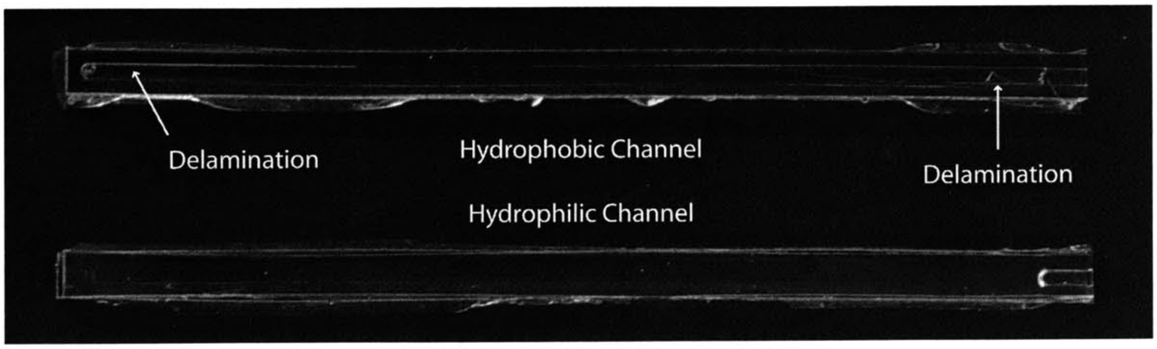

Figure 2.13. Picture comparing curing in a hydrophobic channel and curing in a hydrophilic channel. Delamination of the core from the cladding results during curing since the polyurethane is not attracted to the PDMS surface and the polyurethane shrinks during curing.

Since the fabricated device is polymer based, efforts to cleave the end faces of the waveguides lead to deformed and rough interfaces. In order to improve coupling into and out of the waveguides, two methods can be used. The first is to cleave and then apply a flat layer of core material to the end faces. After curing the extra core material, the initial rough surface is indexed matched to a smooth surface, leaving a flat and optically smooth waveguide end. The second method is to close the ends of the waveguides with a flat nonstick layer of PDMS during initial cure or with a thin glass layer for protection. While the first method creates a smooth but slightly lensed surface, it is an easier process to use, and can also be used to repair broken waveguide ends or poor cleaves.

Process Step	Time Frame
Polycarbonate Milling	2 hours
Vapor Polishing and Drying	0.5 hours
Curing negative PDMS mold	4 hours
Non-stick surface treatment	4 hours
Curing positive PDMS mold	4 hours
Air Plasma Bonding	0.5 hours
Core injection and curing	4 hours
Total	19 hours

Table 2.1. Summary of the fabrication processes required to create a device and the time frame required for each step.

A summary of the process steps for positive molds and the time frame necessary for full prototyping is given in Table 2.1. It is clear that the entire fabrication process takes under a day to complete for waveguides and also requires only cheap materials.

2.4 Results and Measurements

Channels created from positive polycarbonate molds are expected to be lossier since tool marks occur on three out of four sides in a given cross section of the waveguide. Therefore it is more beneficial to use negative molds when creating waveguides. Important parameters for the waveguides are the surface roughness, waveguide loss, and numerical aperture.

Both waveguides are fabricated with a 1 mm diameter end mill at 2800 RPM and a feed speed of 30 mm/min. A finish pass is taken on each side to reduce roughness.

2.4.1 Waveguide Loss

As shown from Figure 2.3, material losses are small for NOA71 in the visible wavelengths. Therefore, most losses should result from impurities or roughness during fabrication. These losses should be uniform since roughness and impurities per distance should be uniform and random. The waveguide loss is measured using a two step process. The first step involves calibrating the loss at a given wavelength. A monochromatic LED is coupled through the waveguide and a CCD camera is used to take a picture of the scattered light intensity escaping through the core cladding interface. By plotting the intensity versus distance, an exponential curve can be fit to determine the waveguide loss. After the loss coefficient is known for a single wavelength, the LED is replaced with a broad band source and a spectrometer is coupled directly to the output face of the waveguide through a multimode optical fiber to get the relative light intensity versus wavelength. While light from the cladding does appear at the output face, the diameter of the fiber used for collection is smaller than the waveguide cross-section, so the majority of light propagating within the cladding is not collected by the collection fiber. By calibrating the curve measured by the spectrometer using the loss data determined for a single wavelength, a plot of loss versus wavelength can be generated for a waveguide. A schematic of the measurement setup is given in Figure 2.14.

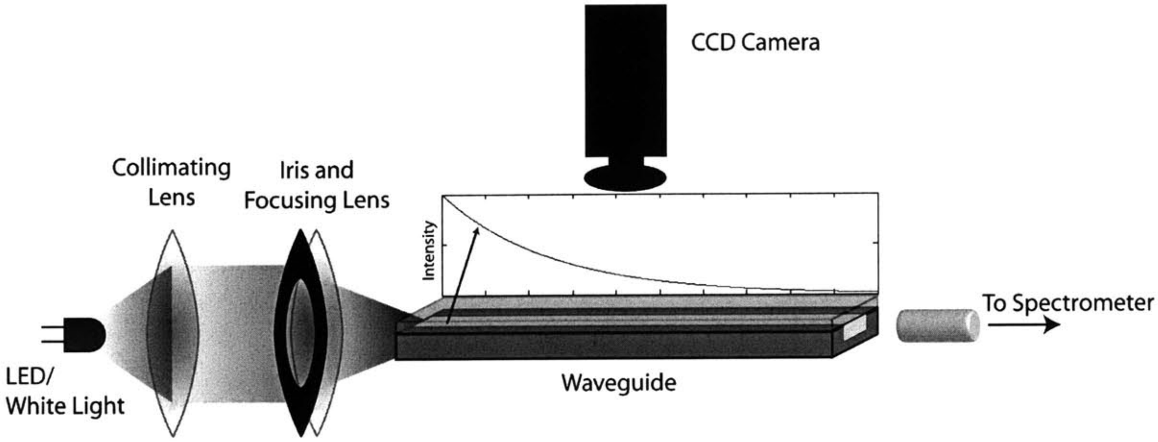

Figure 2.14. A schematic of the measurement setup used for measuring waveguide loss for straight waveguides. An LED is focused into the waveguide and scattered light intensity is measured by the CCD camera. Afterwards the LED is replaced by a white light source and the waveguide output is coupled to a spectrometer.

Since there are two ways to make waveguides, using positive molds or negative molds, waveguides fabricated using both process should be tested. For both positive and negative molds, waveguides are cut to a 1.2x1 mm^2 cross section and are 70 mm in length. Loss plots for both waveguides are shown in Figure 2.15.

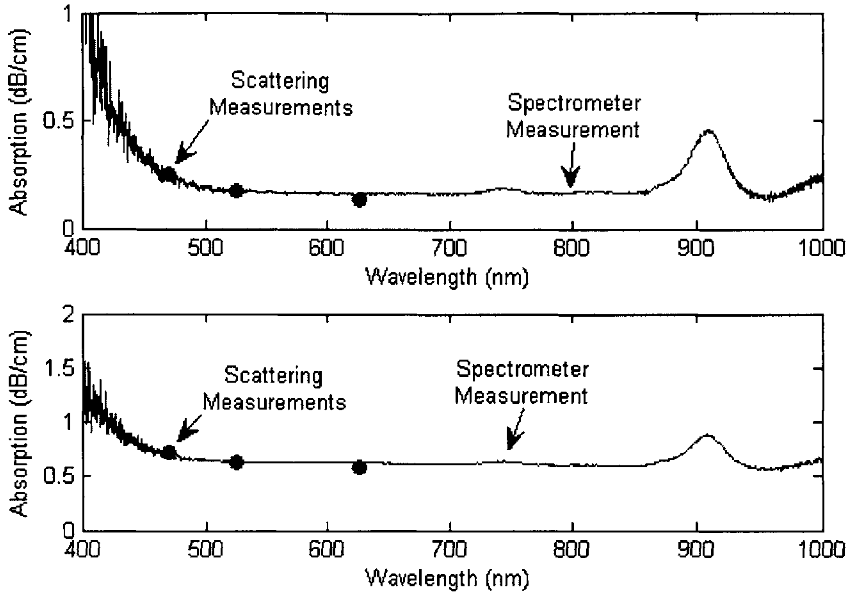

Figure 2.15. A plot of loss vs wavelength for a 1.2x1 mm^2 cross section and 70 mm long waveguide machined with negative mold procedures (top) and positive mold procedures (bottom). Dots on the plot correspond to loss points measured for different wavelengths with the CCD camera while the line measures the loss of the waveguide through a spectrometer.

From Figure 2.15, it is evident that waveguides fabricated with positive molds are lossier than waveguides fabricated with negative molds. At 525 nm excitation, the loss coefficients for the positive and negative molded waveguides are measured to be 0.63 dB/cm and 0.18 dB/cm respectively. While the loss experienced by the positively molded waveguide is higher, the increased loss due to the extra roughness stays relatively independent of wavelength as shown in Figure 2.16.

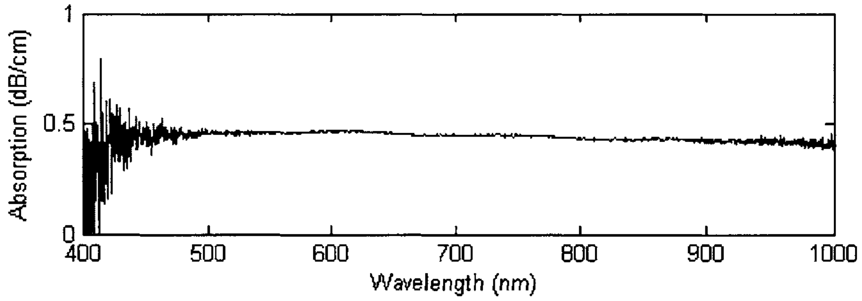

Figure 2.16. Increased absorption of the positively molded waveguide in comparison to the negatively molded waveguide. While the absorption coefficient is increased, it remains relatively independent of wavelength.

2.4.2 Numerical Aperture

Numerical aperture for large core waveguides should depend only on the index contrast between the core and the cladding of the waveguide and the refractive index of the external media, which in most cases is air, as discussed in Section 1.2. By determining the critical angle between the core and the cladding, the maximum input and output angles can be determined. From the refractive indices given for PDMS and NOA71, a numerical aperture of 0.62 is calculated.

To measure the numerical aperture of the waveguide, we place a rotational stage centered at the waveguide output and position a power meter a given distance away from the waveguide output. By measuring the output intensity with respect to angle, the numerical aperture can be determined for the waveguide. The measurement setup is shown in Figure 2.17. In addition, the input aperture can be varied by changing the beam diameter incident on the focusing lens so that the entire input numerical aperture of the waveguide can be filled.

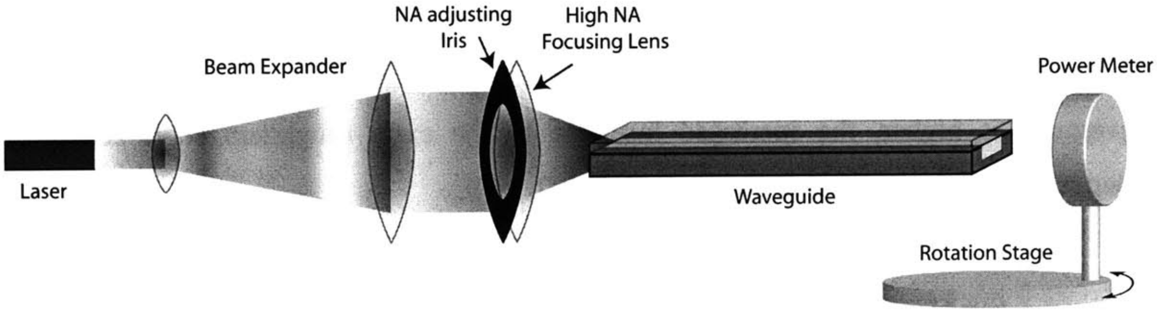

Figure 2.17 Measurement setup for determining the numerical aperture for a straight waveguide. An HeNe laser is coupled into the waveguide through a lens system and a power meter mounted on a rotation stage measures the output intensity at different angles.

Plots of the output intensity versus angle for both the positive and negative molded waveguides are shown in Figure 2.18. It is clear that there is attenuation of the higher order modes which is due to surface roughness for both waveguides. The negative molded waveguide results in a 50% intensity numerical aperture of 0.5, while the positive molded waveguide results in a 50% intensity numerical aperture of 0.27. Since the only difference between the two waveguides is that the positive molded guide contains a rough bottom surface, the extra reduction in numerical aperture for the positive molded guide must be due to the extra rough surface. Similar experiments at IBM with epoxy waveguides also demonstrate a decrease in numerical aperture for large input NA due to mode dependent loss by waveguide imperfections [17] as discussed in Section 1.2.

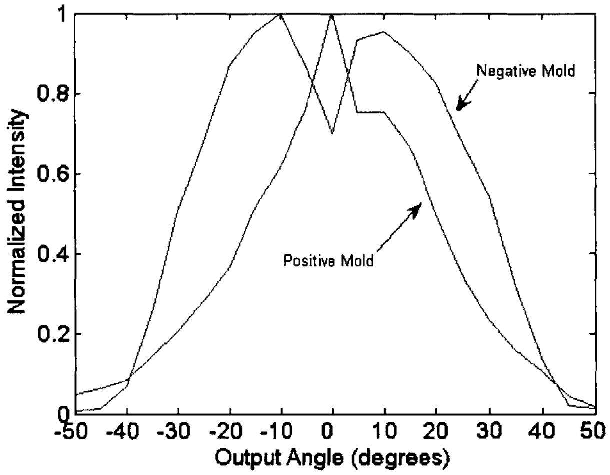

Figure 2.18 The numerical aperture versus output angle is shown for the fabricated positive and negative NOA71 waveguides of 70 mm. The intensity is taken in the plane of the milled surfaces. The 50% intensity numerical apertures for the positive and negative molded waveguides are 0.27 and 0.5 respectively.

Since for waveguides, modes translate into output angles, if there is indeed mode dependent loss due to the surface roughness, then the rate of loss over distance should increase as the output angle increases. This effect was shown in [17], with a smaller loss coefficient for a smaller input aperture. Measurements on the positive molded waveguide show a change in loss from 0.58 dB/cm at an input aperture of 38 degrees to 0.09 dB/cm at an input numerical aperture of 7 degrees.

Since large perturbations in surface smoothness resulting from the fabrication process can not be removed by the polishing procedure without over polishing the device, the mode dependent attenuation should be characterized in order to account for such problems during design of devices. This will be accomplished in Chapter 3.

2.4.3 Surface Roughness

PDMS molds replicated out of cut polycarbonate molds are measured using a Dektak surface profilometer. For positive polycarbonate molds, profile measurements will be taken for the side walls as well as the bottom to determine the degree of scalping caused by the end mill. For negative polycarbonate molds, profile measurements will be taken for just the side walls since the bottom surface is assumed to already be of optical quality.

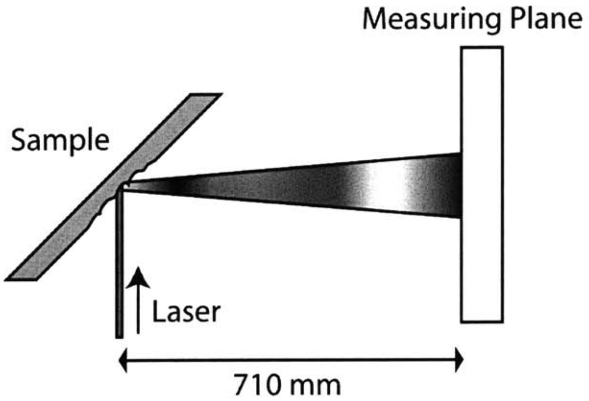

Figure 2.19. Illustration of the measurement setup used to determine the beam divergence caused by vapor polished sample roughness.

In addition, measurements can be taken by reflecting a collimated laser beam off the surface at 45 degrees and looking at the beam spread due to the variations on the surface of the machined part as shown in Figure 2.19.

By measuring the intensity distribution versus angle at the output, a comparison can be made between the measured beam spread of the laser and the measured slope distribution from the profilometer. For waveguides fabricated with a 1 mm diameter end mill at a feed speed of 30 mm/min and a spindle speed of 2800 RPM, measurements taken on the side wall roughness using both measurement techniques yield a half width half max of approximately 0.01.

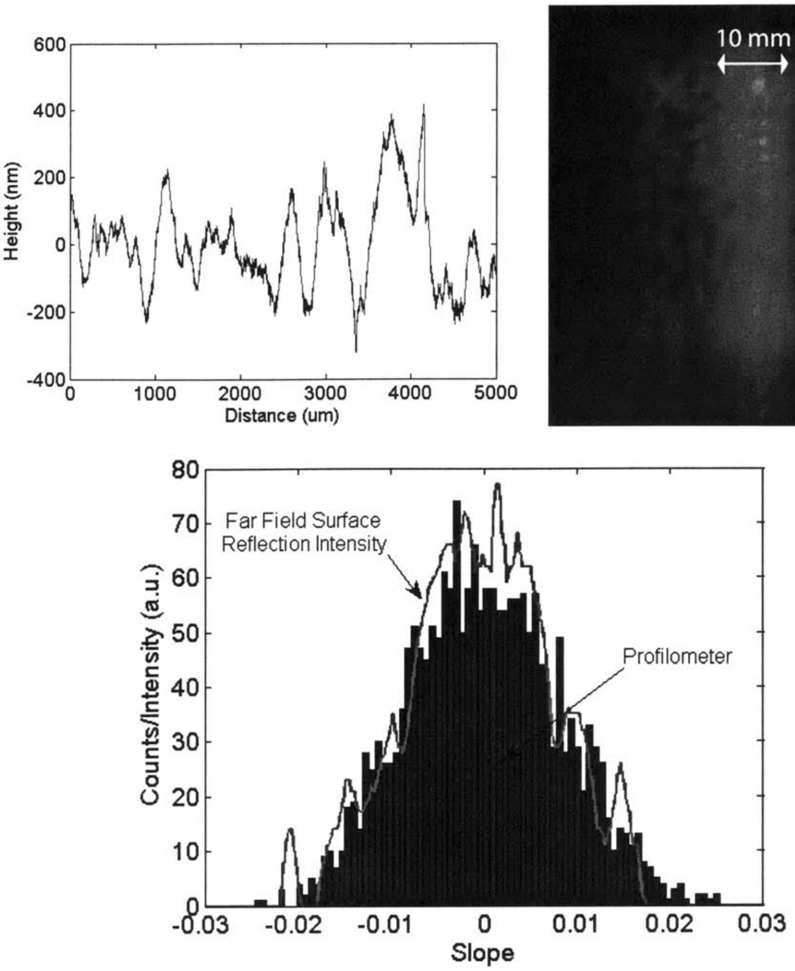

Figure 2.20. Slope distribution for the side wall of a vapor polished waveguide milled with a 1 mm diameter end mill at 30 mm/min and 2800 RPM. A profilometer trace of the surface (Top Left) and a beam divergence measurement (Top Right) are taken. Both measurement techniques yield the same slope distribution (Bottom).

From the beam spread measurement in Figure 2.20, it is clear that for ideally vapor polished surfaces, there is little uniform diffuse reflection, and instead, the roughness slope distribution encountered by an optical beam reflecting off a vapor polished surface directly translates into a beam spread distribution. It is important to note that this implies that vapor polished surfaces are only optically clear, not optically flat. This information will be very useful for when we attempt to model roughness in Chapter 3.

2.5 Summary

A fabrication process has been presented which utilizes polymer materials and CNC machining to prototype polymer based fluidic and optical devices. By applying controlled vapor polishing to machined parts, the quality of surfaces can be enhanced to a degree where even optical signals can be guided. The fabrication process from start to finish takes under 24 hours and is suitable for making devices which either need embedded waveguides, or optically clear surfaces. Although waveguide fabrication with CNC milling has been demonstrated, losses of 1 dB/cm result mainly from surface roughness in the milling process [7]. While the fabrication approach combining CNC machining and vapor polishing is well established [12] and is able to remove most of the milling induced roughness, its application towards waveguide fabrication has never been explored. By utilizing the polishing process for waveguide fabrication, waveguide losses are shown to be reduced to as low as 0.18 dB/cm at 525 nm.

A straight waveguide device fabricated using this process behaves like epoxy waveguides studied previously [17], demonstrating a reduced output numerical aperture for large input apertures. Waveguides fabricated using this process also exhibit low loss, with an increase at lower wavelengths due mainly to Raleigh scattering from impurities within the waveguide cores. Measurements on the surface roughness of these waveguides also show that roughness translates into a beam divergence for reflected optical beams. This is also evidence that waveguide surface roughness on optically clear surfaces translates into mode dependent losses.

Chapter 3

Ray Tracing

3.1 Introduction

A ray tracing program's main advantage is its ability to approximate optical behavior in optically macroscopic systems. This set of systems can include free space optics and multimode waveguides. For most free space optical systems, there is a desired optical path that the user specifies for tracing rays. Rays that stray from this optical path are considered to be blocked in the system. This method of ray tracing is called sequential ray tracing and requires little need for intersection tests between rays and surfaces [18]. Since there is no confusion in the sequence of surface intersections, simulations can be carried out quickly. For other systems where the exact optical path is unknown, such as certain types of prisms or waveguides, a method known as non-sequential ray tracing must be implemented. In this method, every surface in the system must be checked in order to determine which surface will be intersected next. Also since the desired path is unknown, optical effects such as total internal reflection and partial transmission through interfaces must be accounted for. For very large systems with many surfaces, this process can greatly slow down a ray tracing simulation. Unfortunately, for rays propagating in a multimode waveguide, this non-sequential method is necessary throughout the system in order to accurately describe behavior.

For waveguides, it is often more accurate to talk about modes rather than rays. While modes are easily modeled for simple single or slightly multimode systems such as optical fibers or slab waveguides, it is hard to account for non-ideal behavior such as macroscopically rough surfaces and bends. For waveguides of interest in fluorescence applications, waveguide dimensions need to have 1x1 mm^2 cross-sections to collect adequate signal intensity. To estimate the number of modes in the system, the two dimensional cross-section can be treated as a superposition of two 1-dimensional symmetric slab waveguides where each guide has a width of 1 mm. The modes of a 1-D symmetric slab waveguide can be solved exactly, and the number of modes allowed by the waveguide can be determined by solving for the number of plane waves which form standing wave patterns within the waveguide. A derivation of the condition for modes in a 1-D symmetric dielectric waveguide is given in [35] and concludes that the number of TE modes is given by

$$M_{1-D} = \frac{2d}{\lambda}\sqrt{n_1^2 - n_2^2} \qquad 3.1$$

where n_1 is the core refractive index, n_2 is the cladding refractive index, d is the width of the core, and λ is the wavelength of light.

If this equation is solved for a core index of 1.56 and a cladding index of 1.43, 4156 guided TE and TM modes result at 600 nm. Also since the actual guide has a square cross-section, the superposition of slab waveguide modes along two orthogonal directions would result in a total number of modes proportional to M^2. A better estimate from [35], results in an equation for the number of modes in a square dielectric waveguide as

$$M_{2-D} \approx \frac{\pi}{4} M_{1-D}^2 \qquad 3.2$$

which results in over 13.5 million modes.

While a mode based simulation is still feasible with 13.5 million modes, if roughness indeed leads to mode dependent loss as well as mode mixing as described in Section 2.4, then the simulation would become very time consuming. The simulation would now require 13.5 million loss coefficients and the factorial of 13.5 million coupling coefficients in order to determine how mode mixing and mode dependent loss affect propagation in a waveguide with rough sidewalls. If an even more complicated structure was necessary, then that enormous number of variables would need to be calculated for every piece of the structure.

Ray tracing is an alternative method which can be used to approximate behavior within waveguide systems, provided that the waveguides are heavily multimode. In this case, the discrete nature of the propagation vectors of different modes within the waveguide becomes negligible, and a random selection of angles accurately approximates modes of the system. Since the focus of this research is on designing high numerical aperture waveguides, we are easily in this regime. For example, the 1-D waveguide suggested above contains its 4156 modes at 600 nm within a span of 23.6 degrees. In this case, rays with randomly generated angles will most likely approximate a mode of the waveguide system.

By designing a ray tracer for waveguides, non-ideal behavior such as scattering and material loss can be modeled statistically to determine if measured data can be explained by suspected nonideal factors. By statistically modeling propagation, ray tracing simulations can be performed much faster than mode solving simulations. Chapter 3 will introduce an algorithm for ray tracing which is suitable for simple waveguide structures. After developing a fundamental algorithm for tracing rays through waveguide structures, an approach will be discussed for modeling macroscopic roughness within the ray tracing system. Simulations with surface roughness modeled will then be performed in Section 3.3 and compared to measurements in Chapter 2 to determine how surface roughness affects waveguide performance.

3.2 Algorithms and Implementation

3.2.1 Ray Tracer Algorithm

For the particular simulations planned, the ray tracer needed must be able to handle waveguide structures and prisms. The implementation must also handle non-symmetric surface roughness as well as out of plane bends. For this reason, the ray tracer is developed to simulate three dimensional structures.

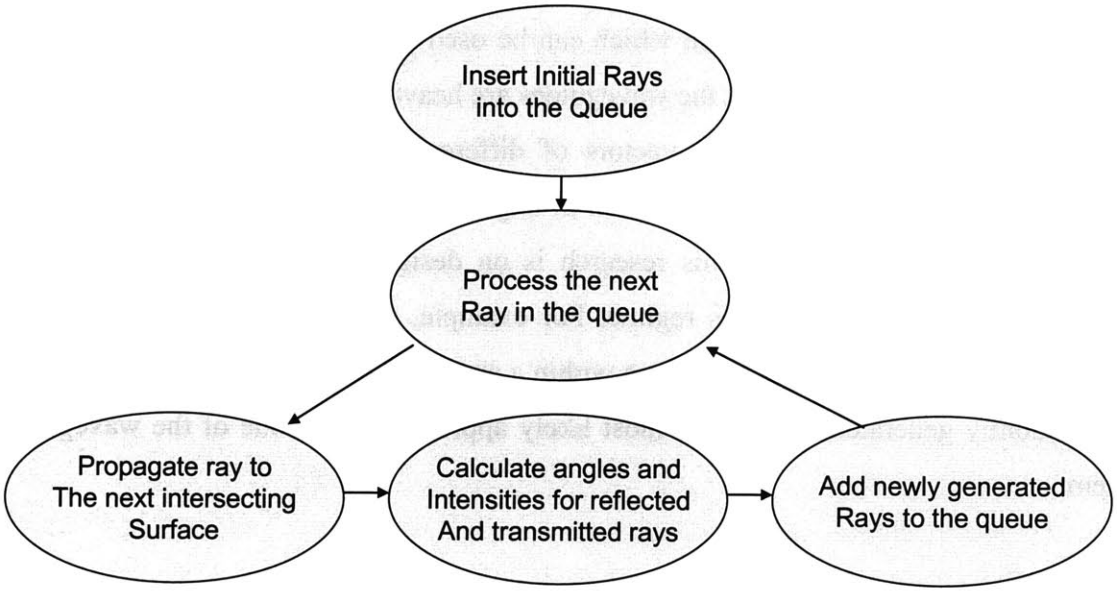
Insert Initial Rays into the Queue
Process the next Ray in the queue
Propagate ray to The next intersecting Surface
Calculate angles and Intensities for reflected And transmitted rays
Add newly generated Rays to the queue

Figure 3.1. Shows the general algorithm for performing a ray trace in an arbitrary structure. This type of algorithm may not converge for arbitrary boundary conditions.

The basic algorithm for the ray tracer is given in Figure 3.1 with a detailed version of the code in Appendix C. It is clear from the implementation that the simulation's convergence can not be estimated for any arbitrary initial conditions or structure designs since new rays are generated from within the computation loop. Therefore, there are two stopping conditions specified by the user when the program is initiated, one of which is a lower bound on allowed ray intensity and the other which is an upper bound on the number of bounces an initial ray can incur before termination. Of all steps required, most time is spent determining how to propagate a ray to the next intersecting surface. The more complicated the structure, the longer it will take to determine which plane is intersected in the shortest distance.

3.2.2 Data Types and Definitions

In order to run the ray tracing algorithm, initial rays must be generated and planes defining the interfaces of the structure must be traced. For this implementation, since a single ray can interact with a boundary to produce both a reflected and refracted ray, a ray is defined as a single line segment intersected at its ends by two bound surfaces. Other implementations that track a ray through its entire propagation path are not suitable since single rays can result in multiple propagation paths. In this way there is no need to keep track of all propagated paths for a single ray during computation. Planes are defined as bound two dimensional surfaces within the context of the program. By only defining the interfaces of the system, it is difficult to consider effects such as graded index materials or materials with birefringence, both of which are related to properties of three dimensional structures. However, since analysis will be performed on very large symmetric structures consisting of amorphous polymers, higher order effects related to inhomogeneous material properties are assumed to be minimal. A summary of parameters is given in Figure 3.2.

Rays	Planes
Source Ray	Plane Number
Bounces Acquired	Plane Type
Starting Point	Boundary Index 1
Direction	Boundary Index 2
Wavelength	Normal Vector
Intensity	Intercept Point
Polarization	Boundary/End Points
Refractive Index	Scattering Directions
Time of Flight	Surface Variances

Figure 3.2. Summarizes the basic variables and attributes that need to be considered in order to correctly perform a ray tracing simulation.

For rays, the polarization variable allows a ray to be considered TE or TM polarized. For non-polarized sources, the value for polarization is chosen randomly as either TE or TM. The main purpose of defining the polarization for a ray is to consider the different Fresnel reflection and transmission coefficients for different polarizations. The time of flight variable is used to determine an estimate for the multimode distortion due to the different angular distribution of rays.

For planes, the declaration of the plane properties is simplified to optimize for intersection comparisons. For simple structures, the boundary conditions for surfaces can be checked by faster methods than boundary conditions of arbitrary structures. This will be discussed in the next section. To account for scattering, planes are also defined with roughness parameters, one along each parallel axis of the surface, as specified by the user. Roughness by this model is described by a variance in the slope of the plane encountered by a ray intersecting the surface. Therefore, an equivalent variance parameter must be specified for each parallel axis of the surface. Measurements of this roughness have been discussed previously in Section 2.4.

3.2.3 Plane Intersections and Boundaries

After the planes and rays are defined, the simulation needs to trace through the system to determine where traveling rays will hit boundaries. This is accomplished by finding the closest intersection between a ray and the defined structure. For an incident ray on a plane, the angle of incidence and the distance the ray must travel to reach the plane are the two important parameters. The equations for a plane and a ray in 3-D space are given in equations 3.3 and 3.4.

$$\langle n_x \quad n_y \quad n_z \rangle \bullet \langle v_x \quad v_y \quad v_z \rangle = c \qquad 3.3$$

$$\langle v_x \quad v_y \quad v_z \rangle = \langle p_x \quad p_y \quad p_z \rangle + m \langle d_x \quad d_y \quad d_z \rangle \qquad 3.4$$

For the plane, c determines the allowed length of all rays $\vec{v}$ when projected onto the normal vector $\vec{n}$ of the plane as shown in Figure 3.3. For the ray, m determines the length of the ray that starts at the point given by vector $\vec{p}$ and points in direction $\vec{d}$.

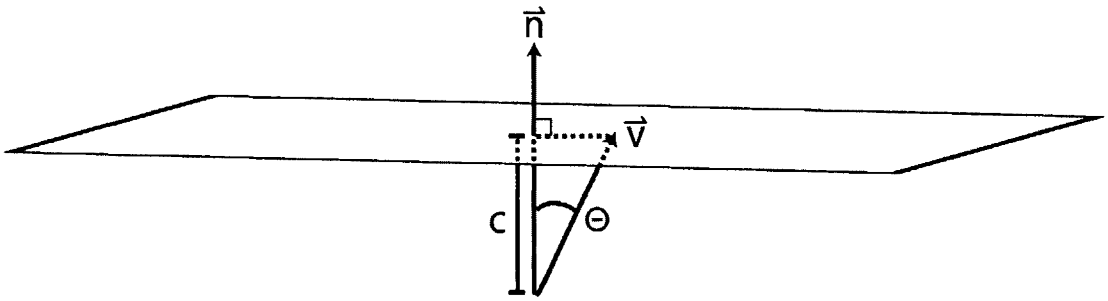

Figure 3.3. Illustration of the plane equation. Only vectors v that have a length c when projected onto the plane normal vector n are allowed.

In general, given any vector equation of a surface, setting ray $\vec{v}$ from the ray equation equal to ray $\vec{v}$ in the surface equation allows you to solve for the intersection point between the ray and the surface. An example of this is shown for the plane-ray intersection. By solving for m, we get the resulting Equation 3.5 where m is the distance from the ray to the plane.

$$m = \frac{c - \langle n_x \quad n_y \quad n_z \rangle \bullet \langle p_x \quad p_y \quad p_z \rangle}{\langle n_x \quad n_y \quad n_z \rangle \bullet \langle d_x \quad d_y \quad d_z \rangle} \qquad 3.5$$

This equation can be explained easily. We can see that the numerator of Equation 3.5 is the distance of a line drawn perpendicular to the plane which also intersects the starting point $\vec{p}$ of the ray. Then if $\vec{d}$ is a unit of travel distance for the ray, the denominator is a unit of travel distance for the ray if it was moving on the normal vector $\vec{n}$. From this we see that m is the amount of travel distance necessary for the ray to intersect the plane.

If we find the travel distance m between the ray and every plane in the system, we can determine where the ray intersects each plane. Unfortunately, to determine which surface the ray intersects first, we need to know not only where the ray intersects each plane, but also if that point of intersection lies inside or outside of the surface that exists on that plane. To accomplish this, two methods for implementing boundary conditions are used. The first method is for on-axis planes which are rectangular in shape. Since these planes are always parallel to two axes, their boundaries can be checked easily by checking if each of the three coordinates of the ray $\bar{v}$ lies between the minimum and maximum of the plane's bounding points.

$$\begin{aligned}\min\left(bounds_x\right) &\le v_x \le \max\left(bounds_x\right)\\ \min\left(bounds_y\right) &\le v_y \le \max\left(bounds_y\right)\\ \min\left(bounds_z\right) &\le v_z \le \max\left(bounds_z\right)\end{aligned} \qquad 3.6$$

For more complicated structures, a different method is used. Structures are broken into subsets of triangles, from which boundaries can be compared. A simple method for determining if a point is within a triangle is shown in Figure 3.4.

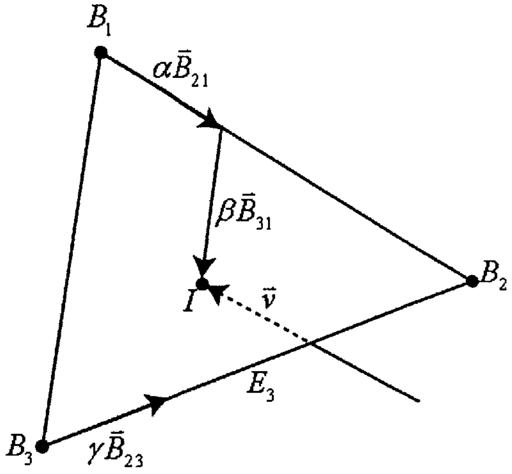

Figure 3.4. Illustration of a method for determining if an intersection point lies within the boundaries of a triangular object.

It should be noted that this method exists in 2-D space and therefore any plane described in 3-D should first be projected into 2-D space. An easy way to do this is to find the dominant axis of the plane normal vector, and remove that dimension from the calculation. This effectively projects the plane into either x-y, x-z, or y-z space while picking the largest projection.

The first thing to note from Figure 3.4 is that the vectors $\vec{B}_{21}, \vec{B}_{31}$, and $\vec{B}_{23}$ span the lengths of each side of the triangle. Therefore, in order to stay within the boundaries of the triangle, the scaling constants must be between zero and one.

$$0 \leq \alpha \leq 1 \qquad 0 \leq \beta \leq 1 \qquad 0 \leq \gamma \leq 1 \tag{3.7}$$

The point of intersection between the ray vector $\vec{v}$ and the plane can then be described using two of the three vectors on the triangle and scaling; in the case of Figure 3.4, the two vectors used to find the intersection point are $\alpha\vec{B}_{21}$ and $\beta\vec{B}_{31}$.

$$I - B_1 = \alpha\vec{B}_{21} + \beta\vec{B}_{31} \tag{3.8}$$

To find the scaling coefficients α and β which are necessary to reach the point I from B_1, we can solve Equation 3.8 for α and β.

$$\begin{pmatrix} \alpha \\ \beta \end{pmatrix} = \begin{pmatrix} \vec{B}_{21} & \vec{B}_{31} \end{pmatrix}^{-1} \bullet (I - B_1) \tag{3.9}$$

To actually determine whether the intersection point, I, made by the plane and ray $\vec{v}$ is inside or outside of the triangle, we need to look at the scaling constants. We already know that staying between the edges defined by $\vec{B}_{21}$ and $\vec{B}_{31}$ means that the scaling constants follow the rules of Equation 3.7. Unfortunately, these conditions alone say nothing about when the third edge defined by $\vec{B}_{23}$ is crossed. Therefore we must also determine what conditions on α and β result in a point on the third edge. We can start by writing the equation for the third edge.

$$E_3 = \vec{B}_{31} + \gamma\vec{B}_{23} \tag{3.10}$$

If we expand this using the end points of the triangle, we can attempt to transform Equation 3.10 in terms of the vectors $\vec{B}_{21}$ and $\vec{B}_{31}$.

$$E_3 = (B_3 - B_1) + \gamma(B_2 - B_3) \qquad 3.11$$

Then we can add and subtract Equation 3.11 by γB_1.

$$E_3 = (B_3 - B_1) + \gamma(B_2 - B_3) + \gamma B_1 - \gamma B_1 \qquad 3.12$$

If we then rearrange Equation 3.12, we see that the third edge E_3 is now expressed in terms of the other two edge vectors $\vec{B}_{21}$ and $\vec{B}_{31}$.

$$E_3 = \gamma(B_2 - B_1) + (1-\gamma)(B_3 - B_1) \qquad 3.13$$

Equation 3.13 looks very similar to Equation 3.8. In fact, Equation 3.13 states the limits on the scaling constants necessary for a point to lie on E_3. We can see that if we add the scaling factors γ and $(1-\gamma)$ we arrive at a maximum of unity. Therefore the last condition on α and β is given.

$$0 \le \alpha + \beta \le 1 \qquad 3.14$$

While planes bounded as triangles could be used to describe all planar features in a system, the number of iterations required to determine an intersection for a planar rectangular system would be increased by a factor of 2, which could slow down simulation time drastically. Therefore, it is more beneficial to design rectangular systems parallel to the Cartesian axes for a ray tracer programmed in this fashion to take advantage of the quick boundary checks available for on axis rectangular surfaces.

In order to correctly model the numerical aperture measurements as well as include gradual bends into the ray tracer, circular planes should also be included into the simulator. This is accomplished by replacing the plane equation with a 2-D circle equation and a boundary condition along the third dimension. The equation for a 2-D circle is given in Equation 3.15

$$(v_x - c_x)^2 + (v_y - c_y)^2 = r^2 \tag{3.15}$$

where c_x and c_y specify the center of the circle and r is the radius.

To find the intersection between the ray and the circular plane, we again need to solve for the separation distance, m, between the circular plane and the ray. To accomplish this, we first insert our description for the ray from Equation 3.4 into Equation 3.15.

$$(p_x + md_x - c_x)^2 + (p_y + md_y - c_y)^2 = r^2 \tag{3.16}$$

Then by rearranging terms, we can form a quadratic equation for m, specifying the distance from the ray to the two possible intersection points on the circle.

$$(d_x^2 + d_y^2)m^2 + 2\left(d_x(p_x - c_x) + d_y(p_y - c_y)\right)m + (p_x - c_x)^2 + (p_y - c_y)^2 - r^2 = 0 \tag{3.17}$$

Constraints for finding a valid distance m are that m is small and real.

Since the normal vector of the circular plane is not predetermined, we must also find the normal vector at the intersection point. This is simply the vector extending from the center of the circle, through the intersection point.

$$n_{circle} = \left| \langle v_x \quad v_y \rangle - \langle c_x \quad c_y \rangle \right| \tag{3.18}$$

After calculating the intersection point and the normal vector, the boundary conditions can be checked just like a regular plane by checking each axis boundary independently as described in Equation 3.6.

3.2.4 Reflection and Refraction

After determining an intersected plane, the incident angle of the ray onto the intersected plane must be determined. This information, combined with the refractive indices of the boundary and the ray polarization, can be used to determine whether reflected and refracted rays should be generated, and what their intensities should be. Since planes are defined with normal vectors, the angle of incidence between the incident ray and the intersected plane is simply related to the dot product given in Equation 3.19.

$$\frac{\vec{n} \bullet \vec{v}}{|\vec{n}||\vec{v}|} = \cos\theta \qquad 3.19$$

Next, it is necessary to determine if the ray experiences total internal reflection through Equation 3.20.

$$\theta_C = \sin^{-1}\left(\frac{n_2}{n_1}\right) \qquad 3.20$$

If not, we need to calculate the reflection coefficient of the reflected and refracted rays assuming either a TE or TM polarization as given in Equation 3.21.

$$\Gamma_{TE} = \frac{n_1 \cos\theta_1 - n_2 \cos\theta_2}{n_1 \cos\theta_1 + n_2 \cos\theta_2} \qquad \Gamma_{TM} = \frac{n_2 \cos\theta_1 - n_1 \cos\theta_2}{n_2 \cos\theta_1 + n_1 \cos\theta_2} \qquad 3.21$$

In equations 3.20 and 3.21, the subscripts 1 and 2 refer to the initial and final medium respectively. In the ray tracer presented, we are only interested in power and not phase or polarization of the propagating light as will be evident in Section 3.2.5. In this sense, the square of the coefficients determined in Equation 3.21 will describe the percent reflected power, and unity minus the reflected power will be the percent transmitted power.

3.2.5 Surface Roughness

To convert the roughness into a beam perturbation within the context of a ray tracer, a model must be developed to account for the viewed roughness. Since roughness measurements presented in Section 2.4.3 show peaks greater than 1 μm and surfaces which are optically clear, the extensive ray tracing approaches discussed earlier in Section 1.2 can not be used. Surface roughness from the vapor polishing procedure, will lead mostly to specular mode coupling rather than diffuse reflections. While mode mixing due to diffuse features still exists, the majority of scattering most likely results from these larger features. Much like the method proposed previously, only a probability resulting from surface roughness can be used for the simulation, making it inappropriate to factor in phase and polarization which require knowledge of the exact surface profile.

Since roughness appears macroscopic in comparison to a wavelength, the simplest approach to model rough surface behavior is to add a probabilistic perturbation to the interface slopes of ray/plane intersections. Rays that intercept a rough plane see a probability distribution of the normal vector of the plane instead of an absolute normal vector as shown in Figure 3.5.

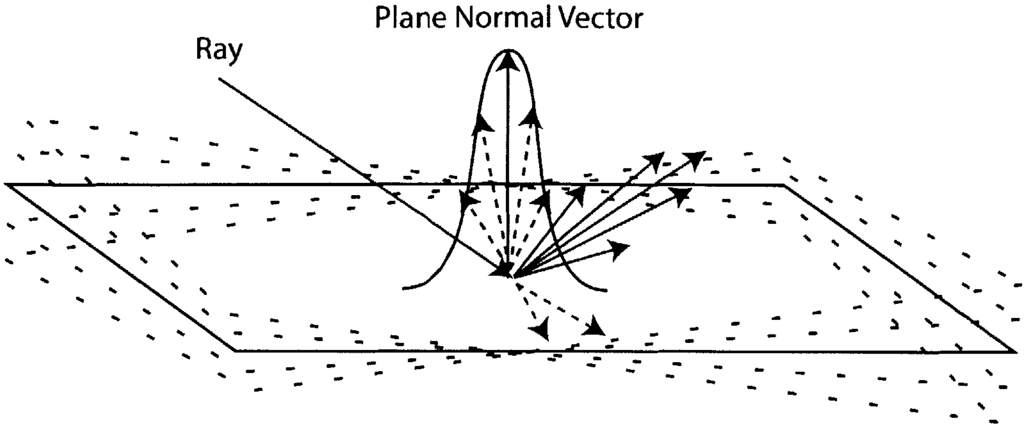

Figure 3.5. Illustration of surface scattering of a ray intersecting a plane with probabilistic surface roughness.

To incorporate this scattering behavior into the ray tracer, a Gaussian probability distribution with a given variance is assumed for the roughness. Measurements of the surface roughness and the distribution of slopes for the waveguide surfaces were shown in Section 2.4.3 to exhibit nearly Gaussian behavior. To create a random number with the desired distribution, a function is necessary to convert a uniform distribution into the desired distribution. For a Gaussian random variable, the PDF is given as

$$PDF(x) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.22}$$

where μ is the mean and σ is the variance of the Gaussian distribution. To use this distribution, we need to first integrate the PDF into a CDF.

$$CDF(x) = \frac{1}{2}\left(1+erf\left(\frac{x-\mu}{\sqrt{2}\sigma}\right)\right) \tag{3.23}$$

Then if we generate values of the CDF uniformly using a uniform distribution between 0 and 1, we can solve for values of x which are automatically distributed in a Gaussian fashion.

$$x = \sqrt{2}\sigma_x erf^{-1}\left(2CDF(x)-1\right) \tag{3.24}$$

The CDF of the Gaussian distribution is inverted to convert the uniform random variable into a Gaussian random variable.

Since the milling process in general creates different degrees of roughness depending on the cutting axis, two Gaussian distributions, one for each parallel axis to the plane, are necessary. The slopes chosen by the random variables are added to the plane's normal vector before the incident angle of the ray on the plane is calculated. An example of a trace including surface roughness is shown in Figure 3.6.

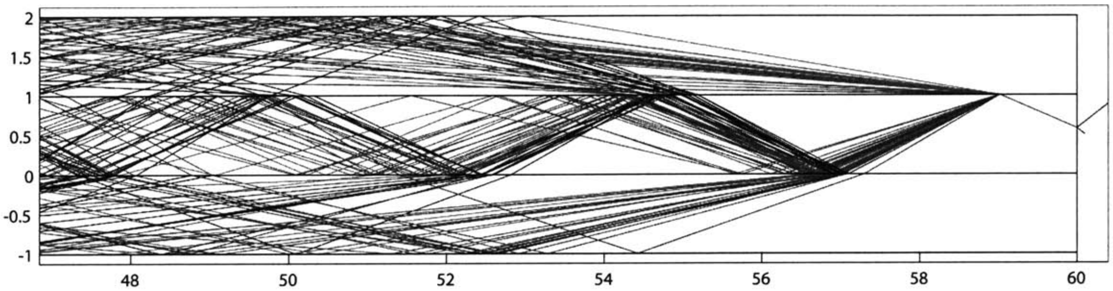

Figure 3.6. Ray tracing simulation for a waveguide structure with a core index of 1.56 and a cladding index of 1.43. The slope variance for the core-cladding interface is chosen as 0.019.

From the simulation, it is evident that roughness leads directly to light scattering from the core into the cladding and vice versa. Even rays that do not scatter into the cladding can have their refraction angle perturbed.

3.2.6 Input Distributions

To accurately trace through the system and determine the output distribution, the input distribution must be specified to describe the light source. For uniform point sources, rays need to be generated that are spherically uniform in intensity. This type of distribution applies to fluorescent particles as well as uniform intensity plane waves focused through lenses. Since our experimental measurements performed in Section 2.4 used LEDs focused through lenses, the spherically uniform intensity profile is a decent approximation for the intensity distribution versus angle incident on the waveguide input.

To create a spherically uniform random variable, two uniform random variables are required, one for each angular axis as shown in Figure 3.7.

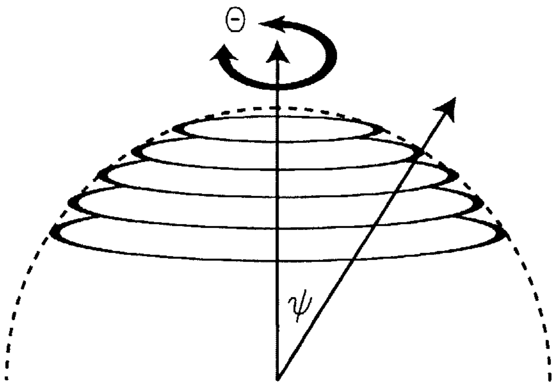

Figure 3.7. Illustration of the dependence of the spherical number generation on the angles y and Q. While the dependence on Q is uniform, the dependence on y is proportional to the circumference.

Since the Q angle simply describes the rotation around a selected circular plane, the choice of this angle should also be a uniform distribution. The dependence on y, however, is proportional to the circumference of the circle lying on the cross-section described by y, which is proportional to the radius, sin y. First we must start with normalized sine distribution in an interval from 0 degrees to y_{max} degrees.

$$P(\psi) = \frac{\sin\psi}{1-\cos\psi_{\max}} \qquad 3.25$$

To relate the distribution in y to a uniform distribution, we need to map the uniform variable to the y variable. Since we assume that there is a function that can convert the uniform distribution into the y distribution, the probabilities of each variable should be the same over differential distances.

$$P(\psi)d\psi = P(x)dx \qquad 3.26$$

We can then plug in the expressions for the spherical probability and the uniform probability distributions.

$$\frac{\sin\psi}{1-\cos\psi_{\max}}d\psi = \frac{1}{\psi_{\max}}dx \qquad 3.27$$

By integrating both sides and solving for y, we arrive at the function we originally assumed to exist, which describes how to convert a uniform distribution in x, into a spherical distribution in y.

$$\psi = \cos^{-1}\left[1 - x\left(\frac{1-\cos\psi_{\max}}{\psi_{\max}}\right)\right] \qquad 3.28$$

A plot of a spherical distribution generated using a uniform distribution for Q and the distribution from Equation 3.28 for y is given in Figure 3.8.

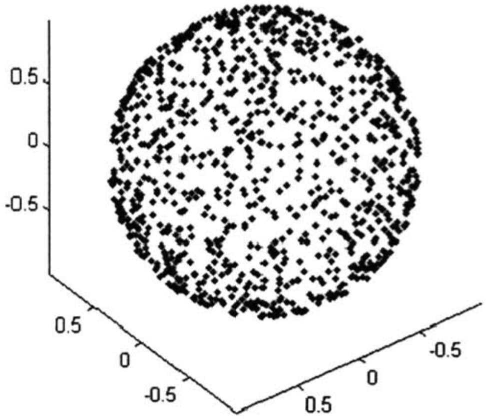

Figure 3.8. A plot of a randomly generated uniform spherical distribution with $\theta_{\max} = 2\pi$ and $\psi_{\max} = \pi$.

Since most other light sources or intensity dependent distributions are only axially symmetric, such as LEDs, we can follow the same procedure used to generate the uniform spherical distribution, but add an additional weighted probability to the y angle. While the approach is straight forward, solving for arbitrary probability distributions can be difficult and in many cases unsolvable. In general, for any additional weighting function $F(\psi)$,

$$P(\psi)=\frac{\sin(\psi)F(\psi)}{\int_0^{\psi_{\max}}\sin(\psi)F(\psi)d\psi} \tag{3.29}$$

Since the difficulty in evaluating the normalization integral depends on the function $F(\psi)$, some distributions cannot be solved analytically.

3.3 Waveguide Simulations

3.3.1 Ray Tracer Verification

Before proceeding with simulations using the ray tracer, it is necessary to verify whether the ray tracer is working correctly. To verify the ray tracer, simple and complex structures are tested against the OSLO ray tracer.

The first structure tested was a simple 2-D spherical lens to test if curved surfaces were programmed correctly. From Figure 3.9, it is clear that the programmed ray tracer behaves exactly like the OSLO ray tracer.

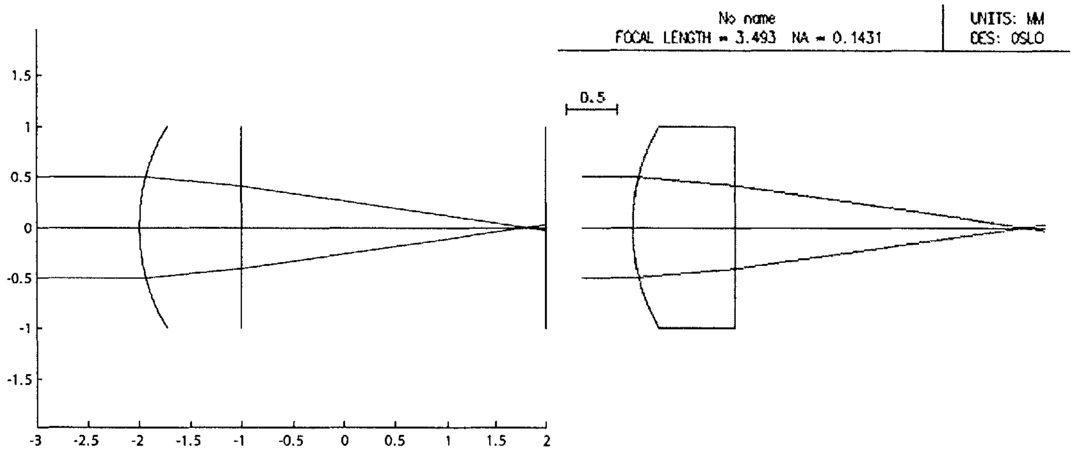

Figure 3.9. Ray tracing comparison between the programmed ray tracer and commercially available OSLO. Both ray tracers behave the same for single plano-convex lenses.

Since the ray tracer will be used mainly for tracing waveguides and bend type structures, the ray tracer is also verified with OSLO for a system containing lenses, total internal reflection prisms and mirrors.

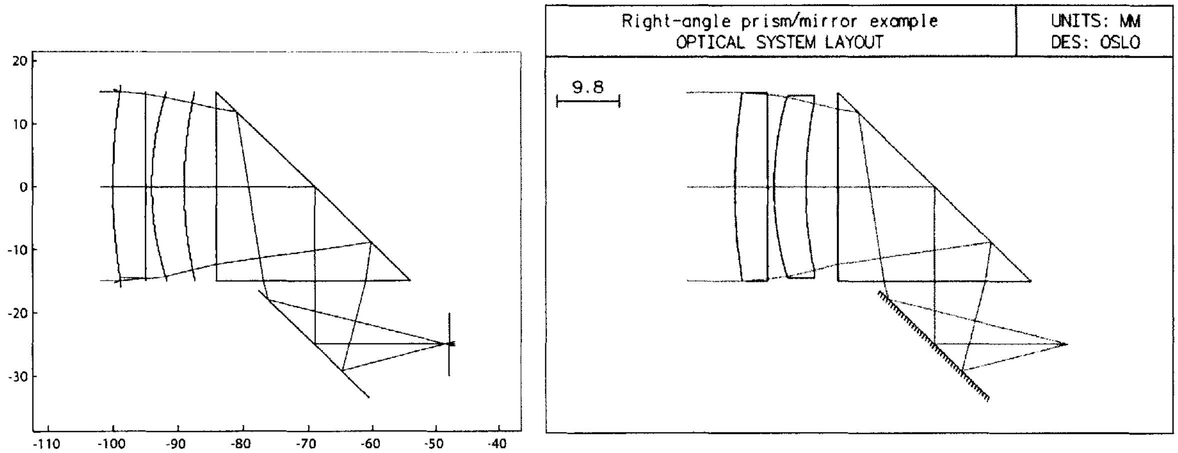

Figure 3.10. Ray tracing comparison between the programmed ray tracer and OSLO for a system containing lenses, prisms, and mirrors.

From the two traces shown in Figure 3.10, it is clear that the programmed ray tracer is capable of tracing through all objects necessary for simulating fabricated waveguides.

3.3.2 Straight Waveguide

Simulating a straight waveguide with the ray tracer will show what the output profile should look like for an ideal square waveguide. By comparing simulations of an ideal square waveguide with a rough waveguide, we can determine how roughness affects the output numerical aperture. In order to compare roughness simulations with the roughness from fabricated waveguides, two waveguides with dimensions of 1x1x75 mm^3 are fabricated with different feed speeds of 30 mm/min and 200 mm/min to simulate differences in roughness. The roughness traces for both guides are given in Figure 3.11.

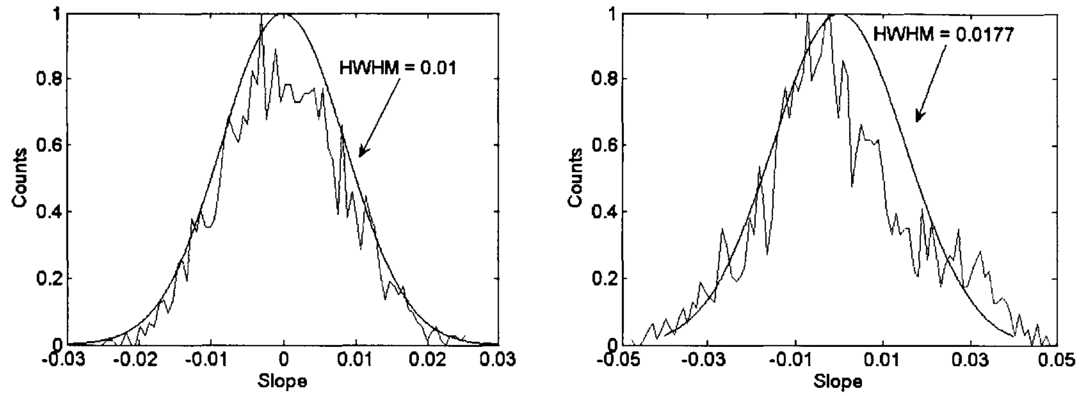

Figure 3.11. Slope histograms resulting from a surface profilometer trace of a sidewall milled at 30 mm/min (left) and a sidewall milled at 200 mm/min (right).

For the simulation, a waveguide with dimensions of 1x1x75 mm is chosen to closely match the fabricated waveguides. A spherically uniform random distribution of 5000 rays with a numerical aperture of 0.62 is propagated through the waveguide structure with the variance set to 0.01 and 0.0177 for the 30 mm/min and 200 mm/min waveguides respectively. Hit statistics are gathered at a circular plane 75 mm away from the output face of the waveguide, matching the power meter distance from numerical aperture measurements from Section 2.4.2.

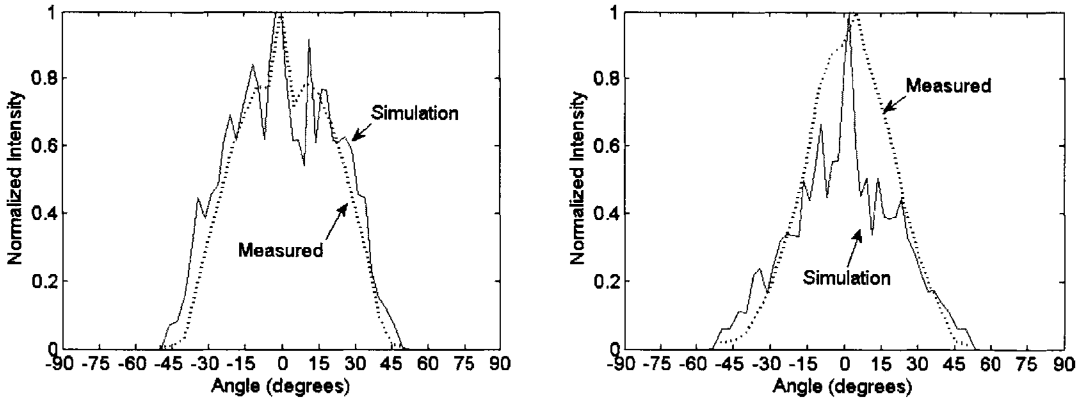

Figure 3.12. Plots of the measured numerical aperture versus a simulation for (left) a waveguide with a slope variance of 0.01 and (right) a waveguide with a slope variance of 0.0177. It can be seen that an increase in roughness decreases the numerical aperture and the resulting output distribution can be predicted through simulations.

From the plots shown in Figure 3.12, it is evident that numerical aperture simulations utilizing a Gaussian slope roughness model accurately describe the measured numerical aperture reduction resulting from side wall roughness. From the change in shape of the output intensity profile for the rougher waveguide, we can see that there is definitely mode dependent loss, and higher order modes suffer this loss more than low order modes.

The next step is to determine how the roughness translates to loss in the waveguide. To model this, hit planes are placed inside the waveguide to generate intensity statistics at locations within the waveguide. This method simulates the cutback method for determining loss within a waveguide. The loss coefficients extrapolated from the simulations are shown in Figure 3.13.

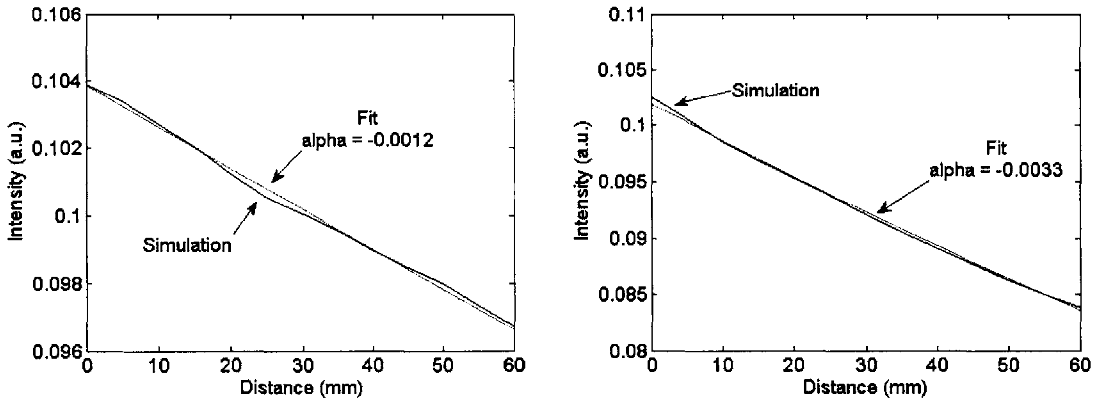

Figure 3.13. Loss plot due to surface roughness from the waveguide simulation for a roughness variance of 0.01 (left) and 0.0177 (right) and an input aperture of 38°. The simulated loss due to roughness increases as the roughness increases.

From the simulation, the roughness related loss is not a function of wavelength, and can only be a function of wavelength if the beam spread due to roughness is a function of wavelength. In addition, the loss due to roughness from the simulation depends greatly on the input numerical aperture. As shown from measurements in Section 2.4.2, a change in the input numerical aperture from 7° to 38° increased the loss by 0.43 dB/cm. Loss increases for larger numerical apertures simply because there are increased numbers of large angle rays incident on the side walls.

The fitted loss for the waveguides milled at 30 mm/min and 200 mm/min are 0.052 dB/cm and 0.14 dB/cm respectively. Both of these losses are below the minimum loss measured by normal scattering measurements as shown in Figure 3.14.

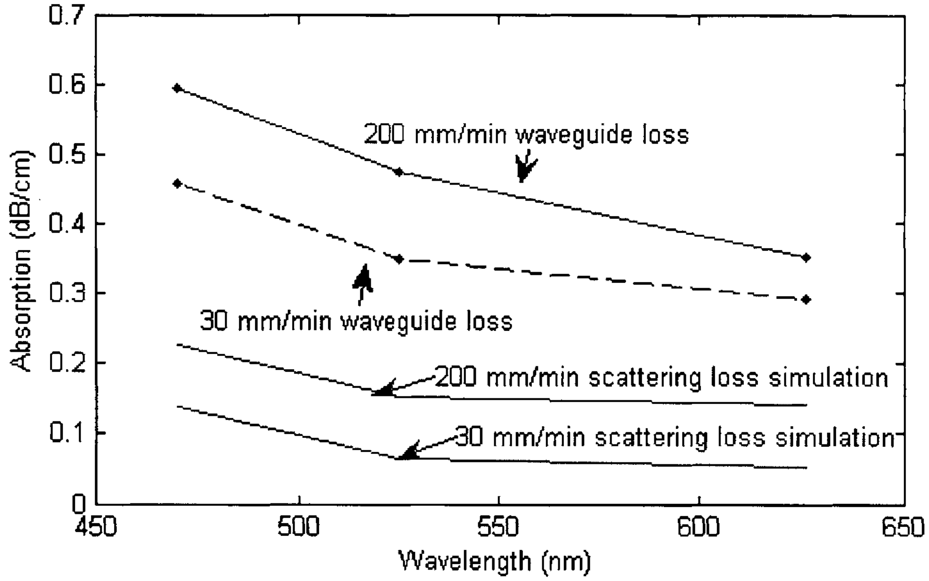

Figure 3.14. Plot of simulated roughness loss versus measured loss. Both side wall scattering loss lower bounds are below the measured loss values for the waveguides. The measured increase in loss for the 200 mm/min waveguide is similar to the expected increase from simulation

While the absolute loss for both waveguides is higher than the simulated surface roughness loss, it should be noted that the relative increase in loss from the 30 mm/min waveguide to the 200 mm/min waveguide is similar to the relative increase determined through simulations, with simulations yielding an increase of 0.09 dB/cm and measurements at 470 nm and 525 nm yielding an increase of 0.13 dB/cm. It is not appropriate to include measurements taken at 626 nm since the scattering loss measured required turning the camera gain to maximum in order to detect scattered light. Data points at 626 nm plotted in Figure 3.14 are a result of a best estimate, with actual data points ranging from 0.2 dB/cm to 0.44 dB/cm for the 200 mm/min waveguide and similar variance in the 30 mm/min waveguide.

In addition to loss and numerical aperture, the ray tracing simulation can give important results about the bandwidth of the waveguide. Rays propagating within a waveguide structure follow trajectories which are close to the group velocity of the mode described by a propagation vector with the same angle of inclination as shown in Figure 3.15.

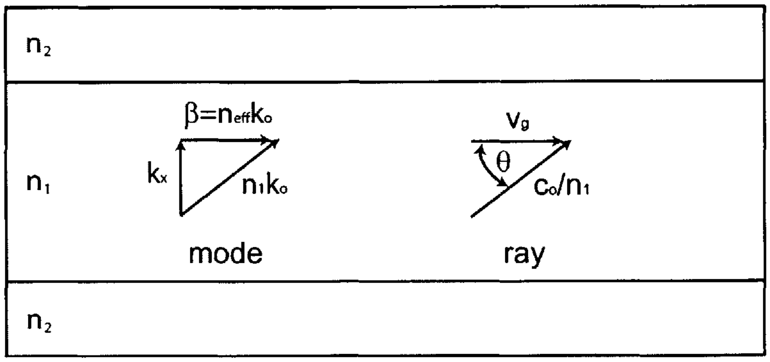

Figure 3.15. Illustration of the propagation vectors for modes and rays within a symmetric slab waveguide.

For rays within a straight slab waveguide, the group velocity is equal to the velocity of the ray in the forward direction.

$$v_{g-ray} = \frac{c}{n_1}\cos\theta \qquad 3.30$$

In the ray description, c is the speed of light in vacuum, n_1 is the refractive index of the waveguide core, and θ is the angle of inclination for the ray with respect to the waveguide axis. The angle of inclination θ within the waveguide can also be written in terms of the mode effective index that contains the same angle propagation vector through trigonometric identities, which shows how the group velocity of the ray in Equation 3.30 depends on the effective index n_{eff}.

$$v_{g-ray} = \frac{c}{n_1}\cos\theta = \frac{c\beta}{n_1^2 k_0} = \frac{cn_{eff}}{n_1^2} \qquad 3.31$$

Where k_0 is the free space propagation vector, also written as $2\pi/\lambda$, where λ is the wavelength of light, and all other parameters which are described in Figure 3.15.

In a mode description, the group velocity of the mode also depends on the effective index of the mode.

$$v_{g\text{-mode}} = \frac{\partial w}{\partial k} = \frac{c_0}{n_{eff} + w\dfrac{\partial n_{eff}}{\partial w}} \qquad 3.32$$

Where w and n_{eff} are related to the transverse propagation vector k_x through the symmetric step index waveguide transcendental equation and the vector based equation.

$$\tan^2\left(\frac{k_x d}{2}\right) + \left(k_x d\right)^2 = \left(\frac{wd}{c_0}\sqrt{n_1^2 - n_2^2}\right)^2 \qquad 3.33$$

$$n_{eff} = \sqrt{n_1^2 - \left(\frac{k_x}{k_0}\right)^2} \qquad 3.34$$

By choosing values of k_x, the group velocity versus frequency for both the ray description and the mode description can be determined.

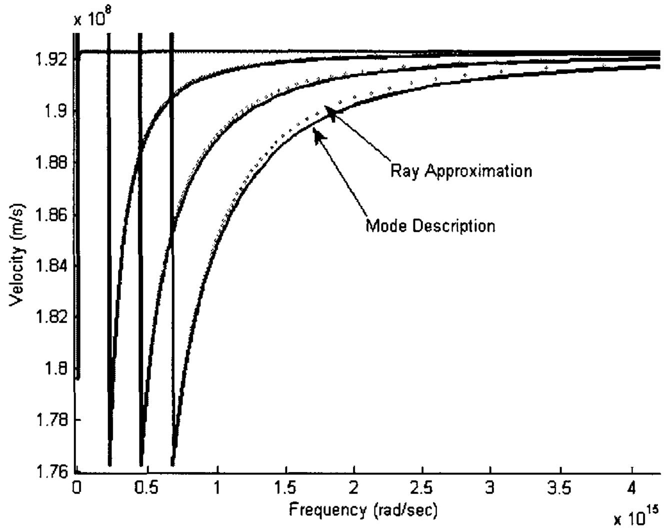

Figure 3.16. A plot of group velocity versus frequency for modes 0, 150, 300, and 450. The ray description is a good approximation of the modal group velocity.

The plot in Figure 3.16 shows how the ray description of group velocity matches the mode description. For all of the modes in the system, the increased path length of the angled ray within the waveguide system accounts for the slower group velocity of higher order modes. This can also be seen analytically by solving 3.30 and 3.32 in terms of n_{eff}.

We first write n_{eff} in terms of the frequency w and the transverse propagation vector k_x.

$$n_{eff} = \sqrt{n_1^2 - \left(\frac{k_x}{k_o}\right)^2} = \sqrt{n_1^2 - \left(\frac{k_x c}{w}\right)^2} \qquad 3.35$$

Then we can find the group index with respect to frequency by taking the derivative of Equation 3.35.

$$\frac{\partial n_{eff}}{\partial w} = \frac{k_x^2 c^2}{w^3 \sqrt{n_1^2 - \left(\frac{k_x c}{w}\right)^2}} = \frac{k_x^2 c^2}{w^3 n_{eff}} \qquad 3.36$$

By plugging in the evaluated form of the group index into Equation 3.32, we can express the mode group velocity solely in terms of the frequency w and the transverse propagation vector k_x,

$$v_{g-\mathrm{mode}} = \frac{c}{n_{eff} + w\frac{\partial n_{eff}}{\partial w}} = \frac{c n_{eff}}{n_{eff}^2 + \left(\frac{k_x w}{c}\right)^2} \qquad 3.37$$

If we solve Equation 3.35 for n_1^2, we see that we can directly simplify the denominator of Equation 3.37.

$$v_{g-\mathrm{mode}} = \frac{c n_{eff}}{n_1^2} \qquad 3.38$$

We see that Equation 3.38 is equivalent to Equation 3.31, showing that the ray picture for a waveguide correctly approximates the dynamics of light propagation so long as rays chosen within the system obey the transcendental equation of the waveguide structure.

Using the ray group velocity equation, we can then construct an equation for the time delay between the longest distance ray and the shortest distance ray to determine how much bandwidth the waveguide can support. This effect exhibits different longitudinal speeds of rays within the waveguide and is called multimode distortion as discussed in Section 1.2. For the fastest ray, the ray travels directly down the waveguide at an angle of 0 degrees.

$$t_{fast} = \frac{n_1 L}{c} \qquad 3.39$$

We see that the ray just travels down a waveguide of length L at the speed of light within the core medium. For the slowest ray, the ray travels down the waveguide at the critical angle.

$$t_{slow} = \frac{n_1 L}{c\sin\left(\theta_C\right)} = \frac{n_1^2 L}{n_2 c} \qquad 3.40$$

Since we would like to know how the roughness induced numerical aperture affects the bandwidth of our waveguide, the slow ray can be rewritten in terms of the numerical aperture, as shown in [17].

$$t_{slow} = \frac{n_1 L}{c\sqrt{1-\cos^2\left(\theta_C\right)}} = \frac{n_1 L}{c\sqrt{1-\left(\frac{NA}{n_1}\right)^2}} \qquad 3.41$$

Now that we have the fast and slow rays of the system, we can determine the bandwidth of the multimode guide. In general, the distance is divided to make the equation independent of distance.

$$\Delta t = \frac{n_1}{c}\left[\frac{1}{\sqrt{1-\left(NA/n_1\right)^2}}-1\right] \qquad 3.42$$

For the waveguide system fabricated in Chapter 2, with a core index of 1.56 and a cladding index of 1.43, the multimode distortion which results is 473 ps/m.

Figure 3.17 shows how an impulse broadens in time at the output by simulating multimode distortion and plotting the length of time taken for rays to reach the output face of the waveguide. Since the simulation assumes that the refractive index is uniform over all wavelengths, other dispersion effects that result in loss of bandwidth can not be modeled. The same generated point source is used for two waveguides, one with a smooth surface and one with a rough surface.

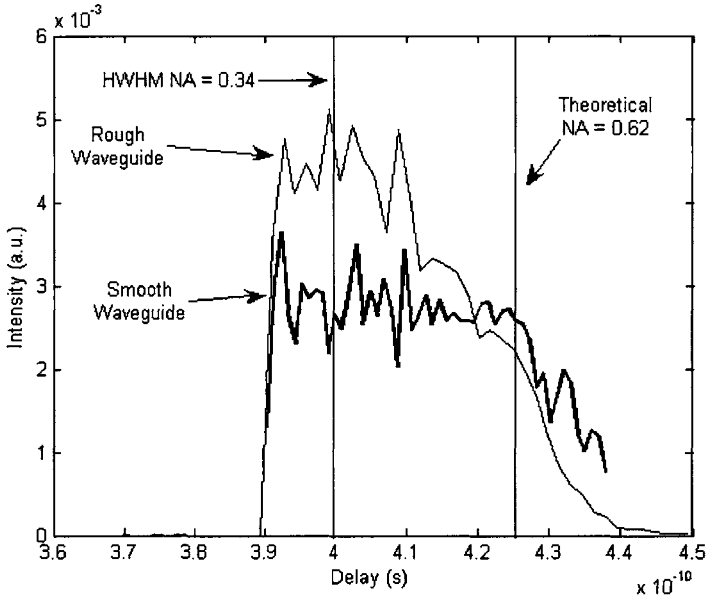

Figure 3.17. Signal intensity at the waveguide output versus time due to a point source impulse at the input. Propagation delay for a 1x1x75 mm waveguide with 0 surface roughness is compared to a waveguide with 0.017 variance in surface roughness. The arrival times of the input signal range from 390 ps to 438 ps for both smooth and rough waveguides.

From the simulation, we can see that for a 1x1x75 mm waveguide, the original impulse with a half angle spread of 45 degrees arrives at the waveguide output over a time length of 48 ps. However, the calculated time delay due to multimode distortion for a 75 mm long waveguide is 35.5 ps. The extra length is due to the overfilling of the waveguide. As shown in Figure 3.17, the intensity starts to decay for rays traveling above the theoretical numerical aperture. For a simulated waveguide with surface roughness, we see that the intensity also starts to decay for rays traveling at angles larger than the estimated numerical aperture due to roughness. It is also important to note that the intensity of faster rays increases for a rough waveguide over a smooth waveguide due to mode mixing from the higher order modes into the low order modes. Even with including roughness, there appears to be no significant improvement in the bandwidth, since 50% intensity occurs at nearly the same delay time, which is what we would expect from [17].

If the delay of 35.5 ps from simulations is treated as a single exponential time constant, optical signals modulated at frequencies greater than 4.5 GHz will not be observable due to multimode distortion. However, for waveguides, multimode distortion with minimal mode dependent loss has a frequency response more closely resembling a sinc function since there is no attenuation of delayed signals. The exponential picture is inaccurate since there is a sharp cut-off of measurable intensity past the critical angle. For a sinc frequency response, the 75 mm long waveguide has a 3dB bandwidth is 17 GHz. The associated bandwidth distance product for the waveguide is 1.3 MHz-km, which is slightly smaller than the bandwidth of a large core multimode fiber described in Chapter 1 due to the larger numerical aperture. Since the bandwidth due to multimode distortion is so low, it is app:opriate to neglect other dispersion related bandwidth limitations.

To test if the simulated bandwidth of the waveguide is accurate, a bandwidth measurement is performed on the negatively molded straight waveguide section fabricated in Chapter 2 with dimensions of 1.2x1x70 mm^3. The bandwidth measurement setup is shown in Figure 3.18.

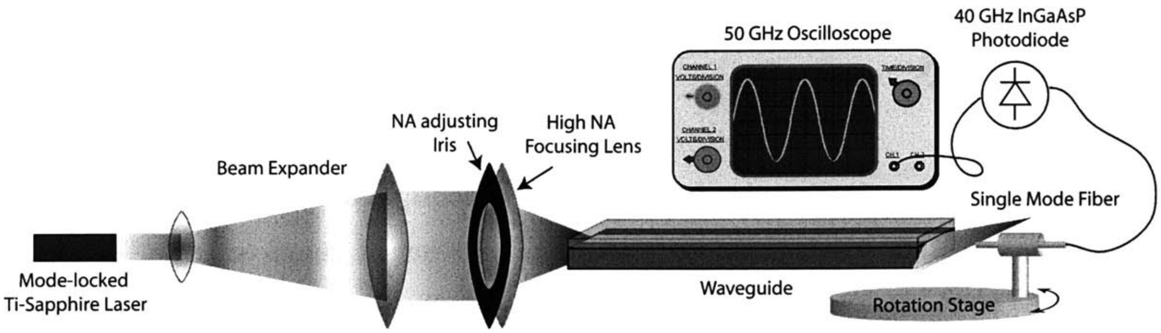

Figure 3.18. Schematic of the measurement setup used to determine the multimode distortion of the waveguide. An oscilloscope, triggered by the mode-locked laser, measures the pulse output from the waveguide. The modal delay is determined by comparing the delay between a pulse measured at 0 degrees and a pulse measured at 40 degrees.

By triggering the oscilloscope with the mode-locked Ti-Sapphire laser operating at 900 nm, the delay of the pulse out of the waveguide relative to the pulse generated by the laser can be measured. The modal distortion can be determined by measuring the pulse delay of the low order modes versus the high order modes which occur at 0 degrees and 38 degrees respectively. In the actual measurement, pulses are measured for the fiber at 0 degrees and 40 degrees with respect to the waveguide. The measured pulses are shown in Figure 3.19. The measured delay of 31 ps agrees with the calculated delay of 33.1 ps using Equation 3.42.

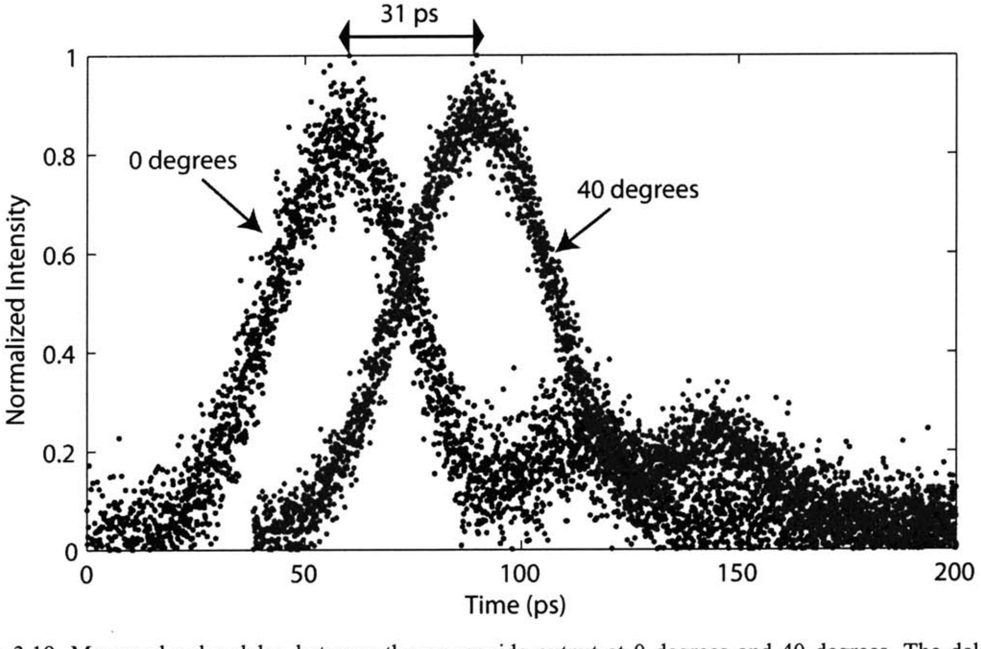

Figure 3.19. Measured pulse delay between the waveguide output at 0 degrees and 40 degrees. The delay of 31 ps agrees with the calculated delay of 33.1 ps.

3.4 Summary

A ray tracer has been developed to both aid in optical design for future devices as well as provide a model for understanding scattering behavior of rough side walls on an elementary level. By modeling rough surfaces as planes with probabilistic slopes, simulated straight waveguides exhibited behavior similar to fabricated and measured waveguides. Factors dependent on surface roughness such as loss and numerical aperture were examined. In addition, waveguide loss due to surface roughness was determined through simulations and sets a lower bound on waveguide loss in fabricated devices. An upper bound on waveguide bandwidth was also determined for a 75 mm long waveguide and it was found that surface roughness had little effect on bandwidth.

Chapter 4

Integrated Polymer Optical Devices

4.1 Introduction

Since CNC milling relies on physical cutting with a circular blade, there is always a macroscopic roughness or waviness which results from the machining parameters. As a result, any measurements taken on machined waveguides that rely on properties of light such as polarization, phase, or coherence will be lost due to random surface roughness from the milling process. However, any sensors which are lifetime based or intensity based as discussed in Chapter 1, can be measured using large core waveguides. While intensity measurements have no limitations other than signal strength, lifetime based measurements have a bandwidth requirement limited by the multimode distortion of the waveguide as discussed in Section 3.3.2. Fortunately, the bandwidth available for such short lengths is high enough to accommodate lifetimes as small as 1 ns.

Integrating waveguides into a polymer based sensing system is helpful for performing fluorescence lifetime measurements, but it is not enough to help in miniaturizing systems. Since we are limited to measuring intensity with the waveguides, other intensity based functions which can be incorporated into the system, such as light sources and filters, will be very useful. Structures such as power combiners, power splitters, and bends will also be useful in minimizing the amount of external components necessary to carry out detection. In this chapter, we will focus on developing these devices and work towards integrating them into polymer chips.

In Section 4.2, we will explore two ways to modify waveguide cores. The first method, involving organic dyes, can yield an in-place method of performing wavelength selective filtering, which is essential for a good signal to noise ratio. The second modification involves suspending nanocrystals into the core polymer. By creating a core material embedded with highly fluorescent nanocrystals, light sources can be integrated within the waveguide structures. Enhancements such as on-chip integrated filters and light sources can help greatly with system integration and scaling.

There are also a variety of useful passive components which need to be integrated into the waveguide system in order to improve scaling capabilities. These components will be explored in Section 4.3. The first passive component necessary is a small area bend. In order to create an optical system which does not affect design of fluidic systems, it is essential to design a method for coupling light into and out of plane. The other passive components necessary for aiding scalability are signal splitters and signal combiners. Both elements can add multiplexing capabilities which will greatly simplify external detection elements.

4.2 Polymer Modifiers

An approach to adding functionality to polymer based optical devices is to modify the polymer core or the polymer cladding with particulates. These particles could be either non-fluorescing dyes to provide filtering properties or high quantum yield dyes and nanocrystals to provide sources of light. Since adding particles into a polymer system can increase scattering greatly, it is necessary to choose components that dissolve readily into polymers.

4.2.1 Organic Dye Filters

While using dyes in polymer based photoresist films to provide wavelength selective filtering has been achieved [20], the goal is to incorporate these types of filters into waveguides to provide filtering without sacrificing signal intensity. To accomplish this, dyes must be dissolved into the core polymer in order to provide filtering and guiding simultaneously.

The most common approach to combining organic dyes and polymers is to dissolve both substances into a solvent to improve mixing. After sufficient dissolving occurs, the solvent is evaporated, leaving the dye incorporated into the polymer. This method works well for applications such as spin coating or direct placement, where mixtures are exposed to air and can readily evaporate to reharden the polymer. For situations where there are other components in contact with the solution which are incompatible with the solvent used, there are other approaches. For multipart thermal or ultraviolet cured polymer systems, the dye can be dissolved into the prepolymers. If this is done with a solvent, then the solvent can be evaporated before prepolymer mixing. In this approach, situations where the solvent is difficult to remove, or where the dye and solvent are incompatible with the cross-linker can occur.

A study has already been conducted by the company that manufactures NOA, Norland Products, regarding the strength of the cured polymer against a variety of solvents, but to incorporate dyes into the prepolymer requires solvent compatibility with the NOA prepolymer as well. For NOA, three different types of solvents were tested to determine compatibility. The results are summarized in Table 4.1.

Solvent	Dissolves	Effects
Methanol	N	Upon evaporation of solvent, white crystalline compounds precipitate out of optical adhesive. Polymer is still curable.
Acetone	Y	Solvent interacts with cross-linker. Efforts to cure polymer under UV illumination were unsuccessful.
Toluene	Y	Good dissolvability and polymer still curable. Solvent hard to completely remove, resulting in cracks on the cured surface.

Table 4.1. Summary of the effects of solvents on uncured NOA prepolymer.

Results from Table 4.1 are comparable results given by Norland Products, with the exception that some solvents, such as methanol, do not even dissolve readily with the polymer. Therefore, even though both methanol and toluene do not affect the curing properties of NOA, only toluene can be used to aid in mixing between NOA and dyes.

Some dyes, in particular cyanine based dyes, dissolve into NOA without the aid of a solvent. This method of incorporation is preferred over solvent aided mixing. To test the effects of toluene on dye solubility in NOA, two dyes were chosen, 1,1'-Diethyl-2,2'-carbocyanine iodide (PCI), and Copper(II) 5,9,14,18,23,27,32,36-octabutoxy-2,3-naphthalocyanine (CopperNC). First, both dyes are placed in a solution of toluene. From Figure 4.1, it is clear that PCI does not dissolve well in toluene due to noticeable precipitation while it does dissolve readily in NOA. Since toluene and PCI are incompatible, we do not expect the solubility to increase from a mixture which includes toluene.

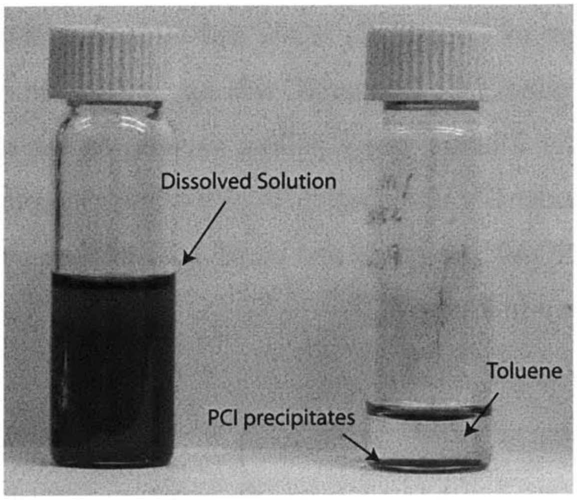

Figure 4.1. Comparison between PCI dissolved in NOA (left) and PCI sitting in toluene (right). Even though toluene and NOA are compatible, PCI does not dissolve into toluene.

CopperNC, however, experiences the opposite effect. Addition of toluene to CopperNC in powder form results in a dissolved solution while a direct mixture of CopperNC and NOA does not result in a dissolved solution. To test if the addition of toluene increases the solubility of CopperNC into NOA, the solution of CopperNC and toluene is mixed with NOA. Figure 4.2 shows the comparison of a solution of CopperNC and NOA and a solution of CopperNC, toluene, and NOA.

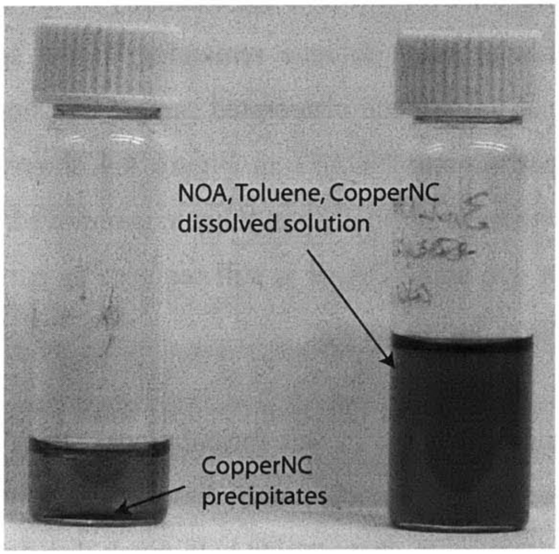

Figure 4.2. Comparison between CopperNC dissolved in NOA (left) and CopperNC dissolved in toluene/NOA (right). Addition of toluene into the mixture allows the CopperNC particles to dissolve into solution. Without toluene, there exists a suspension of CopperNC particles resulting in a cloudy mixture.

After creating a solution of CopperNC, NOA, and toluene, it is necessary to evaporate out the toluene to determine if the CopperNC will stay in solution. This is done by baking the solution at 60° C for 2 hours, or by pulling vacuum on the solution. Then a 1 mm thick sample is cured under UV illumination. A plot of the absorption characteristics for a sample of dye doped NOA71 prepared and cured with toluene partially evaporated and fully evaporated is shown in Figure 4.3.

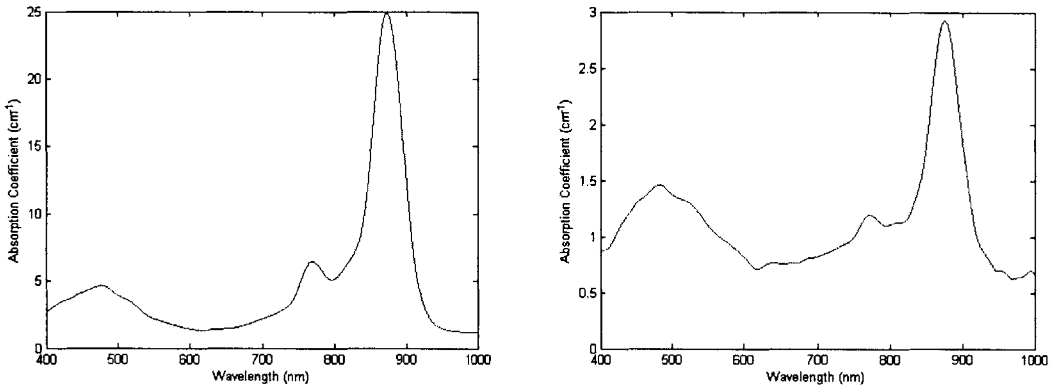

Figure 4.3. Comparison of dye absorption in NOA with the addition of toluene. (Left) Absorption coefficient for a sample cured with toluene in the mixture. (Right) Absorption coefficient for a sample prepared with toluene but with toluene evaporated from the mixture before curing.

From Figure 4.3, it is clear that toluene aids the solubility of the dye into NOA, but only if the sample is cured with some toluene remaining in the solution. A comparison between the absorption of the toluene evaporated sample of CopperNC and the directly mixed sample of CopperNC into NOA71 in Figure 4.4 shows that the solubility of CopperNC in NOA71 returns to normal when toluene is removed. While toluene aids the mixing process between dye and NOA, it is still required as part of the solution for an increased dye concentration during curing.

Since many of the cyanine dyes are in salt formulations, where the dye is bound to a halogen such as chlorine or iodide, and dissolve readily into NOA through experimentation, it is necessary to characterize how much dye dependent absorption can be obtained at the saturation limit of dye in NOA. To accomplish this, a variety of dyes were mixed with NOA, and the resulting combinations were heated to 60° C for 24 hours to enhance mixing. Toluene was not used to aid in the solubility process since the dye dissolvability into NOA appears to be dependent on the toluene concentration after solution mixing. After the dyes were thoroughly dissolved, the saturated NOA compounds were centrifuged to remove any precipitates that formed during cooling. A 1 mm thick slab of each mixture was cured between two glass slides and a 1 mm slab of NOA was also cured between two glass slides for reference, and the absorption spectrums in Figure 4.4 were obtained for each dyed NOA sample to find the maximum absorption for each dye/NOA combination. The 7 dyes investigated are given in Table 4.2 and an extensive list of dyes is given in Appendix B.

Organic Dye	Absorption Peak (nm)
1,1′-Diethyl-2,2′-cyanine chloride (PIC Chloride)	524
1,1′-Diethyl-2,2′-carbocyanine chloride (Pinacyanol Chloride) (PCC)	604
1,1'-Diethyl-2,2'-carbocyanine iodide (Pinacyanol Iodide) (PCI)	614
1,1’-Diethyl-4,4’-carbocyanine iodide (Cryptocyanine) (CYC)	703
1,1’-Diethyl-2,2’-dicarbocyanine iodide (DDI)	707
2,11,20,29-Tetra-tert-butyl-2,3-naphthalocyanine (TetraT)	784
Copper(II) 5,9,14,18,23,27,32,36-octabutoxy-2,3-naphthalocyanine (CopperNC)	853

Table 4.2. List of organic dyes explored as possible dye filters. The maximum absorption peaks are quoted from Sigma Aldrich.

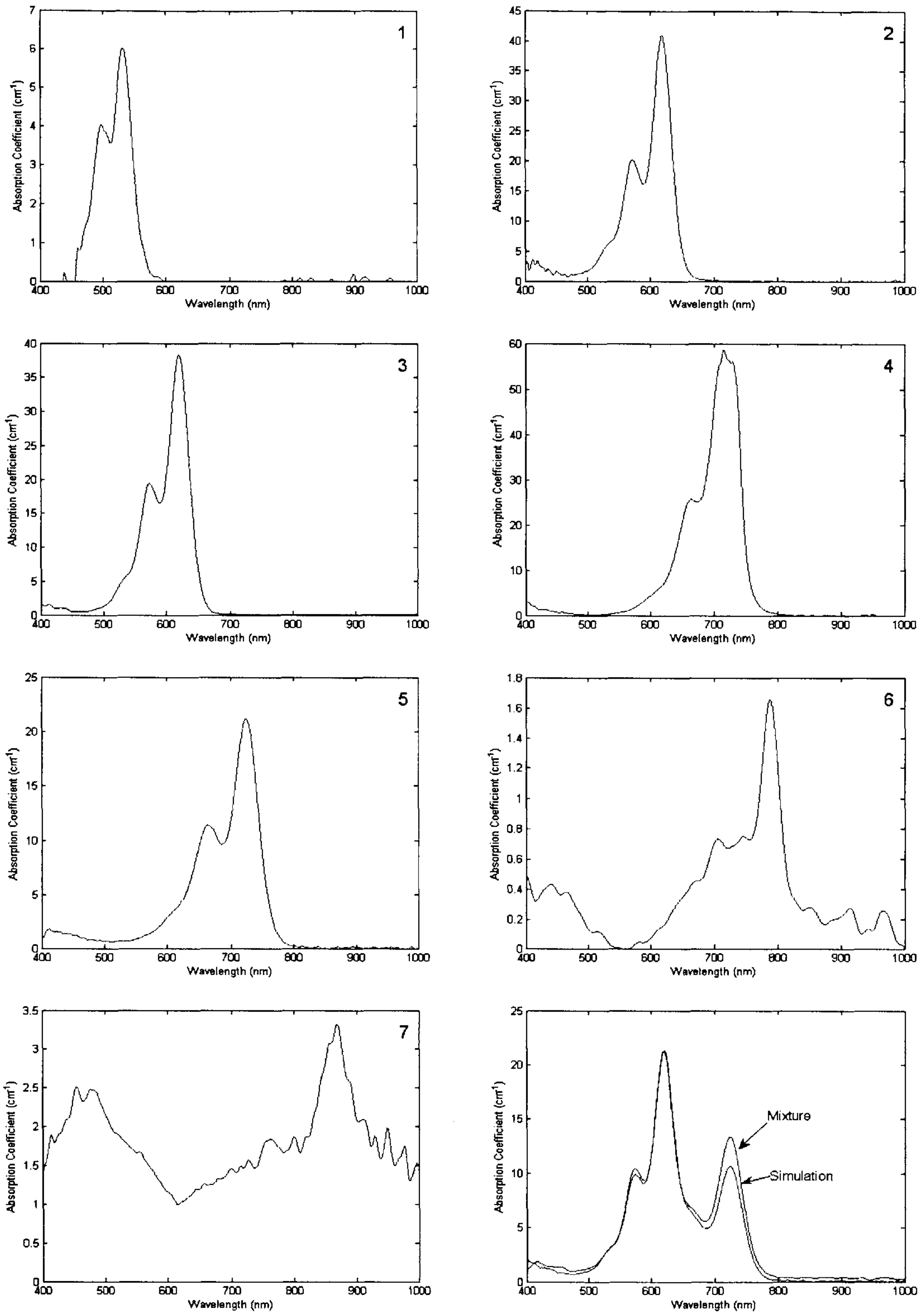

Figure 4.4. Plots of the absorption coefficient for different dyes saturated in NOA71 polymer. Plots are normalized to the spectrum of NOA71. Seven different dyes are tested, PIC Chloride (1), PCC (2), PCI (3), CYC (4), DDI (5), TetraT (6), and CopperNC (7). The last plot is a 1:1 mixture of DDI and PCI (8) showing the near linear behavior of combinations of dyes.

Clearly, some dyes, such as Copper NC and TetraT do not perform well as filters. Others, such as PCI, PCC, CYC, and DDI, which are all cyanine dyes, have very large absorption bands and little Raleigh scattering when dissolved into NOA, and should perform well as filters. Waveguide dye filters can be fabricated in the same way that regular waveguides are fabricated, with the core material replaced by a dyed core material. An example of this is shown in Figure 4.5 where a cured waveguide section containing PCI dye is shown.

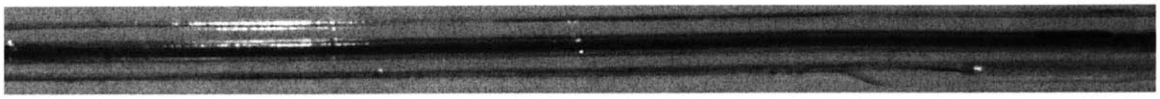

Figure 4.5. Picture of a waveguide embedded with Pinacyanol Iodide dye.

Dyes only act as good filters if their quantum yield is low. In this respect, many dyes in the visible wavelengths are not useful since they are manufactured specifically to fluoresce in applications such as laser dyes. Many cyanine dyes, in comparison, have very low quantum yields, with PCI at 0.001 and CYC at 0.007 [21]. A variety of other cyanine dyes with absorption peaks centered between 500 nm and 800 nm also have quantum yields under 0.01 [22].

To test the fluorescence output, a waveguide filled with PCI dye is excited at the input with a HeNe laser at 633 nm. A BG-3 color glass filter is placed at the waveguide output to block the excitation wavelength. The measured output fluorescence power to input excitation power ratio is 0.0005, indicating that PCI filters with little fluorescence.

In order to successfully attenuate the excitation source of a fluorescence measurement, the fluorescence due to the filter must be negligible in comparison to the sensor signal intensity. Since the dyes are embedded into the core of the waveguide, the amount of dye fluorescence actually captured by the waveguide is proportional to the numerical aperture of the waveguide. This decreases the unwanted fluorescence. The percentage of light collected by an emitting point source is the percentage of the emanating sphere captured by the waveguide given by

$$Efficiency = \frac{\int_0^{2\pi}\int_0^{\frac{\pi}{2}-\theta_c} \sin\psi\partial\psi\partial\theta}{4\pi} \qquad 4.1$$

where θ_c is the critical angle. For the waveguides fabricated with core refractive index 1.56 and cladding refractive index 1.43, the dye fluorescence capture efficiency is 4%. Therefore, for a dye that has a quantum yield of 0.001, the intensity of an absorbed light source at the filtered waveguide output is 0.00004 the original signal.

The expected guided fluorescence efficiency is lower than the measured efficiency by an order of magnitude. The measured quantum yield of PCI was 0.001 in methanol, where as the dye in the waveguide is dissolved in an entirely different material. It is most likely that the quantum yield of PCI when dissolved in NOA is higher than the quantum yield when dissolved in methanol.

4.2.2 Quantum Dot Light Sources

In the same way that low quantum yield organic dyes can be embedded into waveguides to provide filtering without coupling losses, high quantum yield dyes or nanoparticles can be embedded into waveguides to provide excitation with minimal coupling loss. With dyes, there is still the problem of selecting a proper dye which emits in the desired wavelengths, but with quantum dots, any color within the visible spectrum is possible to within 10 nm of accuracy. While this approach suffers from the same capture efficiency set by the index contrast of the waveguide, the ability to provide a single UV excitation for a variety of excitation wavelengths can reduce chip complexity if many different sensors are required.

Since the lifetime of the CdSe nanocrystals is on the order of 20 ns at room temperature [36], which is 3 orders of magnitude slower than the delay caused by the waveguide, the modulation frequency which can be supported by the nanocrystals in a waveguide will be limited by the nanocrystals themselves. A single exponential time constant of 20 ns results in a 3 dB frequency of 8 MHz, which is much faster than the required modulation frequency of 5 KHz for dyes such as PtOEPK. Also, since the collection efficiency of a point source emission in the waveguide is so low and 20 ns translates into a propagation distance of 3.8 m for light within the waveguide, other bandwidth reducing effects such as reabsorption and simultaneous emission from different spatial locations are negligible.

Processes for embedding quantum dots into polymers have been demonstrated [23], and companies that manufacture quantum dots such as Evident Technologies and Quantum Dot Corporation (QDC) offer dots in a variety of forms such as polymer mixtures, powders, and microspheres. A suspension of quantum dots into NOA has also been achieved through the aid of toluene, but small scale aggregation of quantum dots still occurs, leading to an increase in Raleigh scattering and poor optical quality as shown in Figure 4.6. For successful integration of quantum dots into NOA, the surface chemistry of the quantum dots would need to be changed to make a suspension without aggregation or precipitation.

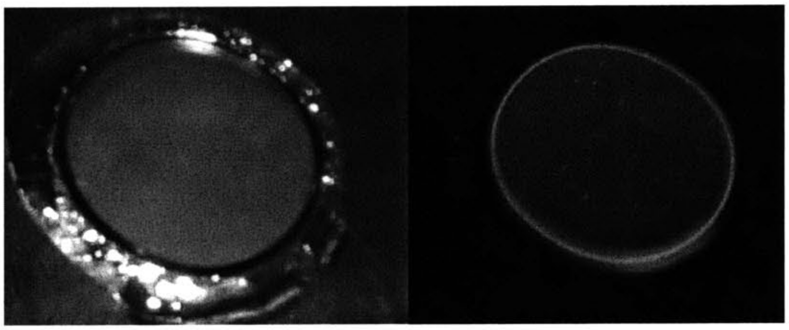

Figure 4.6. Cured 1mm thick sample of 585 nm emission quantum dots from QDC suspended in NOA71. (Left) A clouded solution results from the incompatibility between the NOA and the dots. (Right) Even though the dots noticeably aggregate, there is still fluorescence.

Since providing light sources in the form of quantum dots is indirect, it is necessary to determine how efficiently power can be collected and guided through the waveguide given an excitation LED. A model for quantum dot excitation and emission within a waveguide is given in Figure 4.7.

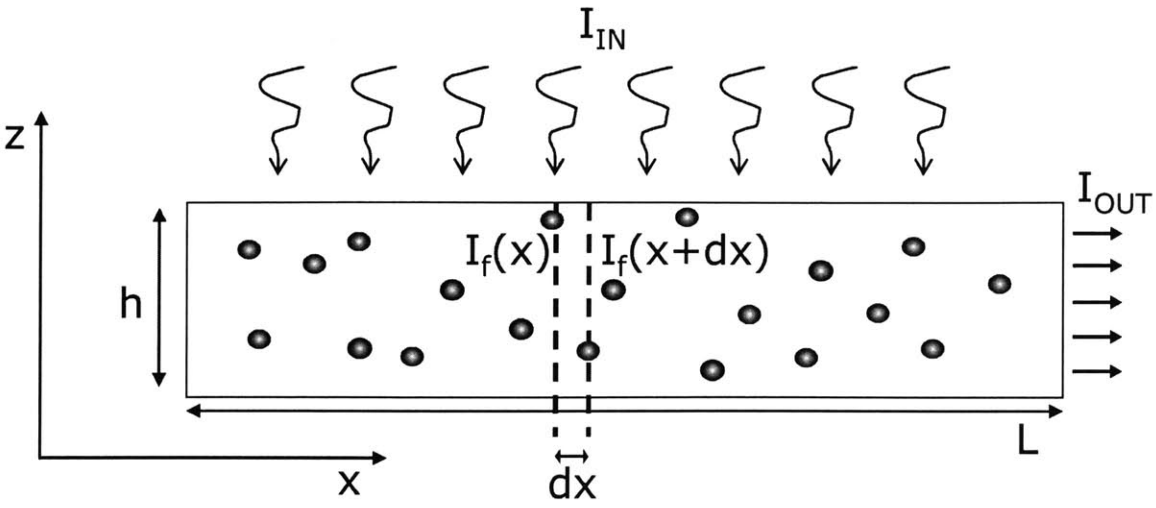

Figure 4.7. Model for determining the output efficiency of a quantum dot embedded waveguide given uniform overhead illumination. Quantum dot emission is collected by the waveguide and guided to the output.

From the model shown above, the absorption and emission properties of the quantum dots are treated as a bulk property of the material rather than as properties of individual particles. This will allow for a continuous model which is valid at high concentrations where the discrete locations of particles are not important. For this model to be accurate, the quantum dots must be uniformly distributed within the polymer. Two quantities must be known of the quantum dots in order to successfully model their behavior within the waveguide, their absorption coefficient versus wavelength and their quantum efficiency. These quantities will determine how much of the pump light will be absorbed and how much emission will be guided to the output.

Assuming that the quantum dot embedded waveguide section is under uniform ultraviolet illumination in the z direction, we can obtain the average intensity incident on the dots in the x direction taking into account UV dot absorption

$$I_{0avg} = \frac{1}{h}\int_0^h I_{IN}\exp\left(-\mu_{dot}^{uv} z\right)dz = \frac{I_{IN}}{\mu_{dot}^{uv} h}\left[1-\exp\left(-\mu_{dot}^{uv} h\right)\right] \qquad 4.2$$

Where μ_{dot}^{uv} is the absorption coefficient of the quantum dot at the pump wavelength, I_{IN} is the uniform input intensity over the entire surface of the waveguide, and h is the height of the waveguide.

Next, we determine the amount of absorbed and emitted light that occurs in a differential distance along the waveguide direction x, which is given by

$$I_f\left(x+dx\right)=I_f\left(x\right)-I_f\left(x\right)\mu_{dot}^{em}dx+\zeta\Phi I_f\left(x\right)\mu_{dot}^{em}dx-I_f\left(x\right)\mu_{poly}dx+\zeta\Phi I_{0avg}\mu_{dot}^{uv}dx \tag{4.3}$$

Where $I_f\left(x\right)$ is the intensity of light at the emission wavelength, μ_{dot}^{em} is the absorption coefficient at the emission wavelength, ζ is the collection efficiency of the waveguide, Φ is the quantum yield of the quantum dot, and μ_{poly} is the absorption coefficient of the polymer at the emission wavelength.

After collecting terms and taking the limit as dx → 0, we arrive at a simple first order differential equation.

$$\frac{dI_f}{dx}=I_f\left(\zeta\Phi\mu_{dot}^{em}-\mu_{dot}^{em}-\mu_{poly}\right)+\zeta\Phi I_{0avg}\mu_{dot}^{uv} \tag{4.4}$$

Solving with the initial condition of no pump light in the x direction, or $I_f\left(0\right)=0$, we obtain a solution of the form

$$I_f\left(x\right)=\frac{\zeta\Phi I_{IN}}{\left(\zeta\Phi\mu_{dot}^{em}-\mu_{dot}^{em}-\mu_{poly}\right)h}\times\left[1-\exp\left(-\mu_{dot}^{uv}h\right)\right]\times \\ \left[\exp\left[\left(\zeta\Phi\mu_{dot}^{em}-\mu_{dot}^{em}-\mu_{poly}\right)x\right]-1\right] \tag{4.5}$$

In order to apply Equation 4.5, material parameters for both the polymer and the quantum dots must be measured. Quantum yield is high for quantum dots, but is still hard to measure accurately since actual quantum yield depends on processing parameters such as curing time and exposure to ambient air. Absorption coefficients versus wavelength can be measured following the same approach used for organic dyes and plots of the absorption coefficients for different dot-polymer resins provided by Evident Technologies are given in Figure 4.8.

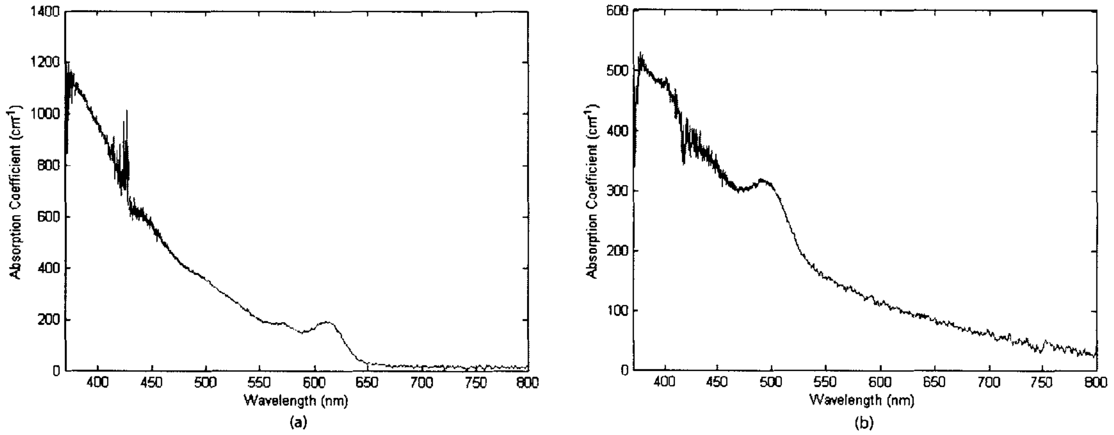

Figure 4.8. Plot of the absorption coefficient versus wavelength for a red quantum dot (a) and a green quantum dot (b) at room temperature suspended in a cured polymer. The noise at 450 nm occurs because two light sources were necessary to extend the measurement below 450 nm. Both the white light source and the UV led used had very low intensity in the noisy region.

For a red quantum dot, the concentrated solution yields absorption coefficients at 380 nm and 630 nm as approximately 1000 cm^{-1} and 100 cm^{-1} respectively. If the concentration is changed, these coefficients should scale linearly as per Beer's law. Assuming uniform illumination of the quantum dot embedded waveguide section, it is clear that the maximum power output will occur for an infinitely long waveguide so it is more appropriate to talk about the efficiency of the waveguide. Converting Equation 4.5 into power efficiency, we arrive at

$$\frac{I_f(L)\times w\times h}{I_{IN}\times w\times L}=\frac{\zeta\Phi}{\left(\zeta\Phi\mu_{dot}^{em}-\mu_{dot}^{em}-\mu_{poly}\right)L}\times\left[1-\exp\left(-\mu_{dot}^{uv}h\right)\right]\times \left[\exp\left[\left(\zeta\Phi\mu_{dot}^{em}-\mu_{dot}^{em}-\mu_{poly}\right)L\right]-1\right] \tag{4.6}$$

where $w \times h \times L$ specify the dimensions of the quantum dot embedded waveguide. A plot of the power efficiency for a 10:1 weight dilution of concentrated dots in polymer resin is given in Figure 4.9 using the waveguide parameters calculated from simulation.

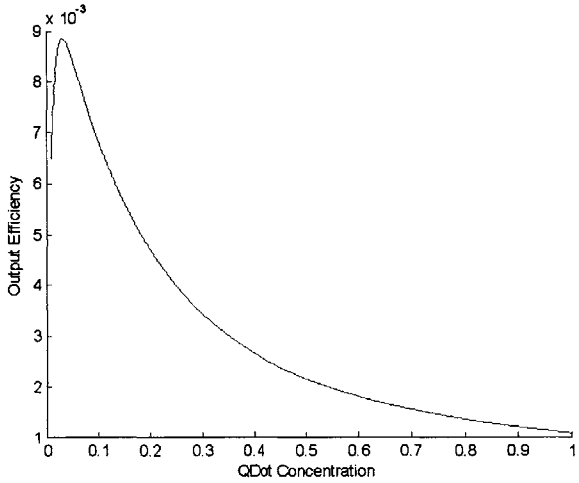

Figure 4.9. Plot of the output power efficiency for uniform overhead illumination of a quantum dot embedded waveguide section. The waveguide dimensions are 1x1x1 mm^3, the 380 nm and 630 nm absorption coefficients are 1000 cm^{-1} and 100 cm^{-1} respectively, the dot concentration is 10%, and the quantum yield and waveguide collection efficiency are assumed to be 0.9 and 0.012 respectively.

The plot shown uses parameters given in the absorption curve of Figure 4.8 and the collection efficiency resulting from the effective numerical aperture measured in Figure 2.18. It can be seen that the power efficiency of the waveguide section has a maximum which depends on the concentration of dots in the waveguide. Therefore it is important to optimize the concentration for a given set of waveguide dimensions.

By combining the fabrication process used to make waveguides with the quantum dot embedded waveguide sections, devices can be made which contain many different wavelength excitation sources that only require a single LED to activate. This is demonstrated with the waveguide fan shown in Figure 4.10 using quantum dots pre-suspended into polymer by Evident Technologies. The polymer suspension provided was incompatible with NOA, requiring that quantum dot sections and NOA sections were cured separately.

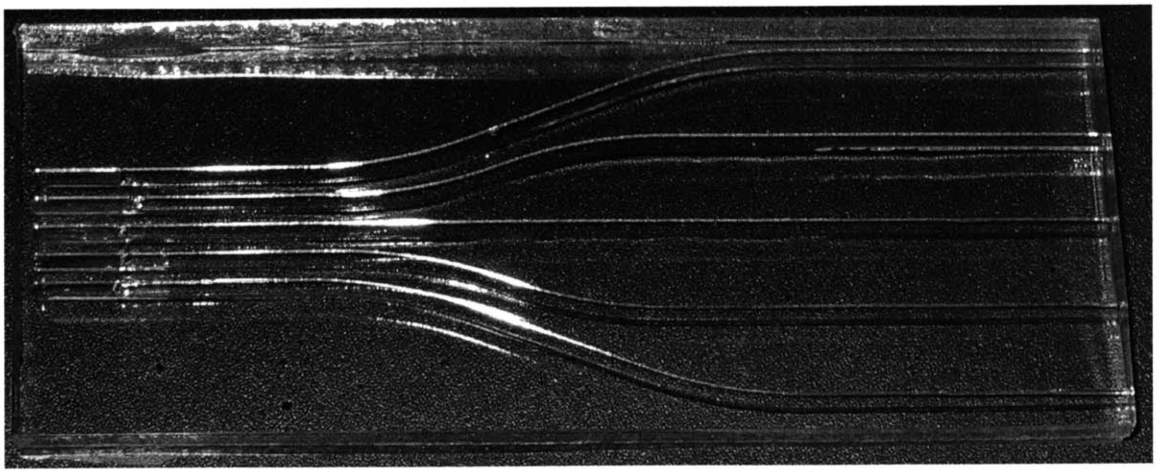

Figure 4.10. Picture of the waveguide quantum dot light source device. A high power ultraviolet LED is used to illuminate the section concentrated with quantum dot embedded waveguides. Dot emission is collected by the waveguide structure and guided to each output.

The five sections are compacted into a square at the input end and the emitted light is captured and expanded at the output end. All five quantum dot polymer sections are diluted 10:1 from the initial concentrations. Focused illumination of the first millimeter of the red waveguide shows an efficiency of 0.0023 which is on the same order as the calculated efficiency of 0.0068 for a 1 mm long sample of the same cross section dimension shown in Figure 4.9. The normalized spectrum of each waveguide output is shown in Figure 4.11.

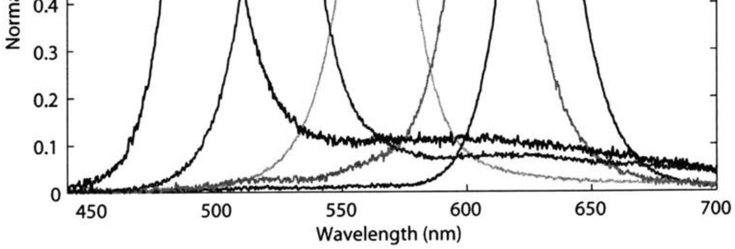

Figure 4.11. Plot of the emission intensity versus wavelength for the five quantum dot outputs. Emission tails that stretch into the infrared are more prominent for shorter wavelength dots.

While most of the light is centered at the emission wavelength, there are large tails that leak into the infrared for smaller dots. While this is not seen clearly on a linear scale, a log scale as shown in Figure 4.12 for the 565 nm emission dot shows a 1% tail extending well into the near infrared spectrum. These large tails are most likely due to surface imperfections or oxidation resulting from curing the samples in air.

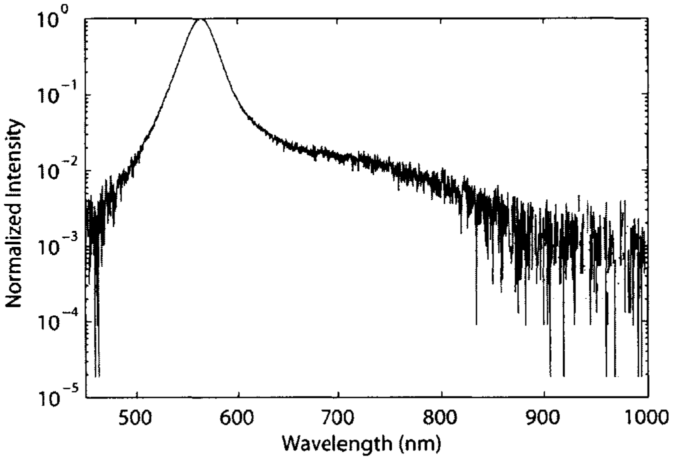

Figure 4.12. Log plot of the emission intensity versus wavelength for the 565 nm emitting quantum dot clearly showing a long wavelength tail.

This can create noise in sensitive fluorescence measurements wishing to use quantum dots as excitation sources. For example, for a fluorescent dye sensor that absorbs at 570 nm and fluoresces at 760 nm, filtering out the quantum dot's main excitation source will be insufficient since the source also emits at the fluorescence wavelength. In this case, a filter to block 760 nm light must be placed immediately after the quantum dot source in addition to the 570 nm filter placed at the fluorescence output in order to be certain that there is no intensity noise coming from the source.

For quantum dots to be truly successful as on-chip excitation sources, either appropriate waveguide filters must be integrated to suppress the tails of each dot emission, or the quantum dots inside the polymer resins must be of higher quality.

4.3 Passive waveguide devices

In addition to filtering signals and providing excitation sources, successful delivery and collection of signals to and from on chip sensors is necessary. For chips that require very few sensors, excitation and collection is usually performed using fiber bundles, which then route the signals to photodiodes that can measure the sensors with less background noise. Unfortunately, for larger arrays of on-chip systems, this approach becomes not only expensive, but also large since most measurements must be done perpendicular to the chip plane. The ability to provide sensing with waveguides can reduce the number of fibers necessary for the detection system and also provide devices such as power splitters and power combiners cheaply, which will be able to merge or expand signals and reduce off chip components. In an ideal design, one could have a single LED and photodiode for each type of sensor in an arbitrarily large array of devices.

4.3.1 Waveguide Bends

There have been many efforts to provide in-plane optical sensing. Direct addition of fiber coupled waveguides alongside microfluidic channels [3, 4] allow for both fluorescence and density measurements, but the design of the fluidic system directly affects waveguide placement. Other approaches, such as molecularly imprinted polymer waveguides or evanescent-wave sensors [24, 8], suffer from the fact that detection sites on the fluidic chip must be in contact with the waveguide sensors to function, which can result in contamination. Therefore, an approach that is in-plane but able to provide the functionality and versatility of out-of-plane detection is useful. The most logical approach to achieving this goal is to integrate bends into waveguides. Two possible methods exist for bending light in multimode waveguides at sharp angles, total internal reflection and mirrors.

Total internal reflection bends require a large index contrast to be effective. To achieve a 45 degree bend with a core index of 1.56, the cladding index must be at most 1.1, which is not feasible with normal polymer materials or even with nanoporous dielectric materials [25]. For a 30 degree bend, the cladding index necessary becomes 1.35, which is still low, barely approaching the lower refractive index polymers such as Teflon and other fluorinated polymers. Since it is difficult to find materials with such low index of refraction, the alternative is to provide air interface bends to create the index difference. The use of air interface bends for low index contrast waveguides has been explored [26], and while air interface bends can be very effective, the fabrication approach used for waveguide fabrication outlined in Section 2.3 is strictly an additive process after initial polycarbonate mold fabrication. Therefore it is very difficult to create TIR based bends. A fabrication approach which can accomplish this is shown in Figure 4.13.

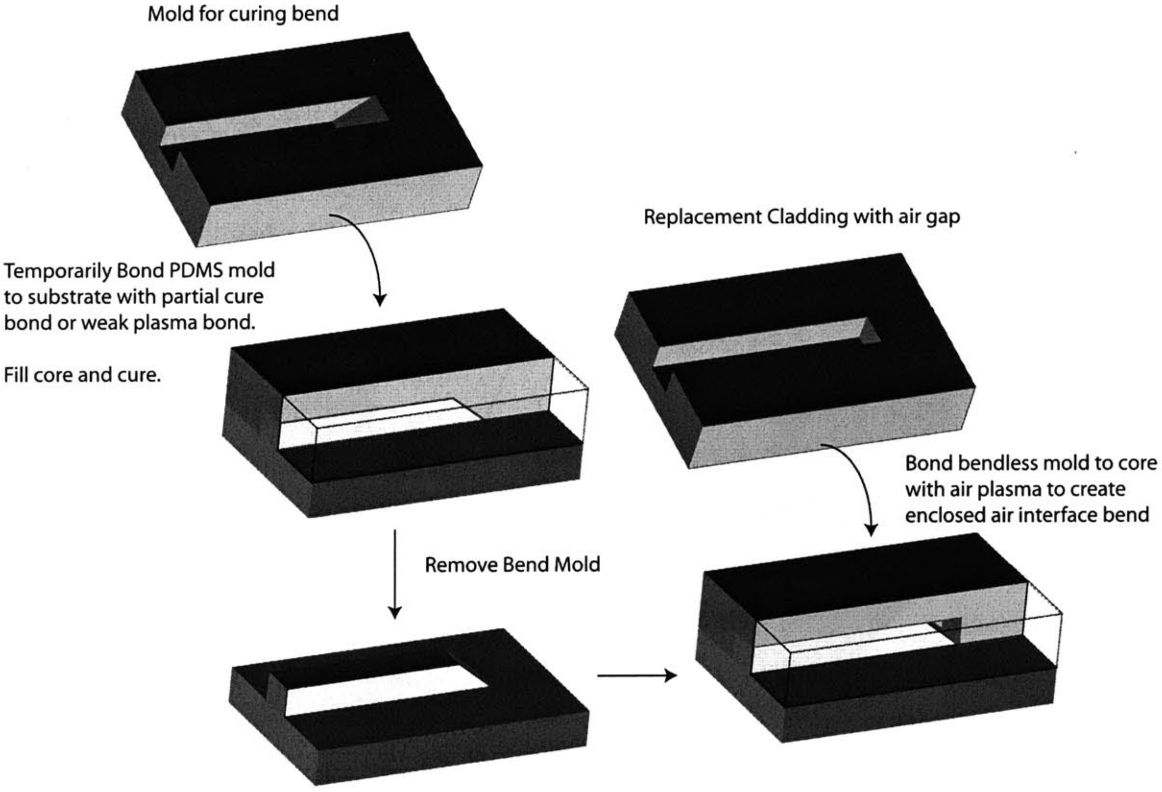
Mold for curing bend
Temporarily Bond PDMS mold to substrate with partial cure bond or weak plasma bond.
Fill core and cure.
Remove Bend Mold
Replacement Cladding with air gap
Bond bendless mold to core with air plasma to create enclosed air interface bend

Figure 4.13. Illustration of the fabrication process necessary to create waveguide bends which utilize total internal reflection.

As can be seen in Figure 4.13, a mold that has the negative bend structure is necessary to cure the core material into the shape of the bend. Then an elaborate process is performed where the bend mold is removed and a new PDMS mold with an extra space for an air gap is bonded. While this is feasible, this type of processing will result in easy delamination between polyurethane and PDMS since bonding between the core and the cladding occurs when both are already in their cured solidified state. In addition to delamination, there is now an extra alignment step which must be performed to bond the final cladding to the substrate containing waveguides.

Mirror bends act the same way as TIR bends, except that there is no longer a critical angle which restricts the bending angle. Using this approach, the index contrast between the core and the cladding is no longer important and power leakage through the bend interface becomes negligible. Even though leakage through the bend is removed, additional leakage from light that is over-reflected becomes an issue as can be seen in Figure 4.14. Ray tracing simulations estimate that the TIR bend efficiency is 60% where as the mirror bend efficiency is 80%. According to simulations, the mirror bend is more effective and does not require structurally unstable, fabrication extensive structures such as air gaps.

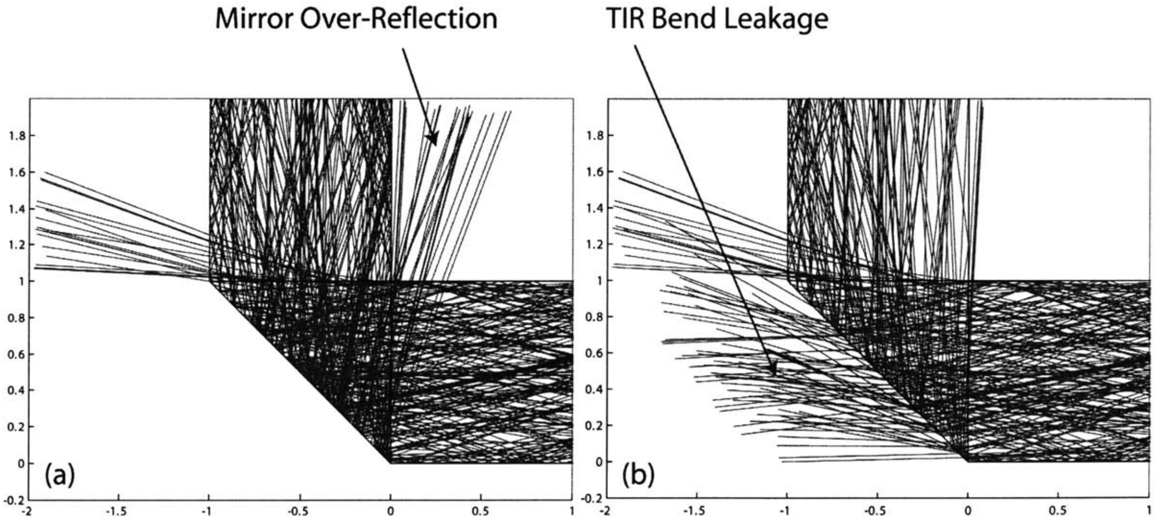

Figure 4.14. Ray tracing simulations of a 45 degree bend structure using a mirror interface (a) and an air interface (b).

Effective mirrors in the visible spectrum can be made out of metals such as silver and gold, and studies of evaporating metals on PDMS have been performed [27]. Silver evaporates like gold, with a grainy surface quality when directly evaporated on PDMS, and a more mirror-like surface quality when PDMS is first treated with oxygen plasma to increase its surface energy as shown in Figure 4.15.

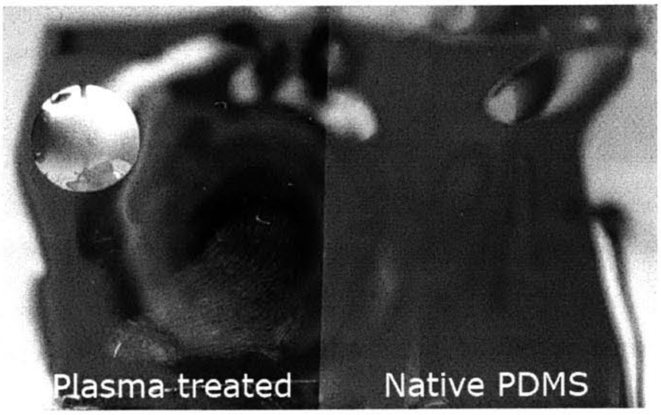

Figure 4.15. Results of E-beam evaporated silver mirrors on (left) an air plasma treated PDMS surface and (right) a native PDMS surface. The camera reflection from the native PDMS surface is much more diffuse than the camera reflection from the plasma treated PDMS surface.

Unlike gold on PDMS, silver adhesion does not seem to increase when directly evaporated on an oxygen plasma treated PDMS substrate. In experiments, silver films evaporated on natural and air plasma activated PDMS can be ripped off by sticking tape to and then removing the tape from the silver surface. Since the silver adhesion to PDMS is poor, it is very easy to fracture the layer, leading to cracks and roughness. Therefore it is important to reduce the stress experienced by the stamp after evaporation. While the interface between the silver and core NOA polymer matters more than the interface between the silver and PDMS, the lack of adhesion between the silver and PDMS causes problems during NOA curing. Since NOA shrinks by 4% after curing and also binds strongly to metals such as silver, the shrinking process actually stresses the silver and causes cracks. A comparison of the flat silver layer before and after NOA curing is shown in Figure 4.16.

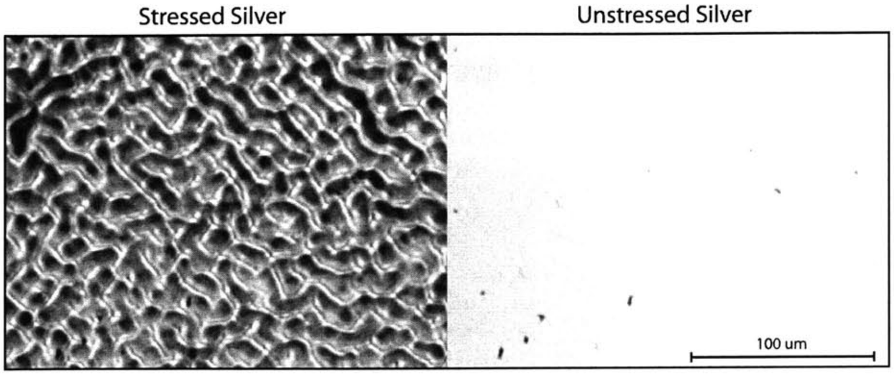

Figure 4.16. Comparison between (left) silver surface stressed by cured NOA and (right) the original silver surface.

Stress induced fractures are visible on the silver film after the NOA is cured. Reflectivity measurements also determine a reflectivity of 90%, a reduction in reflectivity from the originally evaporated silver film reflectivity of 98%, and a large spread angle in the reflected beam, indicating that the microcracks affect the beam spread angle similar to the surface roughness from the polycarbonate. For out-of-plane bends, the beam spread resulting from the rough silver surface does not greatly hinder performance since after the beam reflects off the silver surface, it is expected to leave the waveguide.

The process for making out-of-plane bending mirrors is straight forward using E-beam evaporation as shown in Figure 4.17.

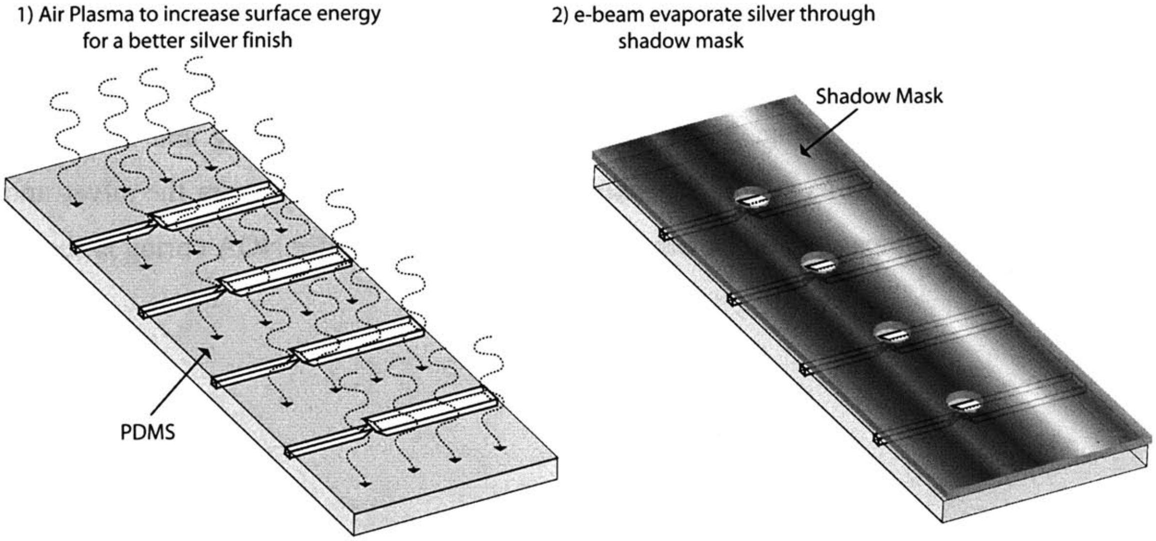
1) Air Plasma to increase surface energy for a better silver finish
PDMS
2) e-beam evaporate silver through shadow mask
Shadow Mask

Figure 4.17. Illustration of the silvering process required for making mirrors on PDMS bend interfaces.

To determine the reflectivity of silver evaporated onto PDMS, air plasma treated samples of PDMS were metallized with 250 nm of silver in an e-beam evaporator. Reflectivity and slope variance measurements conducted with the setup shown in Figure 2.19 show a silver mirror reflectivity of 98% and 95% at 632 nm with a HeNe laser for the left and right samples in Figure 4.15 respectively, and a slope variance of nearly 0, indicating a very smooth and flat surface. However, these reflectivity values can be decrease dramatically for fractured mirror surfaces which allow light penetration through cracks. Fractures can occur easily due to the elastic nature of PDMS.

In addition to providing silver mirrors, the actual bend structures on which the silver will be evaporated must also be fabricated. With other fabrication procedures, making angled structures requires very special control such as KOH etching of specially cleaved silicon wafers, razor blade cutting, or angled laser ablation. In comparison, direct machining can accomplish bends easily since tools of arbitrary angles can be made. If a particular out-of-plane bend angle is required, a tool cut to that angle can be used to create the angled structure. For in-plane bends, the bend angle is mostly independent of the milling process. Exceptions include bend radii which are smaller than the tool radius, sharp 45 degree bends in a positive molding scheme, and sharp 90 degree corners in a negative molding scheme. While perfect 45 degree bends result in 80% transmission, they are not easy structures to fabricate in a conventional milling process. Therefore it is also appropriate to look at gradual bends as an alternative. Simulations of the transmission efficiency for gradual bends are shown in Figure 4.18. At the local maximum, the in-plane bend achieves a transmission of 78%, which is comparable to the sharp 45 degree bend transmission of 80%. This local maximum occurs at a smaller bending radius because the silver bend acts as a reflector for all modes, unlike the TIR bend. In this case, the smaller bend radius would act more like a 45 degree mirror than a gradual bend and would approach the efficiency of an actual 45 degree bend.

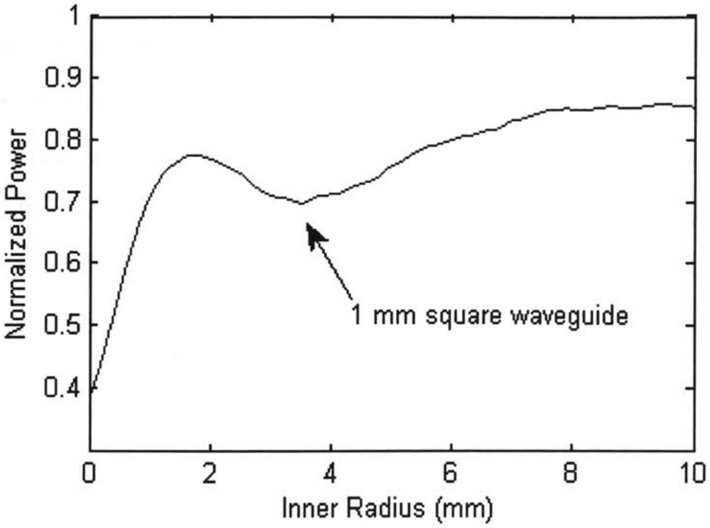

Figure 4.18. Ray tracing simulation of the transmitted power efficiency versus inner bending radius for a 1mm square waveguide. A local maximum in power transfer occurs at 1.7 mm and a local minimum in power transfer occurs at 3.5 mm.

Fabricated in-plane silvered bends and out-of-plane silvered bends with a 250 nm and 300 nm e-beam evaporated silver layers respectively are shown in Figure 4.19. The in-plane bend radii are chosen at the local maximum and local minimum intensity as predicted by simulations shown in Figure 4.18 from the ray tracer. The bottom silvered surfaces for the in-plane bends show higher degrees of cracking after curing NOA as discussed previously, while the out-of-plane bend with a thicker silver layer does not exhibit as much deformation and cracking. This indicates that the stress induced cracking can be minimized by evaporating thicker silver films (> 300 nm).

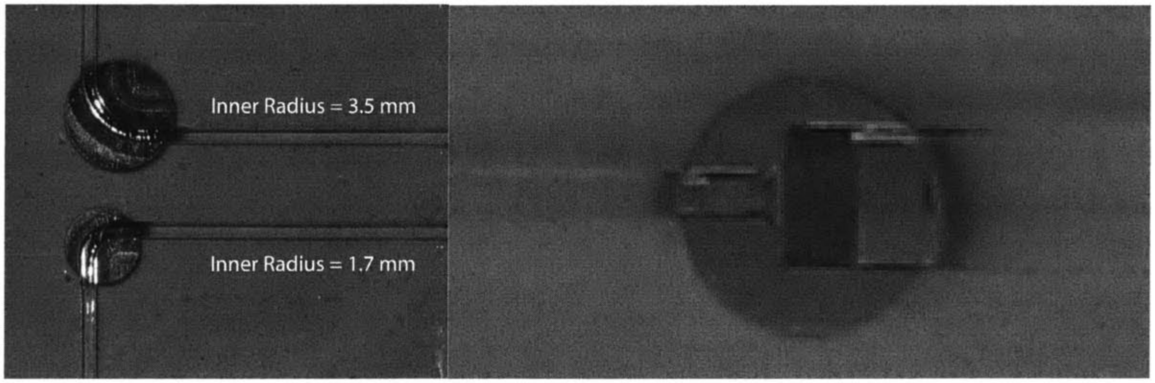

Figure 4.19. Pictures of fabricated in-plane silvered bends (left) and out-of-plane silvered bends (right). NOA induced stress is also clearly visible in the diffuse reflection of the out-of-plane silvered bend.

Measurements of the transmission through the in-plane bends were performed by coupling a HeNe laser at the input with a 38 degree half angle spread and measuring the transmitted power at the output. The measured power efficiency for the 1.7 mm inner radius bend and 3.5 mm inner radius bend were 8% and 4% respectively. In comparison, for the same bends without silvered sidewalls, the simulated efficiencies were 17% and 31% and measured efficiencies from a fabricated version were 14% and 28% respectively. This suggests that the majority of loss for the silvered bends actually results from the silver deposition. Since e-beam evaporation is very directional, the amount of silver actually deposited is too thin to contribute reflective properties. This was verified by physically removing the waveguide core and examining the degree of silver deposition on the sidewalls. All in-plane silvered bends show a translucent deposition of silver on the sidewalls, indicating that the deposited layer thickness is very thin (~10 nm). The combined effects of a translucent silver sidewall and the cracks visible on the bottom silver surface result in an output transmission through the silver bend which is worse than the TIR bend. Table 4.3 summarizes the simulated and measured results for the two bends.

	Simulated	Measured
Silvered R = 1.7 mm	78%	8%
TIR R = 1.7 mm	17%	14%
Silvered R = 3.5 mm	70%	4%
TIR R = 3.5 mm	31%	28%

Table 4.3. Summary of the simulated and measured power efficiencies for silvered and TIR in-plane bends.

In contrast to the low in-plane silvered bend transmission, the vertical silvered bend transmission is very high at about 72%. The reduction in reflectivity for the vertical bend from a flat evaporated silver mirror on PDMS results mainly from machined surface roughness rather than from a non-ideal silver deposition. The non-ideal silver deposition for the in-plane bends occurs because the e-beam evaporation direction is parallel to the surface being silvered. This does not occur for the vertical bends since the surface being silvered is mostly perpendicular to the evaporation direction.

4.3.2 Power Splitters and Combiners

In order to perform measurements in a multiplexed fashion, it is necessary to incorporate either power splitters or power combiners to reduce either the number of LED's or photodetectors. To demonstrate the possibility of power combiners on the described fabrication platform, a 2 to 1 power combiner utilizing the silvered bends was also fabricated on the same mold as the in-plane bends. The purpose of this was to make sure the silvered sidewall measurements for the bends and the power combiner would match.

In order to create a power combiner which has little loss, the combined waveguide should be twice the size of the input waveguides, in order to accommodate all of the modes which enter the combined waveguide. For multiple inputs, this is impractical since the combined waveguide can become larger than the detection area of the photodiode. Therefore an alternative approach is to taper the combined guide to reduce its size as shown in Figure 4.20.

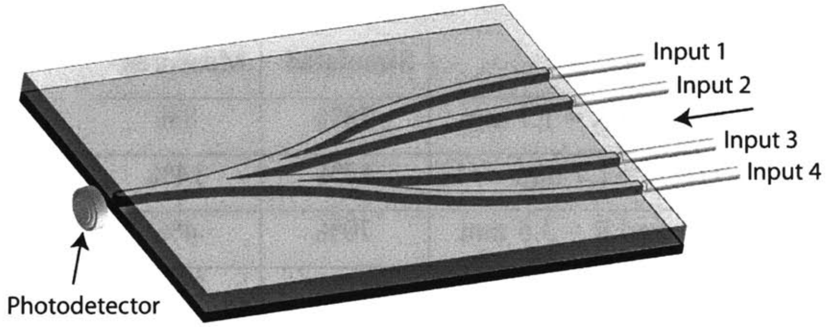

Figure 4.20. Illustration of a tapered waveguide power combiner. Four different fluorescence signals can be combined on-chip and sent to a single photodetector.

This type of design provides benefits for both serial and parallel detection. For serial detection, multiple sensors can be routed to the same detector with each sensor using the detector in sequence. For parallel detection, the excitation source for each sensor can be modulated at a different frequency and signals can be separated in software with proper filtering, rather than using a single photodetector for each fluorescence sensor.

While fabrication of this device is simple using a positive molding scheme, it was shown previously that a positive molding scheme generates additional loss due to increased surface roughness. Gradual combiners with a negative molding scheme are much more difficult, since the resolution of separation distance between two waveguides is limited by the size of the end mill. While this does not make the design of a combiner impossible, it results in a larger taper, in the case of Figure 4.21, a taper from a 3 mm width to a 1 mm width since each waveguide is 1 mm wide and the end mill is 1 mm in diameter.

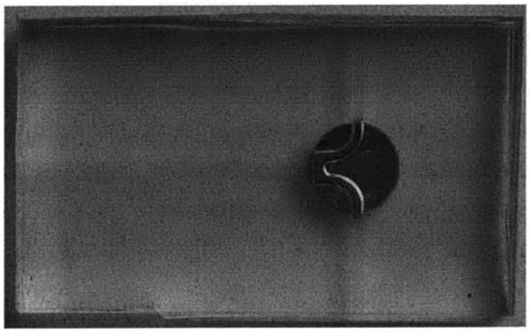

Figure 4.21. Picture of the bend integrated waveguide power combiner. A 3 mm to 1 mm taper is implemented to reduce the final combined waveguide size to a 1 to 1 aspect ratio.

Measurements were performed by coupling in a HeNe laser at each of the two inputs and measuring the resulting output power at the combined waveguide output. As with the silvered in-plane bends described previously, the 250 nm deposition of silver on the bend interfaces did not result in a thick enough layer of silver on the sidewalls to enable efficient reflection. The leakage resulting at the bend interface is visible in Figure 4.22. Each input waveguide results in a transmission through the taper of 9%, which, in comparison to the loss measured for the single bends without silver, indicates that silvering the sidewalls with e-beam evaporation reduced the reflectivity.

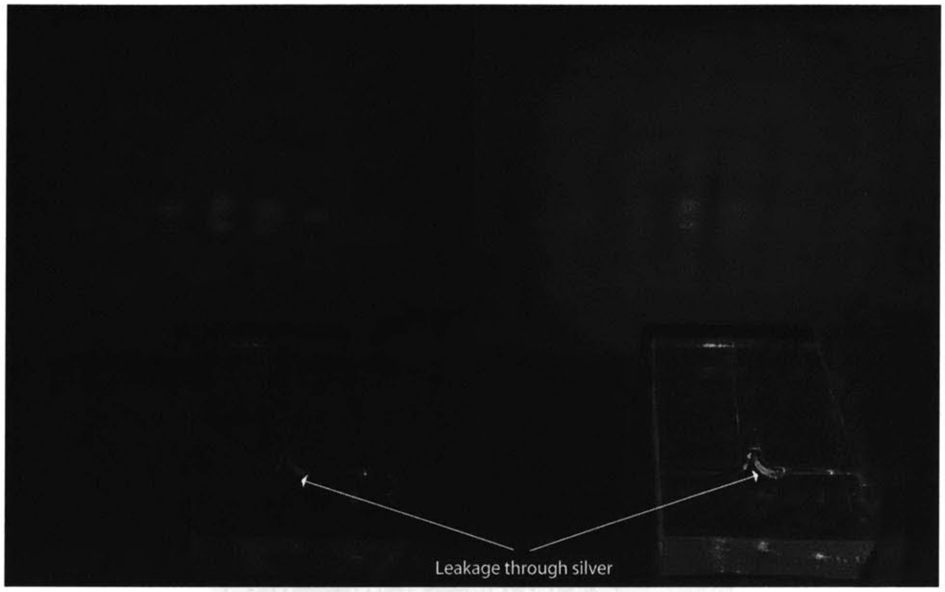

Figure 4.22. Picture of the bend integrated power combiner. From the picture, it is clear that through transmission occurs at the silver bend interface, indicating that the intended silver mirror is not thick enough to act as a mirror surface. The output mode profile for an input of 7 degrees (left) and an input of 38 degrees (right) suggests that the taper is responsible for the output profile.

A ray tracing simulation of the integrated silvered bend and taper structure shows that there should be a total transmission of 30%, suggesting that the taper is quite lossy since the silvered bend transmission should be 78%. Even though the measured loss through the fabricated combiner structure is well below the expected value due to the poor silver deposition, the visible output profile demonstrates that the structure is performing as expected. Simulations of the output mode profile shown in Figure 4.23 very well match the experimentally observed mode profiles from Figure 4.22. It is also clear that this mode profile is due to the taper rather than the bend, since experimentally measured silver bends made in the same process have no indication of a discrete output mode pattern and varying the input numerical aperture does not change the horizontal mode spread angle.

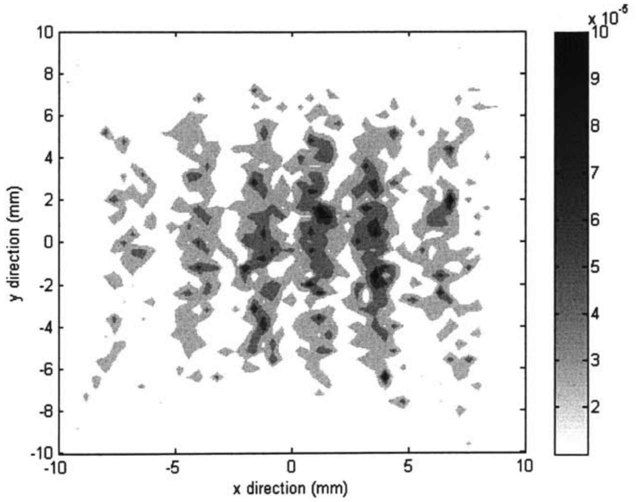

Figure 4.23. Output intensity 10 mm away from the tapered output face showing the discrete line profile due the silvered bend tapered combiner.

4.4 Summary

A variety of devices are possible within a polymer platform through modification of polymers either internally or externally. The addition of these devices, filters, light sources, bends, and power splitters, allow for the development of compact optical sensing systems which can be integrated directly onto polymer chips. Integration of optical components combined with proper design will improve the scalability of polymer based arrays requiring optical sensors.

It has been shown that effective filters can be produced by dissolving dyes into polyurethane, either directly or with the aid of toluene. In the same way, self-aligned light sources can be created by dissolving quantum dots into polyurethane. These types of modifications to the waveguide core result in multifunctional polymer materials, improving the integrability of optical components with polymer chips.

Utilizing the fabrication process developed, direct fabrication of molds containing varying depth structures is possible, allowing for the integration of optical elements such as prisms and bends. In addition, power splitters and power combiners have been demonstrated with efficiency currently limited by the bend silvering process and the need for a taper to reduce the physical aperture of the combined waveguide.

Chapter 5

Integrated Oxygen Sensors

5.1 Introduction

Waveguide based devices are useful for integrating optical devices with sensors since they are compact and able to provide multiplexing functionality. With the large set of components developed in Chapter 4, it is necessary to choose an application which demonstrates the scalability and functionality of a multimode waveguide platform. The goal is to create an optical backplane which exists separately from the microfluidic chip, but can be aligned and integrated with the chip to provide waveguide functionality. In this way, the design of the microfluidic system is not affected by design of the optical system.

A device which can be miniaturized with the aid of an integrated optical platform is the cell culture array [28]. Sensors used in this device consist of dyes that have chemically sensitive fluorescence lifetimes. As stated in Chapter 1, measurements with lifetime based dyes require sufficient intensity modulation such that the phase delay due to the dye lifetime is measurable. Since photodetectors are more suitable for detecting higher frequency signals, waveguides are beneficial for reducing the complexity of photodetector design and placement.

The first step towards creating a fluorescence sensing platform is to develop an efficient waveguide design. Section 5.2 explores a few different waveguide designs and optimizes a particular design for maximum collection efficiency of sensor fluorescence. After designing the waveguide system, Section 5.3 discusses the fabrication of the device, which combines milling, vapor polishing, and mirror bends. After device fabrication, measurements will be performed in Section 5.4 using a specific oxygen sensor, PtOEPK, which exhibits low photobleaching [29]. The sensors will first be calibrated, and then the entire waveguide device will be integrated onto a fluidic channel to perform measurements on the diffusion of oxygen through PDMS into a flowing liquid.

5.2 Design and Simulations

The optical backplane is intended to provide the same functionality available with fiber bundles. Out-of-plane sensor excitation and fluorescence collection makes fiber bundles ideal for optical sensors since the fibers do not interfere with microfluidics. But in order to make waveguides which can be integrated with the microfluidic chip, the waveguides must be level with and not perpendicular to the fluidics. By integrating bends to the end of the waveguides, light can be forced out-of-plane while maintaining an overall planar structure.

Many waveguide designs are capable for performing excitation and collection and the most important parameter of any excitation-collection system is the signal to noise ratio of the system with respect to excitation leakage. Therefore, it is necessary to determine how much excitation light leaks through the system and reaches the output. Assuming that the sensor is made of a transparent material, leakage in the system is dominated by the reflection at the sensor interface. Even though this light can be filtered, filter fluorescence is an issue, and reducing the excitation leakage will still improve the signal to noise ratio of the system. Simulations of the excitation leakage are performed by sending a uniform pulse through the input faces and summing the intensities of the resulting rays that exit the systems.

For all designs explored, it is assumed that the sensors are 1 mm above the waveguide plane, most likely enclosed in either a channel or a well. Excitation area profiles are taken by generating hit statistics at the sensor plane with a random 10000 ray trace assuming a uniform point source emitting at the waveguide input. Collection area profiles are generated by creating a grid of point sources emitting into the waveguide bends from the sensor plane. Each point source represents a location on the sensor, and the total collected signal at the output is found for each point to determine the collection efficiency at each sensor location.

Since passive waveguides are reciprocal devices, a simple approach would be to use the same excitation waveguide as the collection waveguide and fabricate a 45 degree mirror at the excitation site. In this way, the largest intensity excitation arrives at a location where a large angle of fluorescence can be collected. A simulation of the system is shown in Figure 5.1.

While the excitation incident on the fluorophore overlaps the collection area of the waveguide, a 2 to 1 power splitter is still necessary to channel the collected fluorescence away from the excitation source, so a 50% decrease in power must be factored into the total collected intensity. Assuming that the fluorophore has an infinite absorption cross section and a 100% quantum yield, multiplying the excitation area by the collection area yields the total collected fluorescence efficiency, which 0.0081.

Since the simulations indicate a maximum intensity near the end of the 45 degree mirror, an alternative approach is to place a collection waveguide near the excitation waveguide to eliminate the need for a splitter. From a fabrication point of view, the easiest design is to place the collection guide opposite the input guide as shown in Figure 5.2.

While this method removes the need for a 2 to 1 power splitter, collected signal intensity is reduced due to the mismatch between the excitation and collection areas. This results in a total conversion efficiency of around 0.004 assuming a perfect fluorophore. To fix this problem, the bending angle can be adjusted so that the excitation and collection areas again overlap. This occurs near a bending angle of 30 degrees. Again, a ray trace of the guide as well as the excitation and collection areas is shown in Figure 5.3.

By overlapping the excitation and collection areas, the overall efficiency is increased by a factor of 3. While the pump leakage also increases with respect to the 45 degree bend case, the signal spread resulting from propagating between the bends and the sensor help to suppress coupling of excitation light into the collection waveguide.

Looking at the excitation area, the overall signal intensity could be increased if the collection area could be widened to encompass the excitation area more fully. This can be accomplished by simply widening the collection waveguide. However, since pump leakage is suppressed by the small collection guide, widening the collection area should also increase the collection of reflected excitation light. A simulation with a 3 times larger collection waveguide width is shown in Figure 5.4.

It is clear that enlarging the collection area is only beneficial so long as the excitation area notices the change. Even though the collection efficiency is increased for this system, the leakage is increased as well. The four systems described are summarized in Table 5.1 with their conversion efficiency and excitation leakage. The results of a ray tracing simulation of the efficiency and leakage of the fiber bundle from Chapter 1, consisting of a central 1 mm diameter fiber and 9 surrounding 500 μm diameter fibers, are also shown in comparison.

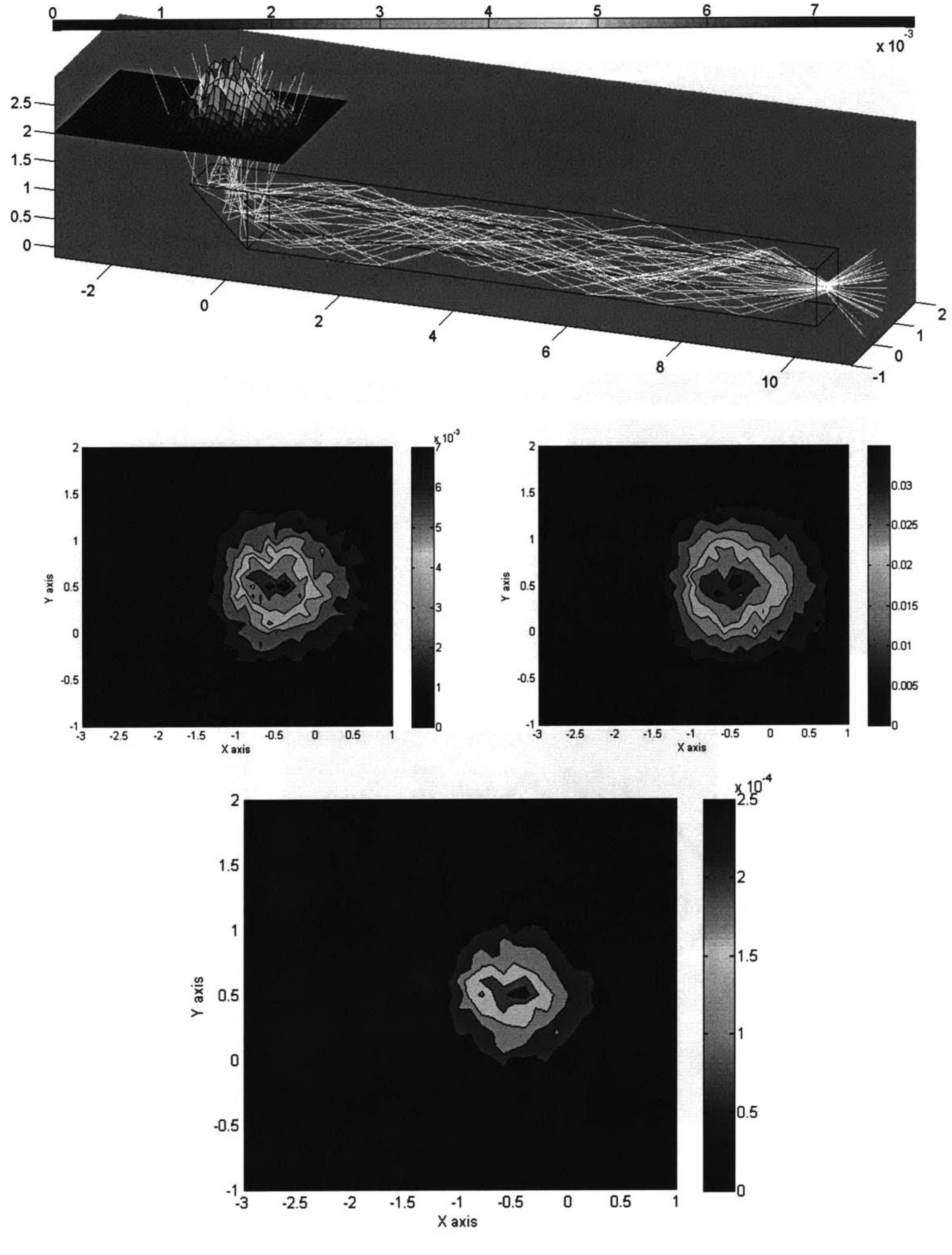

Figure 5.1. Simulations of the fluorescence back collection system with a 45 degree. The sensor excitation (left) and collection (right) areas overlap perfectly, and lead to a high total efficiency (bottom) for fluorophores located on the sensor plane. Unfortunately the difficulty is in separating the excitation from the fluorescence.

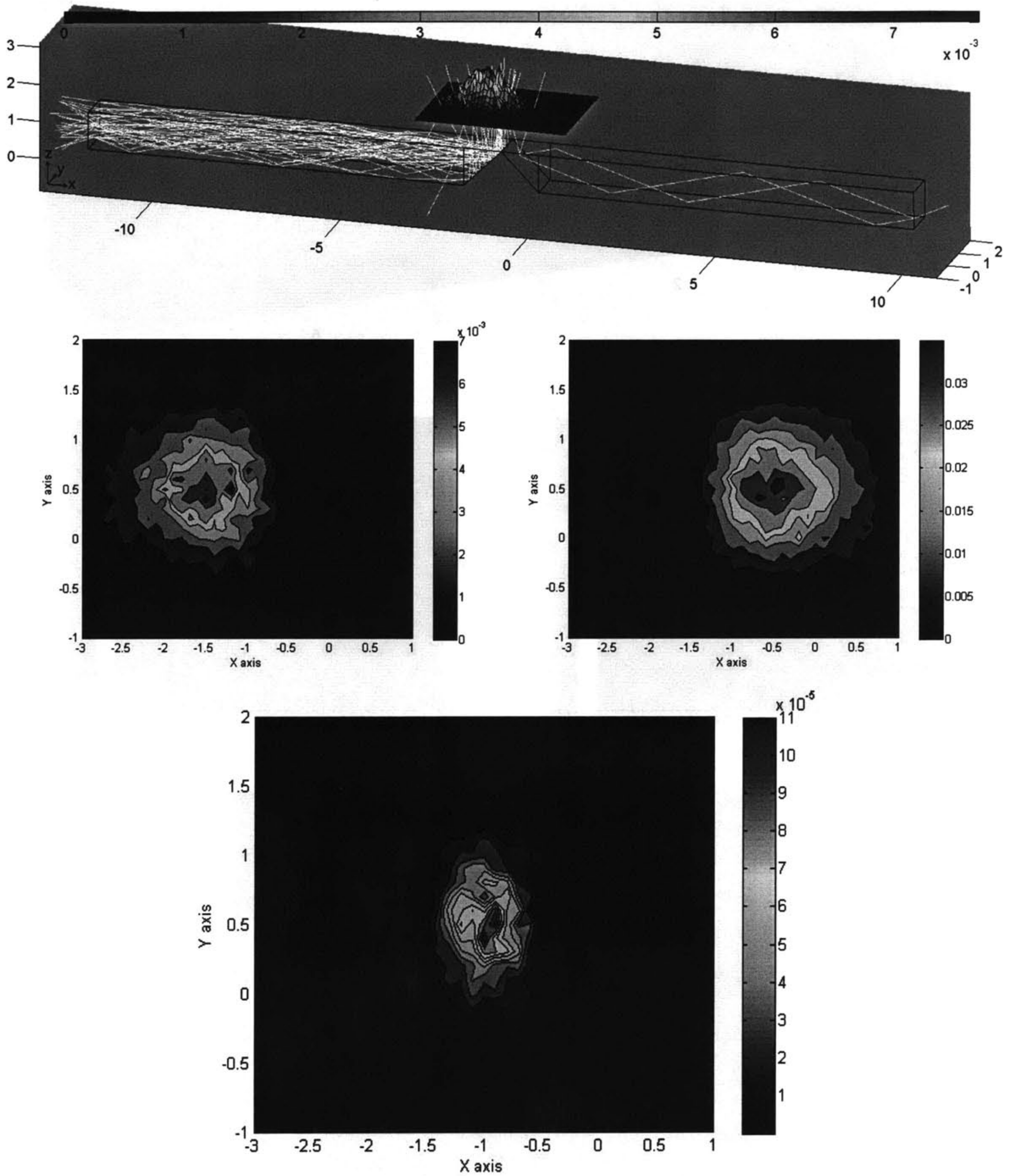

Figure 5.2. Simulations of the fluorescence forward collection system with 45 degree mirrors. The sensor excitation (left) and collection (right) areas do not overlap well, and lead to a low total efficiency (bottom) for fluorophores located on the sensor plane.

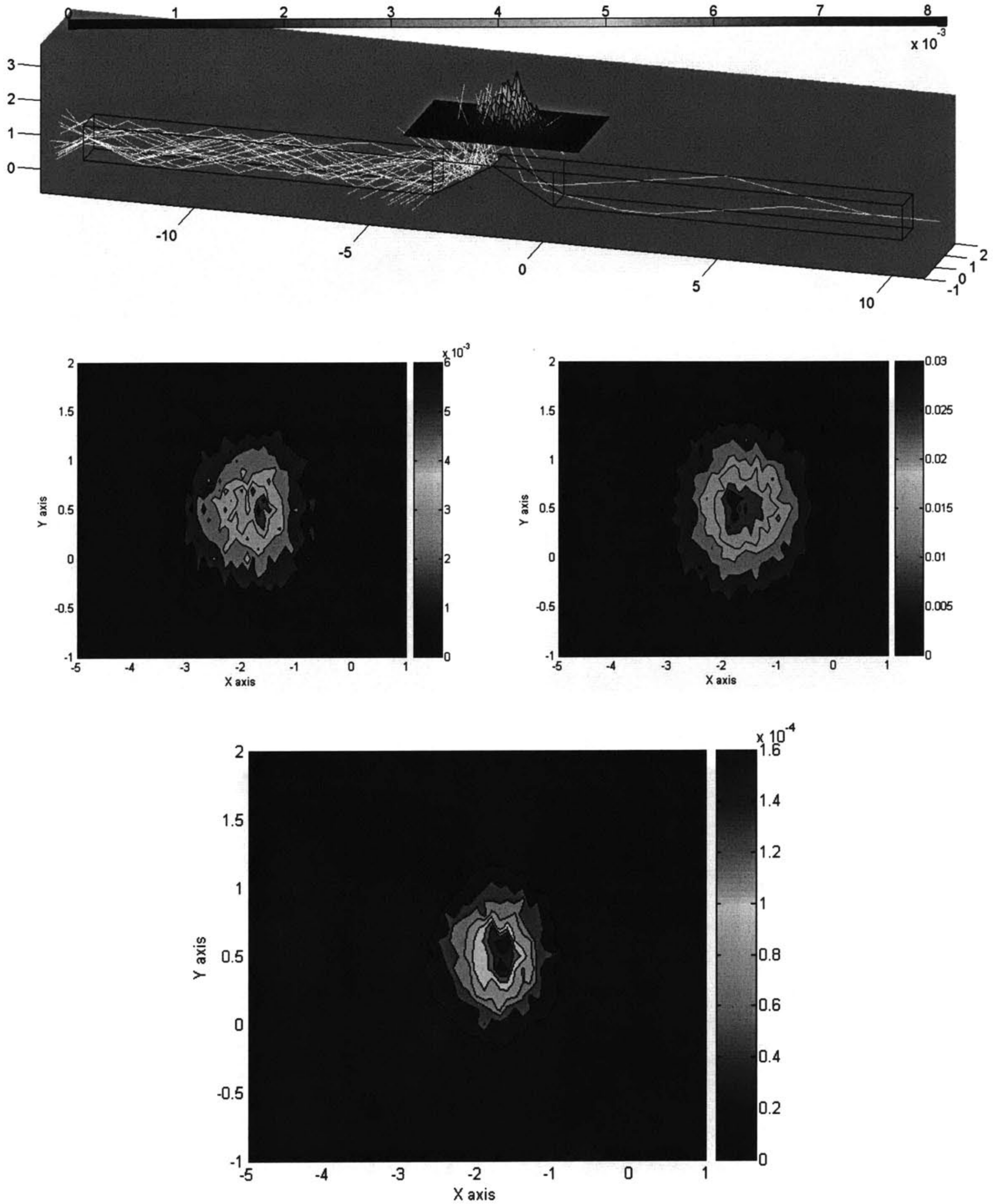

Figure 5.3. Simulations of the fluorescence forward collection system with 30 degree mirrors. The sensor excitation (left) and collection (right) areas overlap unlike the 45 degree case. The total sensor efficiency (bottom) is increased since the excitation and collection areas are again overlapped.

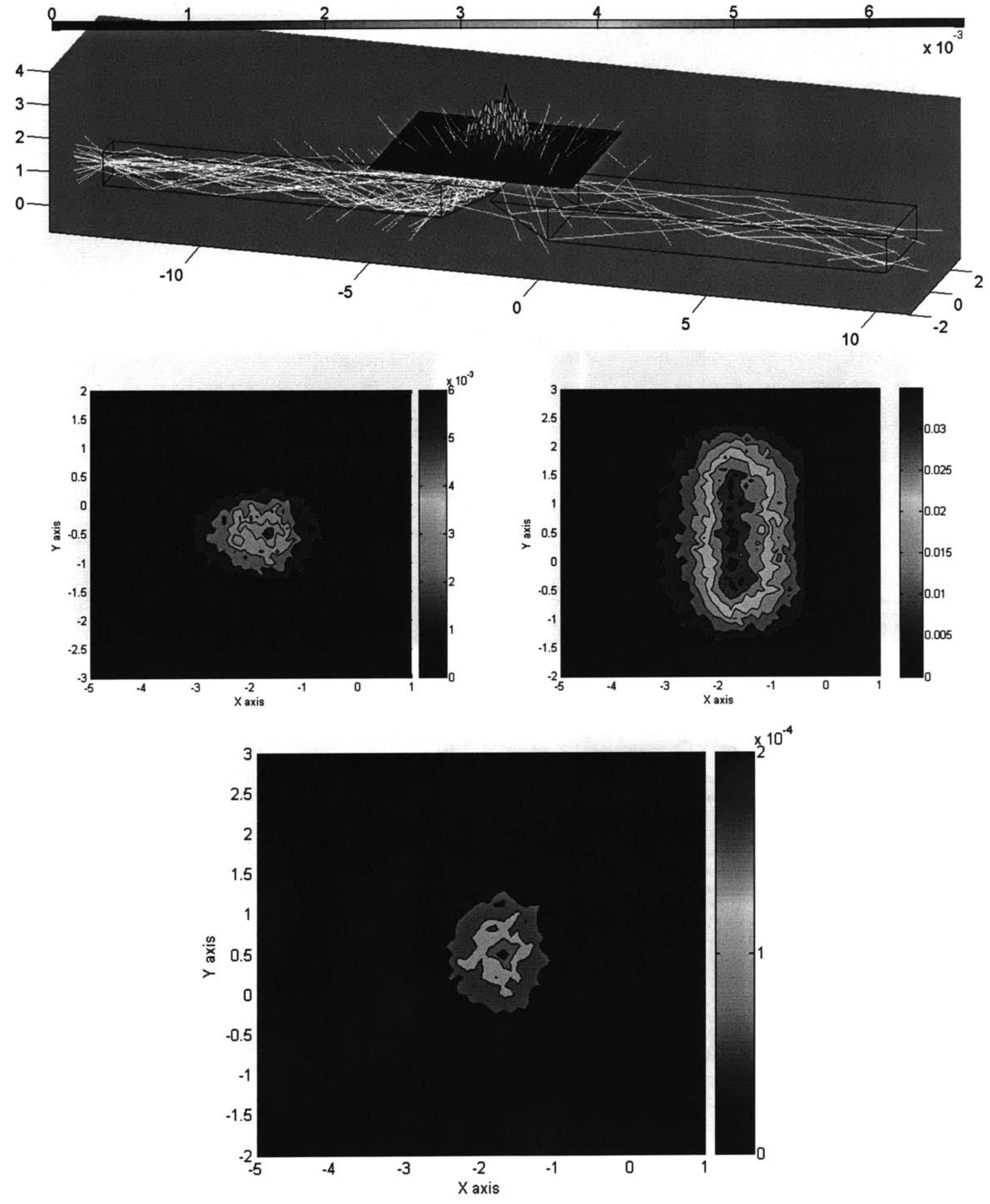

Figure 5.4. Simulations of the fluorescence forward collection system with 30 degree mirrors and a wide collection guide. The sensor excitation (left) and collection (right) areas are shown. The increase in the collection guide width leads to a higher sensor efficiency (bottom) versus the smaller collection guide width design.

Design	Efficiency	Leakage	SNR	SNR w/ PtOEPK
Fiber Bundle (Chapter 1)	0.028	0.013	2.2	0.17
45° back collection + splitter	0.0081	0.0009	9.0	0.70
45° forward collection	0.0041	0.00028	14.6	1.14
30° forward collection	0.0120	0.00052	23.1	1.80
30° forward w/ wide collection	0.0196	0.0013	15.1	1.18

Table 5.1 Summary of the collection efficiencies and performance of four different waveguide designs. SNR is a measure of the collection efficiency if a perfect dye is used divided by the leakage. The last column gives the SNR if PtOEPK oxygen sensors are used. The fiber bundle is measured experimentally using the design shown in Chapter 1.

Although the signal to noise ratio for the last system has dropped, the total collected signal is increased. Given a decent filter with low auto-fluorescence, it is more beneficial to design the system for optimum collection efficiency.

All of the above configurations assumed that the sensor was able to convert excitation light into fluorescence with 100% efficiency, which is untrue for all fluorescence based sensors. The sensors which will be used for oxygen detection in this integrated waveguide system, are made from a 1 to 1 mixture of PtOEPK [29] saturated in toluene and 2% weight polystyrene dissolved in toluene, and are fabricated using the same procedure as used for the sensors in the miniature bioreactor array system [28]. Fabricated sensors measure a total absorption of 65% for 590 nm and these sensors have a quantum yield of 12%. With these inefficiencies, the total collected signal for any system must be scaled by 0.078, which greatly reduces the overall fluorescence signal intensity. Since these sensors operate with a 590 nm excitation, auto-fluorescence of filters is not a major issue. Therefore, it is appropriate to choose the widened 30 degree design for the waveguide structures.

5.3 Device Fabrication

To take advantage of the pre-polished polycarbonate surface, polycarbonate molds are fabricated as negative molds. Waveguides are cut out with a 1 mm diameter carbide end mill spinning at 2800 rpm with a feed speed of 0.5 mm/sec. A finish pass of 0.2 mm is taken across the waveguide with climb-milling to reduce the roughness even more before polishing. A moat is then cut out between the waveguides to form the supporting structure of the backplane. The waveguide ends are left open for easy core filling. A drill mill with a 30 degree face is then used to create the bends and a finish pass is again taken to reduce roughness. The milling process is shown in Figure 5.5.

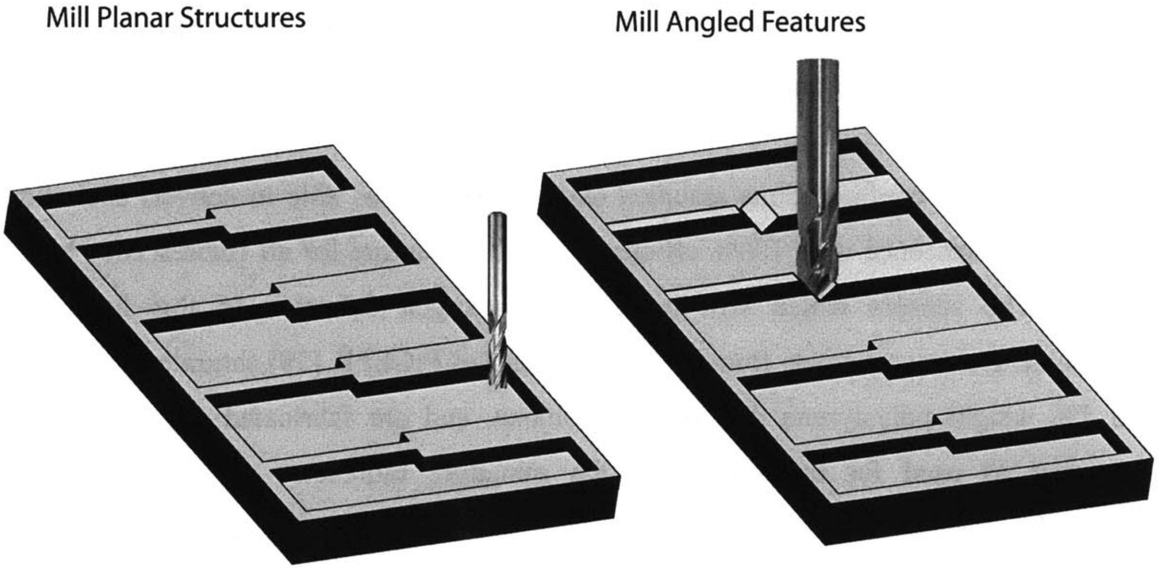

Figure 5.5. Illustration of the milling process for making the oxygen sensor mold. Straight waveguide sections are first machined on a CNC milling machine (left), then a drill mill is used to add bends (right).

The device is then placed into a vapor polishing chamber for 3 minutes at 19.4 inHg of methylene chloride. After polishing, the roughness is measured by determining the laser beam spread resulting from reflection. The estimated slope variance for the sidewalls is 0.009 as was measured in Figure 2.20 and 0.076 for the bend interface as shown in Figure 5.6.

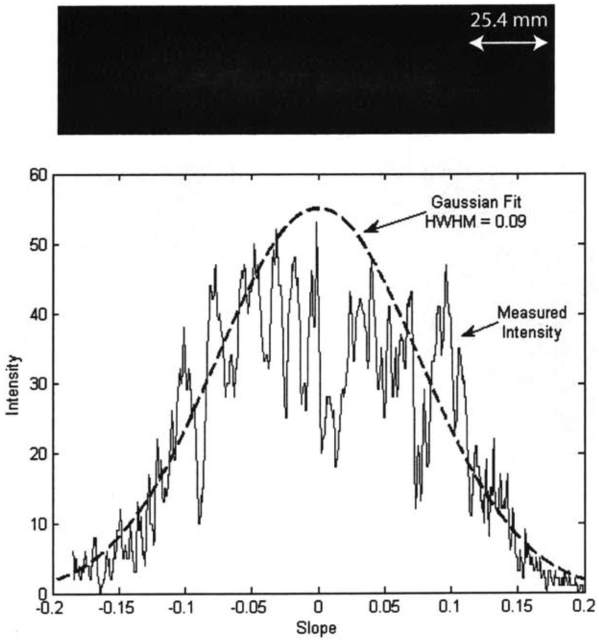

Figure 5.6. Beam spread due to reflection off of the bend surface of the PDMS replica of the machined and polished part (Top). The fit of the intensity profile shows a variance of 0.076 in the slope (bottom).

The increased roughness on the bend face is a result of the slightly rough cutting edge of the drill mill and not a result of process parameters such as milling speeds since this roughness is on the angled direction and not the cutting direction. The final milled and vapor polished polycarbonate mold is shown in Figure 5.7.

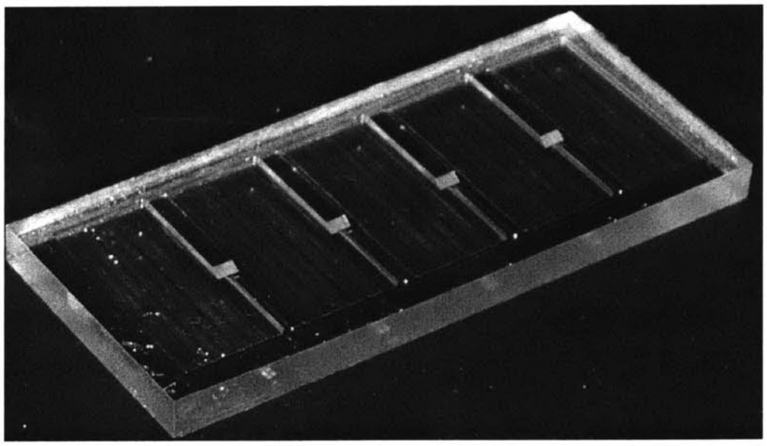

Figure 5.7. Picture of the fabricated and vapor polished waveguide device.

Since the roughness of the bend is pronounced even after vapor polishing, a simulation taking into account the measured roughness should be performed to determine the effect of the bend roughness on the sensor excitation area. A comparison between the smooth and rough profiles is shown in Figure 5.8.

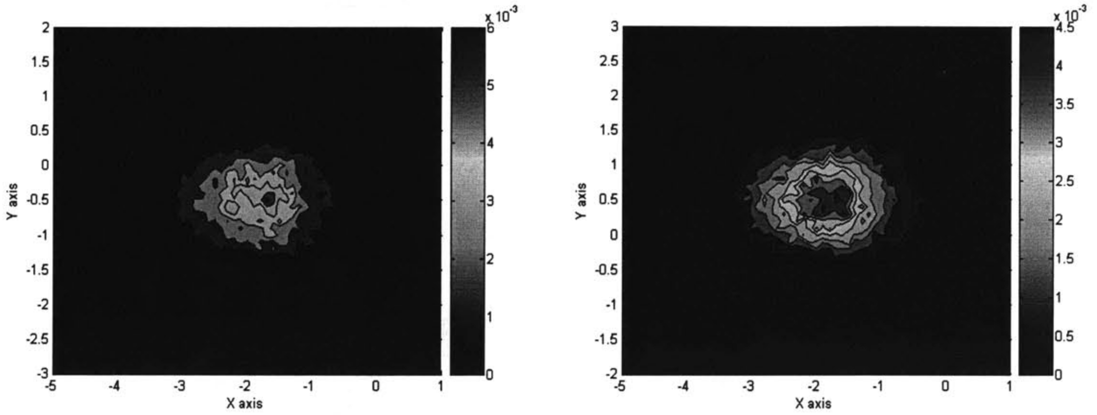

Figure 5.8. Plots of the excitation area for a smooth device (left) and a rough device (right). Roughness variance values were measured for the side walls and bends of the fabricated polycarbonate mold.

Although the excitation area increases slightly in size, the small distance between the bends and the sensor plane suppresses the effect of angular dispersion due to roughness. The calculated total collection efficiency with a rough excitation waveguide is 0.0183 as compared to 0.0196 for a roughness free excitation waveguide, showing that there is little effect on the total output. A simulation of the leak ratio for a rough waveguide also shows a decrease from 0.0013 to 0.0011. While the bend roughness does not greatly affect this particular design beyond a slight attenuation, waveguides designed for sensors farther from the waveguide plane will be less effective at collecting fluorescence.

After vapor polishing, PDMS molds are cast from the polycarbonate masters. These molds are then subject to air plasma to create a thin hydrophilic layer which will enhance e-beam evaporation. A silver layer 300 nm thick is then e-beam evaporated onto the PDMS through a shadow mask so that only the bends and surrounding areas are silver plated. Since the silver layer does not adhere to the PDMS, further processing steps should be performed carefully as not to fracture the silver layer.

After e-beam evaporation, a 1 mm thick slab of PDMS is air plasma bonded over the waveguide channels and NOA71 is injected in through the openings to fill the waveguide cores. The device is then UV-baked for 24 hours to ensure hardening of the polyurethane.

After the device is fabricated, the waveguide ends are cleaved with a razor blade and then a thin film of NOA is cured on the rough waveguide ends to smooth the end faces and improve coupling efficiency. A schematic and picture of the final waveguide device is shown in Figure 5.9.

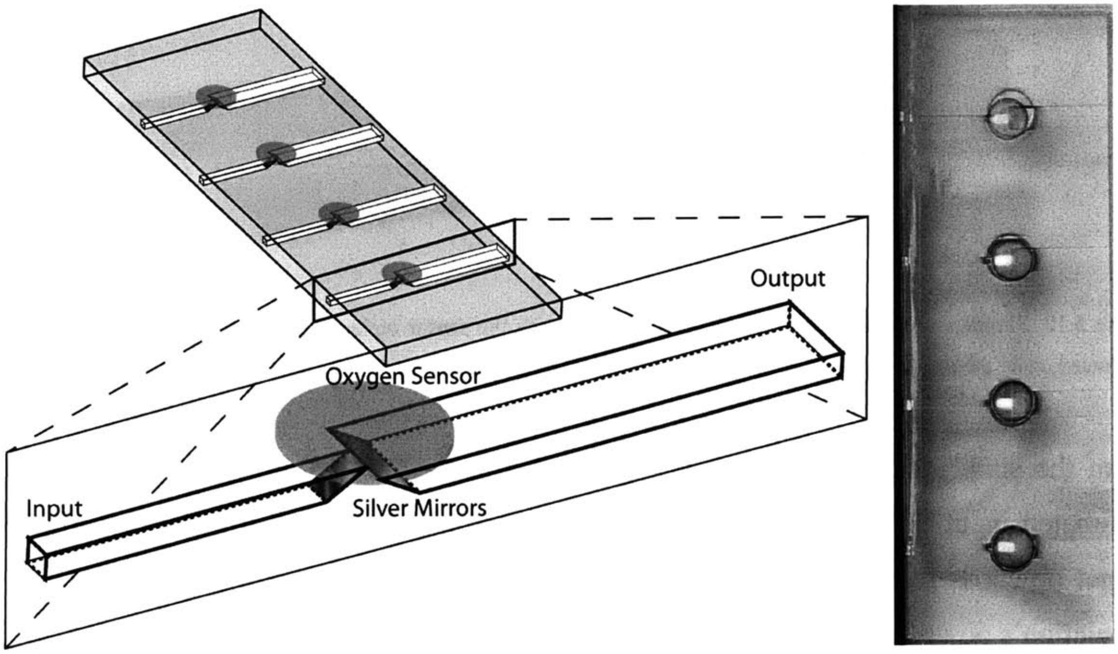

Figure 5.9. Schematic and picture of the fabricated waveguide array. Oxygen sensors placed above the center of the waveguide bends are illuminated by the thin waveguides and fluorescence is collected by the thick waveguides.

The final device consists of four sensors resting above waveguide structures using the 30 degree wide collection design. The picture shows an example of sensor illumination for excitation from the input waveguides.

5.4 Measurements

Before performing sensor measurements, the fabricated device is tested to determine how well it compares to the simulation. PDMS samples with evaporated silver measure reflectivies of 72% at 45 degrees from a HeNe laser at 630 nm.

To test the effectiveness of the waveguide device, the sensors are replaced with a high reflectivity silver mirror and the ratio of output intensity to input intensity is measured. A simulation including measured roughness is performed in Figure 5.10. Since a laser is used as the excitation source, a beam divergence of 0.5° is taken.

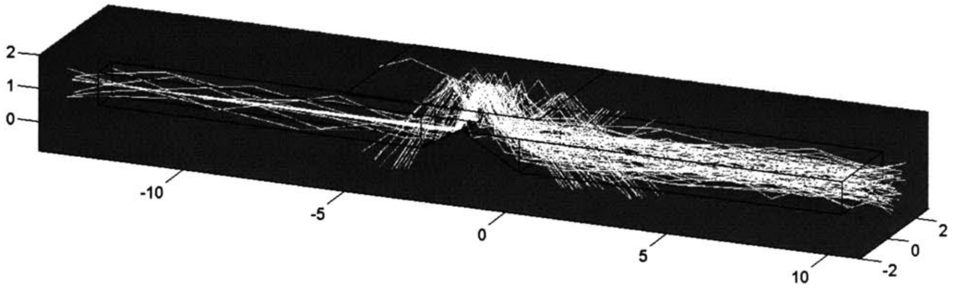

Figure 5.10. Simulation of the fabricated waveguide sensor with the sensor replaced by a mirror. The simulation is conducted under the assumption of a collimated beam input with a divergence of 0.5 degrees.

From the simulation, the total output power should be 57% of the input power for a collimated input. However, combined with the bend mirror reflectivities, the simulated output power is reduced to 30%. The measured output power of 28% agrees with the simulation's power estimate.

It is also necessary to show that fluorescence collection can be performed with the waveguides. In order to accomplish this, four PtOEPK sensors are placed above the four waveguide devices. Since there is no fluidic device above the waveguides, the sensors are all resting above the chip and exposed to air. In order to perform a lifetime measurement, four 585 nm LEDs, which are modulated at 5 kHz, are coupled to multimode fibers for easy alignment with the waveguide input faces. A long pass colored glass filter (RG-9) is then placed at the waveguide array output to remove any extra leakage caused by the excitation light. Four PIN photodiodes are then placed directly behind the glass filter to measure the fluorescence emission from each of the sensors. To measure changes in oxygen concentration, a tube of compressed nitrogen is placed above the exposed sensors. The measurement setup is shown in Figure 5.11.

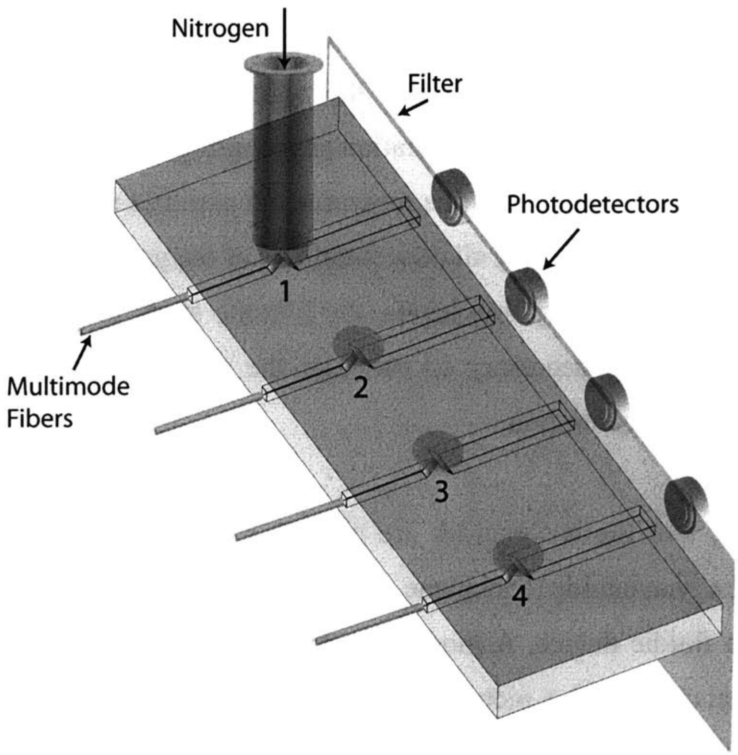

Figure 5.11. Schematic of the measurement setup around the waveguide device. Fibers are used to couple light into the input waveguides and photodetectors are placed at each collection waveguide behind an excitation filter to measure fluorescence. A nitrogen tube placed above a sensor induces a measurable change in oxygen concentration.

By exposing each sensor to the nitrogen source in sequence, it is possible to determine the maximum response of each sensor in relation to air versus nitrogen and measure the diffusion of nitrogen into the ambient air.

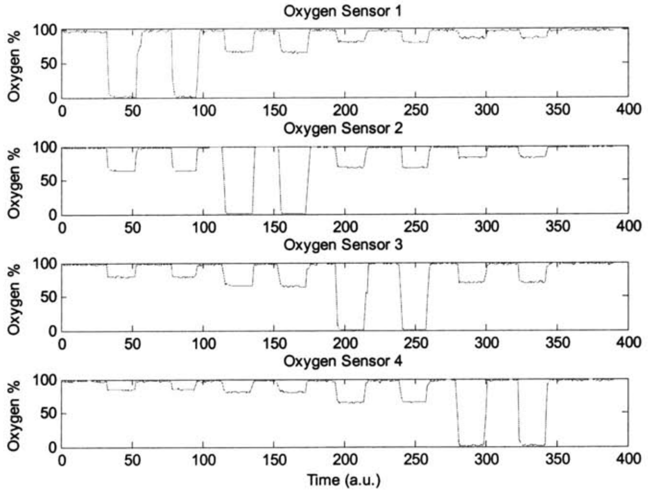

Figure 5.12. A plot of the measured oxygen concentration for all four sensors as a pulsed stream of nitrogen is moved from the left to the right. Sensor 1 refers to the leftmost sensor and Sensor 4 refers to the rightmost sensor.

Figure 5.12 demonstrates measured data for the four sensors while a nitrogen stream is pulsed serially from the leftmost sensor to the rightmost sensor. There is a noticeable decrease in the measured oxygen concentration as the nitrogen stream moves closer to the given sensor. It should be noted that this measurement assumes that the lifetime response for the PtOEPK sensor is linear for all concentrations of oxygen. However, as the oxygen concentration decreases to very low levels, the lifetime starts to increase more quickly [29]. Therefore it is also necessary to calibrate the sensors before use to accurately measure the oxygen concentration.

After determining that the waveguides are operating successfully, a fluidic channel is placed above the waveguide backplane to demonstrate how the backplane can be integrated with a fluidic device. A modified schematic of the backplane integrated with the fluidic channel along with a picture of the device is shown in Figure 5.13.

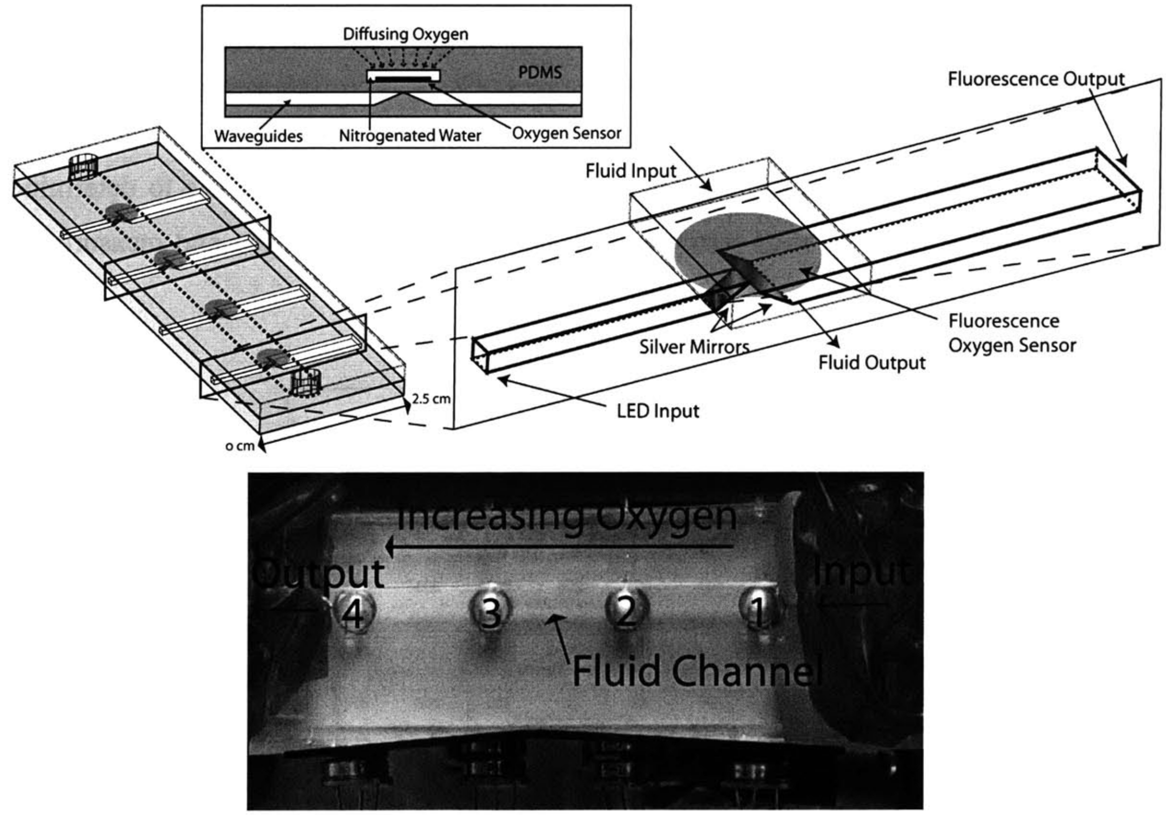

Figure 5.13. Modified schematic (top) of the waveguide backplane including the fluidic channel. With this system it is possible to measure the rate of oxygen diffusion into water for a flowing liquid. As shown in the picture (bottom), fluid is flowed through the channel from right to left and the increase of dissolved oxygen is measured.

Since PDMS is oxygen permeable, it is possible to use the line of sensors to measure the rate of oxygen diffusion into water for a flowing fluid system. By flowing nitrogenated water through the channel and measuring the oxygen concentration of the water, it should be possible to determine the flow rate of the liquid assuming that the rate of diffusion throughout the chip is known. A plot of the dissolved oxygen concentration of water for a flow rate of 0.5 ml/min is given in Figure 5.14.

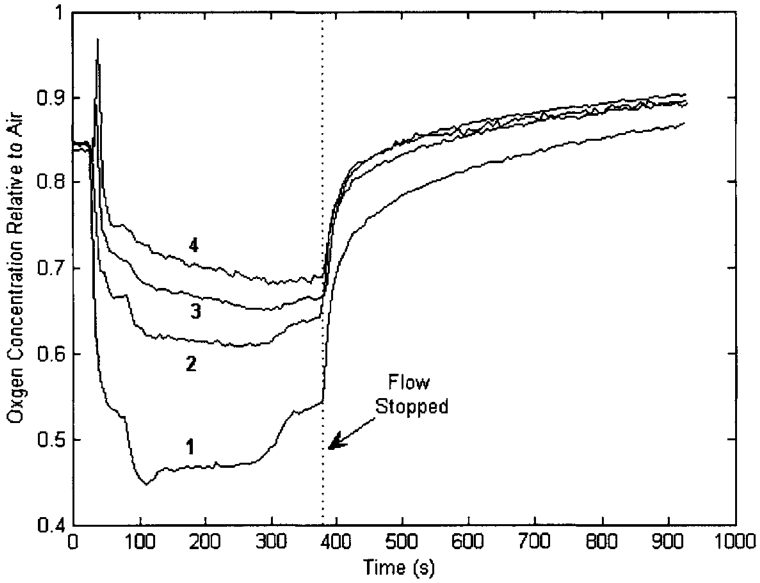

Figure 5.14. Oxygen data for the integrated fluidic and optical device. Sensors 1 through 4 are placed with even spacing along the channel from the input to the output respectively. Nitrogenated water is flowed through the channel at 0.5 ml/min starting at 38 seconds. The flow rate is stopped at 380 seconds and diffusion of oxygen through PDMS is measured for each sensor. Measurements of the four sensors during flow show similar diffusion behavior as measurements taken while flow is stopped.

While the concept seems simple, the diffusion rate throughout the chip is not uniform. From a lack of symmetry, sensors near the chip edge most likely see a different diffusion profile from sensors near the middle of the chip. In addition, the sensors are elevated inside the channel, leading to a small degree of unpredictable chaotic flow behavior. Even though the flow rate can not be determined from the measurement, it can be seen that the same diffusion behavior occurs for the dynamic and stationary flow case.

5.5 Summary

By utilizing the milling and vapor polishing fabrication procedure, we have shown that it is possible to fabricate integrated devices containing waveguide structures and bends. Four different device designs were considered for the purpose of performing fluorescence measurements, and it was concluded that a forward collection scheme with 30 degree bends and a widened collection waveguide would provide the largest fluorescence signal with minimal noise. Fabricated waveguides based on this design performed as predicted by ray tracing simulations. While roughness limited the performance of the system, it could be modeled to give results consistent with experimental measurements. The device worked well enough to measure gradients in oxygen concentration for both an opened gas system and a closed fluid system.

Chapter 6

Conclusions and Future Work

6.1 Summary and Conclusions

The goal of this thesis was to develop an optical backplane consisting of optical components relevant to the miniaturization of optical detection systems for fluidic applications. In order to accomplish this goal, it was necessary not only to select a variety of compatible materials, but also to design a variety of devices capable of performing optical measurements with performance better than existing systems. To design such devices, it was also necessary to develop a ray tracing design tool capable of simulating desired devices including relevant non-ideal effects caused by the fabrication process.

6.1.1 Waveguide Materials and Fabrication

A variety of materials were explored for compatibility with current microfluidic fabrication processes. It was concluded that a cladding material of PDMS should be used in order to maintain compatibility with existing devices. While a variety of materials were available for use as a waveguide core, polyurethane provided by Norland Products was chosen due to its improved compatibility over other PDMS based alternatives when placed in contact with PDMS.

A fabrication process was then introduced which could create molds for the production of large core waveguides. While conventional milling yielded poor surface roughness results, a polishing procedure utilizing solvent vapor was shown to improve the surface to optical quality. Combining this mold making procedure with the already established transfer molding techniques created a powerful fabrication system for the production of both fluidics and waveguides.

In order to determine the effectiveness of the new fabrication system, straight waveguides were manufactured using the process. Waveguides created with square cores exhibited losses below 1 dB/cm in the visible and below 0.2 dB/cm for most of the visible and IR spectrum. This loss was much lower than loss measured in previous experiments in conventional waveguide fabrication, and numerical aperture measurements resulting in only a reduction from 0.62 to 0.5 indicated decent roughness values.

6.1.2 Ray Tracing

A ray tracer was developed to model surface roughness in an attempt to explain the numerical aperture results measured by the experiment. The simulation program modeled surface roughness as a Gaussian distribution in slope, reflecting the slope distribution measured from experimental samples. This slope translated to an angular distribution of rays leaving a macroscopically flat surface due to variations in roughness and was experimentally verifiable. By tracing rays through a simulation including roughness, it was determined that the effect of surface roughness was a numerical aperture decrease, agreeing with measured data. By analyzing the roughness simulation, it was determined that surface roughness also attributed to loss and mode mixing.

Measurements on waveguide samples with different roughness values were performed and verified that ray tracing simulations accurately described mode dependent losses within the waveguide structure. In addition, the signal stretching due to multimode distortion was measured, indicating that the waveguide could support modulation frequencies of 1.3 MHz-km as predicted by simulations.

6.1.3 Integrable Optical Components

A variety of integrable optical components were developed for use in the waveguide platform. The first components explored were cyanine dyes. Taking advantage of the low quantum yield of many red absorbing cyanine dyes, the dyes were mixed with polyurethane core prepolymer to create doped polymers for use as filters. By making waveguides with this doped core, the structure acted simultaneously as a waveguide and a filter, providing filtering characteristics with virtually no coupling loss.

Seeing as how low quantum yield dye dopants could be utilized as filters, high quantum yield dopants were also explored for use as light sources. Different sized CdSe quantum dots were purchased in a premade suspension of polymer resin. By curing these mixtures into waveguides, it was also demonstrated that light sources could be made. Unfortunately, due to the numerical aperture of the waveguides, only a small percentage of the emitted light could be coupled into the waveguide system. Modeling of the bulk waveguide structure also revealed that creating doped waveguide light sources was an optimization process dependent on the dot concentration and the waveguide length.

The last devices explored were structural devices rather than polymer modifications. It was shown that silver mirrors could be successfully plated onto PDMS with low fracturing with the aid of air plasma. These mirrors reflected light with a reflectivity of 98% on flat PDMS and 72% on a fabricated out-of-plane bend, allowing for the realization of sharp bends without the aid of an air interface. By plating the bend with a high reflectivity metal instead of utilizing a TIR air interface, the waveguides could be protected on all sides by a cladding material and would not be subject to degradation due to ambient air. Structural stability and fabrication strategies were more accessible since there was no longer a need to create air gaps in the device structure. In-plane bends and power combiners were also fabricated to test if compact routing schemes could be utilized in the backplane platform. Bend efficiencies were low due to the directional nature of the silver evaporation process and although both devices were functional, these losses dominated the transmission characteristics.

6.1.4 Integrated Optical Backplane for Oxygen Sensing

Since the fabrication of bends was successful, a compact design for waveguide fluorescence sensing could be achieved. After iterating through a few designs, a design was found maximizing overall collected signal intensity. While it was important to achieve a good signal to noise ratio, it was determined that signal intensity was more important since filters could be used to remove any excess excitation leakage through the device.

The fabricated device consisted of four waveguide sensors, each capable of delivering excitation light and collecting fluorescence and an application to oxygen sensing was demonstrated. To show that the system was independent but attachable to a fluidic chip, a separate fluidic channel was designed and bonded to the optical backplane. Successful measurements of dissolved oxygen through the flowing channel demonstrated the usefulness of the backplane technology.

6.2 Future Work

Waveguide devices fabricated by the conventional milling process utilized a bench top milling machine from Sherline Products. In addition to the inherent vibrations accompanying such a small machine, it was discovered afterwards that the majority of the end mill displacement resulted from a loose collet. After purchasing a new collet and draw bar, the maximum displacement per revolution reduced from 100 μm to 10 μm. While it was mentioned in Chapter 2 that this does not necessarily translate into a reduction of the displacement variance, it will most likely help. Waveguides should be fabricated under more stable tool conditions to determine if there are improvements to surface roughness.

Also, while the polishing process was characterized, there were some unexplained observations which resulted during optimization. First, humidity seems to play a major role in the final polishing appearance. Increased humidity tends to lead to a very hazy surface. Second, in many cases of over polishing, the tooled features on the side facing up are the only areas which become hazy. In the case of negative molds, side walls and pre-polished surfaces remain optically clear, while sections which contain tool marks from the bottom of the end mill polish to a hazy appearance. Third, the degree of polishing also has some dependence on the feed speed and spindle speed of the cutter. When comparing a channel milled with a high frequency of cuts per distance versus a channel milled with a lower frequency of cuts per distance, there appears to be almost no polishing and optimal polishing respectively. There are clearly many more factors which contribute to the polishing quality outside of just the exposure time, vapor pressure, and substrate temperature and if waveguides are to be manufactured using this process, all of the anomalies observed above need to be addressed.

In the ray tracer, an attempt was made to model sidewall roughness. While ray tracing simulations of the numerical aperture reduction due to roughness closely matched output intensity profiles measured for fabricated waveguides, there was a discrepancy between the measured loss versus the simulated loss due to roughness. This unaccounted loss of roughly 0.25 dB/cm can not be explained as material absorption loss, since it was shown that for small numerical apertures, the loss could be reduced to 0.09 dB/cm, well below the 0.25 dB/cm difference. The source of this extra loss should be determined.

With regard to integrated devices, further research should be performed to determine if quantum dots are usable as light sources. While it was shown during the course of this research that they can be used as emitters within a waveguide, the emission quality was poor, due mainly to two aspects. The first was the numerical aperture of the waveguide, which restricted collection efficiency of the omnidirectional quantum dot emission. The second was the long tail associated with the dots purchased by Evident Technologies, which were the only dots available suspended in polymer. Since it was shown that a leakage of greater than 0.001 exists through the waveguide, and the collected signal at best is 0.002, quantum dot tails extending into the infrared at emission intensities of 1% or 2% of the peak will be noticeable. In addition, much of the quantum dot emission that is not collected by the waveguide numerical aperture will travel directly to the detector through the cladding in a forward-collection geometry, further increasing the detected tail intensity. Efforts to reduce tail emissions and decrease cladding coupling should be pursued if dots are to be used as wavelength specific light sources.

While it was shown that silvering bends greatly increased the coupling efficiency for out-of-plane bending structures, efforts to create silvered in-plane bends were not successful due to the unidirectional behavior of e-beam evaporation. Since it is essential for compact routing schemes to implement tight bends, such as the local maximum radial bend or the sharp 45 degree bend, other silver deposition schemes should be explored. In particular, sputtering should be looked at as an alternative to e-beam evaporation due to its conformal nature. In addition, the crack induced diffuse surface which results from the NOA shrinking and stressing the silver film should be addressed. If a deposition procedure can be determined which allows silver to form a smooth surface and also adhere to the PDMS substrate, the NOA induced stress will most likely be minimized.

If in-plane bends can be fabricated successfully, the next step towards achieving a compact measurement platform would be to implement multiplexed detection. In a serial detection algorithm, the modulated LED's would simply turn on in a time domain multiplexed method. In a parallel detection algorithm, setting a different modulation frequency per sensor would allow a single detector to distinguish between fluorescence signals by band pass filtering.

The key to miniaturizing the detection system of a biochip is to provide as many functions on-chip as possible. The ability to add waveguides for optical sensing, multiplexed detection, integrated filters, and light sources, would greatly decrease the off-chip complexity.

Appendix A

Chemical Resistance to Solvents

A full list can be accessed at http://us.burkert.com/img_article/US_Chemical_Resistance.pdf but a short list is given below with resistance values for chemicals which act as solvents for polycarbonate.

Abbreviations

NBR	Nitrile rubber
EPDM	Ethylene propylene
FKM	Fluorine rubber (Viton)
FFKM	Perfluorinated elastomers (Simnz, Kalrez, Chemraz)
CR	Chloroprene rubber
PVC	Polyvinylchloride
PP	Polypropylene
PA	Polyamide
PVDF	Polyvinylidene fluoride
PPS	Polyphenylsulfide
PEEK	Polyetheretherketone
MS	Brass
RG	Leaded red brass
GG	Cast Iron
GS	Cast Steel
1.4401	Stainless steel

1.4305 Stainless steel

Resistance degrees

+	Material little or not affected by chemical: suitable
0	Various attack grades depending on conditions: limited suitability
-	Material shows severe attack: unsuitable

Material Chemical	NBR	EPDM	FKM	FFKM	CR	PVC	PP	PA	PVDF	PPS	PEEK	MS	RG	GG	GS	1.4401	1.4305
Acetic Acid	-	0	-	0	-	0	0	0	+	+	+	-	0	0	0	0	0
Acetic Anhydride	-	0	-	0	-	-	-	-	-	+		-	0	0	0	0	0
Acetophenome	-		-	+	-	-		+	0	0		+	+	+	+	+	+
Amyl acetate	-	0	-	+	-	-	0	+	+	+	+	+	+	0	0	+	+
Benzaldehyde	0	+	+	+	-	-	+	0	0	0	+	0	0	0	0	+	+
Butyl acetate	-	+	-	+	-	-	-	+	+	+	+	0	+	0	0	+	+
Butyl alcohol	+	+	0	+	0	0	+	+	+	+	+	+	+	0	0	+	+
Carbon Tetrachloride	-	-	0	-	-	0	-	+	+	0	+	0	0	0	0	+	+
Chloroform	-	-	0	+	-	-	-	-	+	0	+	0	0	0	0	+	0
Cyclohexane	-	-	0	+	-	+	-	+	+	+	+	+	+	+	+	+	+
Cyclohexanone	-	-	-	+	-	-	-	+	0	+	+	0	0	0	0	+	+
Dimethyl formamide	-	-	-	+	-	-	+	-	-	0	+	0	0	0	0	+	+
Dioxane	-	0	-	+	-	-	-	+	-	+		+	+	+	+	+	+
Ethyl acetate	-	0	-	+	-	-	0	0	0	+	+	0	+	0	0	+	+
Ethylene chloride	-	-	0	0	-	-	+	0	+	0	0	+	+	+	+	+	+
Ethylene glycol	+	+	+	+	+	0	0	0	+	+	+	0	0	0	0	+	0
Hexane	+	-	+	+	+	+	-	+	+	+	+	+	+	+	+	0	0
Methyl ether ketone	-	0	-	+	-	-	-	0	-	0	0	+	+	0	0	+	+
Methylene Chloride	-	-	0	+	-	-	0	-	-	0	+	0	0	0	0	0	0
Phenol	0	0	0	+	0	+	+	-	0	+	0	0	0	0	0	+	+
Tetrahydrofuran	+	-	-	+	-	-	0	+	-	0	+					+	+
Toluene	-	-	-	+	-	-	0	+	0	0	+	0	0	0	0	+	+
Xylene	-	-	-	+	-	-	-	+	0	0	+	0	0	0	+	+	0

From this list, it is clear that brass, cast iron, and steel are suitable for most polycarbonate solvents. From a chemical resistance standpoint, the best plastic material for tubing to pull vacuum and inject solvent vapor is either FFKM or PEEK. In addition, to create a vacuum sealable box, it is necessary to incorporate an o-ring between the lid and box. While FFKM o-rings are available, they are very expensive. Instead, FKM o-rings are used. Even though their resistance to most polycarbonate solvents is unsuitable, the surface area of the o-ring in contact with the solvent vapor during compression is minimal, and in addition, the exposure times are short.

Appendix B

Chemical Dyes Filters

Below is a list of a variety of dyes available from Sigma Aldrich with the associated peak absorption wavelengths and part numbers.

Chemical Name	Sigma Aldrich #	Peak Wavelength 1 (nm)	Peak Wavelength 2 (nm)
3,3'-(4,4'-Biphenylene)bis(2,5-diphenyl-2H-tetrazolium chloride),	B3,510-5	250	
Indium(III) phthalocyanine chloride, 70% (dye content)	55,623-8	280	
2,5-Diphenyl-1,3,4-oxadiazole, 97%	D21,021-8	282	
2,5-Diphenyloxazole, 99%, scintillation grade	D21,040-4	303	
5-Phenyl-2-(4-pyridyl)oxazole, 99%	22,903-2	321	
4-Chloro-7-chlorosulfonyl-2,1,3-benzoxadiazole, 97%	55,650-5	324	
Silver phthalocyanine, 75% (dye content)	55,227-5	327	664
2-(4-Biphenylyl)-6-phenylbenzoxazole, 98%	23,536-9	331	
Cadmium phthalocyanine, approx. 95% (dye content)	55,621-1	347	
1,4-Bis(2-methylstyryl)benzene, 99.5+%	25,740-0	349	
Carbostyril 124, 99%	36,330-8	350	
7-Amino-4-methylcoumarin, 99%	25,737-0	354	
1,4-Bis(5-phenyl-2-oxazolyl)benzene, 99+%	25,739-7	362	
Aluminum 2,3-naphthalocyanine chloride, 80% (dye content)	40,449-7	381	
2,3,7,8,12,13,17,18-Octaethyl-21H,23H-porphine, 97%	25,240-9	401	
2-[4-(Dimethylamino)styryl]-1-methylpyridinium iodide, 95% (dye content)	28,013-5	466	

4-(Dicyanomethylene)-2-methyl-6-(4-dimethylaminostyryl)-4*H*-pyr an, 98% (dye content)	41,049-7	468	
3,3′-Diethyloxacarbocyanine iodide, 98%	32,068-4	483	
Rhodamine 6G, 99% (dye content)	25,243-3	524	
1,1′-Diethyl-2,2′-cyanine iodide	323764	524	
Rhodamine 6G perchlorate, 99% (dye content)	25,244-1	528	
Sulforhodamine G, 60% (dye content)	23,056-1	529	
Sulforhodamine B, acid form, laser grade, 95% (dye content)	34,173-8	558	
3,3′-Diethylthiacarbocyanine iodide, 95% (dye content)	17,373-8	560	
Boron subphthalocyanine chloride, 90% (dye content)	42,554-0	565	
Styryl 7, 95% (dye content)	37,182-3	565	
Styryl 9M tetrafluoroborate, 99% (dye content)	42,533-8	584	
Styryl 9M, 99% (dye content)	41,702-5	584	
1,1′-Diethyl-2,2′-carbocyanine chloride	201715	604	560
Copper(II) phthalocyanine-tetrasulfonic acid tetrasodium salt	27,401-1	609	692
1,1'-Diethyl-2,2'-carbocyanine iodide	166510	614	
Oxazine 170 perchlorate, 95% (dye content)	37,205-6	624	
Nile Blue A perchlorate, 95% (dye content)	37,008-8	628	
Copper phthalocyanine-3,4',4",4"'-tetrasulfonic acid, tetrasodium salt tetrasodium salt, 85% (dye content)	24,535-6	630	667
Iron(III) phthalocyanine-4,4′,4″,4‴	45,252-1	635	
2-[5-(1,3-dihydro-1,3,3-trimethyl-2H-indol-2-ylidene)-1,3-pentadienyl]-1,3,3-trimethyl-3H-indolium iodide, 99%	25,733-8	637	
1,3-Bis[4-(dimethylamino)-2-hydroxyphenyl]-2,4-dihydroxycyclobutenediylium dihydroxide, bis(inner salt), 97% (dye content)	42,528-1	638	
Zinc(II)tetranitrophthalocyanine	54,832-4	642	
Oxazine 1 perchlorate, 85% (dye content)	37,009-6	644	
1,1′-Diethyl-4,4′-carbocyanine iodide, 96%	D9,153-5	648	703
Disodium phthalocyanine, 90% (dye content)	40,274-5	649	721
Iron(III) phthalocyanine chloride, ~95% (dye content)	37,957-3	654	
3,3′-Diethylthiadicarbocyanine iodide, 98%, 98% (dye content)	17,375-4	655	
Iron(II) phthalocyanine bis(pyridine) complex, 95% (dye content)	41,016-0	655	
Iron(II) phthalocyanine, ~90% (dye content)	37,954-9	657	
Nickel(II) phthalocyanine-tetrasulfonic acid tetrasodium salt	27,490-9	657	
Dilithium phthalocyanine, 70% (dye content)	36,254-9	658	
2,9,16,23-Tetraphenoxy-29*H*,31*H*-phthalocyanine, 98% (dye content)	41,815-3	668	703
Magnesium phthalocyanine, 90% (dye content)	40,273-7	668	
Silicon(IV) phthalocyanine bis(trihexylsilyloxide), 98% (dye content)	42,815-9	669	
Nickel(II) phthalocyanine, 85% (dye content)	36,063-5	670	

Nickel(II) tetrakis(4-cumylphenoxy)phthalocyanine, 97% (dye content)	41,544-8	670	
Propyl Astra Blue Iodide, 90% (dye content)	36,337-5	670	
Chloroaluminum chlorophthalocyanine, ~65% (dye content)	55,228-3	671	
Copper(II) 2,9,16,23-tetra-*tert*-butyl-29*H*,31*H*-phthalocyanine, 97% (dye content)	42,316-5	672	
Methylsilicon(IV) phthalocyanine hydroxide, 95% (dye content)	51,682-1	672	
Silicon phthalocyanine dihydroxide, 75% (dye content)	40,648-1	672	
Boron sub-2,3-naphthalocyanine chloride, 75% (dye content)	42,775-6	673	
IR-676, 98% (dye content)	54,349-7	675	
IR-676 iodide, 97% (dye content)	47,750-8	676	
Cobalt(II) phthalocyanine, β-form, 97% (dye content)	30,769-6	677	716
Copper(II) 2,3,9,10,16,17,23,24-octakis(octyloxy)-29*H*,31*H*-phthalocyanine, 95% (dye content)	45,971-2	677	
Zinc 2,3,9,10,16,17,23,24-octakis(octyloxy)-29*H*,31*H*-phthalocyanine, 95% (dye content)	45,972-0	677	
Zinc 2,9,16,23-tetra-*tert*-butyl-29*H*,31*H*-phthalocyanine, 97% (dye content)	43,099-4	677	
Cobalt(II) 1,2,3,4,8,9,10,11,15,16,17,18,22,23,24,25-hexadecafluoro-29*H*,31*H*-pht halocyanine, 95% (dye content)	44,664-5	678	
Copper(II) phthalocyanine, >99% (dye content), Sublimation Grade	54,668-2	678	
Copper(II) phthalocyanine, 99% (dye content)	54,667-4	678	
Copper(II) phthalocyanine, β-form, 97% (dye content)	25,298-0	678	
Nickel(II) 2,9,16,23-tetraphenoxy-29*H*,31*H*-phthalocyanine, 97% (dye content)	41,437-9	678	
Silicon 2,9,16,23-tetra-*tert*-butyl-29*H*,31*H*-phthalocyanine dihydroxide, 80% (dye content)	46,557-7	678	
Aluminum phthalocyanine chloride, ~85% (dye content)	36,253-0	680	
Aluminum 2,9,16,23-tetraphenoxy-29*H*,31*H*-phthalocyanine hydroxide, 90% (dye content)	41,898-6	681	686
Copper(II) tetrakis(4-cumylphenoxy)phthalocyanine, 97% (dye content)	43,294-6	681	
Iron(II) 1,2,3,4,8,9,10,11,15,16,17,18,22,23,24,25-hexadecachloro-29*H*,31*H*-pht halocyanine, 80% (dye content)	44,804-4	681	
Silicon 2,3,9,10,16,17,23,24-octakis(octyloxy)-29*H*,31*H*-phthalocyanine dihydroxide, 95% (dye content)	46,351-5	682	
2,9,16,23-Tetrakis(phenylthio)-29*H*,31*H*-phthalocyanine, 90% (dye content)	41,818-8	683	715
Zinc 1,2,3,4,8,9,10,11,15,16,17,18,22,23,24,25-hexadecafluoro-29*H*,31*H*-pht halocyanine, 95% (dye content)	44,452-9	686	
1,8,15,22-Tetraphenoxy-29*H*,31*H*-phthalocyanine, 95% (dye content)	41,814-5	687	719
Copper(II) 1,2,3,4,8,9,10,11,15,16,17,18,22,23,24,25-hexadecafluoro-29*H*,31*H*-pht halocyanine, 80% (dye content)	44,665-3	689	
Copper(II) 2,3-naphthalocyanine, 85% (dye content)	38,191-8	692	
Titanium(IV) phthalocyanine dichloride, 95% (dye content)	51,174-9	692	
Titanyl phthalocyanine, 95% (dye content)	40,455-1	692	
Titanyl phthalocyanine, β-modification	55,618-1	692	
Titanyl phthalocyanine, γ-modification	55,620-3	692	

Zinc 2,9,16,23-tetrakis(phenylthio)-29*H*,31*H*-phthalocyanine, 85% (dye content)	41,823-4	692	
Tin(IV) phthalocyanine oxide, 90% (dye content)	55,624-6	693	
Gallium(III)-phthalocyanine chloride, 97% (dye content)	40,880-8	694	
2,9,16,23-Tetra-*tert*-butyl-29*H*,31*H*-phthalocyanin e, 97% (dye content)	42,315-7	695	
Gallium(III) phthalocyanine hydroxide, 75% (dye content)	44,108-2	695	
Methylsilicon(IV) phthalocyanine chloride	43,346-2	695	
Tin(II) phthalocyanine, 97% (dye content)	40,454-3	695	
Aluminum phthalocyanine hydroxide, 85% (dye content)	44,663-7	698	
Lead(II) phthalocyanine, 80% (dye content)	37,956-5	698	
29*H*,31*H*-Phthalocyanine, β (modification), ~99% (dye content)	55,625-4	699	
29*H*,31*H*-Phthalocyanine, β-form, 98%	25,310-3	699	
Tetrakis(4-cumylphenoxy)phthalocyanine, 97%	41,206-6	699	
2,3,9,10,16,17,23,24-Octakis(octyloxy)-29*H*,31*H*-phthaloc yanine	46,220-9	701	
Aluminum 2,9,16,23-tetraphenoxy-29*H*,31*H*-phthalocyanine chloride, 90% (dye content)	41,825-0	701	
Zinc phthalocyanine, 97% (dye content)	34,116-9	701	
Silicon phthalocyanine dichloride, ~85% (dye content)	28,776-8	702	
Tin(IV) phthalocyanine dichloride, 97% (dye content)	40,647-3	703	
1,1′-Diethyl-2,2′-dicarbocyanine iodide, 97%	39,219-7	707	
Vanadyl 2,9,16,23-tetraphenoxy-29*H*,31*H*-phthalocyanine, 98% (dye content)	41,438-7	710	
1,8,15,22-Tetrakis(phenylthio)-29H,31H-phthalocyanine, 90% (dye content)	41,816-1	711	738
2,3-Naphthalocyanine, 95% (dye content)	38,189-6	712	
Lead(II) tetrakis(4-cumylphenoxy)phthalocyanine, 90% (dye content)	40,887-5	713	
Naphthol Green B, 50% (dye content)	11,991-1	714	
Aluminum 2,9,16,23-tetrakis(phenylthio)-29*H*,31*H*-phthalocyanine chloride, 90% (dye content)	41,827-7	725	
Manganese(III) phthalocyanine chloride, 85% (dye content)	39,393-2	726	
Manganese(II) phthalocyanine	37,955-7	727	
Silicon 2,3-naphthalocyanine bis(trihexylsilyloxide), 95% (dye content)	38,993-5	729	774
Cobalt(II) 2,3-naphthalocyanine, 85% (dye content)	38,192-6	731	
Zinc 1,4,8,11,15,18,22,25-octabutoxy-29*H*,31*H*-phthalocyanine	38,381-3	736	
1,1′,3,3,3′,3′-Hexamethylindotricarbocyanine iodide, 98%	25,203-4	740	
1,1′,3,3,3′,3′-Hexamethylindotricarbocyanine perchlorate, 97%	40,381-4	740	
Copper(II) 1,4,8,11,15,18,22,25-octabutoxy-29*H*,31*H*-phthalocyanine, 95% (dye content)	38,661-8	740	
Bis[5-[[4-(dimethylamino)phenyl]imino]-8(5*H*)-quinolinone]nicke l(II), 70% (dye content)	52,324-0	742	
1,1′-Diethyl-2,2′-quinotricarbocyanine iodide, 95% (dye content)	38,193-4	743	
Nickel(II) 1,4,8,11,15,18,22,25-octabutoxy-29*H*,31*H*-phthalocyanine, 97% (dye content)	40,888-3	743	
IR-746, 85% (dye content)	54,320-9	746	

Aluminum 1,8,15,22-tetrakis(phenylthio)-29*H*,31*H*-phthalocyanine chloride, 90% (dye content)	41,826-9	759
3,3′-Diethylthiatricarbocyanine perchlorate, 99%	38,991-9	760
1,4,8,11,15,18,22,25-Octabutoxy-29*H*,31*H*-phthalocyanine, 95% (dye content)	38,380-5	762
3,3′-Diethylthiatricarbocyanine iodide, 99%	38,130-6	765
Aluminum 1,4,8,11,15,18,22,25-octabutoxy-29*H*,31*H*-phthalocyanine triethylsiloxide	38,662-6	766
Nickel(II) 2,11,20,29-tetra-*tert*-butyl-2,3-naphthalocyanine, 95% (dye content)	43,295-4	766
2,4-Di-3-guaiazulenyl-1,3-dihydroxycyclobutenediylium dihydroxide bis(inner salt), 99% (dye content)	39,121-2	768
Copper(II) 3,10,17,24-tetra-*tert*-butyl-1,8,15,22-tetrakis(dimethylamino)-29*H*,31 *H*-phthalocyanine, 80% (dye content)	39,388-6	768
IR-768 perchlorate, 85% (dye content)	42,745-4	768
Zinc 2,11,20,29-tetra-*tert*-butyl-2,3-naphthalocyanine, 90% (dye content)	43,221-0	769
Indocyanine green, 80% (dye content)	22,886-9	775
IR-775 chloride, approx. 90% (dye content)	54,491-4	775
IR-786 iodide, 97% (dye content)	42,413-7	775
IR-786 perchlorate, 98% (dye content)	40,711-9	775
IR-780 iodide, 99% (dye content)	42,531-1	780
Silicon 2,3-naphthalocyanine dichloride, 85% (dye content)	38,992-7	780
IR-783, 90% (dye content)	54,329-2	783
2,11,20,29-Tetra-*tert*-butyl-2,3-naphthalocyanine, 97% (dye content)	42,527-3	784
Silicon 2,3-naphthalocyanine dihydroxide, 80% (dye content)	40,653-8	785
IR-792 perchlorate, 99% (dye content)	42,598-2	792
IR-797 chloride, 80% (dye content)	54,337-3	797
IR-797 perchlorate, 97% (dye content)	54,335-7	797
Silicon 2,3-naphthalocyanine dioctyloxide, 90% (dye content)	40,767-4	798
Dimethyl{4-[1,5,5-tris(4-dimethylaminophenyl)-2,4-pentadienylidene]-2 ,5-cyclohexadien-1-ylidene}ammonium perchlorate, 97%	39,236-7	800
IR-806, 90% (dye content)	54,334-9	806
Vanadyl 2,11,20,29-tetra-*tert*-butyl-2,3-naphthalocyanine, 95% (dye content)	43,296-2	808
Vanadyl 3,10,17,24-tetra-*tert*-butyl-1,8,15,22-tetrakis(dimethylamino)-29*H*,31*H*-phthalocyanine, 80% (dye content)	39,389-4	810
IR-813 chloride, 90% (dye content)	54,340-3	813
IR-813 perchlorate, 60% (dye content)	54,338-1	813
1,1′-Diethyl-4,4′-dicarbocyanine iodide, 90% (dye content)	39,220-0	814
Gallium(III) 2,3-naphthalocyanine chloride, 90% (dye content)	40,884-0	814
Vanadyl 2,3-naphthalocyanine	45,872-4	817
IR-820, 80% (dye content)	54,336-5	820
IR-140	26,093-2	823

Tin(IV) 2,3-naphthalocyanine dichloride, 95% (dye content)	40,651-1	828
Tin(II) 2,3-naphthalocyanine, 85% (dye content)	40,881-6	829
Vanadyl 5,14,23,32-tetraphenyl-2,3-naphthalocyanine, 70% (dye content)	39,317-7	846
Nickel(II) 5,9,14,18,23,27,32,36-octabutoxy-2,3-naphthalocyanine, 98% (dye content)	41,885-4	848
Copper(II) 5,9,14,18,23,27,32,36-octabutoxy-2,3-naphthalocyanine, 97% (dye content)	41,528-6	853
5,9,14,18,23,27,32,36-Octabutoxy-2,3-naphthalocyanine, 95% (dye content)	41,207-4	867
Dimethyl{4-[1,7,7-tris(4-dimethylaminophenyl)-2,4,6-heptatrienylidene]-2,5-cyclohexadien-1-ylidene}ammonium perchlorate, 97%	39,237-5	895
Poly(copper phthalocyanine), 60% (dye content)	52,762-9	921
IR-27, 99% (dye content)	40,610-4	988
IR-1040, 95% (dye content)	40,513-2	###
IR-1048, 97% (dye content)	40,517-5	###
IR-1050, 97% (dye content)	40,518-3	###
IR-1051, 95% (dye content)	40,516-7	###
IR-1061, 80% (dye content)	40,512-4	###
IR-1100, 95% (dye content)	40,514-0	###
1,4-Bis(2-methylstyryl)benzene, 99%	22,244-5	
1,4-Bis(4-methyl-5-phenyl-2-oxazolyl)benzene, scintillation grade, 98%	22,291-7	
2-(4-Biphenylyl)-5-(4-*tert*-butylphenyl)-1,3,4-oxadiazole, 99+%	22,400-6	
2-(4-Biphenylyl)-5-phenyl-1,3,4-oxadiazole, 99%	25,785-0	
2-(4-Bromophenyl)-5-phenyl-1,3,4-oxadiazole, 96%	55,364-6	
2,1,3-Benzoxadiazole-5-carboxylic acid	55,661-0	
2,2′-Bipyridine-3,3′-diol, 98%	34,008-1	
2,5-Bis-(4-aminophenyl)-1,3,4-oxadiazole, 90% (dye content)	33,352-2	
2,5-Bis(5-*tert*-butyl-benzoxazol-2-yl)thiophene, scintillation grade, 99%	22,399-9	
5-(4-Methoxyphenyl)-1,3,4-oxadiazole-2-thiol	54,896-0	
5-(4-Methylphenyl)-1,3,4-oxadiazole-2-thiol	54,895-2	
5-(4-Pyridyl)-1,3,4-oxadiazole-2-thiol, 97%	43,856-1	
5-Phenyl-1,3,4-oxadiazole-2-thiol, 98%	39,180-8	
7-Diethylamino-4-methylcoumarin, 99%	D8,775-9	
Copper(II) 4,4¢,4¢¢,4¢¢¢-tetra	31,020-4	

Appendix C

Matlab Code

C.1 Ray Tracer

C.1.1 Ray propagation Functions

Ray Tracer Main Program - trace_rays.m

```
function raysout = trace_rays(planes,rays,int_bound,bounce_bound,messages)
tic
time = clock;
% boundary of ray intensity to consider negligible
%int_bound = .2;
% boundary on number of bounces from initial ray before we stop tracing
%bounce_bound = 100;

numplanes = length(planes);
numrays = length(rays);
numraystemp = 0;
% trace all rays
done = false;
rays_traced = 0;
loss = 0;
silverloss = .98;
zerothresh = 1e-10;
% set all plane values precision to 10 decimals
for n=1:length(planes)
  planes(n).normal = round(planes(n).normal.*(1/zerothresh))./(1/zerothresh);
  planes(n).intercept = round(planes(n).intercept.*(1/zerothresh))./(1/zerothresh);
  planes(n).bounds = round(planes(n).bounds.*(1/zerothresh))./(1/zerothresh);
  planes(n).endpoints = round(planes(n).endpoints.*(1/zerothresh))./(1/zerothresh);
end
while(done == false)
  if (messages == true)
    if ((rays_traced - floor(rays_traced/1000)*1000) == 0)
      toc
      fprintf('[Trace_Rays] Number of Rays Traced: %d of %d\n',rays_traced,numrays+numraystemp)
      tic
    end
  end

  rays_traced = rays_traced+1;
 %
 %
 %  trying self replication approach
    [isect normal pln distnc]= find_intersect(planes,rays(rays_traced),zerothresh);
    isect;
    normal;
    pln;
    distnc;
    % check if the ray is done
%     if (pln == -1)
      % mark the flag
%       done = true;
%     else
      % rays(rays_traced).start(:,size(rays(rays_traced).start,2)+1) = isect;
      %rays(rays_traced).surface(:,size(rays(rays_traced).surface,2)+1) = pln;
      rays(rays_traced).start(:,2) = isect;
      rays(rays_traced).surface(:,2) = pln;
      % mm scale defined, calculate time traveled
      rays(rays_traced).TOF = rays(rays_traced).TOF + distnc/1000*rays(rays_traced).index/3e8;
      % if we define loss, calculate new intensity
      if (rays(rays_traced).index == 1.56)
        rays(rays_traced).intensity = rays(rays_traced).intensity*exp(loss*distnc);
      end
      % if we intersected a plane, run snells law
      if (pln > 0)
        [transray reflectray] = snells_law(rays(rays_traced),planes(pln),isect,normal);
        % check for our stop conditions and set the starting plane
        if (transray.intensity > int_bound && transray.bounce < bounce_bound)
          numraystemp = numraystemp+1;
          transray.surface = pln;
          transray.surfnormal = normal;
          raystemp(numraystemp) = transray;
        end
        if (reflectray.intensity > int_bound && reflectray.bounce < bounce_bound)
          numraystemp = numraystemp+1;
          reflectray.surface = pln;
          reflectray.surfnormal = normal;
          raystemp(numraystemp) = reflectray;
        end
      end
      % section is used to speed up array creation and destruction
      % if we hit 5000 stored, move them into our rays array
      if (numraystemp == 5000)
        rays(numrays+1:numrays+numraystemp) = raystemp(1:numraystemp);
        numrays = numrays + numraystemp;
        numraystemp = 0;
      end
      % if we have no more rays to trace
      if (rays_traced == numrays)
        % check if our temp rays are empty
        if (numraystemp > 0)
          rays(numrays+1:numrays+numraystemp) = raystemp(1:numraystemp);
          numrays = numrays + numraystemp;
          numraystemp = 0;
        else
          done = true;
        end
      end
end
raysout = rays;
if (messages == true)
fprintf('[Trace_Rays] Total Time Elapsed: %d seconds\n\n',etime(clock,time));
end
```

Ray propagation and plane intersection – find_intersect.m

```
function [intersect normalvar plane distance]= find_intersect(planes,ray,zerothresh)
% intersect is 3x1 intersection point
% angle is the angle made with respect to the next intersected plane normal
% vector
% plane is the plane intersected
numplanes = length(planes);
mag = Inf;
for n=1:numplanes
```

```
  % get the normal vector for the nth plane
  normal = planes(n).normal;
  % get plane coefficient
  coeff = planes(n).intercept;
  % get the most recent intersection point
  point = ray.start;
  % get the most recent direction vector
  direction = ray.direction;
  % are we not parallel?
  dotp = normal*direction;
  check = true;
  if (dotp == 0)
     check = false;
  else
     % if we have a rectangle or a triangle
     if (strcmp(planes(n).type,'rect') == 1 || strcmp(planes(n).type,'tri') == 1)
        % find scalar that leads to intersection
        m = coeff - normal*point;
        m = m/(dotp);
        % make intersection point
        isect(:,n) = point + m*direction;
        % get rid of error prone precision
        isect(:,n) = round(isect(:,n)*(1/zerothresh))/(1/zerothresh);
        % we must round the distance for our zero checks
        m = round(m*(1/zerothresh/10000))/(1/zerothresh/10000);


        % check for special planes (lenses, bends)
     % if we have a circle
     elseif (strcmp(planes(n).type,'circle')==1)
        % get the axial axis
        depy = abs(planes(n).sp.direction);
        % determine the independent axis
        depx = abs(cross(planes(n).sp.commonparallel,planes(n).sp.direction));
        % solve for the intersection distances along the circle
        quada = dot(direction,depx)^2+dot(direction,depy)^2;
        quadb = 2*(dot(direction,depx)*(dot(point,depx)-dot(planes(n).sp.center,depx))+dot(direction,depy)*(dot(point,depy)-
dot(planes(n).sp.center,depy)));
        quadc = (dot(point,depx)-dot(planes(n).sp.center,depx))^2+(dot(point,depy)-dot(planes(n).sp.center,depy))^2-
planes(n).sp.radius^2;
        m1 = (-quadb + sqrt(quadb^2-4*quada*quadc))/2/quada;
        m2 = (-quadb - sqrt(quadb^2-4*quada*quadc))/2/quada;
        % if we do not reduce precision, checks against zero will fail
        % since we already set a zerothresh, we must pick a higher
        % threshold set to zerothresh*10000
        m1 = round(m1*(1/zerothresh/10000))/(1/zerothresh/10000);
        m2 = round(m2*(1/zerothresh/10000))/(1/zerothresh/10000);
        % if first solution is not valid
        if ((isreal(m1) == 0) || m1 <= 0 || sign(dot(point,depy)+m1*dot(direction,depy)-dot(planes(n).sp.center,depy)) ~=
sign(dot(planes(n).sp.direction,depy)))
           m1 = Inf;
     %      normal = (depy*-(dot(point,depx)+m1*dot(direction,depx)-dot(planes(n).sp.center,depx))/sqrt(planes(n).sp.radius^2-...
     %         (dot(point,depx)+m1*dot(direction,depx)-dot(planes(n).sp.center,depx))^2))+depx;
     %      normal = normal/norm(normal);
        end
        % if second solution is not valid
        if ((isreal(m2) == 0) || m2 <= 0 || sign(dot(point,depy)+ m2*dot(direction,depy)-dot(planes(n).sp.center,depy)) ~=
sign(dot(planes(n).sp.direction,depy)))
           m2 = Inf;
     %      normal = (depy*-(dot(point,depx)+m2*dot(direction,depx)-dot(planes(n).sp.center,depx))/sqrt(planes(n).sp.radius^2-...
     %         (dot(point,depx)+m2*dot(direction,depx)-dot(planes(n).sp.center,depx))^2))+depx;
     %      normal = normal/norm(normal);
        end
        % figure out which is the real intersection
        if (intersect_test(planes(n),point+m1*direction) == false)
           m1 = inf;
        end
        if (intersect_test(planes(n),point+m2*direction) == false)
           m2 = inf;
        end
```

```
      m = min([m1 m2]);
      if (m < Inf)
        isect(:,n) = point+m*direction;
        isect(:,n) = round(isect(:,n)*(1/zerothresh))/(1/zerothresh);
        normal = depy*(dot(point,depy)+m*dot(direction,depy)-
dot(planes(n).sp.center,depy))+depx*(dot(point,depx)+m*dot(direction,depx)-dot(planes(n).sp.center,depx));
        normal = normal/norm(normal);
       % if (m < .001)
       %   m
       % end
      else
        isect(:,n) = [0 0 0];
        normal = [0 0 0];
      end

    end
    % make escape sequence if we want to remove a plane from our declaration
    if (planes(n).number == -1)
      check = false;
    else
      % check if point positive,smaller than minimum, and within bounds,
      % or intersects itself or another plane of same angle
      % m set to < 0 incase we need a double bounce off a plane
      % overlap
      % old check (planes(n).number == ray.surface(1)) ||

      if ( m < 0 || m > min(mag) || intersect_test(planes(n),isect(:,n)) == false)
        check = false;
      elseif (m == 0) % if we do hit a double bounce, check if we are actually supposed to bounce, or if we have an overlap
        if (ray.surface(1) ~= 0)   % make sure we have a surface to compare
          overlap = round(sum(cross(normal,ray.surfnormal))*(1/zerothresh))/(1/zerothresh);
        else
          overlap = 1;
        end
        if (overlap == 0)
          check = false;
        end
      end
    end
  end
  % find the length of the vectors
  if (check == false)
    mag(n) = Inf;
   %  ang(n) = -1;
    normalout(n,:) = normal;
  else
    mag(n) = m;
   %  ang(n) = acos(abs(normal*direction));
    normalout(n,:) = normal;
  end
end
% find the smallest magnitude
[y i] = min(mag);
% if we are successful assign return parameters
if (y < Inf)
  intersect = isect(:,i);
 %  angle = ang(i);
  plane = planes(i).number;
  normalvar = normalout(i,:);
  distance = y;
else
  %if we do not intersect, propagate a magnitude of 1 and stop
  intersect = point+direction;
  normalvar = normalout(i);
  plane = -1;
  distance = 0;
end
```

Surface boundary checker – intersect_test.m

```
function check = intersect_test(plane,isect)
check = true;
if (strcmp(plane.type,'rect')==1)
  if (plane.bounds(1,1) > isect(1) || plane.bounds(1,2) < isect(1) || ...
        plane.bounds(2,1) > isect(2) || plane.bounds(2,2) < isect(2) || ...
        plane.bounds(3,1) > isect(3) || plane.bounds(3,2) < isect(3))
    check = false;
  end
elseif (strcmp(plane.type,'tri')==1)
  % get the points
  % find maximum projection angle xy,xz,yz
 % m(1) = plane.normal*[0;0;1];
 % m(2) = plane.normal*[0;1;0];
 % m(3) = plane.normal*[1;0;0];
  [maximum index] = max(plane.normal);
  switch index
    case 3, % xy plane
      v0 = plane.endpoints(1:2,1);
      v1 = plane.endpoints(1:2,2);
      v2 = plane.endpoints(1:2,3);
      p = isect(1:2);
    case 2, % xz plane
      v0 = planc.cndpoints(1:2:3,1);
      v1 = plane.endpoints(1:2:3,2);
      v2 = plane.endpoints(1:2:3,3);
      p = isect(1:2:3);
    case 1, % yz plane
      v0 = plane.endpoints(2:3,1);
      v1 = plane.endpoints(2:3,2);
      v2 = plane.endpoints(2:3,3);
      p = isect(2:3);
  end
  mat1 = [v1-v0 v2-v0];
  vectp = p-v0;
  coeff = inv(mat1)*vectp;
  if (coeff(1) < 0 || coeff(2) < 0 || coeff(1)+coeff(2) > 1)
    check = false;
  end
elseif (strcmp(plane.type,'circle')==1)
  % we can do a rectangular check since we already know our boundaries
  if (plane.bounds(1,1) > isect(1) || plane.bounds(1,2) < isect(1) || ...
        plane.bounds(2,1) > isect(2) || plane.bounds(2,2) < isect(2) || ...
        plane.bounds(3,1) > isect(3) || plane.bounds(3,2) < isect(3))
    check = false;
  end
end
```

Reflected and transmitted ray direction and power calculator – snells_law.m

```
function [transray reflectray] = snells_law(ray, plane, intersect, planenormal)
% inputs = a single ray, a single plane, an intersection point, and an
% intersection angle relative to normal
% outputs = transray
  n1 = ray.index;
  if (n1 == plane.material1)
    n2 = plane.material2;
  else
    n2 = plane.material1;
  end
  % find the critical angle
  if (n2 < n1)
    ang_crit = asin(n2/n1);
  else
    ang_crit = -1;
  end
  % create scatter slope with x and y scatter probability
% in future, assign sigma-x and sigma-y to each plane
scat = rand(1,2);
slopes1 = 2^.5 * plane.variance1* erfinv(2*scat(1) - 1);
slopes2 = 2^.5 * plane.variance2* erfinv(2*scat(2) - 1);
% modify plane normal to account for perturbation
% rotate vector xang and yang degrees about plane's parallel axes
% planes need to have specified orthogonal directions of surface
% roughness
if (strcmp(plane.type,'rect')==1 || strcmp(plane.type,'tri') ==1)
   planenormal = planenormal + slopes1*plane.scatvect1 + slopes2*plane.scatvect2;
   planenormal = planenormal / norm(planenormal);
elseif (strcmp(plane.type,'circle')==1)
   planenormal = planenormal + slopes1*plane.sp.commonparallel+slopes2*cross(plane.sp.commonparallel,planenormal);
   planenormal = planenormal / norm(planenormal);
end
% generate the incident angle
angle = acos(abs(planenormal*ray.direction));
% find transmission and reflection coefficients
% take into account TIR
if (ang_crit >= 0 && ang_crit <= angle)
   refc = 1;
% if TE ray
elseif (ray.polarization <= .5)
   refc = (sqrt(1-sin(angle)^2)-n2/n1*sqrt(1-(n1^2/n2^2)*sin(angle)^2))...
      /(sqrt(1-sin(angle)^2)+n2/n1*sqrt(1-(n1^2/n2^2)*sin(angle)^2));
% if TM ray
else
   refc = (sqrt(1-sin(angle)^2)-n1/n2*sqrt(1-(n1^2/n2^2)*sin(angle)^2))...
      /(sqrt(1-sin(angle)^2)+n1/n2*sqrt(1-(n1^2/n2^2)*sin(angle)^2));
end
% convert to power coefficients
refc = refc^2;
transc = 1-refc;
% set the reflect ray
reflectray = ray;
reflectray.start = intersect;
% find the new direction
% dot the normal to get the normal component of our direction
reflectray.direction = ray.direction - 2*(planenormal*ray.direction)*(planenormal');
reflectray.direction = reflectray.direction/norm(reflectray.direction);
reflectray.intensity = reflectray.intensity*refc;
reflectray.bounce = reflectray.bounce+1;
% set the transray
transray = reflectray;
% if transmission coefficient = 0, set intensity and leave
if (transc == 0)
   transray.intensity = 0;
else
   % find transmission angle
   ang_trans = asin(n1/n2*sin(angle));
   % get the normal going the direction of the vector
   normal = planenormal*ray.direction*(planenormal');
   % get parallel vector to plane
   parallel = ray.direction - normal;
   % normalize them
   if(norm(normal) ~= 0)
      normal = normal/norm(normal);
   end
   if(norm(parallel) ~= 0)
      parallel = parallel/norm(parallel);
   end
   % combine parallel and normal to form correct angle and normalize
   transray.direction = (normal + tan(ang_trans)*parallel)/norm(normal + tan(ang_trans)*parallel);
   transray.intensity = ray.intensity*transc;
   transray.index = n2;
end
```

C.1.2 Plotting Functions

Plot the ray end points – plot_endpoints.m

```
function plot_endpoints(rays)
figure
for n=1:length(rays)
   if (rays(n).bounce == 0)
      plot3(rays(n).direction(1),rays(n).direction(2),rays(n).direction(3),'.')
      if (ishold == false)
         hold
      end
   end
end
axis equal
```

Plot the rays and planes in 3D – plot_rays.m

```
function plot_rays(rays,planes)
figure;
if (ishold == 0)
   hold
end
fig = gcf;
ax = gca;
set(ax,'color',[.35 .35 .35]);
int_bound = .0000000001;
numplanes = length(planes);
for n=1:numplanes
   if (strcmp(planes(n).type,'circle')==1)
      if (dot(planes(n).sp.commonparallel,[0 0 1]) ~= 0)
         z = ones(1,1000)*min(planes(n).endpoints(3,:));
         z(1001:2000) = ones(1,1000)*max(planes(n).endpoints(3,:));
         if (dot(planes(n).sp.direction,[1 0 0]) ~= 0)
            y = linspace(min(planes(n).endpoints(2,:)),max(planes(n).endpoints(2,:)),1000);
            if (planes(n).sp.direction(1) == 1)
               x = sqrt(planes(n).sp.radius^2-(y-planes(n).sp.center(2)).^2)+planes(n).sp.center(1);
            else
               x = -sqrt(planes(n).sp.radius^2-(y-planes(n).sp.center(2)).^2)+planes(n).sp.center(1);
            end
         else
            x = linspace(min(planes(n).endpoints(1,:)),max(planes(n).endpoints(1,:)),1000);
            if (planes(n).sp.direction(2) == 1)
               y = sqrt(planes(n).sp.radius^2-(x-planes(n).sp.center(1)).^2)+planes(n).sp.center(2);
            else
               y = -sqrt(planes(n).sp.radius^2-(x-planes(n).sp.center(1)).^2)+planes(n).sp.center(2);
            end
         end
         plot3(x,y,z(1:1000),'k');
         plot3(x,y,z(1001:2000),'k');
      elseif (dot(planes(n).sp.commonparallel,[0 1 0]) ~= 0)
         y = ones(1,1000)*min(planes(n).endpoints(2,:));
         y(1001:2000) = ones(1,1000)*max(planes(n).endpoints(2,:));
         if (dot(planes(n).sp.direction,[1 0 0]) ~= 0)
            z = linspace(min(planes(n).endpoints(3,:)),max(planes(n).endpoints(3,:)),1000);
            if (planes(n).sp.direction(1) == 1)
               x = sqrt(planes(n).sp.radius^2-(z-planes(n).sp.center(3)).^2)+planes(n).sp.center(1);
            else
               x = -sqrt(planes(n).sp.radius^2-(z-planes(n).sp.center(3)).^2)+planes(n).sp.center(1);
            end
         else
            x = linspace(min(planes(n).endpoints(1,:)),max(planes(n).endpoints(1,:)),1000);
            if (planes(n).sp.direction(3) == 1)
               z = sqrt(planes(n).sp.radius^2-(x-planes(n).sp.center(1)).^2)+planes(n).sp.center(3);
            else
               z = -sqrt(planes(n).sp.radius^2-(x-planes(n).sp.center(1)).^2)+planes(n).sp.center(3);
            end
      end
      plot3(x,y(1:1000),z,'k');
      plot3(x,y(1001:2000),z,'k');
    elseif (dot(planes(n).sp.commonparallel,[1 0 0]) ~= 0)
      x = ones(1,1000)*min(planes(n).endpoints(1,:));
      x(1001:2000) = ones(1,1000)*max(planes(n).endpoints(1,:));
      if (dot(planes(n).sp.direction,[0 1 0]) ~= 0)
        z = linspace(min(planes(n).endpoints(3,:)),max(planes(n).endpoints(3,:)),1000);
        if (planes(n).sp.direction(2) == 1)
          y = sqrt(planes(n).sp.radius^2-(z-planes(n).sp.center(3)).^2)+planes(n).sp.center(2);
        else
          y = -sqrt(planes(n).sp.radius^2-(z-planes(n).sp.center(3)).^2)+planes(n).sp.center(2);
        end
      else
        y = linspace(min(planes(n).endpoints(2,:)),max(planes(n).endpoints(2,:)),1000);
        if (planes(n).sp.direction(3) == 1)
          z = sqrt(planes(n).sp.radius^2-(y-planes(n).sp.center(2)).^2)+planes(n).sp.center(3);
        else
          z = -sqrt(planes(n).sp.radius^2-(y-planes(n).sp.center(2)).^2)+planes(n).sp.center(3);
        end
      end
      plot3(x(1:1000),y,z,'k');
      plot3(x(1001:2000),y,z,'k');
    end
  else
    plot3(planes(n).endpoints(1,:),planes(n).endpoints(2,:),planes(n).endpoints(3,:),'k')
  end
end
numrays = length(rays);
%get colormap

intmax = rays(1).intensity;
for n=1:numrays
  if (rays(n).intensity > intmax)
    intmax = rays(n).intensity;
  end
end

for n=1:numrays
  if (rays(n).intensity > int_bound)
    % red blue scheme
    plot3(rays(n).start(1,:),rays(n).start(2,:),rays(n).start(3,:),'color',[rays(n).intensity/intmax 0 1-rays(n).intensity/intmax])
    % white
    %plot3(rays(n).start(1,:),rays(n).start(2,:),rays(n).start(3,:),'color',[1 1 1])
  end
end
axis equal
```

Plot rays incident on a plane for hit statistics – surfaceplot.m

```
function [zstore planerays scatarray] = surfaceplot(rays,plane,startsurf,endsurf,bouncebound,graphon,xres,yres)
% surface definitions .... -1 defined as infinity surface
%              -2 defined as any surface
% find all rays that originate on the selected surface and exit
numrays = length(rays);
newcount = 0;
for n=1:numrays
  if((rays(n).surface(1) == startsurf || startsurf == -2) && (endsurf == -2 || rays(n).surface(2) == endsurf) && rays(n).bounce >
bouncebound)
    newcount = newcount+1;
    surfacerays(newcount) = rays(n);
  end
end

if (newcount == 0)
  zstore = 0;
  planerays = 0;
  scatarray = 0;
```

```
else
planerays = surfacerays;
if (graphon == true)
   % find largest translation plane
   % xy?
   m(1) = plane.normal*[0;0;1];
   % xz?
   m(2) = plane.normal*[0;1;0];
   % yz?
   m(3) = plane.normal*[1;0;0];
   [y index] = max(m);

   % flatten surface rays and 2d-ize points
   switch index
      case 1,
         for n=1:newcount
            surfacerays(n).start = surfacerays(n).start(1:2,:);
         end
         minx = min(plane.endpoints(1,:));
         maxx = max(plane.endpoints(1,:));
         miny = min(plane.endpoints(2,:));
         maxy = max(plane.endpoints(2,:));

      case 2,
         for n=1:newcount
            surfacerays(n).start = surfacerays(n).start(1:2:3,:);
         end
         minx = min(plane.endpoints(1,:));
         maxx = max(plane.endpoints(1,:));
         miny = min(plane.endpoints(3,:));
         maxy = max(plane.endpoints(3,:));
      case 3,
         for n=1:newcount
            surfacerays(n).start = surfacerays(n).start(2:3,:);
         end
         minx = min(plane.endpoints(2,:));
         maxx = max(plane.endpoints(2,:));
         miny = min(plane.endpoints(3,:));
         maxy = max(plane.endpoints(3,:));
   end
   % get x's
   for n=1:newcount
      if (plane.number == startsurf)
         x(n) = surfacerays(n).start(1,1);
         y(n) = surfacerays(n).start(2,1);
         z(n) = surfacerays(n).intensity;
      elseif (plane.number == endsurf)
         x(n) = surfacerays(n).start(1,2);
         y(n) = surfacerays(n).start(2,2);
         z(n) = surfacerays(n).intensity;
      end
   end
   array = [x' y' z'];

   %plane.endpoints
   figure
   scatter(x,y,6,z,'filled')
   axis equal


   xboxes = xres;
   yboxes = yres;
   xaxis = linspace(minx,maxx,xboxes);
   yaxis = linspace(miny,maxy,yboxes);
   [xnum ynum] = size(array);
   % initialize count for each box
   xstore = zeros(xboxes,1);
   zstore = zeros(xboxes,yboxes);
   for n=1:xnum
      % find the shortest distance between this x and a box
```

```
    dist = abs(array(n,1)-xaxis);
    [val ind] = min(dist);
    % increase the count
    xstore(ind,1) = xstore(ind,1)+1;
    % add the index of this entry into the xstore
    xstore(ind,xstore(ind,1)+1) = n;
  end
  for n=1:xboxes
    % get the length of the current x
    lengthx = xstore(n,1);
    % find the y index that each x will belong to
    if (lengthx > 0)
      for m=2:lengthx+1
        dist = abs(array(xstore(n,m),2)-yaxis);
        [val ind] = min(dist);
        % add the intensity to this box
        zstore(n,ind) = array(xstore(n,m),3)+zstore(n,ind);
      end
    end
  end
  figure
  surf(xaxis,yaxis,zstore')
  figure
  contour(xaxis,yaxis,zstore')
  % scatter plot the surface
  scatcount = 0;
  for n=1:xboxes
    for m=1:yboxes
      scatcount = scatcount+1;
      scatarray(scatcount,:) = [xaxis(n) yaxis(m) zstore(n,m)];
    end
  end
  %figure
  %scatter(scatarray(:,1),scatarray(:,2),6,scatarray(:,3))

else
  zstore = 0;
  scatarray = 0;
  for n=1:newcount
    zstore = zstore + surfacerays(n).intensity;
  end
end
end
```

Plot vectorized hit statistics into an intensity surface contour – surfaceconvert.m

```
function zstore = surfaceconvert(array,xboxes,yboxes)
minx = min(array(:,1));
miny = min(array(:,2));
maxx = max(array(:,1));
maxy = max(array(:,2));
xaxis = linspace(minx,maxx,xboxes);
yaxis = linspace(miny,maxy,yboxes);
xnum = length(array(:,1));
xstore = zeros(xboxes,1);
  zstore = zeros(xboxes,yboxes);
  for n=1:xnum
    % find the shortest distance between this x and a box
    dist = abs(array(n,1)-xaxis);
    [val ind] = min(dist);
    % increase the count
    xstore(ind,1) = xstore(ind,1)+1;
    % add the index of this entry into the xstore
    xstore(ind,xstore(ind,1)+1) = n;
  end
  for n=1:xboxes
    % get the length of the current x
    lengthx = xstore(n,1);
  % find the y index that each x will belong to
  if (lengthx > 0)
    for m=2:lengthx+1
      dist = abs(array(xstore(n,m),2)-yaxis);
      [val ind] = min(dist);
      % add the intensity to this box
      zstore(n,ind) = array(xstore(n,m),3)+zstore(n,ind);
    end
  end
end
figure
surf(xaxis,yaxis,zstore')
figure
contourf(xaxis,yaxis,zstore')
```

C.1.3 Ray Filtering Functions

Finding a plotted ray – find_ray.m

```
function index = find_ray(rays,point1,point2)
index = 0;
numrays = length(rays);
n = 1;
if (length(point2) == 3)
  %run 2 point search
  while(n <= numrays && index == 0)
    a = rays(n).start == [point1 point2];
    b = rays(n).start == [point2 point1];
    if (all(all(a)) || all(all(b)))
      index = n;
    end
    n = n+1;
  end
else
  %run 1 point search
  while(n <= numrays && index == 0)
    a = rays(n).start(:,1) == point1;
    b = rays(n).start(:,2) == point1;
    if (all(a) || all(b))
      index = n;
    end
    n = n+1;
  end
end
```

Regenerate initial rays based on hit statistic rays – backtracerays.m

```
function rays = backtracerays(planerays,raysold)
len = length(planerays);
% generate a sortable array
for m=1:len
  sources(m) = planerays(m).source;
end

mini = Inf;
count = 1;
% sort it
for m=1:len
  [val ind] = min(sources);
  % if we have a duplicate or bad source, remove it from list
  if (val == mini || val == Inf)
    sources(ind) = Inf;
  else
    rays(count) = raysold(val);
    count = count+1;
    mini = val;
  end
end
```

Filter out ray groups based on ray origins – filter_traced_rays.m

```
function raystore = filter_traced_rays(trays,outrays)
% code to filter out intersected rays and save them
% given a set of rays emitted from a given surface, this function will find
% all parent and children
rayscount = 0;
for k=1:length(outrays)
  for m=1:length(trays)
    if (trays(m).source == outrays(k).source)
      rayscount = rayscount+1;
      raystore(rayscount) = trays(m);
    end
  end
end
if (rayscount == 0)
  raystore = 0;
end
```

C.1.4 Postprocessing Functions

Ray traversal intensity versus delay histogram – timeofflight.m

```
function [TOF intensity] = timeofflight(planerays,res);
for n=1:length(planerays)
  TOF(n,:) = [planerays(n).TOF planerays(n).intensity];
end
figure
scatter(TOF(:,1),TOF(:,2),6);
% do a bin count to determine intensity per time
tmin = min(TOF(:,1));
tmax = max(TOF(:,1));
x = linspace(tmin,tmax,res);
intensity = zeros(res,1);
for n=1:length(TOF(:,1))
  dist = x-TOF(n,1);
  [y i] = min(abs(dist));
  if (dist(i) < 0)
    i2 = i+1;
  elseif (dist(i) > 0)
    i2 = i-1;
  else
    i2 = i;
  end
  i2 = i;
  prop = y/(x(2)-x(1));
  intensity(i) = intensity(i)+(1-y)*TOF(n,2);
  intensity(i2) = intensity(i2)+y*TOF(n,2);
end
figure
plot(x,intensity)
```

Surface roughness induced spread angle for output rays – modecompare.m

```
function rayangle = modecompare(planerays,rays,plane,direction)
% rayangle = modecompare(planerays,rays,plane,direction
% planerays = output rays
% rays = input rays
% plane = central axis
% direction = 1 or -1
```

```
for n=1:length(planerays)
   rayangle(n,1) = acos(plane.normal*planerays(n).direction*direction)*180/pi;
   rayangle(n,2) = acos(plane.normal*rays(planerays(n).source).direction*direction)*180/pi;
end
[yhist xhist] = hist(rayangle(:,1)-rayangle(:,2));
figure;plot(xhist,yhist);
xlabel('Change in Angle')
ylabel('Counts')
figure;plot(rayangle(:,2),rayangle(:,1)-rayangle(:,2),'.')
xlabel('Original Angle')
ylabel('Change in Angle')
```

C.1.5 Ray Generators

Uniform point source ray burst -z direction – ray_burst.m

```
function rays = ray_burst(points, center,phimax,intensity)
spherepercentage = (1-cos(phimax))/2;
x = rand(1,points)*phimax;
phi = acos(1-(1-cos(phimax))./(phimax)*x);
theta = rand(1,points)*2*pi;
% get directions
direction = [sin(phi).*cos(theta);sin(phi).*sin(theta);-cos(phi)];
%scatter(direction(1,:),direction(2,:),6);
raynum = size(direction,2);
rayint = intensity*spherepercentage/raynum;
% power in s polarization
polarization = rand(points,1);
for i=1:points
   rays(i) = struct('source',i,...
          'bounce',0,...
          'surface',0,...
          'start',center,...
          'direction',direction(:,i),...
          'wavelength',600,...
          'intensity',rayint,...
          'spower',polarization(i),...
          'ppower',(1-polarization(i)),...
          'phase',0,...
          'index',1.56,...
          'TOF',0);
end
```

Uniform point source ray burst -x direction – ray_burstx.m

```
function rays = ray_burstx(points, center,phimax,intensity)
spherepercentage = (1-cos(phimax))/2;
x = rand(1,points)*phimax;
phi = acos(1-(1-cos(phimax))./(phimax)*x);
theta = rand(1,points)*2*pi;
% get directions
direction = [-cos(phi);sin(phi).*cos(theta);sin(phi).*sin(theta)];
raynum = size(direction,2);
rayint = intensity*spherepercentage/raynum;
% power in s polarization
polarization = rand(points,1);
%randomx = rand(points,1)*.5;
randomx = rand(points,1)*0;
for i=1:points
   rays(i) = struct('source',i,...
          'bounce',0,...
          'surface',0,...
          'surfnormal',[0 0 0],...
          'start',center+[randomx(i);0;0],...
          'direction',direction(:,i),...
```

```
        'wavelength',.6,...
        'intensity',rayint,...
        'polarization',polarization(i),...
        'phase',0,...
        'index',1,...
        'TOF',0);
end
```

Uniform point source ray burst +x direction – ray_burstx_reverse.m

```
function rays = ray_burstx(points, center,phimax,intensity)
spherepercentage = (1-cos(phimax))/2;
x = rand(1,points)*phimax;
phi = acos(1-(1-cos(phimax))./(phimax)*x);
theta = rand(1,points)*2*pi;
% get directions
direction = [cos(phi);sin(phi).*cos(theta);sin(phi).*sin(theta)];
raynum = size(direction,2);
rayint = intensity*spherepercentage/raynum;
% power in s polarization
polarization = rand(points,1);
for i=1:points
   rays(i) = struct('source',i,...
        'bounce',0,...
        'surface',0,...
        'surfnormal',[0 0 0],...
        'start',center,...
        'direction',direction(:,i),...
        'wavelength',.6,...
        'intensity',rayint,...
        'polarization',polarization(i),...
        'phase',0,...
        'index',1,...
        'TOF',0);
end
```

Collimated beam in x direction – ray_collx.m

```
function rays = ray_collx(points,center,theta,phi,square,intensity)
% get directions
direction = [cos(phi);sin(phi).*cos(theta);sin(phi).*sin(theta)];

% power in s polarization
polarization = rand(points,1);
loc = rand(points,2)*square-square/2;
for i=1:points
   rays(i) = struct('source',i,...
        'bounce',0,...
        'surface',0,...
        'surfnormal',[0 0 0],...
        'start',center-[0;loc(i,1);loc(i,2)],...
        'direction',direction,...
        'wavelength',.6,...
        'intensity',intensity,...
        'polarization',polarization(i),...
        'phase',0,...
        'index',1,...
        'TOF',0);
end
```

C.1.6 Plane Generation Functions

General surface definition – makeplane.m

```
function output = makeplane(number,type,normal,intercept,index1,index2,endpoints,scatvect1,variance1,scatvect2,variance2,sp)
% bounds = [xmin xmax;ymin ymax;zmin zmax]
% normal = [x y z]
if (strcmp(type,'rect') == 1 || strcmp(type,'tri') == 1)
   sp = normalplane();
end
output = struct('number',number,...
            'type',type,...
            'material1',index1,...
            'material2',index2,...
            'normal',normal,...
            'intercept',intercept,...
            'bounds',[min(endpoints(1,:)) max(endpoints(1,:));min(endpoints(2,:)) max(endpoints(2,:));min(endpoints(3,:))
max(endpoints(3,:))],...
            'endpoints',endpoints,...
            'scatvect1',scatvect1,...
            'variance1',variance1,...
            'scatvect2',scatvect2,...
            'variance2',variance2,...
            'sp',sp);
```

Circle Specific definition – circleplane.m

```
function specialstruct = circleplane(radius,center,direction,commonparallel)
%only works for circles located within cartesian planes...
specialstruct.radius = radius;
specialstruct.center = center;
specialstruct.direction = direction;
specialstruct.commonparallel = commonparallel;
```

C.1.7 Defined Structures

Multimode Optical Fiber with single collection fiber probe – fiber.m

```
function planes = fiber(radius,center,length)

index1 = 1.4036;
index2 = 1.49;
indexglass = 1.52;
indexair = 1;
planes(1) = makeplane(1,'rect',[1 0 0],0,1,index2,[0 0 0 0 0;-radius+center(2) -radius+center(2) -radius+center(2) radius+center(2)
radius+center(2);radius+center(3) -radius+center(3) radius+center(3) radius+center(3) -radius+center(3)],[1 0 0],0,[0 0 1],0);
planes(2) = makeplane(2,'rect',[1 0 0],length,1,index2,[length length length length length;radius+center(2) -radius+center(2) -
radius+center(2) radius+center(2) radius+center(2);-radius+center(3) -radius+center(3) radius+center(3) radius+center(3) -
radius+center(3)],[1 0 0],0,[0 0 1],0);

planes(3) = makeplane(3,'circle',[0 0 1],0,index1,index2,[length 0 0 length length;radius+center(2) -radius+center(2) -radius+center(2)
radius+center(2) radius+center(2);0+center(3) 0+center(3) -radius+center(3) -radius+center(3) 0+center(3)],[1 0 0],0,[0 1
0],0,circleplane(radius,center,[0 0 -1],[1 0 0]));
planes(4) = makeplane(4,'circle',[0 0 1],0,index1,index2,[length 0 0 length length;radius+center(2) -radius+center(2) -radius+center(2)
radius+center(2) radius+center(2);0+center(3) 0+center(3) radius+center(3) radius+center(3) 0+center(3)],[1 0 0],0,[0 1
0],0,circleplane(radius,center,[0 0 1],[1 0 0]));

radius = 0.25;
center = [0 0 0.8];
length = 10;

planes(5) = makeplane(5,'rect',[1 0 0],0,1,index2,[0 0 0 0 0;-radius+center(2) -radius+center(2) -radius+center(2) radius+center(2)
radius+center(2);radius+center(3) -radius+center(3) radius+center(3) radius+center(3) -radius+center(3)],[1 0 0],0,[0 0 1],0);
planes(6) = makeplane(6,'rect',[1 0 0],length,1,index2,[length length length length length;radius+center(2) -radius+center(2) -
radius+center(2) radius+center(2) radius+center(2);-radius+center(3) -radius+center(3) radius+center(3) radius+center(3) -
radius+center(3)],[1 0 0],0,[0 0 1],0);
```

```
planes(7) = makeplane(7,'circle',[0 0 1],0,index1,index2,[length 0 0 length length;radius+center(2) -radius+center(2) -radius+center(2)
radius+center(2) radius+center(2);0+center(3) 0+center(3) -radius+center(3) -radius+center(3) 0+center(3)],[1 0 0],0,[0 1
0],0,circleplane(radius,center,[0 0 -1],[1 0 0]));
planes(8) = makeplane(8,'circle',[0 0 1],0,index1,index2,[length 0 0 length length;radius+center(2) -radius+center(2) -radius+center(2)
radius+center(2) radius+center(2);0+center(3) 0+center(3) radius+center(3) radius+center(3) 0+center(3)],[1 0 0],0,[0 1
0],0,circleplane(radius,center,[0 0 1],[1 0 0]));

% substrate
len = 2;
planes(9) = makeplane(9,'rect',[1 0 0],-1,1,indexglass,[-1 -1 -1 -1 -1;len len -len -len len; len -len -len len len],[1 0 0],0,[0 0 1],0);

%back = 240;
%planes(5) = makeplane(5,'rect',[1 0 0],-back/2,indexair,indexair,[-back/2 -back/2 -back/2 -back/2 -back/2;-back -back back+1
back+1 -back;-back back+1 back+1 -back -back],[0 1 0],0,[0 0 1],0);
```

Ideal straight waveguide – straight_guide.m

```
function planes = straight_guide()
index1 = 1.43;
index2 = 1.56;
indexair = 1;
len = 75;
planes(1) = makeplane(1,'rect',[0 1 0],0,index1,index2,[0 0 len len 0;0 0 0 0 0;0 1 1 0 0],[1 0 0],0,[0 0 1],0);
planes(2) = makeplane(2,'rect',[0 0 1],0,index1,index2,[0 0 len len 0;0 1 1 0 0;0 0 0 0 0],[1 0 0],0,[0 1 0],0);
planes(3) = makeplane(3,'rect',[0 1 0],1,index1,index2,[0 0 len len 0;1 1 1 1 1;0 1 1 0 0],[1 0 0],0,[0 0 1],0);
planes(4) = makeplane(4,'rect',[0 0 1],1,index1,index2,[0 0 len len 0;0 1 1 0 0;1 1 1 1 1],[1 0 0],0,[0 1 0],0);
planes(5) = makeplane(5,'rect',[1 0 0],0,indexair,index2,[0 0 0 0 0;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0 1],0);
planes(6) = makeplane(6,'rect',[1 0 0],len,indexair,index2,[len len len len len;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0 1],0);

back = 240;
planes(7) = makeplane(7,'rect',[1 0 0],-back/2,indexair,indexair,[-back/2 -back/2 -back/2 -back/2 -back/2;-back -back back+1 back+1 -
back;-back back+1 back+1 -back -back],[0 1 0],0,[0 0 1],0);
```

Straight waveguide with scattering – straight_guide_scat.m

```
function planes = straight_guide()

index1 = 1.43;
index2 = 1.56;
indexair = 1;
len = 75;
var = 0.01;
planes(1) = makeplane(1,'rect',[0 1 0],0,index1,index2,[0 0 len len 0;0 0 0 0 0;0 1 1 0 0],[1 0 0],var,[0 0 1],var);
planes(2) = makeplane(2,'rect',[0 0 1],0,index1,index2,[0 0 len len 0;0 1 1 0 0;0 0 0 0 0],[1 0 0],0,[0 1 0],0);
planes(3) = makeplane(3,'rect',[0 1 0],1,index1,index2,[0 0 len len 0;1 1 1 1 1;0 1 1 0 0],[1 0 0],var,[0 0 1],var);
planes(4) = makeplane(4,'rect',[0 0 1],1,index1,index2,[0 0 len len 0;0 1 1 0 0;1 1 1 1 1],[1 0 0],0,[0 1 0],0);
planes(5) = makeplane(5,'rect',[1 0 0],0,indexair,index2,[0 0 0 0 0;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0 1],0);
planes(6) = makeplane(6,'rect',[1 0 0],len,indexair,index2,[len len len len len;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0 1],0);
%back = 240;
%planes(7) = makeplane(7,'rect',[1 0 0],-back/2,indexair,indexair,[-back/2 -back/2 -back/2 -back/2 -back/2;-back -back back+1
back+1 -back;-back back+1 back+1 -back -back],[0 1 0],0,[0 0 1],0);
radius = 75;
center = [0 .5 .5];
dist = 60;
planes(7) = makeplane(7,'circle',[1 0 0],0,indexair,indexair,[0+center(1) 0+center(1) -radius+center(1) -radius+center(1)
0+center(1);radius+center(2) -radius+center(2) -radius+center(2) radius+center(2) radius+center(2);-dist dist dist -dist -dist],[1 0
0],0,[0 1 0],0,circleplane(radius,center,[-1 0 0],[0 0 1]));

% spacing = 0;
% res = 15;
% for n=1:res-1
% spacing = spacing + len/res;
% planes(7+n) = makeplane(7+n,'rect',[1 0 0],len-spacing,index2,index2,[len-spacing len-spacing len-spacing len-spacing len-
spacing;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0 1],0);
% end
```

Waveguide backplane structure – bend_guide_total.m

```
function planes = bend_guide_total()

angle = 45*pi/180;
normx = cos(angle)*sin(angle)^2;
normz = (cos(angle)^2)*sin(angle);
x = -1/tan(angle);
index1 = 1.43; % cladding
index2 = 1.56; % core
index3 = 0; % bend interface
%single
%--------------------------------------------------------
planes(1) = makeplane(1,'rect',[0 1 0],0,index1,index2,[0 0 10 10 0;0 0 0 0 0;0 1 1 0 0],[1 0 0],0,[0 0 1],0);
planes(2) = makeplane(2,'rect',[0 0 1],0,index1,index2,[0 0 10 10 0;0 1 1 0 0;0 0 0 0 0],[1 0 0],0,[0 1 0],0);
planes(3) = makeplane(3,'rect',[0 1 0],1,index1,index2,[0 0 10 10 0;1 1 1 1 1;0 1 1 0 0],[1 0 0],0,[0 0 1],0);
% closed faced
%planes(4) = makeplane(4,'rect',[0 0 1],1,index1,index2,[x x 10 10 x;0 1 1 0 0;1 1 1 1 1],[1 0 0],0,[0 1 0],0);
% Open faced
planes(4) = makeplane(4,'rect',[0 0 1],1,index1,index2,[0 0 10 10 0;0 1 1 0 0;1 1 1 1 1],[1 0 0],0,[0 1 0],0);

planes(5) = makeplane(5,'rect',[normx 0 normz]/norm([normx 0 normz]),0,index3,index2,[0 0 x x 0;0 1 1 0 0;0 0 1 1 0],[0 1
0],0,[normz 0 -normx]/norm([normz 0 -normx]),0);
planes(6) = makeplane(6,'rect',[1 0 0],10,1,index2,[10 10 10 10 10;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0 1],0);
planes(7) = makeplane(7,'tri',[0 1 0],0,index1,index2,[0 0 x 0;0 0 0 0;0 1 1 0],[1 0 0],0,[0 0 1],0);
planes(8) = makeplane(8,'tri',[0 1 0],1,index1,index2,[0 0 x 0;1 1 1 1;0 1 1 0],[1 0 0],0,[0 0 1],0);
%dist = 10;
%planes(9) = makeplane(9,'rect',[0 0 1],dist,index1,index2,[-5 -5 1 1 -5;-1 2 2 -1 -1;dist dist dist dist dist],[1 0 0],0,[0 1 0],0);


%--------------------------------------------------------
% triple
%--------------------------------------------------------
% planes(1) = makeplane(1,'rect',[0 1 0],-1,index1,index2,[0 0 10 10 0;-1 -1 -1 -1 -1;0 1 1 0 0],[1 0 0],0.0085,[0 0 1],0.0085);
% planes(2) = makeplane(2,'rect',[0 0 1],0,index1,index2,[0 0 10 10 0;-1 2 2 -1 -1;0 0 0 0 0],[1 0 0],0,[0 1 0],0);
% planes(3) = makeplane(3,'rect',[0 1 0],2,index1,index2,[0 0 10 10 0;2 2 2 2 2;0 1 1 0 0],[1 0 0],0.0085,[0 0 1],0.0085);
% planes(4) = makeplane(4,'rect',[0 0 1],1,index1,index2,[x x 10 10 x;-1 2 2 -1 -1;1 1 1 1 1],[1 0 0],0,[0 1 0],0);
%
% planes(5) = makeplane(5,'rect',[normx 0 normz]/norm([normx 0 normz]),0,index3,index2,[0 0 x x 0;-1 2 2 -1 -1;0 0 1 1 0],[0 1
0],0.0085,[normz 0 -normx]/norm([normz 0 -normx]),0.0764);
% planes(6) = makeplane(6,'rect',[1 0 0],10,1,index2,[10 10 10 10 10;-1 -1 2 2 -1;0 1 1 0 0],[0 1 0],0,[0 0 1],0);
% planes(7) = makeplane(7,'tri',[0 1 0],-1,index1,index2,[0 0 x 0;-1 -1 -1 -1;0 1 1 0],[1 0 0],0.0085,[0 0 1],0.0085);
% planes(8) = makeplane(8,'tri',[0 1 0],2,index1,index2,[0 0 x 0;2 2 2 2;0 1 1 0],[1 0 0],0.0085,[0 0 1],0.0085);
% dist = 2;


%--------------------------------------------------------
% sensor face
%--------------------------------------------------------
%planes(9) = makeplane(9,'rect',[0 0 1],dist,index1,index3,[-5 -5 1 1 -5;-2 3 3 -2 -2;dist dist dist dist dist],[1 0 0],0,[0 1 0],0);

%--------------------------------------------------------
% input guide
%--------------------------------------------------------
planes(10) = makeplane(10,'rect',[0 1 0],0,index1,index2,[2*x 2*x 2*x-10 2*x-10 2*x;0 0 0 0 0;0 1 1 0 0],[1 0 0],0.0085,[0 0
1],0.0085);
planes(11) = makeplane(11,'rect',[0 0 1],0,index1,index2,[2*x 2*x 2*x-10 2*x-10 2*x;0 1 1 0 0;0 0 0 0 0],[1 0 0],0,[0 1 0],0);
planes(12) = makeplane(12,'rect',[0 1 0],1,index1,index2,[2*x 2*x 2*x-10 2*x-10 2*x;1 1 1 1 1;0 1 1 0 0],[1 0 0],0.0085,[0 0
1],0.0085);
% closed faced
%planes(13) = makeplane(13,'rect',[0 0 1],1,index1,index2,[2*x-10 2*x-10 x x 2*x-10;0 1 1 0 0;1 1 1 1 1],[1 0 0],0,[0 1 0],0);
% open faced

planes(13) = makeplane(13,'rect',[0 0 1],1,index1,index2,[2*x-10 2*x-10 2*x 2*x 2*x-10;0 1 1 0 0;1 1 1 1 1],[1 0 0],0,[0 1 0],0);

planes(14) = makeplane(14,'rect',[-normx 0 normz]/norm([-normx 0 normz]),(-normx*x+normz)/norm([-normx 0
normz]),index3,index2,[2*x 2*x x x 2*x;0 1 1 0 0;0 0 1 1 0],[0 1 0],0.0085,[normx 0 normz]/norm([normx 0 normz]),0.0764);
planes(15) = makeplane(15,'rect',[1 0 0],2*x-10,1,index2,[2*x-10 2*x-10 2*x-10 2*x-10 2*x-10;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0
1],0);
planes(16) = makeplane(16,'tri',[0 1 0],0,index1,index2,[2*x x 2*x 2*x;0 0 0 0;0 1 1 0],[1 0 0],0.0085,[0 0 1],0.0085);
planes(17) = makeplane(17,'tri',[0 1 0],1,index1,index2,[2*x x 2*x 2*x;1 1 1 1;0 1 1 0],[1 0 0],0.0085,[0 0 1],0.0085);
```

```
%--------------------------------------------------------
% hit plane
%--------------------------------------------------------
%planes(18) = makeplane(18,'rect',[1 0 0],30,1,1,[30 30 30 30 30;-30 -30 31 31 -30; -30 31 31 -30 -30],[0 1 0],0,[0 0 1],0);

%--------------------------------------------------------
% vertical guiding
%--------------------------------------------------------
offh = 1
planes(18) = makeplane(18,'rect',[1 0 0],0,index1,index2,[0 0 0 0 0;0 1 1 0 0;1+offh 1+offh 1 1 1+offh],[0 1 0],0,[0 0 1],0);
planes(19) = makeplane(19,'rect',[1 0 0],2*x,index1,index2,[2*x 2*x 2*x 2*x 2*x;0 1 1 0 0;1+offh 1+offh 1 1 1+offh],[0 1 0],0,[0 0
1],0);
planes(20) = makeplane(20,'rect',[0 1 0],0,index1,index2,[2*x 2*x 0 0 2*x;0 0 0 0 0;1+offh 1 1 1+offh 1+offh],[1 0 0],0,[0 0 1],0);
planes(21) = makeplane(21,'rect',[0 1 0],1,index1,index2,[2*x 2*x 0 0 2*x;1 1 1 1 1;1+offh 1 1 1+offh 1+offh],[1 0 0],0,[0 0 1],0);
planes(9) = makeplane(9,'rect',[0 0 1],1+offh,1,index2,[2*x 2*x 0 0 2*x;1 0 0 1 1;1+offh 1+offh 1+offh 1+offh 1+offh],[1 0 0],0,[0 1
0],0);
%--------------------------------------------------------
```

Gradual in-plane bend – gradual_bend.m

```
function planes = gradual_bend(ir)

index1 = 1.43;
index2 = 1.56;
indexair = 1;
indexm = 0;
len = 15;
planes(1) = makeplane(1,'rect',[0 1 0],0,index1,index2,[0 0 len len 0;0 0 0 0 0;0 1 1 0 0],[1 0 0],0,[0 0 1],0);
planes(2) = makeplane(2,'rect',[0 0 1],0,index1,index2,[-(ir+1) -(ir+1) len len -(ir+1);0 (ir+1) (ir+1) 0 0;0 0 0 0 0],[1 0 0],0,[0 1 0],0);
planes(3) = makeplane(3,'rect',[0 1 0],1,index1,index2,[0 0 len len 0;1 1 1 1 1;0 1 1 0 0],[1 0 0],0,[0 0 1],0);
planes(4) = makeplane(4,'rect',[0 0 1],1,index1,index2,[-(ir+1) -(ir+1) len len -(ir+1);0 (ir+1) (ir+1) 0 0;1 1 1 1 1],[1 0 0],0,[0 1 0],0);
%planes(5) = makeplane(5,'rect',[1 0 0],0,indexair,index2,[0 0 0 0 0;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0 1],0);
planes(6) = makeplane(6,'rect',[1 0 0],len,indexair,index2,[len len len len len;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0 1],0);

planes(5) = makeplane(5,'circle',[1 0 0],0,indexm,index2,[0 0 -ir -ir 0;1 1 (ir+1) (ir+1) 1;1 0 0 1 1],[1 0 0],0,[0 1 0],0,circleplane(ir,[0
(ir+1) .5],[-1 0 0],[0 0 1]));
planes(7) = makeplane(7,'circle',[1 0 0],0,indexm,index2,[0 0 -(ir+1) -(ir+1) 0;0 0 (ir+1) (ir+1) 0;1 0 0 1 1],[1 0 0],0,[0 1
0],0,circleplane((ir+1),[0 (ir+1) .5],[-1 0 0],[0 0 1]));
planes(8) = makeplane(8,'rect',[1 0 0],-(ir+1),index1,index2,[-(ir+1) -(ir+1) -(ir+1) -(ir+1) -(ir+1);(ir+1) (ir+1) (ir+1)+len (ir+1)+len
(ir+1);0 1 1 0 0],[0 1 0],0,[0 0 1],0);
planes(9) = makeplane(9,'rect',[1 0 0],-ir,index1,index2,[-ir -ir -ir -ir -ir;(ir+1) (ir+1) (ir+1)+len (ir+1)+len (ir+1);0 1 1 0 0],[0 1 0],0,[0
0 1],0);
planes(10) = makeplane(10,'rect',[0 1 0],(ir+1)+len,indexair,index2,[-(ir+1) -(ir+1) -ir -ir -(ir+1);(ir+1)+len (ir+1)+len (ir+1)+len
(ir+1)+len (ir+1)+len;0 1 1 0 0],[1 0 0],0,[0 0 1],0);
planes(11) = makeplane(11,'rect',[0 0 1],0,index1,index2,[-(ir+1) -(ir+1) -ir -ir -(ir+1);(ir+1) (ir+1)+len (ir+1)+len (ir+1) (ir+1);0 0 0 0
0],[1 0 0],0,[0 1 0],0);
planes(12) = makeplane(12,'rect',[0 0 1],1,index1,index2,[-(ir+1) -(ir+1) -ir -ir -(ir+1);(ir+1) (ir+1)+len (ir+1)+len (ir+1) (ir+1);1 1 1 1
1],[1 0 0],0,[0 1 0],0);

dist = 60;
planes(13) = makeplane(13,'rect',[0 1 0],(ir+1)+len+dist,indexair,indexair,[-(ir+1)-dist -(ir+1)-dist -ir+dist -ir+dist -(ir+1)-
dist;(ir+1)+len+dist (ir+1)+len+dist (ir+1)+len+dist (ir+1)+len+dist (ir+1)+len+dist;0-dist 1+dist 1+dist 0-dist 0-dist],[1 0 0],0,[0 0
1],0);
```

Bend integrated power combiner – bend_power_combiner.m

```
function planes = gradual_bend(ir)

index1 = 1.43;
index2 = 1.56;
indexair = 1;
indexm = 0;
len = 10;
planes(19) = makeplane(19,'rect',[0 1 0], -10,indexair,indexair,[-10 -10 10 10 -10;-10 -10 -10 -10 -10; -10 10 10 -10 -10],[1 0 0],0,[0 0
1],0);
```

```
planes(1) = makeplane(1,'rect',[1 0 0],-.5,index1,index2,[-.5 -.5 -.5 -.5 -.5; 0 0 6.2 6.2 0; -.5 .5 .5 -.5 -.5],[0 1 0],0,[0 0 1],0);
planes(2) = makeplane(2,'rect',[1 0 0],.5,index1,index2,[.5 .5 .5 .5 .5; 0 0 6.2 6.2 0; -.5 .5 .5 -.5 -.5],[0 1 0],0,[0 0 1],0);
planes(3) = makeplane(3,'rect',[0 1 0],0,indexair,index2,[.5 .5 -.5 -.5 .5; 0 0 0 0 0; -.5 .5 .5 -.5 -.5],[0 1 0],0,[0 0 1],0);
perp = cross([0 0 1],[1 -20 0]);
perp = perp/norm(perp);
planes(4) = makeplane(4,'rect',perp,dot(perp,[-.5 6.2 0]),index1,index2,[-.5 -1.5 -1.5 -.5 -.5;6.2 26.2 26.2 6.2 6.2;-.5 -.5 .5 .5 -.5],[0 0
1],0,[1 0 0],0);
perp = cross([0 0 1],[1 20 0]);
perp = perp/norm(perp);
planes(5) = makeplane(5,'rect',perp,dot(perp,[.5 6.2 0]),index1,index2,[.5 1.5 1.5 .5 .5;6.2 26.2 26.2 6.2 6.2;-.5 -.5 .5 .5 -.5],[0 0 1],0,[1
0 0],0);
planes(6) = makeplane(6,'circle',[0 1 0],26.2,indexm,index2,[-1.5 -3.2 -3.2 -1.5 -1.5;26.2 27.9 27.9 26.2 26.2;-.5 -.5 .5 .5 -.5],[1 0
0],0,[0 0 1],0,circleplane(1.7,[-3.2 26.2 0],[0 1 0],[0 0 1]));
planes(7) = makeplane(7,'circle',[0 1 0],26.2,indexm,index2,[1.5 3.2 3.2 1.5 1.5;26.2 27.9 27.9 26.2 26.2;-.5 -.5 .5 .5 -.5],[1 0 0],0,[0 0
1],0,circleplane(1.7,[3.2 26.2 0],[0 1 0],[0 0 1]));
planes(8) = makeplane(8,'rect',[0 1 0],27.9,index1,index2,[-3.2 -12.2 -12.2 -3.2 -3.2; 27.9 27.9 27.9 27.9 27.9; -.5 -.5 .5 .5 -.5],[1 0
0],0,[0 0 1],0);
planes(9) = makeplane(9,'rect',[0 1 0],27.9,index1,index2,[3.2 12.2 12.2 3.2 3.2; 27.9 27.9 27.9 27.9 27.9; -.5 -.5 .5 .5 -.5],[1 0 0],0,[0
0 1],0);
planes(10) = makeplane(10,'rect',[0 1 0],28.9,index1,index2,[-3.2 -12.2 -12.2 -3.2 -3.2; 28.9 28.9 28.9 28.9 28.9; -.5 -.5 .5 .5 -.5],[1 0
0],0,[0 0 1],0);
planes(11) = makeplane(11,'rect',[0 1 0],28.9,index1,index2,[3.2 12.2 12.2 3.2 3.2; 28.9 28.9 28.9 28.9 28.9; -.5 -.5 .5 .5 -.5],[1 0
0],0,[0 0 1],0);
planes(12) = makeplane(12,'circle',[0 1 0],26.2,indexm,index2,[-.5 -3.2 -3.2 -.5 -.5;26.2 28.9 28.9 26.2 26.2;-.5 -.5 .5 .5 -.5],[1 0
0],0,[0 0 1],0,circleplane(2.7,[-3.2 26.2 0],[0 1 0],[0 0 1]));
planes(13) = makeplane(13,'circle',[0 1 0],26.2,indexm,index2,[.5 3.2 3.2 .5 .5;26.2 28.9 28.9 26.2 26.2;-.5 -.5 .5 .5 -.5],[1 0 0],0,[0 0
1],0,circleplane(2.7,[3.2 26.2 0],[0 1 0],[0 0 1]));
planes(14) = makeplane(14,'circle',[0 1 0],26.2,indexm,index2,[-.5 .5 .5 -.5 -.5;26.2 25.7 25.7 26.2 26.2;-.5 -.5 .5 .5 -.5],[1 0 0],0,[0 0
1],0,circleplane(.5,[0 26.2 0],[0 -1 0],[0 0 1]));
planes(15) = makeplane(15,'rect',[1 0 0],-12.2,indexair,index2,[-12.2 -12.2 -12.2 -12.2 -12.2; 27.9 28.9 28.9 27.9 27.9; -.5 -.5 .5 .5 -
.5],[0 1 0],0,[0 0 1],0);
planes(16) = makeplane(16,'rect',[1 0 0],12.2,indexair,index2,[12.2 12.2 12.2 12.2 12.2; 27.9 28.9 28.9 27.9 27.9; -.5 -.5 .5 .5 -.5],[0 1
0],0,[0 0 1],0);

planes(17) = makeplane(17,'rect',[0 0 1],.5,index1,index2,[-12.2 -12.2 12.2 12.2 -12.2; 0 28.9 28.9 0 0;.5 .5 .5 .5 .5],[1 0 0],0,[0 1
0],0);
planes(18) = makeplane(18,'rect',[0 0 1],-.5,index1,index2,[-12.2 -12.2 12.2 12.2 -12.2; 0 28.9 28.9 0 0;-.5 -.5 -.5 -.5 -.5],[1 0 0],0,[0 1
0],0);

%planes(1) = makeplane(1,'circle',[1 0 0],0,indexair,index2,[0 -.5 -.5 0 0;0 0 1 1 0;0 1 1 0 0],[0 1 0],0,[0 0 1],0,circleplane(.5,[0 .5
.5],[-1 0 0],[0 0 1]));
%back = 240;

%planes(7) = makeplane(7,'rect',[1 0 0],-back/2,indexair,indexair,[-back/2 -back/2 -back/2 -back/2 -back/2;-back -back back+1
back+1 -back;-back back+1 back+1 -back -back],[0 1 0],0,[0 0 1],0);
```

C.1.8 OSLO Verification Structures

Planoconvex Lens – lens_planoconvex.m

```
function planes = lens_planoconvex()

index1 = 1.5725;
indexair = 1;
planes(1) = makeplane(1,'circle',[1 0 0],0,indexair,index1,[-1 -2 -2 -1 -1;-1 -1 1 1 -1;-1 1 1 -1 -1],[0 1 0],0,[0 0 1],0,circleplane(2,[0 0
0],[-1 0 0],[0 0 1]));
planes(2) = makeplane(2,'rect',[1 0 0],-1,indexair,index1,[-1 -1 -1 -1 -1;-1 -1 1 1 -1;-1 1 1 -1 -1],[0 1 0],0,[0 0 1],0);
planes(3) = makeplane(3,'rect',[1 0 0],2,indexair,indexair,[2 2 2 2 2;-1 -1 1 1 -1;-1 1 1 -1 -1],[0 1 0],0,[0 0 1],0);
```

Lens set, right angle prism, and mirror – prismirroslo.m

```
function planes = prismirroslo()
index1 = 1.5725;
lasf35 = 2.022041;
sf15 = 1.698952;
indexair = 1;
```

```
planes(1) = makeplane(1,'circle',[1 0 0],0,indexair,lasf35,[-95 -100 -100 -95 -95;-16 -16 16 16 -16;-1 1 1 -1 -1],[0 1 0],0,[0 0
1],0,circleplane(100,[0 0 0],[-1 0 0],[0 0 1]));
planes(2) = makeplane(2,'rect',[1 0 0],-95,indexair,lasf35,[-95 -95 -95 -95 -95;-15 -15 15 15 -15;-1 1 1 -1 -1],[0 1 0],0,[0 0 1],0);
planes(3) = makeplane(3,'circle',[1 0 0],0,indexair,lasf35,[-89 -94 -94 -89 -89;-15 -15 15 15 -15;-1 1 1 -1 -1],[0 1 0],0,[0 0
1],0,circleplane(50,[-44 0 0],[-1 0 0],[0 0 1]));
planes(4) = makeplane(4,'circle',[1 0 0],0,indexair,lasf35,[-84 -89 -89 -84 -84;-15 -15 15 15 -15;-1 1 1 -1 -1],[0 1 0],0,[0 0
1],0,circleplane(69.212293,[-89+69.212293 0 0],[-1 0 0],[0 0 1]));
planes(5) = makeplane(5,'rect',[1 0 0],-84,indexair,sf15,[-84 -84 -84 -84 -84;-15 -15 15 15 -15;-1 1 1 -1 -1],[0 1 0],0,[0 0 1],0);
normvect = [1 1 0]/norm([1 1 0]);
planes(6) = makeplane(6,'rect',normvect,dot(normvect,[-69 0 0]),indexair,sf15,[-54 -84 -84 -54 -54;-15 15 15 -15 -15;-1 -1 1 1 -1],[0 1
0],0,[0 0 1],0);
planes(7) = makeplane(7,'rect',[0 1 0],-15,indexair,sf15,[-54 -84 -84 -54 -54;-15 -15 -15 -15 -15;-1 -1 1 1 -1],[0 1 0],0,[0 0 1],0);
planes(8) = makeplane(8,'rect',normvect,dot(normvect,[-69 -25 0]),indexair,0,[-60.5 -77.5 -77.5 -60.5 -60.5;-33.5 -16.5 -16.5 -33.5 -
33.5;-1 -1 1 1 -1],[0 1 0],0,[0 0 1],0);
planes(9) = makeplane(9,'rect',[1 0 0],-48,indexair,indexair,[-48 -48 -48 -48 -48;-30 -30 -20 -20 -30;-1 1 1 -1 -1],[0 1 0],0,[0 0 1],0);
```

C.1.9 Power Efficiency Solvers

Fiber collection efficiency – collectionfiber_eff.m

```
function collection_eff(burstsize,filename,planes)
time = clock;
% make center points
z = [-2:.1:2];
y = [-2:.1:2];
% set height
x = ones(1,length(y))*-1;
for n=1:length(z)
   cpoints(:,(n-1)*length(y)+1:n*length(y)) = [x;y;z(n)*ones(1,length(y))];
end
% trace out each rayburst and find the output intensity given absorption
% intensity
total = length(z)*length(y);
% initialize total intersected rays count
rayscount = 0;
for n=1:total
   rays = ray_burstx_reverse(burstsize,cpoints(:,n),pi/2,1);
   raysout = trace_rays(planes,rays,2e-7,100,false);
   [zstore planerays] = surfaceplot(raysout, planes(2),2,-1,1,false,1,1);
   if (zstore > 0)
      fprintf('[Collection_eff] %d rays intersected for %d power.\n',length(planerays),zstore)
   else
      fprintf('[Collection_eff] 0 rays intersected.\n');
   end
   fid = fopen(filename,'a');
   intdist = [cpoints(2:3,n)' zstore];
   fprintf(fid,'%12.1f %12.1f %24.12f\n',intdist);
   fclose(fid);
   if ((n - floor(n/100)*100) == 0)
   fprintf('[Collection_eff] %d of %d completed.\n\n',n,total)
   end

end
fprintf('[Collection_eff] Total Time Elapsed: %d seconds\n\n',etime(clock,time));
```

Waveguide backplane collection efficiency – collection_eff.m

```
function collection_eff(burstsize,filename,planes)
time = clock;
% load planes
%planes = bend_guide_total;
%planes = bend_guide45;
% for this waveguide the input face is at x = [-1 -2] y = [0 1]
% make center points
x = [-5:.1:1];
y = [-2:.1:3];
```

```
% SPECIFY FILENAME
%filename = 'bend_guide45_2000burst_mirrorback.txt';
% set height
z = ones(1,length(y))*2.001;
for n=1:length(x)
   cpoints(:,(n-1)*length(y)+1:n*length(y)) = [x(n)*ones(1,length(y));y;z];
end
% trace out each rayburst and find the output intensity given absorption
% intensity
total = length(x)*length(y);
% initialize total intersected rays count
rayscount = 0;
for n=1:total
   rays = ray_burst(burstsize,cpoints(:,n),pi/2,1);
   raysout = trace_rays(planes,rays,2e-7,100,false);
   [zstore planerays] = surfaceplot(raysout, planes(6),6,-1,2,false,1,1);
   if (zstore > 0)
      fprintf('[Collection_eff] %d rays intersected for %d power.\n',length(planerays),zstore)
   else
      fprintf('[Collection_eff] 0 rays intersected.\n');
   end
   fid = fopen(filename,'a');
   intdist = [cpoints(1:2,n)' zstore];
   fprintf(fid,'%12.1f %12.1f %24.12f\n',intdist);
   fclose(fid);
   if ((n - floor(n/100)*100) == 0)
   fprintf('[Collection_eff] %d of %d completed.\n\n',n,total)
   end

end
fprintf('[Collection_eff] Total Time Elapsed: %d seconds\n\n',etime(clock,time));
```

C.2 Waveguide Measurements

Straight waveguide scattering loss exponential fit – lossplot.m

```
function [P P2] = lossplot(filename,cindex,pixellength)
conversion = pixellength;
pic = imread(filename);
points = double(pic(:,:,cindex));
points2 = min(points);
points = mean(points);
[ysize xsize] = size(points);
xaxis = linspace(1,xsize,xsize);
xaxis = xaxis/conversion;
logpoints = log(points);
logpoints2 = log(points2);
[P S] = polyfit(xaxis,logpoints,1);
[P2 S] = polyfit(xaxis,logpoints2,1);
fit = polyval(P,xaxis);
fit2 = polyval(P2,xaxis);
figure;plot(xaxis,logpoints,'k',xaxis,fit,'k',xaxis,logpoints2,'b',xaxis,fit2,'b');
figure;plot(xaxis,points,'k',xaxis,exp(fit),'k',xaxis,points2,'b',xaxis,exp(fit2),'b')
figure;image(pic)
```

Sidewall roughness profilometer data handling – dataplot.m

```
function [Ra slopes yfit Yfreq]= dataplot(filename,len,points,fit,res)
feed = 150000/60;
x = load(filename);
y = reformatdata(x);
p = polyfit(1:length(y),y,fit);
yfit = y-polyval(p,1:length(y));
yfitnorm = yfit - mean(yfit);
```

```
Y = fft(yfitnorm,65536);
figure
xaxis = linspace(0,len,points);
plot(xaxis,yfitnorm/10)
figure
plot(xaxis,y/10,xaxis,polyval(p,1:length(y))/10);
xdiff = xaxis(2)-xaxis(1);
feed = feed/.04/60*1000; %um/s
freq = 1/xdiff;
faxis = linspace(0,freq,65536);
figure
Yfreq = Y.*conj(Y)/max(Y.*conj(Y));
plot(faxis,Yfreq)
slopes = slopedist(yfit,len,points,res);
Ra = sum(abs(yfit))/points;
```

Profilometer to matlab data conversion – reformatdata.m

```
function y = reformatdata(x)
[row col] = size(x);
for n=1:row
   y((n-1)*col+1:(n-1)*col+col) = x(n,:);
end
```

Sidewall roughness slope histogram – slopedist.m

```
function slopes = slopedist(data,length,points,res)
% slopedist(data,length,points,resolution)
% Plots a distribution of the roughness as slope

ydiff = diff(data)/10000;
x = linspace(0,length,points);
xdiff = diff(x);
slopes = ydiff./xdiff;
[y x] = hist(slopes,res);
y = y/max(y);
figure;plot(x,y)
```

Gaussian Fitting – plotgaussian.m

```
function plotgaussian(center,HWHM,Height,min,max,points,newfig)
if (newfig == true)
   figure
else
   hold on;
end
x = linspace(min,max,points);
y = Height.*exp(-(x-center).^2/(HWHM/(log(2)^.5))^2);
plot(x,y,'k')
```

C.3 Quantum Dot Waveguide Efficiency

```
function EC = qdotwg(xstart, xend, resolution, concstart, concend, concres) %conc)
% qdotwg(length start, length end, length spacing, concentration start, concentration end
%        concentration spacing) cms, percentage from base
% Return: maximum value, concentration occurence

plotstyles = plotstyle;
%efficiency = .13087; theoretical
%quantum dot used = 600 nm
```

```
W = .1;
H = .1;
%ceff = .0417; %theoretical
ceff = .012; % measured
%ad = 134;%1470;           % dot absorption at base concentration
                          % alpha = molar extinction * concentration * ln(10)
ax = .07;                   % waveguide absorption of dot emission wavelength

af = 100;                 % dot absorption of dot emission
auv = 1000;                 % dot absorption at uv 380
qyield = .5;               % dot quantum yield
flickrate = 1;
% unused variables
Acsuvdot = 7e-18*100^2; %absorption cross section at 350 nm in cm^2
startingmolarratio = 75.7e-9; %nanoMoles/ml *1 Mole/10^9 nanoMoles = Moles/ml = Moles/cm^-3
aduv = Acsuvdot*startingmolarratio; %base concentration absorption from
Awg = W*H;
az = .347;

% data crunching
conc = linspace(concstart,concend,concres);
L = linspace(xstart, xend, resolution);
for n=1:resolution
   for m=1:concres
      EC(n,m) = qyield*ceff*flickrate*(1-exp(-conc(m)*H*auv))*(exp((-ax+(qyield*ceff*flickrate*af-af)*conc(m))*L(n))-1)/((-
ax+(qyield*ceff*flickrate*af-af)*conc(m)))/L(n);
   end
end
figure;
HtoF = GCF;
if(ishold == 0)
  hold
end
%for n=1:concres
%   plot(L,EC(:,n),plotstyles(n,:));
%end
for n=1:resolution
   plot(conc,EC(n,:),plotstyles(n,:));
end
[maximum(:,1) maximum(:,2)] = max(EC,[],2); % intensity, concentration value
maximum(:,2) = (maximum(:,2)*(concend-concstart)/concres)+concstart; %change index to concentration
maximum(:,3) = L';
%plot(maximum(:,3),maximum(:,1))
```

Appendix D

G-Code

D.1 Straight Waveguide Positive Mold

```
G21
G00 Z 1
X 6
Y -2
G01 Z -1 F 30
Y 78.2
X 6.1
Y -2
X 5.9
Y 78.2
X 6.1
G00 Z 1
X 12
G01 Z -1 F 30
Y -2
X 11.9
Y 78.2
X 12.1
Y -2
X 11.9
G00 Z 1
X 18
G01 Z -1 F 30
Y 78.2
X 18.1
Y -2
X 17.9
Y 78.2
X 18.1
G00 Z 10
```

D.2 Straight Waveguide Negative Mold

```
G21
G00 X 4.8
G00 Y 0
G00 Z 1
G01 Z -1 F 30
Y 76.2
X 4.9
Y 0
X 0 F 200
Y 76.2
X 4.8
X 4.5
Y .5
X .5
Y 75.7
X 4
Y 1
X 1
Y 75.2
X 3.5
Y 1.5
X 1.5
Y 74.7
X 3
Y 2
X 2
Y 74.2
X 2.5
Y 2.5
G00 Z 1
Y 76.2
X 7.2
G01 Z -1 F 30
Y 0
X 7.1
Y 76.2
G01 X 10.8 F 200
G01 Y 0 F 30
G01 X 7.2 F 200
X 7.5
Y 76.2
X 10.9
Y 0 F 30
X 8 F 200
Y 75.7
X 10.5
Y 0.5
X 8.5
Y 75.2
X 10
Y 1
X 9
Y 74.7
X 9.5
```

```
Y 1.5
G00 Z 1
Y 76.2
X 13.2
G01 Z -1 F 30
Y 0
X 13.1
Y 76.2
X 16.8 F 200
Y 0 F 30
X 13.2 F 200
X 13.5
Y 76.2
X 16.9
Y 0 F 30
X 14 F 200
Y 75.7
X 16.5
Y .5
X 14.5
Y 75.2
X 16
Y 1
X 15
Y 75.7
X 15.5
Y 1.5
G00 Z 1
Y 76.2
X 19.2
G01 Z -1 F 30
Y 0
X 19.1
Y 76.2
X 25.4 F 200
Y 0
X 19.2
X 19.5
Y 75.7
X 25
Y .5
X 20
Y 75.2
X 24.5
Y 1
X 20.5
Y 74.7
X 24
Y 1.5
X 21
Y 74.2
X 23.5
Y 2
X 21.5
Y 73.7
X 23
Y 2.5
```

```
X 22
Y 73.2
X 22.5
Y 3
G00 Z 10
```

D.3 Straight Waveguide Varying Surface Roughness

```
G21
G00 Z 5
X 0
Y 0
Z 1
G01 Z -1 F 30
Y 76.2 F 200
X 4.2
Y 0
X 0
X .5
Y 76.2
X 1
Y 0
X 1.5
Y 76.2
X 2
Y 0
X 2.5
Y 76.2
X 3
Y 0
X 3.5
Y 76.2
X 4
Y 0
G00 Z 1
X 6.2
M00
G01 Z -1 F 30
Y 76.2 F 200
X 9.2
Y 0 F 200
X 6.2 F 200
X 6.7
Y 76.2
X 7.2
Y 0
X 7.7
Y 76.2
X 8.2
Y 0
X 8.7
Y 76.2
X 9
Y 0
```

```
G00 Z 1
X 11.2
M00
G01 Z -1 F 30
Y 76.2 F 200
X 14.2 F 200
Y 0 F 30
X 11.2 F 100
X 11.7
Y 76.2
X 12.2
Y 0
X 12.7
Y 76.2
X 13.2
Y 0
X 13.7
Y 76.2
X 14
Y 0
G00 Z 1
X 16.2
M00
G01 Z -1 F 30
Y 76.2 F 30
X 19.2 F 100
Y 0 F 30
X 16.2 F 100
X 16.7
Y 76.2
X 17.2
Y 0
X 17.7
Y 76.2
X 18.2
Y 0
X 18.7
Y 76.2
X 19
Y 0
G00 Z 1
X 21.2
M00
G01 Z -1 F 30
Y 76.2
X 25.4 F 100
Y 0
X 21.2
X 21.7
Y 76.2
X 22.2
Y 0
X 22.7
Y 76.2
X 23.2
Y 0
X 23.7
```

```
Y 76.2
X 24.2
Y 0
X 24.7
Y 76.2
X 25.2
Y 0
G00 Z 10
```

D.4 Optical Backplane Mold

```
G21
G90
G00 Z 10
X 10.5
Y 12.6
M98 Pwgsub
X 10.5
Y 29.6
M98 Pwgsub
X 10.5
Y 46.6
M98 Pwgsub
X 10.5
Y 63.6
M98 Pwgsub
X 10.5
Y 80.6
M98 PtopC
X 10.5
Y 63.6
M98 Pclearance
X 10.5
Y 46.6
M98 Pclearance
X 10.5
Y 29.6
M98 Pclearance
X 10.5
Y 12.6
M98 PbottomC

OtopC
G00 Z .5
G91
Y -2.7
X -12.2
G01 Z -1.3 F 50
Y -12.1 F 100
Y .5
X 28.8
Y -.5
Y 12.1
```

```
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
Y -.9
X 28.8
Z .5
M00
Z -.5
Y -.9
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
Z .5
M00
Z -.5
Y -.9
X 28.8
Y -.9
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
G90
G00 Z .5
M00
M99


ObottomC
G00 Z .5
G91
Y -2.2
X -12.2
G01 Z -1.3 F 50
Y -12.1 F 100
X 28.8
Y 12.1
Y -.5
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
Y -.9
X 28.8
Z .5
M00
Z -.5
```

```
Y -.9
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
Z .5
M00
Z -.5
Y -.9
X 28.8
Y -.9
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
G90
G00 Z .5
M00
M99

Oclearance
G00 Z .5
G91
Y -2.2
X -12.2
G01 Z -1.3 F 50
Y -12.6 F 100
Y .5
X 28.8
Y -.5
Y 12.6
Y -.5
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
Y -.9
X 28.8
Z .5
M00
Z -.5
Y -.9
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
Y -.9
X 28.8
Y -.9
```

```
X -28.8
Z .5
M00
Z -.5
Y -.9
X 28.8
Y -.9
X -28.8
Y -.9
X 28.8
Y -.9
X -28.8
G90
G00 Z .5
M00
M99



Owgsub
G00 Z .5
G91
Y 1.2
X -12.2
G01 Z -1.3 F 30
X 11.5
Y 1
X 17.3
Y -.2
X -17.1
Y -1
X -11.7
Y 1
X 11.5
G00 Z 1.8
Y -4.2
X 17.3
G01 Z -1.8 F 30
X -17.3
Y 1
X -11.5
Y -.8
X 11.5
X -11.5
Y 1
X 11.7
Y -1
X 17.1
G90
G00 Z .5
M00
M99
```

D.5 Optical Backplane Mold (Bend Cuts)

```
G21
G00 Z 10
X 10.5
Y 65.7
Z 1
G01 Z -.65 F 25
Y 61.5
G01 Z -.85
Y 65.7

G00 Z 1
Y 48.7
G01 Z -.65 F 25
Y 44.5
G01 Z -.85
Y 48.7

G00 Z 1
Y 31.7
G01 Z -.65 F 25
Y 27.5
Z -.85
Y 31.7

G00 Z 1
Y 14.7
G01 Z -.65 F 25
Y 10.5
G01 Z -.85
Y 14.7
G00 Z 10
```

D.6 In-Plane Bends and Power Combiner Mold

```
G21

(bottom left outer arc)
G00 Z 10
X 10
Y 0
Z 1
G01 Z -.9 F 30
G01 Z 1
M00
G01 Z -1 F 30
Y 20
G03 X 5 Y 25 I -5 J 0
G01 X 0

(bottom left cut away)
Y 25.8 F 150
```

```
X 5
G02 X 10.8 Y 20 I 0 J -5.8
G01 Y 0
X 11.6
Y 20
G03 X 5 Y 26.6 I -6.6 J 0
G01 X 0
X 11.6
Y 20
X 10.8
Y 25.8
X 10
Y 26.6
X 25.4
Y 25.8
X 17.6
G03 X 13.6 Y 21.8 I 0 J -4
G01 Y 25.8
X 17.2

X 12.4
Y 0
X 13.2
Y 25.8
X 14
Y 0

(bottom left inner arc)
G00 Z 1
X 0
Y 23
G01 Z -.9 F 30
G01 Z 1
M00
G01 Z -1
X 5
G02 X 8 Y 20 I 0 J -3
G01 Y 0
X 0

(mill out)
X 7.2 F 150
Y 20
G03 X 5 Y 22.2 I -2.2 J 0
G01 X 0
Y 21.4
X 5
G02 X 6.4 Y 20 I 0 J -1.4
G01 Y 0
X 5.6
Y 20
G03 X 5 Y 20.6 I -.6 J 0
G01 X 0
Y 20
X 5
Y 0
X 0
```

```
Y 19.2
X 4.2
Y .8
X .8
Y 18.4
X 3.4
Y 1.6
X 1.6
Y 17.6
X 2.6
Y 2.4
X 2.4
Y 16.8

(outside bottom right)
G00 Z 1
X 25.4
Y 25
G01 Z -.9 F 30
G01 Z 1
M00
G01 Z -1
X 17.6
G03 X 14.4 Y 21.8 I 0 J -3.2
G01 Y 0

(inside bottom right)

G00 Z 1
X 16.4
G01 Z -.9 F 30
G01 Z 1
M00
G01 Z -1
Y 21.8
G02 X 17.6 Y 23 I 1.2 J 0
G01 X 25.4

(mill out)
Y 22.2 F 150
X 17.6
G03 X 17.2 Y 21.8 I 0 J -.4
G01 Y 0
X 25.4
Y 21.4
X 18
Y 0
X 24.6
Y 20.6
X 18.8
Y .8
X 23.8
Y 19.8
X 19.6
Y 1.6
X 23
Y 19
```

```
X 20.4
Y 2.4
X 22.2
Y 18.2
X 21.3
Y 3.2

(combiner taper left side)
G00 Z 1
Y 76.2
X 11.7
G01 Z -.9 F 30
Z 1
M00
G01 Z -1
Y 70
Y 50 X 10.7
G02 X 9.5 Y 48.8 I -1.2 J 0
G01 X 0

(mill out)
Y 49.6 F 150
X 9.5
G03 X 9.9 Y 50 I 0 J .4
G01 Y 76.2
X 10.8
Y 70
Y 50.4 X 9.8
X 0
Y 76.2
X 8.7
Y 51.2
X .8
Y 75.4
X 7.9
Y 52
X 1.6
Y 74.6
X 7.1
Y 52.8
X 2.4
Y 73.8
X 6.3
Y 53.6
X 3.2
Y 73
X 5.5
Y 54.4
X 4
Y 72.2
X 4.7
Y 55.2

(back side)
G00 Z 1
X 0
Y 46.8
```

```
G01 Z -.9 F 30
G01 Z 1
M00
G01 Z -1
X 9.5
G03 X 12.7 Y 50 I 0 J 3.2
G03 X 15.9 Y 46.8 I 3.2 J 0
G01 X 25.4

(mill out)
Y 27.4 F 150
X 0
Y 46
X 9.5
G03 X 12.7 Y 49.2 I 0 J 3.2
G03 X 15.9 Y 46 I 3.2 J 0
G01 X 24.6
Y 28.2
X .8
Y 45.2
X 9.5
G03 X 12.7 Y 48.4 I 0 J 3.2
G03 X 15.9 Y 45.2 I 3.2 J 0
G01 X 23.8
Y 29
X 1.6
Y 44.4
X 9.5
G03 X 12.7 Y 47.6 I 0 J 3.2
G03 X 15.9 Y 44.4 I 3.2 J 0
G01 X 23
Y 29.8
X 2.4
Y 43.6
X 9.5
G03 X 12.7 Y 46.8 I 0 J 3.2
G03 X 15.9 Y 43.6 I 3.2 J 0
G01 X 22.2
Y 30.6
X 3.2
Y 42.8
X 9.5
G03 X 12.7 Y 46 I 0 J 3.2
G03 X 15.9 Y 42.8 I 3.2 J 0
G01 X 21.4
Y 31.4
X 4
Y 42
X 15.9
Y 45.2
X 9.5
Y 44.4
X 15.9
Y 43.6
X 9.5
Y 42.8
X 15.9
```

```
Y 42
X 20.6
Y 32.2
X 4.8
Y 41.2
X 19.8
Y 33
X 5.6
Y 40.4
X 19
Y 33.8
X 6.4
Y 39.6
X 18.2
Y 34.6
X 7.2
Y 38.8
X 17.4
Y 35.4
X 8
Y 38
X 16.6
Y 36.2
X 8.8
Y 37.2
X 15.8
Y 36.7
X 9.6

( top right)
G00 Z 1
X 25.4
Y 48.8
G01 Z -.9 F 30
Z 1
M00
Z -1
X 15.9
G02 X 14.7 Y 50  I 0 J 1.2
G01 Y 70 X 13.7
Y 76.2

( mill out)
X 25.4 F 150
Y 49.6
X 15.9
G02 X 15.5 Y 50 I 0 J .4
G01 Y 76.2
X 14.6
Y 70
X 15.9 Y 50.4
X 24.6
Y 75.4
X 16.3
Y 51.2
X 23.8
Y 74.6
```

```
X 17.1
Y 52
X 23
Y 73.8
X 17.9
Y 52.8
X 22.2
Y 73
X 18.7
Y 53.6
X 21.4
Y 72.2
X 19.5
Y 54.4
X 20.6
Y 71.4
X 20
Y 55.2
Z 10
```

D.7 Quantum Dot Multiplexer

```
G21
G00 Z 10
Y 0
X 12.7
Z 1
G01 Z -1 F 25
Y 79.2
G00 Z 1
Y 0
X 14.7
G01 Z -1
Y 20
G02 X 16.7 Y 31.3426 I 33.1634 J 0
G03 X 18.7 Y 42.6851 I -31.1634 J 11.3426
G01 Y 79.2
G00 Z 1
Y 0
X 16.7
G01 Z -1
Y 18
G02 X 18.7 Y 29.3426 I 33.1634 J 0
G01 X 22.7 Y 40.3325
G03 X 24.7 Y 51.6750 I -31.1634 J 11.3426
G01 Y 79.2
G00 Z 1
Y 0
X 10.7
G01 Z -1
Y 20
G03 X 8.7 Y 31.3426 I -33.1634 J 0
G02 X 6.7 Y 42.6851 I 31.1634 J 11.3426
```

```
G01 Y 79.2
G00 Z 1
Y 0
X 8.7
G01 Z -1
Y 18
G03 X 6.7 Y 29.3426 I -33.1634 J 0
G01 X 2.7 Y 40.3325
G02 X .7 Y 51.6750 I 31.1634 J 11.3426
G01 Y 79.2
```

D.8 0.125 in. thick Lid Mold

```
(cut moat to define edges of mold)
(1/8 inch end mill, go 1/8 inch deep)

(0,0 is the same as for the other programs)

G90 (Absolute positioning)
G00 Z0.1
X-0.125
Y-0.125
Z0.01

G01 Z-0.1 F1
X1.125

G00 Z1   (tool up to clean chips)
M00
G00 Z0.01

G01 Z-0.1 F1
Y1.5

G00 Z1
M00
G00 Z0.01

G01 Z-0.1 F1
Y3.125

G00 Z1
M00
G00 Z0.01

G01 Z-0.1 F1
X-0.125

G00 Z1
M00
G00 Z0.01

G01 Z-0.1 F1
Y1.5
```

```
G00 Z1
M00
G00 Z0.01

G01 Z-0.1 F1
Y-0.125

G00 Z2
```

D.9 1 mm thick Lid Mold

```
G00 X -.0625
G00 Y -.0625
G00 Z .01
G01 Z -.1 F 2
X 1.0625
Y 3.0625
X -.0625
Y -.0625
X 0
```

D.10 Fluidic Y-Branch Channel Mold

```
G00 Z 1
G00 X .3509
G00 Y 2.9
G00 Z .01
G01 Z -.01 F 1
X .4759
X .4134 Y 2.9625
X .3509 Y 2.9
X .4759
Y 0.4
X .3509
Y 2.9
X .4134
Y .4
X .1 Y .1
X .4134 Y .4
X .7268 Y .1
Z -.1
G00 Z .01
X .1
Y .1
G01 Z -.1
G00 Z .01
X .4134
Y 2.9625
G01 Z -.1
G00 Z 1
```

Bibliography

1. T. Vilkner, D. Janasek, A. Manz, "Micro Total Analysis Systems. Recent Developments," Anal. Chem. 76, 3373-3386 (2004).
2. CY Lee, GB Lee, JL Lin, FC Huang, CS Liao, "Integrated microfluidic systems for cell lysis, mixing/pumping and DNA amplification," J. Micromech. Microeng. 15, 1215-1223 (2005)
3. L Cui, T Zhang, H Morgan, "Optical particle detection integrated in a dielectrophoretic lab-on-a-chip," J. Micromech. Microeng. 12, 7-12 (2002)
4. D. A Chang-Yen, B. K. Gale, "An integrated Optical Glucose Sensor Fabricated Using PDMS Waveguides on a PDMS Substrate," Proc. SPIE – Intern. Soc. Optical Eng. v 5345, 98-107 (2004)
5. M. L. Chabinyc, D. T. Chiu, J. C. McDonald, A. D. Stroock, J. F. Christian, A. M. Karger, G. M. Whitesides, "An Integrated Fluorescence Detection System in Poly(dimethylsiloxane) for Microfluidic Applications," Anal. Chem. 73, 4491-4498 (2001)
6. K. Windhorn, L. Meixner, S. Drost, H. Schroder, E. Kilgus, W. Scheel, H. Reichl, F. Ebling, M. Kuchler, J. Nester, T. Otto, T. Gebner, "Polymer BioMEMs with integrated optical and fluidic functionality," Proc. Micro Systems Tech. 2003, 416-423 (2003)
7. D. Snakenborg, G. Perozziello, O. Geschke, J. P. Kutter, "Novel, Fast and Flexible Methods for Fabrication of Polymer-Based Optical Waveguides," Proc. MicroTAS 2005 Conference, vol 2, 1224-1226 (2005)
8. D. A. Chang-Yen, R. K. Eich, B. K. Gale, "A Monolithic PDMS Waveguide System Fabricated using Soft-Lithography Techniques," J. Lightwave Tech. v 23 n 6 2088-2093 (2005)
9. J. M. Engel, J. Chen, D. Bullen, C. Liu, "Polyurethane Rubber as a MEMS Material: Characterization and Demonstration of an All-Polymer Two-Axis Artificial Hair Cell Flow Sensor," 18th IEEE International Conference on MEMS, 279-282 (2005)
10. K. Lee, D. A. Dornfeld, "A Study of Surface Roughness in the Micro-End-Milling Process," Consortium on Deburring and Edge Finishing, (2004)
11. D. H. Solomon, "The Chemistry of Organic Film Formers," New York, Wiley, 30-31 (1967)
12. V. D. McGinniss, "Vaporous Solvent Treatment of Thermoplastic Substrates," United States Patent 4529563 (1985)
13. J. W. Van Dyk, "Process for Removing Surface Defects from Articles Having a Thermoplastic Surface," United States Patent 3684553 (1972)
14. D. C. Duffy, J. C. McDonald, O. J. A. Schueller, G. M. Whitesides, "Rapid Prototyping of Microfluidic Systems in Poly(dimethylsiloxane)," Anal. Chem. 70, 4974-4984 (1998)

15. Y. Xia, E. Kim, G. M. Whitesides, "Micromolding in Capillaries: Applications in Materials Science," J. Am. Chem. Soc. vol 118, 5722-5731 (1996)
16. C. P. Tripp, M. L. Hair, "Reaction of Methylsilanols with Hydrated Silica Surfaces: The Hydrolysis of Trichloro-, Dichloro-, and Monochloromethylsilanes and the Effects of Curing," American Chemical Society, vol 11, 149-155 (1995)
17. C. M. Olsen, J. M. Trewheila, B. Fan, M. M. Oprysko, "Propagation Properties in Short Lengths of Rectangular Epoxy Waveguides," IEEE Photonics Tech. Letters. vol 4, no. 2, 145-148 (1992)
18. Optical Reference OSLO
19. T. Bierhoff, A. Wallrabenstein, A. Himmler, E. Griese, and G. Mrozynski, "Ray tracing technique and its verification for the analysis of highly multimode optical waveguides with rough surfaces," *IEEE Trans.Magn.*, vol. 37, 3307–3310 (2001)
20. D. J. Guerrero, B. DiMenna, T. Flaim, R. Mercado, S. Sun, "Dyed Red, Green, and Blue Photoresist for Manufacture of High Resolution Color Filter Arrays for Image Sensors," Proceedings of the SPIE- The International Society for Optical Engineering, vol. 5017, 298-306 (2003)
21. http://omlc.ogi.edu/spectra/PhotochemCAD/html/
22. R. C. Benson, H. A. Kues, "Absorption and Fluorescence Properties of Cyanine Dyes," Journal of Chemical and Engineering Data, vol 22 no 4, 379-383 (1977)
23. Y. K. Olsson, G. Chen, R. Rapaport, D. T. Fuchs, V. C. Sundar, J. S. Steckel, M. G. Bawendi, A. Aharoni, U. Banin, "Fabrication and optical properties of polymeric waveguides containing nanocrystalline quantum dots," Applied Physics Letters, vol 85 no 19, 4469-4471 (2004)
24. Y. C. Chen, J. J. Barzier, M. Yan, S. A. Prahl, "Evaluation of Molecularly Imprinted Polyurethane as an Optical Waveguide for PAH Sensing," SPIE Proceedings in Nanosensing: Materials and Devices Symposium, vol 5593, 513-520 (2004)
25. W. P. Risk, H. C. Kim, R. D. Miller, H. Temkin, S. Gangopadhyay, "Optical waveguides with an aqueous core and a low-index nanoporous cladding," Optics Express, vol 12 no 26, 6446-6455 (2004)
26. L. Li, G. P. Nordin, J. M. English, J. Jiang, "Small-area bends and beamsplitters for low-index-contrast waveguides," Optics Express, vol 11 no 3, 282-290 (2003)
27. H. Schmid, H. Wolf, R. Allenspach, H. Riel, S. Karg, B. Michel, E. Delamarche, "Preparation of metallic films on elastomeric stamps and their application for contact processing and contact printing," Advanced Functional Materials, vol 13, 145-153 (2003)
28. H. L. T. Lee, R. J. Ram, "Integrated Fluid Injectors and Mixers for pH Control in Miniature Bioreactor Arrays," Proc. MicroTAS conference 2005, vol 1, 34-36 (2005)
29. D. B. Papkovsky, G. V. Ponomarev, W. Trettnak, P. O'Leary, "Phosphorescent Complexes of Porphyrin Ketones: Optical Properties and Application to Oxygen Sensing," Anal. Chem, vol 67, 4112-4117 (1995)
30. T. Barwicz, H. A. Haus, "Three-Dimensional Analysis of Scattering Losses Due to Sidewall Roughness in Microphotonic Waveguides," Journal of Lightwave Technology, vol 23 no 9, 2719-2732 (2005)
31. **https://www.norlandprod.com/adhesives/NOA%2071.html**
32. S. H. Hur, D. Y. Khang, C. Kocabas, J. A. Rogers, "Nanotransfer printing by use of noncovalent surface forces: Applications to thin-film transistors that use single-walled carbon nanotube networks and semiconducting polymers," Applied Physics Letters, vol 85 no 23, 5730-5732 (2004)
33. D. Landolt, "Fundamental aspects of electropolishing," Electrochimica Acta, vol 32 no 1, 1-11 (1987)

34. R. T. Chen, L. Lin, C. Choi, Y. J. Liu, B. Bihari, L. Wu, S. Tang, R. Wickman, B. Picor, M. K. Hibbs-Brenner, J. Bristow, W. S. Liu, "Fully Embedded Board-Level Guided-Wave Optoelectronic Interconnects," Proceedings of IEEE, vol 88 no 6, 780-793 (2000)

35. B. E. A. Saleh, M. C. Teich, "Fundamentals of Photonics," John Wiley and Sons, Inc. (1991)

36. S. A. Crooker, T. Barrick, J. A. Hollingsworth, V. I. Klimov, "Multiple temperature regimes of radiative decay in CdSe nanocrystal quantum dots: Intrinsic limits to the dark-exciton lifetime," Applied Physics Letters, vol 82 no 17, 2793-2795 (2003)