2004

Sequential alignment and position verification system for functional proton radiosurgery

Veysi Malkoc

SEQUENTIAL ALIGNMNET AND POSITION VERIFICATION SYSTEM FOR FUNCTIONAL PROTON RADIOSURGERY

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

Veysi Malkoc

June 2004

SEQUENTIAL ALIGNMNET AND POSITION VERIFICATION SYSTEM FOR FUNCTIONAL PROTON RADIOSURGERY

A Thesis

Presented to the

Faculty of

California State University,

San Bernardino

---

by

Veysi Malkoc

June 2004

[redacted] Approved by:

[redacted]

Yasha Karant, Chair
Computer Science

4 June 2004
Date

[redacted]

Keith Evan Schubert

[redacted]

Ernesto Gomez

[redacted]

Reinhard Schulte,
Loma Linda University Medical Center

ABSTRACT

Traditional neurosurgery procedures require invasive techniques such as opening the patient's skull in order to gain access to the area of interest such as a tumor. These procedures are risky, especially for elderly patients. An alternative technique is proton-beam radiosurgery, which uses high-energy protons to destroy a target in the patient's brain. This non-invasive form of neurosurgery safer and more comfortable for the patient.

For proton radiosurgery, the proton beam is generated by a proton accelerator outside the treatment room which delivers protons through a narrow conduit to a large cylindrical gantry. The gantry has a full 360-degree rotation range about a horizontal axis. At the end of the beam delivery system is a cone that collimates the proton beam to ensure a straight and narrow beam. The collimated proton beam rotates with the gantry on a plane and is always approximately directed at a point on the gantry's rotation axis, called the isocenter.

Prior to the treatment, a circular metal apparatus, called a halo, is firmly affixed to the patient's skull to establish a coordinate system about the patient's head. The halo is then fitted with a box-like frame

called a fiducial system. A Magnetic Resonance Imaging (MRI) is used to determine the location of the target area in the patient's brain relative to the halo coordinate system. Consequently, the position of the target area is known in the halo's coordinate system. The fiducial box is removed from the patient, who will then be placed on a six-degree-of-freedom (6-DOF) table, called a Patient Positioning System (PPS). The PPS is positioned and oriented so that the proton beam path intersects the target area at specific angles.

We chose an optical positioning system (OPS) manufactured by Vicon Motion Systems Inc, has been chosen to verify the correct alignment of the target point with the proton beam axis. The system is accurate to within ±0.1 mm to provide the position measurement component of the sequential alignment and position verification system (SAPVS) for the proton beam radiosurgery procedure. The OPS supplemented with a marker system that enables it to track all the relevant movements in the radiosurgery procedure.

The coordinate transformation between global and the local coordinates, which is required for the alignment

and verification procedure is the major subject of this thesis. Unitary and least square based coordinate transformation were researched and compared in order to find the optimum and most accurate transformation algorithm. A matlab image processing sequence is applied to check the performance of the SAPVS with these transformation algorithms.

It was found that the alignment method based on unitary transformation was significantly more accurate than that based on the least squares coordinate transformation. The alignment accuracy with the unitary transformation was within 5 mm. The precision was about 2 mm (standard deviation). Further improvements of the system are required to reach submillimeter accuracy and precision.

ACKNOWLEDGMENTS

I would like to begin by thanking my advisors, Dr. Y. Karant, Dr. K. Schubert, Dr. E. Gomez and Dr. R. Schulte for being the members of my committee. I am grateful for the guidance and helpful discussions by my advisor Dr. Karant. I thank Dr. Schulte for always been willing to provide encouragement and for entering my life, and for Dr. Schubert for always being extraordinarily generous and for Dr. Gomez for being a great instructor.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

BACKGROUND

1.1 Introduction

The Loma Linda University Medical Center (LLUMC), located in Loma Linda, California, is part of the Loma Linda University Health Sciences Complex. The University, established in 1905, now has six schools: Allied Health Professions, Dentistry, Medicine, Nursing, Public Health, and the Graduate School. The Medical Center opened next to the University in 1967, with the opening of the Children's Hospital following in 1993. LLUMC runs several of the biggest clinical programs in the United States, and recognized as the international leader in infant heart transplantation and proton treatment for cancer.

The LLUMC's Proton Treatment Center was the first hospital-based proton-beam facility in the world, especially designed for patient treatment and research. Their staff of physicians and researchers is the international leader in the use of radiation therapy. From among them, a small field project group was established in 1995. The group's purpose is to develop

proton radiosurgery techniques for practical use in treating cancer and functional disorders such as Parkinson's disease with narrow beams and high doses. From this group, Dr. R. Schulte, Dr. M. Moyers, Dr. R. Levy, Dr. D. Miller and a group of students from Harvey Mudd College, located in Claremont, California: P. Murata, J. Wong, J. Scott, R. Jackson, M. Thomas, A. Malone and their adviser Dr. E. Spjut started up the project called "Sequential Alignment and Positioning Verification System for Functional Proton Radiosurgery ".

1.2 Thesis Background

Traditional neurosurgery procedures require invasive techniques such as opening the patient's skull in order to gain access to the area of interest such as a cancer or tumor. These procedures are risky, especially for elderly patients. An alternative technique being developed is proton-beam radiosurgery with multiple narrow high-energy proton beams to destroy a small target in the patient's brain. This non-invasive form of neurosurgery is expected to be safer and more comfortable for the patient.

The proton beam is generated by channeling protons from a proton accelerator outside the treatment room through a narrow conduit connected to a large cylindrical gantry. The gantry, shown in Figure 1, has a full 360-degree rotation range about a horizontal axis. At the end of the beam delivery system is a cone that collimates the proton beam to ensure a straight and narrow beam. The conduit rotates with the gantry on a plane and will always be approximately directed at a point on the gantry's rotation axis, called the isocenter.

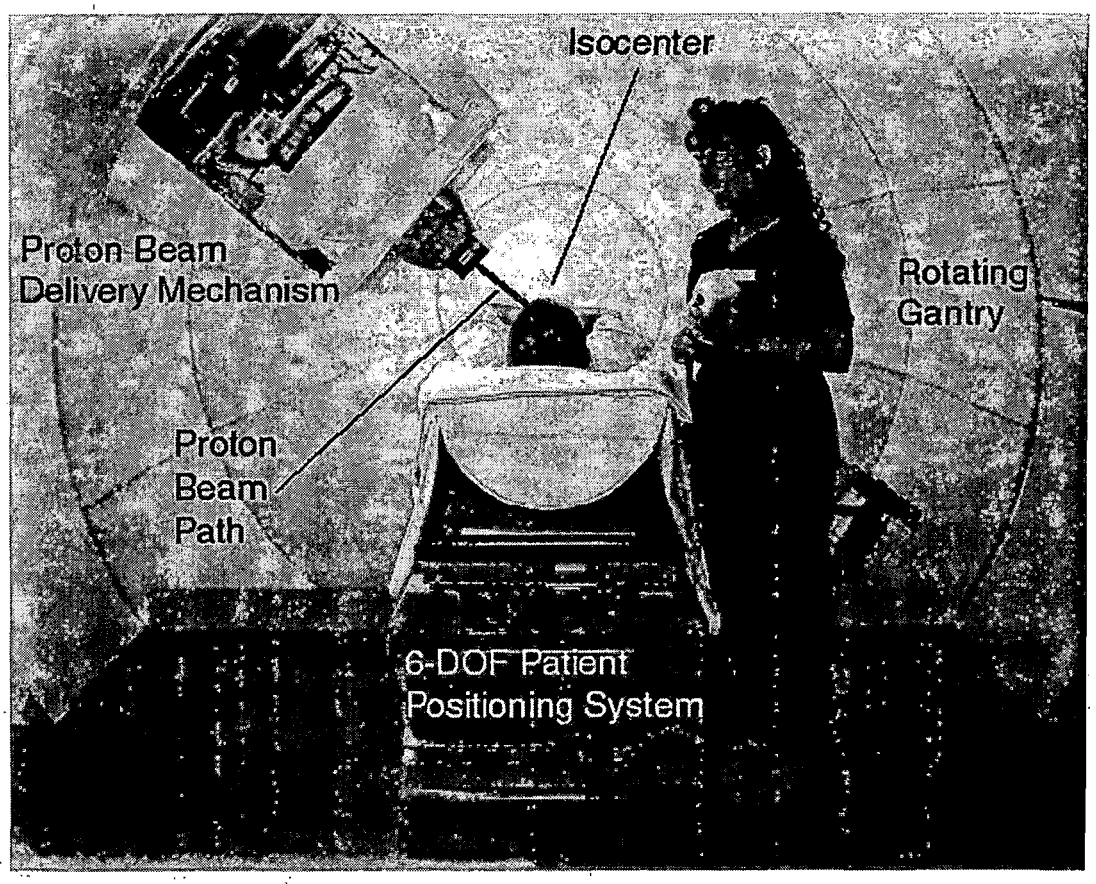


Figure 1. Proton Beam Treatment Gantry and Patient Positioning System

Prior to the treatment, a circular metal apparatus, called a halo, is firmly affixed to the patient's skull to establish a local system about the patient's head. A box-like frame called a fiducial system is attached to the halo prior to imaging the patient for target localization. Both devices are shown in Figure 2. A Magnetic Resonance Imaging (MRI) is used to determine the location of the target area in the patient's brain

relative to the fiducial system. Consequently, the position of the target area will be known in the halo's coordinate system. The fiducial system is removed from the patient, who will then be placed on a six-degree-of-freedom (6-DOF) table, called a Patient Positioning System (PPS).

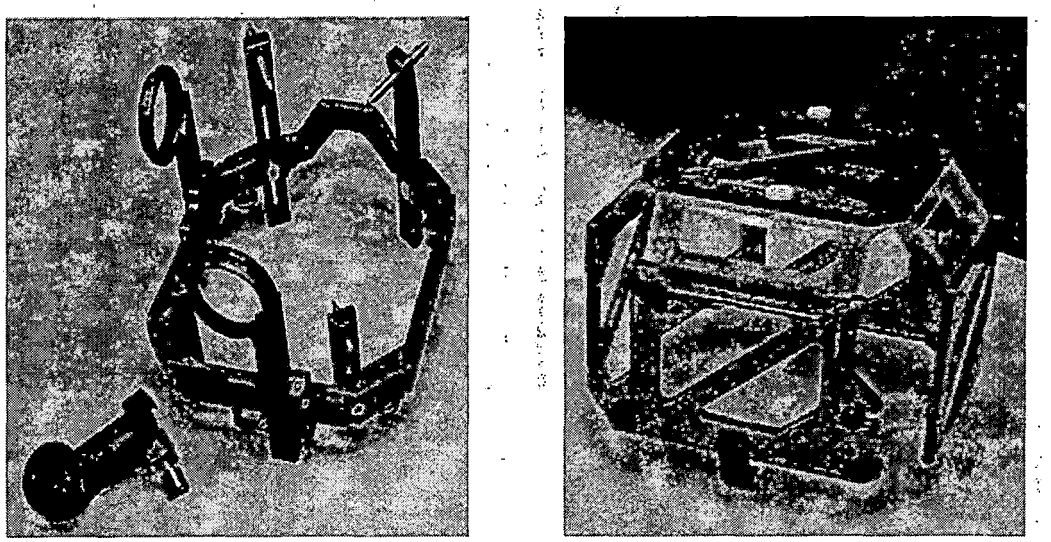


Figure 2. Halo (left) and Fiducial System (right)

The PPS will be positioned and oriented so that the proton beam path intersects the target area at multiple specific angles, creating a highly focused dose distribution. At LLUMC, this functional radiosurgery treatment will use 2-4 mm-wide beams. During the procedure the proton beam radiation will be applied for approximately forty seconds at each orientation. The

Patient Positioning System (PPS) will be repositioned approximately five times. At each PPS position, the radiation will be applied from five to seven different gantry positions, resulting in thirty to thirty-five narrow beams per treatment. Due to the sensitivity of the brain and the risk of applying too much dosage to an untargeted area or too little dosage to target area, the proton beam path must intersect the target area within a tolerance of ± 0.5 mm.

1.2.1 System Effectiveness

The effectiveness of the SAPVS depends on its accuracy, time efficiency, and freedom of movement.

1.2.1.1 Accuracy. Accuracy is a bounded objective, which applies to both position and orientation. The position accuracy requires that the center of the target area must be positioned to within a maximum of ± 0.5 mm from the center of the proton beam; however, even greater accuracy is desirable. Orientation accuracy refers to the angle at which the proton beam intersects the target area. The bound on this objective is to be within ±5° of the desired angle.

1.2.1.2 Time Efficiency. A time efficient system must have a small installation time, calibration time, alignment and verification time, and removal time. The time required for installation and calibration of the system for each patient treatment is limited to 15 minutes. The alignment and verification time for each dosage delivery is constrained to approximately 45 seconds. The removal time of the system is constrained to 5 minutes.

1.2.1.3 Freedom of Movement. The system can in no way reduce or impair the motion of the gantry or the PPS. If the treatment equipment were not guaranteed freedom of movement, the system's effectiveness could be greatly inhibited. The PPS can pitch and roll up to ±5° and yaw up to ±30°. It has two translation capabilities: a rough translation in three directions that allows for large movements on the order of meters, and a fine translation in three directions that allows for adjustments in increments of 0.01 mm.

1.2.2 Equipment Health

All hardware and equipment must be durable throughout the treatment process as well as in the

interim between treatments. If the system fails, it should be repairable within a reasonable amount of time. The system should have a long duty cycle. Repairs should be required as infrequently as possible, leaving the system online and functioning properly most of the time. The components of the system should have a sufficiently long lifetime to not be costly, either in time or in money.

1.2.3 Durability

The system must be durable within its operating environment. One concern is the ionizing radiation within the room due to the proton beam. The SAPVS should be relatively protected from degradation due to the radiation as far as its duty cycle and lifetime are concerned. Another concern is the electromagnetic interference within the room caused by the stray fields from the electromagnets directing the proton beam. These stray fields should not unpredictably affect the system. The system should also be mechanically robust and not brittle or fragile.

1.2.4 Patient Comfort

To provide optimal patient comfort the system seeks to minimize visual, audio, and mechanical disturbances

and the physical restraints placed on the patient. Visual disturbances include flashing lights or lasers in view of the patient. These disturbances are most likely to be found in optical or laser systems. Audio disturbances are audible noises produced by the system. Mechanical disturbances encompass any uncomfortable mechanical movements or vibrations in addition to the repositioning required by the procedure. Collisions between the apparatus and the patient, are not acceptable. Patient constraints are physical encumbrances to the patient, in excess of those imposed by the halo, that restrict his or her movement. An example of a patient constraint is the attachment of the halo to the PPS, which is unavoidable.

1.2.5 Cost

Cost reduction is desired in equipment, manufacturing, maintenance and repair, and user training. Equipment consists of the components of our system. Manufacturing costs account for assembly and any manufacturing work that goes into our final system. Maintenance and repair costs include costs incurred by the replacement of some components, which will most likely be done by hospital staff, and external technical support, which may be necessary for the upkeep of more

intricate components. User training costs include the texts or classes needed to teach practitioners to proficiently operate the system.

1.2.6 Safety

Safety is a major constraint. The system and equipment must not present a threat to the patient, practitioner, or other equipment during the course of the treatment.

1.3 Statement of the Problem

For functional proton radiosurgery, patients are placed at a specified position and orientation so that the desired treatment area in the patient's brain is accurately aligned with the path of the proton beam. The SAPVS in its current form is not accurate enough to achieve the goal of aligning the anatomical target center to within +/- 0.5 mm with respect to the center of the proton beam. Further, essential parts of the software controlling the system have not been written.

1.4 Purpose of the Thesis

The purpose of the thesis is to improve the existing version of the Sequential Alignment and Position

Verification System (SAPVS) for functional proton radiosurgery and to evaluate its performance after improvement. Improvement and evaluation is to be researched by determining the optimum and best accurate coordinate transformation among unitary transformation and least square based transformation. According to the research results, this thesis will require development of new hardware for the system and the software for aligning a specified target point to the central proton beam axis. Finally, I will determine the alignment accuracy that can be achieved with this system after the improvements and software development have been accomplished.

1.5 Significance of the Thesis

Significance of the thesis is to develop a procedure for aligning the anatomical target with respect to the center of the proton beam with a new method that is more accurate and precise than the existing one.

1.6 Assumptions

The following assumptions were made regarding the thesis:

1. The cameras are placed such that they can capture the position of fiducial marker sets

representing the position of the patient's head and the beam delivery system (cone).

2. The patient positioner table moves to given destination points within its specifications. Error introduced by the table movements are not considered in this thesis.
3. The proton beam is simulated by an expanded circular laser beam, which has a 10mm diameter.
4. The center of the 5 mm spherical marker is surrogate for the anatomical target point in space with given halo coordinates.

1.7 Limitations

The following limitations are pertinent to the thesis:

1. The positioner table is accurate in "fine" translational coordinates ±0.4 mm, ±0.02 mm, ± 0.08 mm z, t, s axis respectively.
2. All of the measurements are referenced to the Dimension Inspection Laboratory coordinate values (DIL), which are accurate to within ± 0.1 mm.

3. The treatment cone projects the laser beam to the target point (marker). The projection occurs on a flat surface with minimal distortion of the beam shape and the marker shadow.

1.8 Definition of Terms

The following terms are defined as they apply to the thesis:

- 6 - Degree of Freedom (DOF) - 6 types of movements made by The Patient Positioner Table (PPT); Three orthogonal in translation and three rotational (pitch, roll, yaw).
- Binary Image - Binary images are images whose pixels have only two possible intensity values.
- Edge pixels - Pixels that are belong to the border of an object.
- Thresholding - The technique used to differentiate the object from the background.
- Bragg peak - The point at which protons (and other heavy charged particles) deposit most of their energy. This point occurs at the ends of the protons' paths. By varying the beam's energy,

radiation oncologists can spread this peak to match the contours of tumors or other targets.

- Cancer - Uncontrolled, abnormal growth of cells, which will invade and destroy healthy tissues if not controlled by effective treatment.
- Cobalt 60 - A naturally radioactive substance that is used in machines to treat cancer by external beams.
- Conduit - The proton beam generated by channeling protons from a proton accelerator outside the treatment room connects through a narrow conduit to a large cylindrical gantry.
- Cross - A localization device attached to the treatment cone. It is made of metal, shaped like a cross and has a marker system that has also the shape of a Cross.
- Fiducial system - A box-like frame, which the halo is fitted.
- Gamma rays High-energy rays that come from a radioactive source such as cobalt-60.
- Gantry - In radiation therapy, a device for rotating the radiation delivery apparatus around the patient,

so as to treat from different angles. The gantry has a full 360-degree rotation range about a horizontal axis.

- Halo - A circular metal apparatus, which is firmly affixed to the patient's skull to establish a coordinate system about the patient's head.
- Immobilization device - A device that prevents the patient from moving during radiation treatment. One example, used for proton treatment at Loma Linda, is a
- form-fitting foam liner surrounded by a rigid plastic shell, in which a patient can lie comfortably during treatment.
- Isocenter - At the end of the conduit is a cone that will collimate the proton beam to ensure a straight and narrow beam. The conduit rotates with the gantry on a plane and can move radially but will always be approximately directed at a point on the gantry's rotation axis, called the isocenter.
- Laser beam - A very directional,very tight, very strong and concentrated beam that is formed by the emission of photons.

- Linear accelerator - A machine that creates high-energy radiation to treat cancers, using electricity to form a stream of fast-moving subatomic particles. Also called a megavoltage (MeV) linear accelerator or "linac" (pronounced LYNN-ack).
- Marker - Marker is a plastic sphere covered with retroreflective tape.
- Marker caddy - A frame, which has a marker system on and fixed to the halo in order to track patient's head by the cameras.
- Patient Positioner System(PPS)- A positioner table used for medical purposes, which allows precise and accurate positioning within its specifications.
- Phantom - A device that has pins carrying a target marker and holes where the pins can be placed. In this thesis, it was used to test alignment accuracy.
- Photon - A quantum (energy packet) of electromagnetic radiation; the elementary particle of photon radiation therapy. X rays and gamma rays are photon radiation.
- Proton - Positively charged subatomic particle.

- Proton Radiation Therapy - It is a form of external-beam radiation treatment. Radiation oncologists (physicians who specialize in radiation treatments) can treat the tumors, functional legions, etc. by using an accelerator to generate proton beams.
- Radiosurgery - Radiosurgery is pinpoint precision radiation using multiple, finely-contoured beams from many different angles - all directed at the target and minimizing radiation to normal tissue while the patient's body is maintained in a stable, reproducible position.
- Treatment cone - The actual treatment device that directs and collimates proton radiation beams.
- Tumor - An abnormal mass of tissue. Tumors are either benign or malignant.
- Vicon - The company that produces the cameras used in this thesis.
- X rays - High-energy, ionizing, electromagnetic radiation that can be used at low doses to diagnose disease or at high doses to treat cancer.

1.9 Organization of the Thesis

The thesis is divided into five chapters. Chapter One provides an introduction to the context of the problem, purpose of the thesis, and significance of the thesis, assumptions, limitations and definition of the terms. Chapter two and three discusses the literature. Chapter four documents the methodology used in this thesis. Chapter five presents the results and recommendations.

CHAPTER TWO

RADIATION TREATMENT MODALITIES

2.1 Radiation Therapy and Radiosurgery

Forty years ago, the ability to diagnose and treat an individual with a brain tumor was limited by crude surgical and radiological tools. Modern neurosurgical tools and techniques and advanced imaging modalities such as CT scan, now allow brain tumors to be identified much earlier in the course of the disease. Even when cure is not possible, an earlier diagnosis can result in an improved outcome for the patient through more appropriate utilization of radiation therapy.

Radiation therapy uses high-energy photon beams (X-rays or gamma rays) or charged particles (electron beams or proton beams) to damage critical biological molecules in lesion cells. If enough damage is done to the chromosomes of a cell, it will spontaneously die or it will die the next time it tries to undergo division into two cells. Radiation therapy is usually done on an outpatient basis with treatment occurring each workday for a period of several weeks.

Conventionally administered external beam radiation therapy gives a uniform dose of radiation to the entire region affected by the tumor. There is only a small variation of the dose delivered to various parts of the tumor. There is little chance that not enough or too much radiation will be given to any part of the tumor.

Treatment of brain tumors with external beam radiation therapy has been an area of intense research activity over the past several decades. Through clinical research, conducted on patients, much has been learned about how to appropriately use radiation therapy for various types of brain tumors. External beam radiation therapy is a valuable component of therapy for nearly all brain tumors; treatment can be delivered to any part, or all, of the central nervous system. The ability to assure uniform doses of radiation to the areas being treated is one of the major strengths of modern external beam radiation therapy. The radiation treatment therapies are gamma knife, LINAC and proton treatment.

2.2 Gamma Knife

The Gamma Knife is one of the non-invasive alternative technique for many patients for whom

traditional brain surgery is not an option and it removes the physical trauma and the majority of risks associated with conventional surgery.

The Gamma Knife contains 201 cobalt-60 sources of approximately 30 curies each, placed in a circular array in a heavily shielded unit. The unit directs gamma radiation to a target point. Such target points selected in the brain can be placed at the center of the radiation focus, allowing a tumoricidal radiation dosage to be delivered in one treatment session.

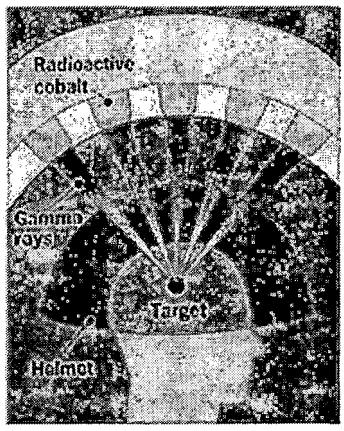


Figure 3. Gamma Knife Technique

2.3 Linac

Linac is short for the term linear accelerator. Linear accelerator machines produce radiation that is referred to as high energy X rays. A linear accelerator machine is designed to be a general-purpose radiation delivery machine and requires modifications to enable it to be used for radiosurgery. Often the modification is the addition of another piece of machinery. There may be dedicated or non-dedicated LINAC machines. The dedicated type has the additional equipment to perform the higher level treatment permanently attached to the radiation couch. This is the preferred method. The non-dedicated LINAC machines may be used for conventional radiation therapy in the morning and the after adding the attachment, are used for higher level treatments in the afternoon. Non-dedicated machines are unable to acquire the same degree of precision and accuracy that dedicated machine may have.

2.4 Proton Beam Therapy

There is, as of yet, no ideal radiation beam. However, the dose pattern of a heavy-charged-particle beam, such as a proton beam, resembles the ideal more

closely than does the pattern of uncharged beams, such as X rays. When protons enter the body they deposit a minimal amount of energy; as they continue to traverse tissues, interact with orbiting electrons, and gradually slow down, the amount of energy they deposit per micron of travel gradually increases until they decelerate to a certain velocity. Then, suddenly, the remaining energy is given up in about a centimeter of distance. This is the Bragg peak. By varying the energy of the beam, the radiation oncologist can spread out the peak to encompass volumes greater than 1 cm. This essentially reverses the photon pattern: protons build their dose up near the end of their travel.

2.4.1 Factors Favoring the Proton Beam

The followings are the factors favoring the proton beam:

2.4.1.1 Charge. The proton's electric charge enables the radiation oncologist to position the Bragg peak. A charged particle offers a potential therapeutic advantage because the charge gives physicians the ability to locate the beam precisely. Neutrons and photons do not have a charge. They cannot be controlled electronically; one must control them mechanically. The radiation oncologist

employing neutrons or photons must shape the fields by putting obstructions in the path of the beam. If a particle has a charge, however, the physician has the opportunity to control the treatment electronically, a much more precise technique. This is true of any charged particle.

2.4.1.2 Mass. The mass of the subatomic particle used in radiation therapy influences its manner of depositing energy within a patient. As the mass of particles diminishes to that of an electron, and continues to zero, as with photons, scattering occurs increasingly. Such scattering defocuses the beam. Because the primary interaction of the incoming beam of particles within the patient is with electrons, the particles in the incoming beam ideally should have a mass much greater than the orbiting electrons, to avoid being scattered as their individual electric fields interact or collide.

A proton's mass is 1,835 times that of an electron; hence, scattering is reduced greatly as compared to an electron or photon beam. This can be compared somewhat to using a bowling ball instead of a cue ball in a game of billiards: a bowling ball rolling across a billiard table will maintain its path; the billiard balls will scatter.

The radiation oncologist's ability to focus the beam of protons on a specific three-dimensional target in a patient depends on the treatment facility's ability to provide the technology that manipulates and modulates the direction and primary energy of each proton. The protons' mass and charge dictate the pattern of energy deposited as the particles travel through the patient to their intended target.

The task of the radiation oncologist is to place energy in targeted cells. This requires three-dimensional control of each beam used; the finer the control the physician has over the therapy beam, the better the treatment the patient will receive.

2.4.1.3.LET. Radiotherapists tend to think of protons and helium ions as "light" ions, in the sense that both are characterized by relatively sparse ionization, or linear energy transfer, (LET) as they pass through tissue. Ions heavier than protons have advantages and disadvantages.

Radiation causes two basic effects in living tissue: one derives from the physical dose distribution; the other, from the biologic effect, which is a function of ionization density (LET). Accelerated particles exhibit a

spectrum of LET, generally referred to as either low or high. Photons, electrons, protons, and helium ions are in the class of low-LET particles; neutrons and all other heavy ions are considered to be high-LET particles in therapeutic terms. When a beam of high-LET particles traverses a cell, it leaves a very dense pathway of ionization; that dense pathway causes much disruption in both normal cells and tumor cells, and overwhelms the cells' natural capability to repair the damage. Low-LET particles, on the other hand, leave a sparsely ionizing pathway, which results in sparse ionization; the cell is left sufficiently intact to repair itself. As the damage to cells increases, the cells' repair ability decreases.

CHAPTER THREE

DIGITAL IMAGING

3.1 Introduction

An image defined in the "real world" is considered to be a function of two real variables, for example, a(x,y) with a as the amplitude (e.g. brightness) of the image at the real coordinate position (x,y). An image may be considered to contain sub-images sometimes referred to as regions-of-interest, or simply regions. This concept reflects the fact that images frequently contain collections of objects each of which can be the basis for a region. In a sophisticated image processing system it should be possible to apply specific image processing operations to selected regions.

The amplitudes of a given image will almost always be either real numbers or integer numbers. The latter is usually a result of a quantization process that converts a continuous range (say, between 0 and 100%) to a discrete number of levels. In certain image-forming processes, however, the signal may involve photon counting which implies that the amplitude would be inherently quantized. In other image forming procedures,

such as magnetic resonance imaging, the direct physical measurement yields a complex number in the form of a real magnitude and a real imaginary phase.

3.2 Digital Image

A digital image a[m,n] described in a 2D discrete space is derived from an analog image a(x,y) in a 2D continuous space through a sampling process that is frequently referred to as digitization. The effect of digitization is shown in Figure 4.

The 2D continuous image a(x,y) is divided into N rows and M columns. The intersection of a row and a column is termed a pixel. The value assigned to the integer coordinates [m,n] with { m=0,1,2,...,M-1 } and {n=0,1,2,...,N-1} is a[m,n]. In fact, in most cases a(x,y) - which we might consider to be the physical signal that impinges on the face of a 2D sensor - is actually a function of many variables including depth (z), color (λ), and time (t). Unless otherwise stated, we will consider the case of 2D, monochromatic, static images in this chapter.

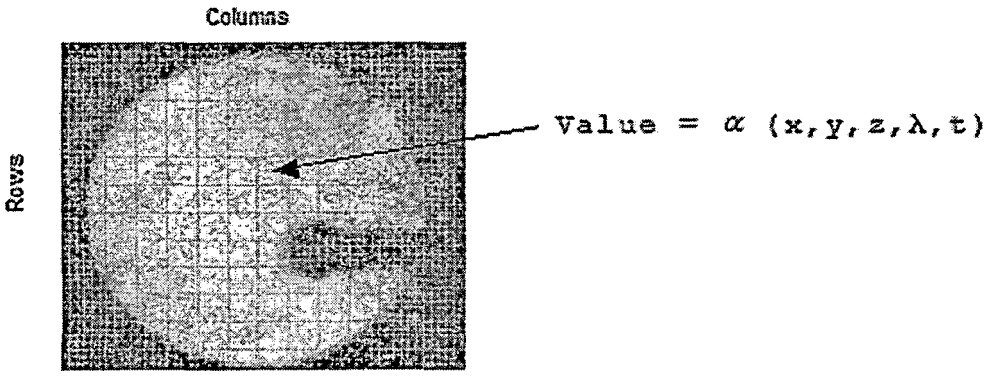


Figure 4. Digitization of a Continuous Image

The process of representing the amplitude of the 2D signal at a given coordinate as an integer value with L different gray levels is usually referred to as amplitude quantization or simply quantization.

3.3 Binary Image

The number of distinct gray levels is usually a power of 2, that is, $L=2^B$ where B is the number of bits in the binary representation of the brightness levels. When B>1 we speak of a gray-level image; when B=1 we speak of a binary image. In a binary image there are just two gray levels which can be referred to, for example, as "black" and "white" or "0" and "1".

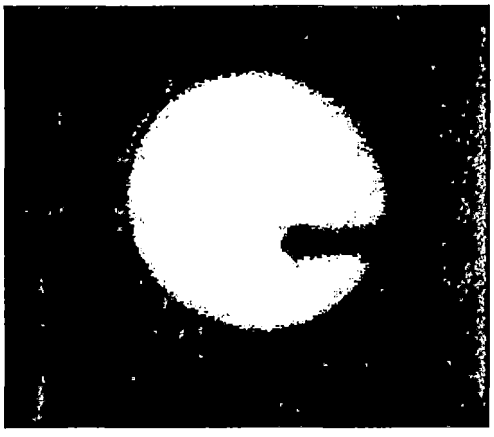


Figure 5. Binary Image

3.4 Thresholding

In the analysis of the objects in images it is essential that we can distinguish between the objects of interest and "the rest." This latter group is also referred to as the background. Thresholding is segmenting the foreground from background.

This technique is based upon a simple concept. A parameter θ called the brightness threshold is chosen and applied to the image a[m,n] as follows:

$if\ a[m,n] \geq \theta$	$a[m,n] = object = 1$
else	$a[m,n] = background = 0$

Figure 6. Thresholding Algorithm

3.5 JPEG Digital Image File Format

JPEG (pronounced "jay-peg") is a standardized image compression mechanism. JPEG stands for Joint Photographic Experts Group, the original name of the committee that wrote the standard.

JPEG is designed for compressing either full-color or gray-scale images of natural, real-world scenes. It works well on photographs, naturalistic artwork, and similar material; not so well on lettering, simple cartoons, or line drawings. JPEG handles only still images. PEG is "lossy," meaning that the decompressed image isn't quite the same as the one you started with. (There are lossless image compression algorithms, but JPEG achieves much greater compression than is possible with lossless methods.) JPEG is designed to exploit known limitations of the human eye, notably the fact that small color changes are perceived less accurately than small changes in brightness. Thus, JPEG is intended for compressing images that will be looked at by humans. If you plan to machine-analyze your images, the small errors introduced by JPEG may be a problem for you, even if they are invisible to the eye.

A useful property of JPEG is that the degree of lossiness can be varied by adjusting compression parameters. This means that the image maker can trade off file size against output image quality. You can make *extremely* small files if you don't mind poor quality; this is useful for applications such as indexing image archives. Conversely, if you aren't happy with the output quality at the default compression setting, you can jack up the quality until you are satisfied, and accept lesser compression.

There are two good reasons to use JPEG: to make your image files smaller, and to store 24-bit-per-pixel color data instead of 8-bit-per-pixel data.

CHAPTER FOUR

METHODOLOGY

4.1 System Components

The system components are the cameras, marker caddy, treatment cone, cross and the patient positioner table.

4.1.1 Optical Positioning System

Optical Positioning Systems (OPS) use a set of cameras to determine the positions of several markers in 3D space. To determine the position and orientation of a target area, the positions of a minimum of three separate markers, not all located in one line required. OPS's are commercially available, require no contact with the patient or other equipment create no audio disturbances, and the use of infrared-light-emitting diodes (IREDs) for target illumination can eliminate visual disturbances.

In this thesis, the cameras used for OPS are manufactured by a company called Vicon. Vicon produces motion capture systems used in biomechanical studies and animation. Their M-Series Cameras system (Mcam) seems most applicable to this thesis. Mcam operates with a resolution

of 1,000,000 pixels and costs $80K for a system using three cameras. The system requires an initial static calibration using an L-frame and a dynamic calibration using a wand. Once the system is calibrated, minimal calibration is necessary, assuming the relative positions of the cameras remain constant.

4.1.2 Camera Placement

Upon consideration of visibility, accuracy, volume of measurement, and patient comfort, the Vicon camera placement was determined as shown in the top illustration in figure 7. The cameras will be placed in an equilateral configuration at the edges of the circular disk at the back of the gantry, 1.62 m from the isocenter. Each camera will be aimed directly at the isocenter, which will allow them to capture all of the markers. The angles resulting from this configuration are shown in the bottom illustration in figure 7. Since the back of the gantry and the proton beam delivery cone rotate as one unit, the position of the cone relative to the cameras would be relatively fixed with only a small deviation due to the sag of the gantry.

The chosen camera placements minimize obstructions to the cameras' visibility, such as the cone of the

proton beam, the collector box, the PPS, and possibly practitioners. The cameras generally point at the top of the patient's head. All the markers that will be detected by the cameras (those on the marker caddy and the cone) lie roughly on a plane passing through the isocenter parallel to the back of the gantry. The camera placement thus allows visibility of as many of the markers as possible.

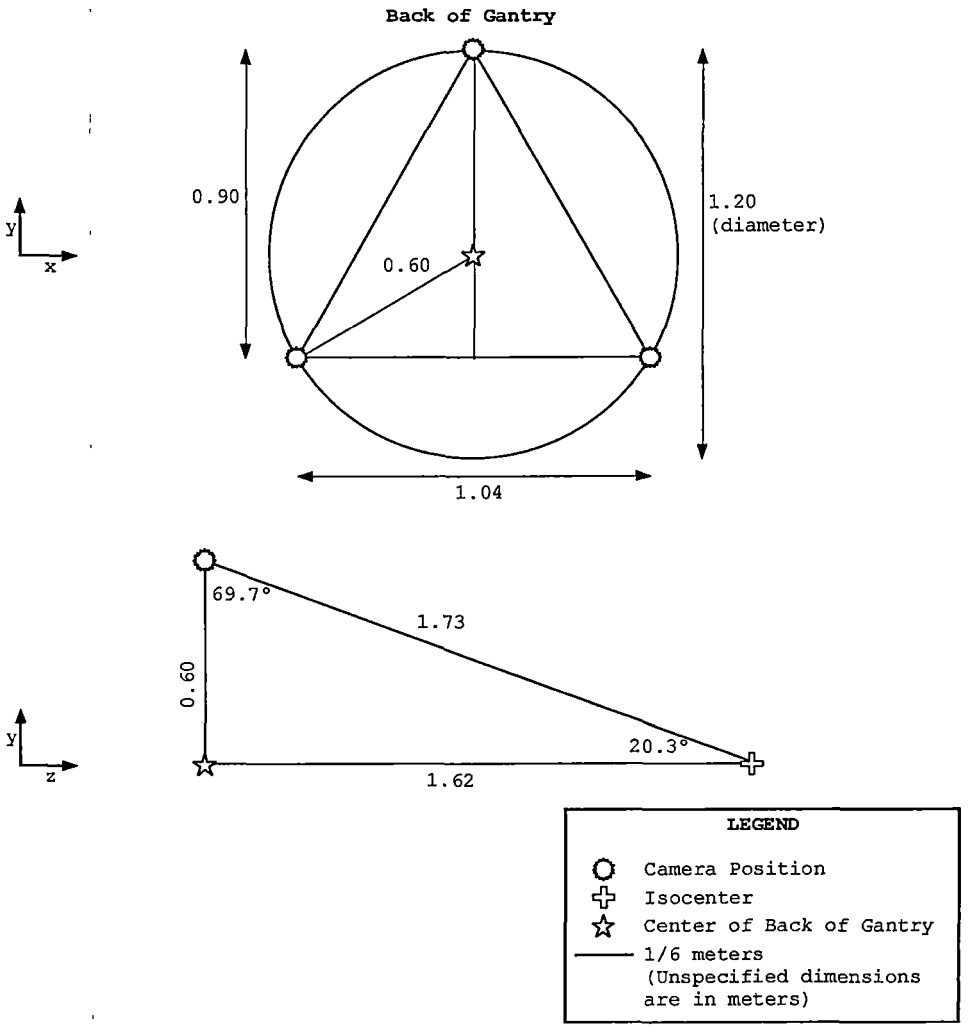


Figure 7. Optimal Vicon Camera Placement

The volume of measurement resulting from the cameras' configuration using 25-mm lenses is shown in Figure 8. The top illustration in Figure 8 shows a plane formed by two of the cameras and the isocenter. The 25-mm lenses achieve higher resolution than the standard 50-mm lenses, but at the sacrifice of field of vision. The 25-

mm lenses provide, at minimum, a square, 20° field of vision (for simplicity, Figure 8 assumes field-of-visions with circular cross-sections inscribed in the actual square cross-sections). The resulting visible areas overlap in the shaded area shown in Figure 8. This overlapping visibility is the same for any two cameras. Therefore, the three cameras' field-of-visions mutually share the cylindrical volume shown in the bottom illustration in Figure 8. The true volume of measurement is actually larger than this, since the cross-section of each camera's field-of-vision is larger than the assumed circular cross-section. This volume allows markers on the plane passing through the isocenter parallel to the back of the gantry, to be up to 0.3 m from the isocenter and is sufficient for our application.

To ensure that position information calculated by the Vicon system remains reliable, the Vicon cameras will have to be mounted such that their relationships to each other remain unchanged during a treatment and between treatments. The cameras will be mounted on individual mounts fixed to a steel ring at the back of the gantry. The mounts should ensure that the cameras are always

fixed relative to each other and that their relative positions do not change even after dismounting and remounting.

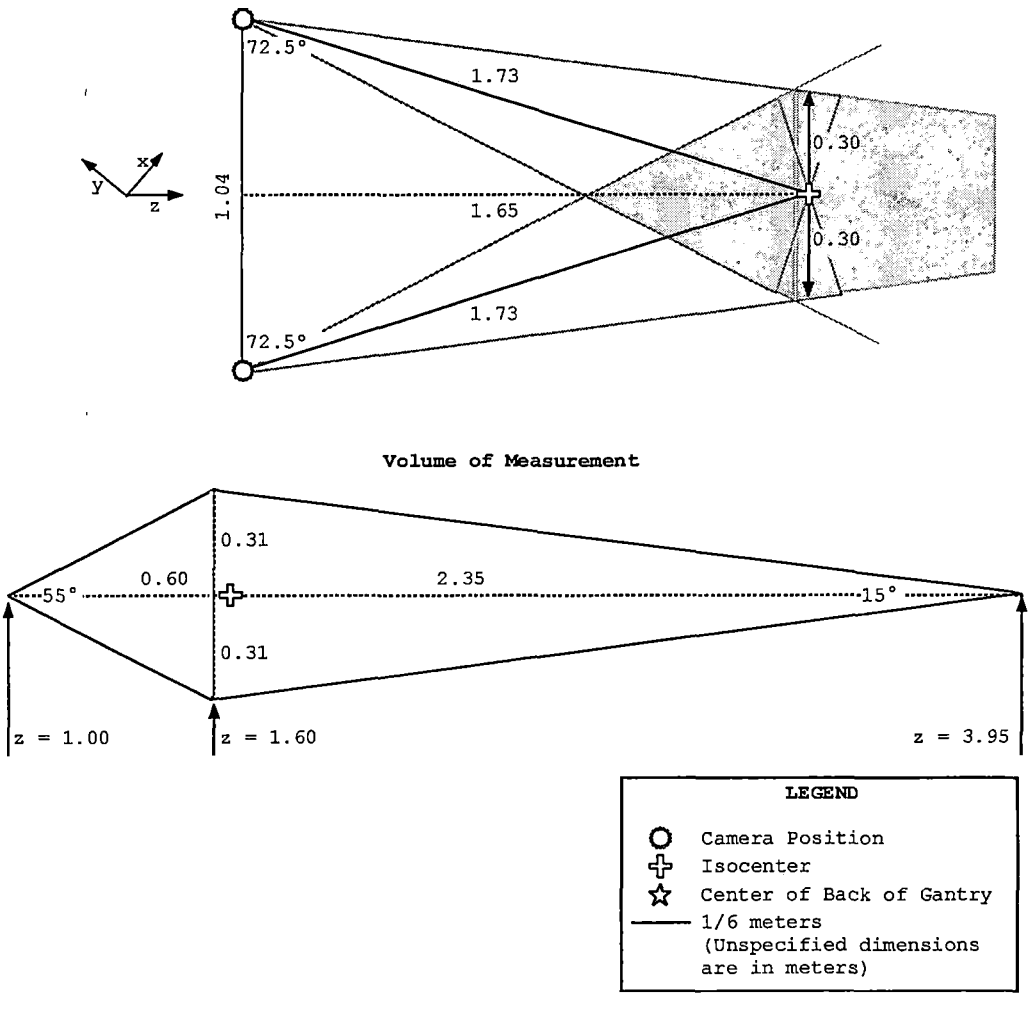


Figure 8. Camera Field of Vision and Volume of Measurement

4.1.3 Marker Caddy

In order for the Vicon system to accurately monitor the position of the target with respect to the proton beam, a marker system that could track the patient's head was developed. The halo that Loma Linda has chosen to use during their proton beam procedures is machined from a non-conducting metallic material to prevent it from causing magnetic disturbances during CT scans. To maintain the functionality of the system, the markers placed on the halo would either have to be non-conducting or be removable during MRI scans. Because of the cost of non-conducting metallic materials and the desire to maintain the versatility of the halo, focus was placed on removable marker systems. Due to the nature of the Vicon system, at least three markers must be visible in two cameras through the entire range of motion. However it would be desirable for at least three markers to be visible by all three cameras at all times for increased accuracy. The markers should stay fixed in reference to each other on the system and the system should be unobtrusive to the patient, block a minimum number of entry angles and be trackable for the entire 180-degree range of motion of the PPS.

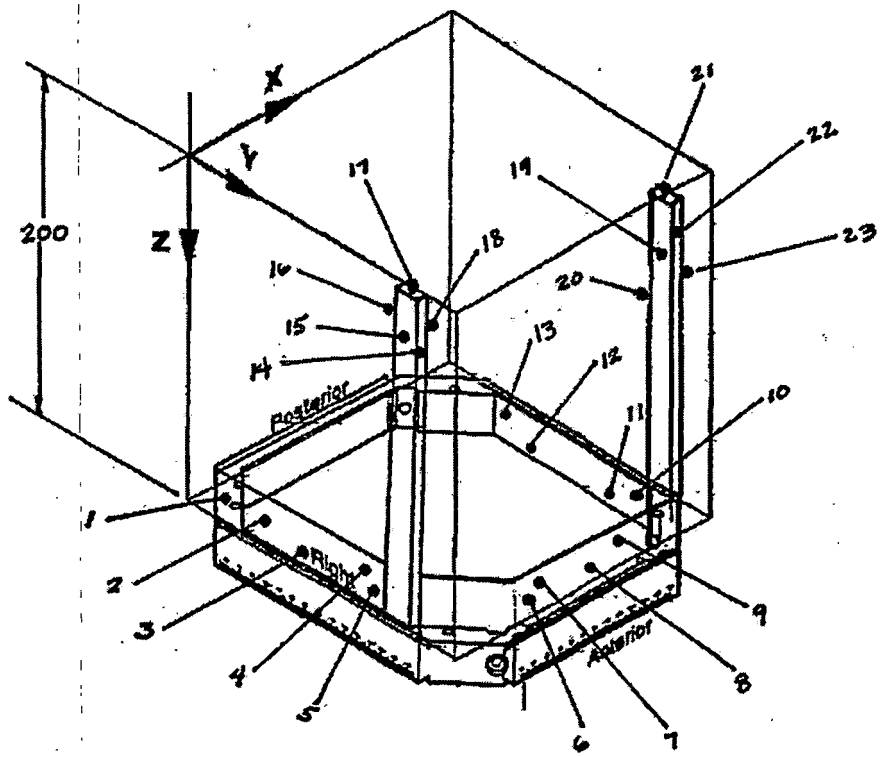


Figure 9. Marker Caddy and Halo Coordinate System

Therefore, the marker caddy consisted of the three major sides of a cut-corner square frame with two posts, one on each side near the top of the frame as shown in figure 9. The fourth side of the frame was eliminated so that the marker caddy could be slid on to the halo without disturbing fixtures that attach the halo to the head. The marker caddy frame rigidly attaches to the halo by having pegs on the caddy inserted into four peg holes already present on the halo and then clamping the

two pieces together, eliminating any relative motion between the caddy and the halo. The marker caddy is similar to the fiducial system since it will represent the halo's coordinate system.

4.1.4 Markers

The markers Vicon typically uses are plastic spheres covered with retroreflective tape. Retroreflective surfaces reflect a large fraction of incident light directly back at the light source (Figure 10).

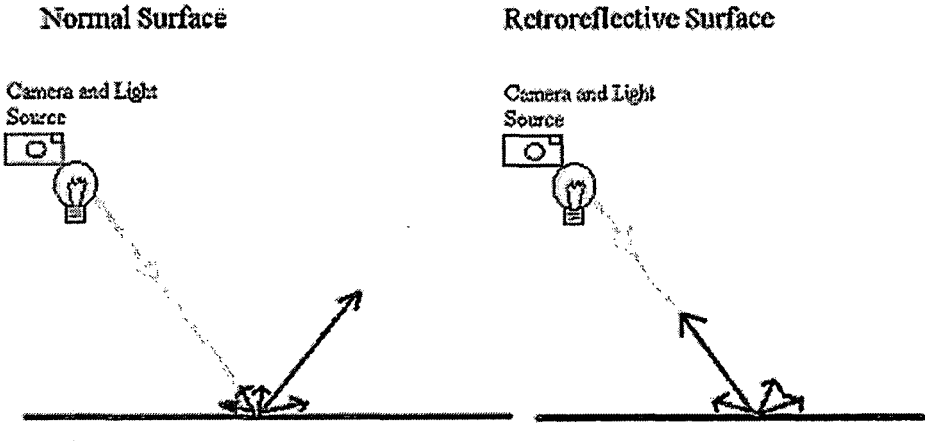


Figure 10. Retroreflectivity

4.1.5 Patient Positioning System

Patient Positioning System allows precise and accurate positioning within its specifications, by a patient positioner table which is used for medical purposes.

4.1.6 Treatment Cone

It is the device (shown in Figure 11) that direct and collimates proton radiation beams at the time of patient treatment.

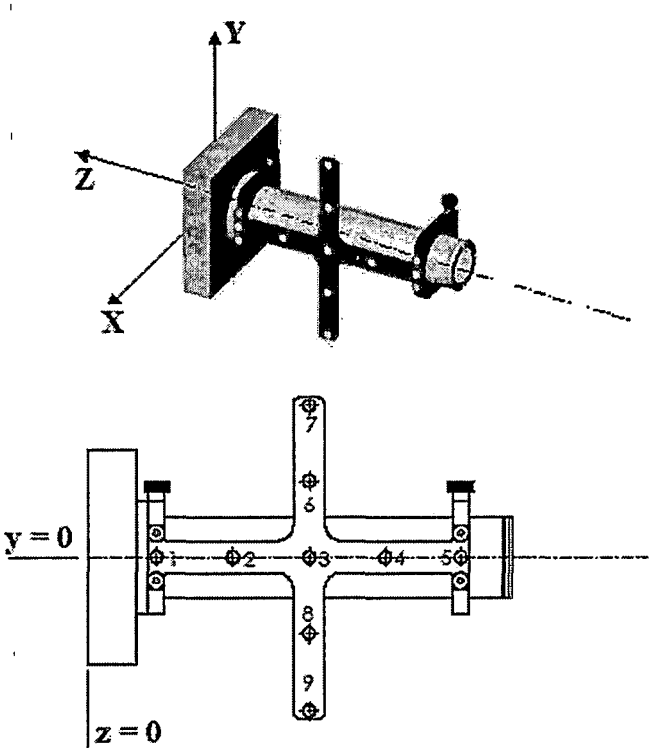


Figure 11. Treatment Cone with Cross Attached

4.1.7 Cross

It is the marker system (shown in Figure 11) attached to the treatment cone. It consists of metallic cross shaped plate and has nine markers attached to it.

4.2 Software

4.2.1 Serial Communication

RS-232 is the external interface for the communication protocol between the positioner table and the computer. Since the operating system is Windows NT, the serial communication between the computer and the table is more complicated.

Particularly in Windows NT, the serial communication can be done in two ways: overlapped or nonoverlapped.

4.2.1.1 Nonoverlapped I/O. It is very straightforward, though it has limitations. An operation takes place while the calling thread is blocked. Once the operation is complete, the function returns and the thread can continue its work. This type of I/O is useful for multithreaded applications because while one thread is blocked on an I/O operation, other threads can still perform work. It is the responsibility of the application to serialize access to the port correctly. If one thread

is blocked waiting for its I/O operation to complete, all other threads that subsequently call a communications API will be blocked until the original operation completes. For instance, if one thread were waiting for a ReadFile function to return, any other thread that issued a WriteFile function would be blocked.

One of the many factors to consider when choosing between nonoverlapped and overlapped operations is portability. Overlapped operation is not a good choice because most operating systems do not support it. Most operating systems support some form of multithreading, however, so multithreaded nonoverlapped I/O may be the best choice for portability reasons.

4.2.1.2 Overlapped I/O. Overlapped I/O is not as straightforward as nonoverlapped I/O, but allows more flexibility and efficiency. A port open for overlapped operations allows multiple threads to do I/O operations at the same time and perform other work while the operations are pending. Furthermore, the behavior of overlapped operations allows a single thread to issue many different requests and do work in the background while the operations are pending.

In both single-threaded and multithreaded applications, some synchronization must take place between issuing requests and processing the results. One thread will have to be blocked until the result of an operation is available. The advantage is that overlapped I/O allows a thread to do some work between the time of the request and its completion. If no work can be done, then the only case for overlapped I/O is that it allows for better user responsiveness. (Please see Appendix A for the serial communication implementation)

4.2.2 Stereotactic Transformations for Functional Proton Radiosurgery

4.2.2.1 Unitary Transformation.

Accurate stereotactic proton beam delivery for functional radiosurgery requires a mathematical transformation of coordinates from local coordinate systems, which change position in space during a treatment session, to a room-fixed global coordinate system, which is defined by the Vicon camera tracking system.

In general, the axes of the different coordinate systems will not be parallel with respect to each other. Therefore, the coordinate transformations mapping each

point of one reference system into another one involves both translations and rotations.

At least three linearly independent points, i.e., points that are not located on one straight line, with known coordinates in both reference systems are needed to calculate the equations for coordinate transformation between the two systems. The mathematical method to determine the coordinate transformation, which will be implemented in the computer code for the sequential alignment and positioning verification system for functional proton radiosurgery, is provided in Appendix B.

4.2.2.2 Least Square Based Transformation. The Least square method (sometimes called regression model) is a statistical approach to estimate an expected value or function with the highest probability from the observations with random errors. The highest probability is derived by minimizing the sum of squared of residuals. (Please see Appendix B for the Least Square Problem Solution applicable to this thesis)

4.2.3 Finding the Shortest Distance from the Target Point to the Beam Axis for Functional Radiosurgery Treatments

In functional radiosurgery, it is required to determine the shortest distance of the anatomical target point from the central beam axis. The target point T is predefined by the physician based on a MRI image study, and is then transformed into the stereotactic reference system. When the patient is set up in the treatment room, the target point coordinates are transformed into a global room-fixed coordinate system, defined by the Vicon cameras. In addition, we measure the actual position of the central beam axis, which is given by a point P_0 on the axis and a unit vector u, which gives the direction of the beam axis. The goal of the alignment process is to bring the target point in coincidence with the beam axis. The mathematical problem is then to find the vector d between the target point and the point H on the axis that has the shortest distance from the target. (Please see Appendix C for mathematical formula)

4.2.4 Checking the Distance of the Markers to each other

Distance verification is a pre-test for the accuracy of the alignment. The marker distances derived from

global coordinates coming from the camera system are compared to distances among local coordinates, which have been measured in a Dimension Inspection Laboratory (DIL). DIL values are the gold standard for the measurements in this thesis.

There are three structures for distance verification algorithm. First, "Point" is the class containing the coordinates of the marker in space. Second, for each marker, there is a vector of "Points" that has the size of the number of frames captured by the camera pertaining to the particular marker. Each frame has x, y, z coordinates respectively that fills in the coordinates in "Point" structure. Third, there is a vector structure that contains all the markers in the system. Later on, for each marker in space an average set of coordinates is obtained by averaging the number of frames arithmetically. (Please see Appendix D for the implementation)

	marker 1	marker 2	marker 3	...
1.	x, y, z	x, y, z	x, y, z	
2.	x, y, z	x, y, z	x, y, z	
3.	x, y, z	x, y, z	x, y, z	
4.	x, y, z	x, y, z	x, y, z	
5.	x, y, z	x, y, z	x, y, z	
.				
.				
.				
.				
N.	x, y, z	x, y, z	x, y, z	

Figure 12. Illustration of the Vector Structure

4.2.5 Image Processing

The following Matlab sequence was applied to images of the laser beam and phantom marker in order to find the distance between the center of these objects. This distance used as a measure of alignment accuracy and precision.

```
Read the image file
Threshold the image
Trace the boundary by initializing a point in the image
Fit a circle to the boundary
```

Figure 13. Matlab Image Processing Sequence

After the thresholding we have a binary image which has pixels valued "0" for the background and "1" for the

objects in the image. Tracing the boundary is done by the "bwtraceboundary" matlab image processing toolbox function. In principal, it detects the location of the pixels where pixel value switches from "0" to "1". Fitting a circle to the boundary is implemented by applying the least square problem solution to the basic equation of a circle. (Please see Appendix E for the Matlab program and least square problem solution for fitting a circle)

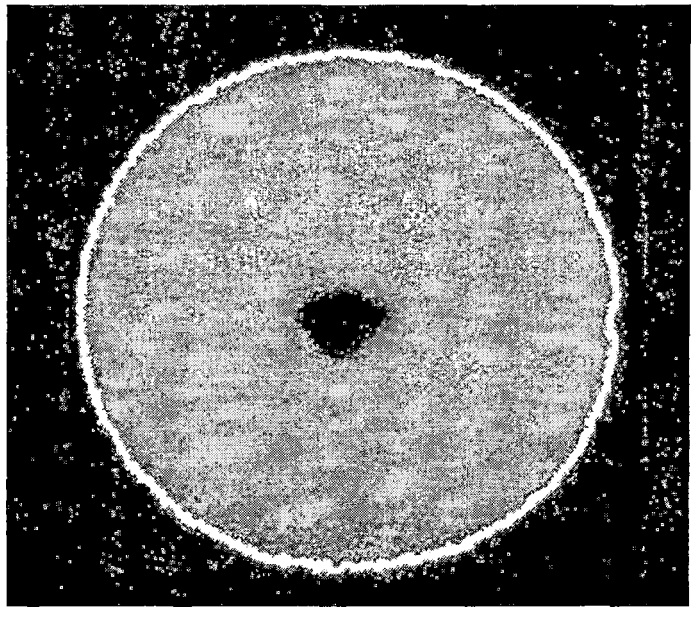


Figure 14. Estimate Bordering

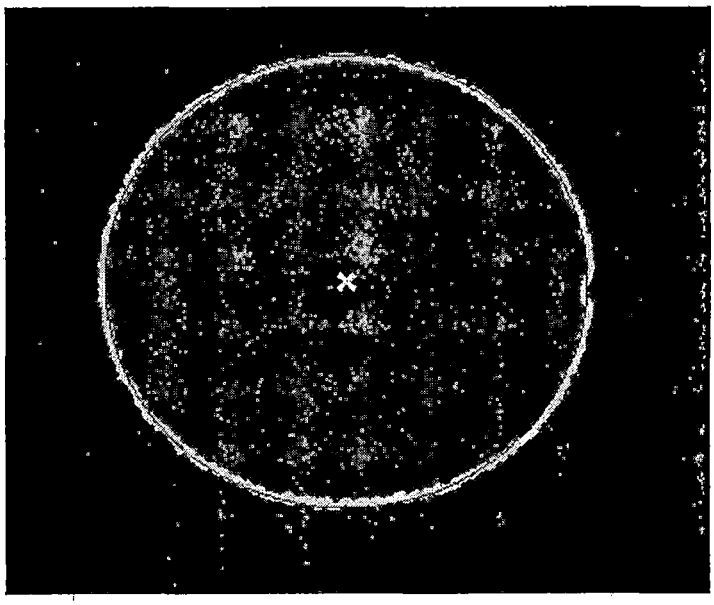


Figure 15. Fit a Circle and Finding the Center

4.3 Experimental Procedure

The phantom base (shown in Figure 16) spherical marker is aligned with the central beam axis of the treatment cone by taking a digital image with a digital camera. For the prototype purposes, the proton beam is assumed to be a laser beam expanded to a circular spot of 10 mm diameter.

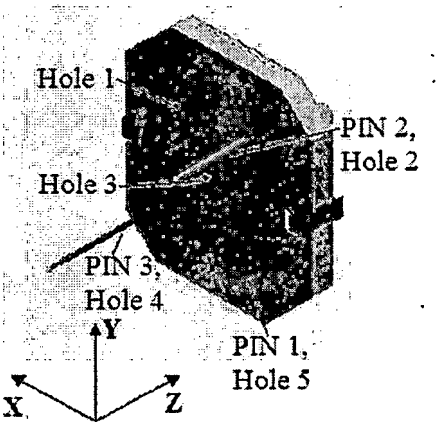


Figure 16. The Phantom Base

The laser beam shoots the spherical marker and the shooting instantaneous is projected to a flat surface. The projection occurs as shown in Figure 14. The Matlab Image Processing Sequence is applied to the digital image to find the offset between the circles. This sequence and alignment is repeated until the satisfactory offset is achieved.

Next, the camera system captures the coordinates for the markers and these coordinates are the input data for the coordinate transformation. After the transformation, the distance of the phantom base marker to the central beam axis is calculated and thus, the alignment error is found. Repeating this test for the same marker position

several times gives the alignment accuracy (mean) and precision(standard deviation).

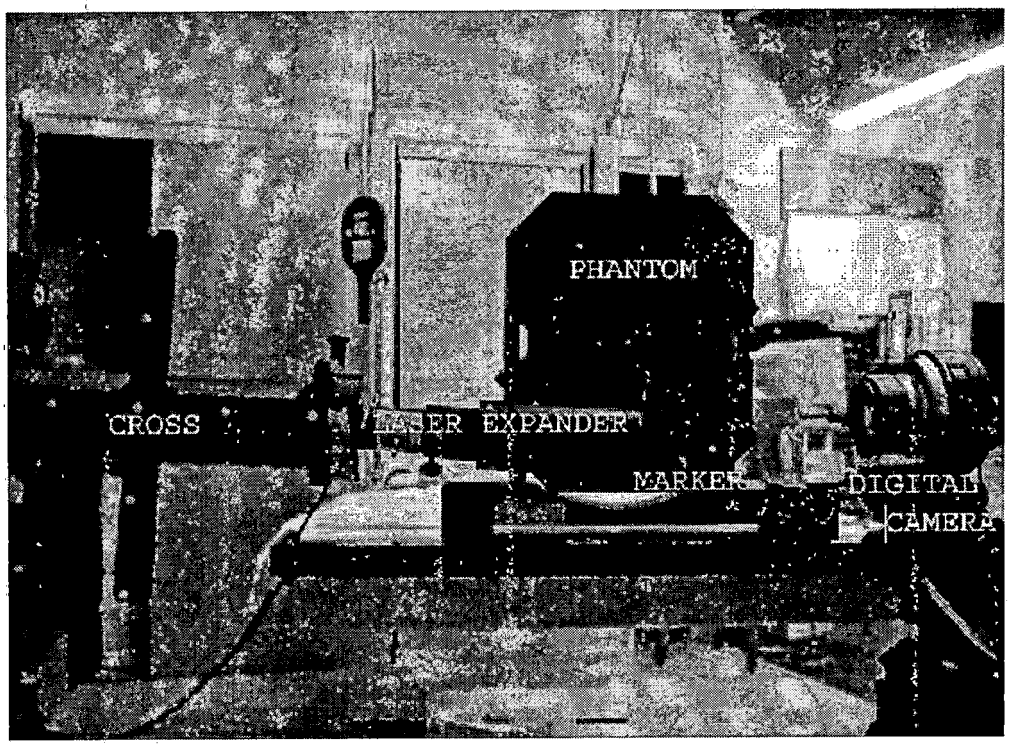


Figure 17. Experimental Setup for Aligning Expanded Laser Beam to Phantom Marker

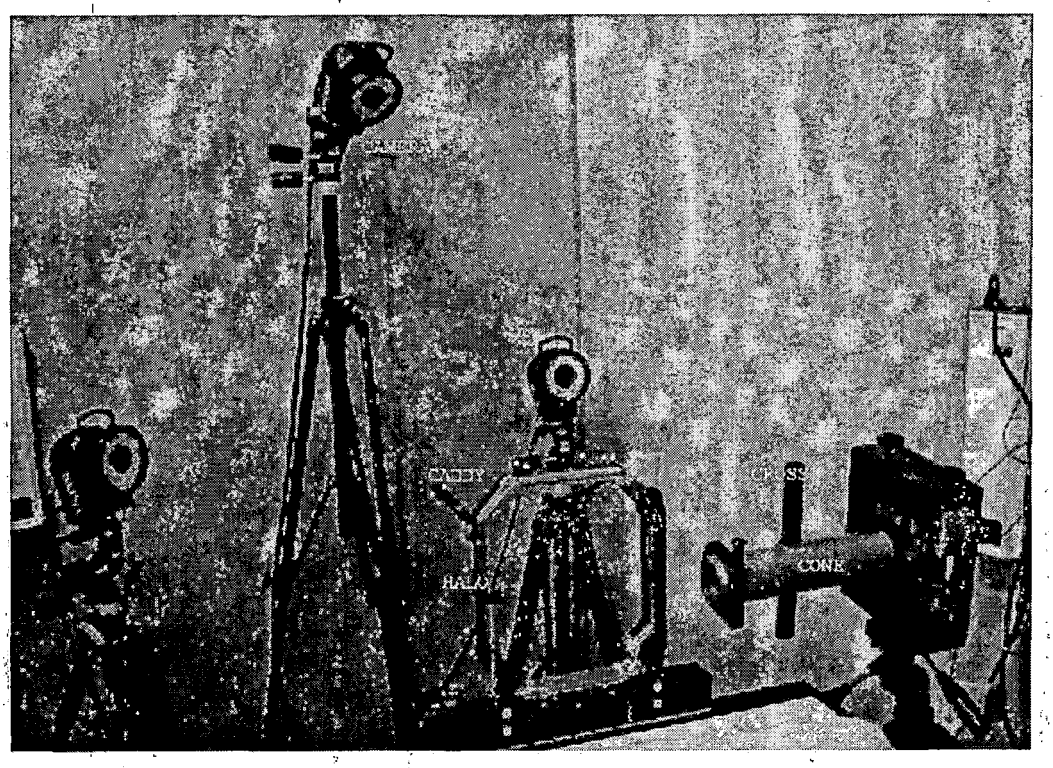


Figure 18. Experimental Setup for Measuring the Alignment Error

4.4 Data Analysis

Using the captured marker data, the coordinate transformations between the two local coordinate systems (halo and cross) and the global coordinate system (Vicon) were calculated using (1) the unitary transformation algorithm and (2) the least squares algorithm. For the first algorithm, the markers that composes the two triangles for the caddy are; 14, 21, 8 and 22, 6, 9; and for the cross are 2, 7, 4 and 3, 6, 5 (see Figure 9).

After the coordinate transformations between the local systems and the global system were established, the local coordinates of the target and the central beam axis were transformed into global coordinates. Next the shortest distance vector between target and beam axis was calculated as well as its length. The distance vector was then projected back into the local cone system to show the vertical and horizontal target point offsets, respectively, in the plane perpendicular to the laser beam.

Mean, standard deviation, and range of all distance results were calculated using standard statistical methods.

CHAPTER FIVE

RESULTS AND DISCUSSION

5.1 Results

5.1.1 Distance Comparison

Table 1 and Table 2 are typical distance error data sets obtained by subtracting the distances between markers derived from Vicon camera and dimensional inspection lab (DIL) measurements for the caddy and the cross, respectively. The largest error was 0.27 mm for the caddy and 1.03 mm for the cross. The mean error and standard deviation for the caddy marker set were -0.10 mm and 0.10 mm, respectively, and -0.30 mm and 0.37 mm, respectively, for the cross data set. The slightly larger distance error for the cross can be explained by the fact that the camera views were centered on the caddy, while the cross was captured in the periphery.

Table 1. Differences in mm Between Measured (DIL) and Observed Marker Distances (Vicon) of the Marker Caddy

Marker	1	2	3	4	5	6
1	0.00					
2	0.03	0.00				
3	-0.21	-0.25	0.00			
4	-0.12	-0.06	-0.22	0.00		
5	-0.25	-0.21	-0.07	-0.23	0.00	
6	-0.15	-0.10	0.04	-0.18	-0.27	0.00

Table 2. Differences in mm Between Measured and Observed Marker Distances of the Cross

Marker	1	2	3	4	5	6
1	0.00					
2	-0.63	0.00				
3	-0.71	-0.09	0.00			
4	-1.03	-0.40	-0.32	0.00		
5	-0.35	0.27	-0.51	-0.67	0.00	
6	-0.15	0.26	-0.61	-0.78	-0.02	0.00

5.1.2 Alignment Accuracy and Precision

Of the 15 performance tests 12 data sets gave useful data. In the remaining data sets the alignment errors obtained both with the unitary transformation and the least-squares (LS) transformation were larger than 50 mm, probably due to a systematic error in the data capture (calibration error). These data sets were excluded from further consideration.

The alignment errors obtained with the unitary and LS transformations are shown in Tables 3-5. Also shown are the sum of squared residuals for the caddy and cone transformations. The true target offset is assumed to be very close to zero, so that the distances derived from the camera measurements can be considered as the true global alignment error of the SAPVS.

For the unitary transformation the alignment error was 2.8 ± 2.0 mm (mean ± standard deviation), ranging from 0.5 to 5.5 mm. For the LS transformation the alignment error was 60.8 ± 33.4 mm, ranging from 8.9 to 127.9 mm.

Table 3. Distances Between Target and Beam Center Obtained with the Unitary Coordinate Transformation

Date	Hole/Pin	Run #	PPS angle	SSRcaddy	SSRcone	Dx	dy	d
5/4/2004	3/1	1	0	5.01	1.73	-0.23	0.47	0.53
5/4/2004	3/3	1	0	3.78	1.72	-1.22	0.65	1.38
5/4/2004	5/1	2	0	0.94	1.27	-0.10	0.97	0.97
5/4/2004	5/1	3	0	0.78	1.04	4.07	3.29	5.23
4/30/2004	5/2	1	0	1.86	0.95	-1.89	0.64	2.00
5/4/2004	5/2	3	0	1.34	1.00	4.04	2.71	4.86
5/5/2004	5/2	4	0	29.99	1.00	-3.11	4.26	5.27
5/5/2004	5/2	5	40	3.98	0.79	2.14	3.17	3.82
5/5/2004	5/2	6	40	0.40	1.50	1.53	1.12	1.90
5/4/2004	5/3	1	0	3.82	1.66	-0.38	0.50	0.63
5/4/2004	5/3	2	0	1.54	1.32	0.48	1.15	1.24
5/4/2004	5/3	3	0	1.76	1.00	3.82	3.95	5.49

The following Table illustrates the summary statistics for the UT and LS transformation.

Table 4. Summary Statistics for the Unitary Transformation and the Least Square Transformation

UT transformation					Least Square Transformation				
	Mean	SD	Range			Mean	SD	Range	
dx (mm)	0.76	2.38	-3.11	4.07	d (mm)	60.83	33.35	8.87	127.90
dy (mm)	1.91	1.45	0.47	4.26					
d (mm)	2.78	1.99	0.53	5.49					

Table 5. Distances Between Target and Beam Center Obtained with the Least-Squares Based Coordinate Transformation

Date	Hole/Pin	Run #	PPS angle	SSRcaddy	SSRcone	d
5/4/2004	3/1	1	0	0.11	0.12	77.56
5/4/2004	3/3	1	0	0.15	0.13	79.41
5/4/2004	5/1	2	0	0.22	0.14	78.08
5/4/2004	5/1	3	0	0.14	0.10	34.55
4/30/2004	5/2	1	0	0.46	0.17	77.50
5/4/2004	5/2	3	0	0.25	0.10	30.53
5/5/2004	5/2	4	0	0.02	0.28	127.90
5/5/2004	5/2	5	40	0.22	0.16	30.73
5/5/2004	5/2	6	40	0.05	0.11	8.87
5/4/2004	5/3	1	0	0.22	0.11	78.07
5/4/2004	5/3	2	0	0.27	0.15	74.71
5/4/2004	5/3	3	0	0.20	0.11	32.02

Table 6. Comparison Between Offsets Obtained with the two Transformation Methods

Hole/Pin	Run #	PPS angle	d LS(mm)	d UT (mm)
3/1	1	0	77.56	0.53
3/3	1	0	79.41	1.38
5/1	2	0	78.08	0.97
5/1	3	0	34.55	5.23
5/2	1	0	77.50	2.00
5/2	3	0	30.53	4.86
5/2	4	0	127.90	5.27
5/2	5	40	30.73	3.82
5/2	6	40	8.87	1.90
5/3	1	0	78.07	0.63
5/3	2	0	74.71	1.24
5/3	3	0	32.02	5.49

Figure 19 shows the distribution of the alignment errors in the plane perpendicular to the laser beam axis. Data points corresponding to the same phantom target location but different experimental runs, are marked with the same symbols. The data is scattered around the central beam axis in x direction and appears to be a systematic offset in the positive y direction pointing to a systematic measurement error in this direction. It is possible that this error is due to different reference systems used by the DIL lab when measuring the halo coordinates of the caddy and phantom base markers, respectively. This needs to be verified. The mean dx and dy errors (± standard deviation) were 0.8 ± 2.4 mm and 1.9 ± 1.5 mm, respectively.

The data shown in Figure 19 correspond to those of in Table 3 which were obtained with the unitary coordinate transformation. Each symbol in Figure 19 corresponds to one hole/pin combination.

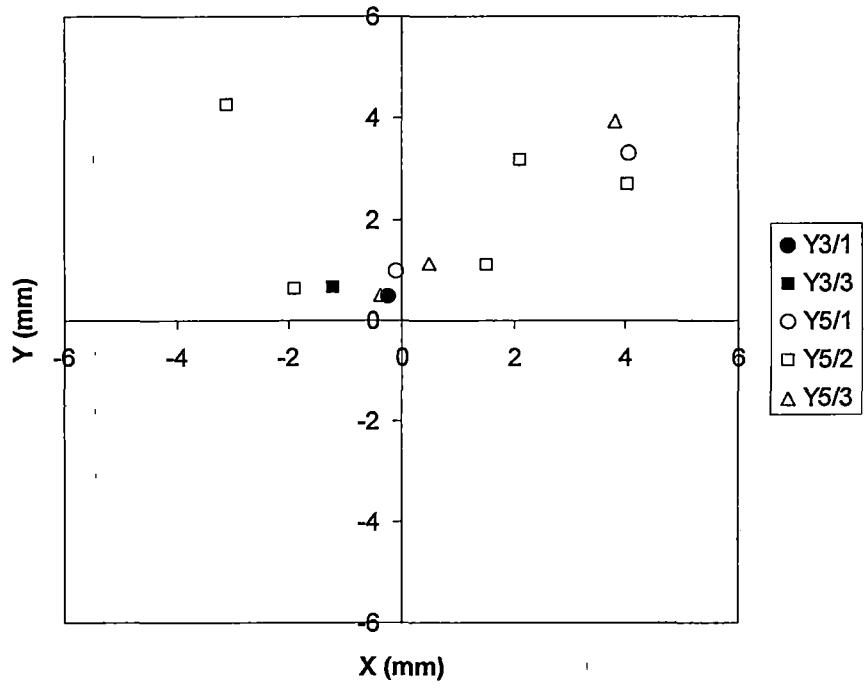


Figure 19. Offset Between Expected and Measured Target Position Assuming Perfect Alignment Between Target and Beam Center

5.2 Discussions and Conclusions

The results clearly show that the unitary transformation should be used rather than the least square based transformation, particularly for our application. The reason the unitary transformation is more accurate is the preservation of length between objects and points even for larger distances. Even though, the least square residuals are small, at the end, it introduces a large error which is of the order of centimeters rather than the desired, which

is the order millimeters. One can explain this by the relatively large distance between the two local reference systems used in our setup (approximately 500mm).

The least square transformation minimizes the residual differences between individual marker coordinates when transformed from the local to the global coordinate system, at the expense of the geometric accuracy of the transformation. On the other hand, since the unitary transformation preserves the length, the distance between the target point and the beam axis is maintained at all times.

The results shown in this thesis do not meet the specifications defined by the clinical application which demands submillimeter accuracy. One of the major limitations was the need of the laser expander for verification purposes. It would be better to place the laser expander inside the cone. In addition, not all markers available were utilized. By adding additional markers, the alignment error can be expected to decrease inversely with $\sqrt{n}$, where n is the number of available markers. Finally, the precision of the SAPVS can be improved by performing a better camera calibration. Only if

the calibration residuals are less than 1 millimeter, one should proceed to use the system.

APPENDIX A

SERIAL COMMUNICATION IMPLEMENTATION

```
/*
FILENAME            CSerialPort.cpp
PURPOSE     This class can read, write and watch one
            serial port.It sends messages to its owner
            when something happends on the port.   The
            class creates a thread for reading and writing
            so the main program is not blocked.

CREATION
DATE        2-01-04

AUTHOR      Veysi Malkoc

*/

#include "stdafx.h"
#include "SerialPort.h"

#include <assert.h>

//
// Constructor
//
CSerialPort::CSerialPort()
{
     m_hComm = NULL;

     // initialize overlapped structure members to zero
     m_ov.Offset = 0;
     m_ov.OffsetHigh = 0;

     // create events
     m_ov.hEvent = NULL;
     m_hWriteEvent = NULL;
     m_hShutdownEvent = NULL;

     m_szWriteBuffer.Empty();

     m_bThreadAlive = FALSE;
}

//
// Delete dynamic memory
//
CSerialPort::~CSerialPort()
```

```
{
     do
     {
          SetEvent(m_hShutdownEvent);
     } while (m_bThreadAlive);

     TRACE("Thread ended\n");

}

//
// Initialize the port. This can be port 1 to 4.
//
BOOL CSerialPort::InitPort(CWnd* pPortOwner, // the owner
(CWnd) of the port (receives message)
                           UINT  portnr,         //
portnumber (1..4)
                           UINT  baud,           //
baudrate
                           char  parity,         //
parity
                           UINT  databits,       //
databits
                           UINT  stopbits,       //
stopbits
                           DWORD dwCommEvents,   //
EV_RXCHAR, EV_CTS etc
                           UINT  writebuffersize)
     // size to the writebuffer
{
     assert(portnr > 0 && portnr < 5);
     assert(pPortOwner != NULL);

     // if the thread is alive: Kill
     if (m_bThreadAlive)
     {
          do
          {
               SetEvent(m_hShutdownEvent);
          } while (m_bThreadAlive);
          TRACE("Thread ended\n");
     }

     // create events
     if (m_ov.hEvent != NULL)
```

```
        ResetEvent(m_ov.hEvent);
    m_ov.hEvent = CreateEvent(NULL, TRUE, FALSE, NULL);

    if (m_hWriteEvent != NULL)
        ResetEvent(m_hWriteEvent);
    m_hWriteEvent = CreateEvent(NULL, TRUE, FALSE,
NULL);

    if (m_hShutdownEvent != NULL)
        ResetEvent(m_hShutdownEvent);
    m_hShutdownEvent = CreateEvent(NULL, TRUE, FALSE,
NULL);

    // initialize the event objects
    m_hEventArray[0] = m_hShutdownEvent;    // highest
priority
    m_hEventArray[1] = m_ov.hEvent;
    m_hEventArray[2] = m_hWriteEvent;

    // initialize critical section
    InitializeCriticalSection(&m_csCommunicationSync);

    // set buffersize for writing and save the owner
    m_pOwner = pPortOwner;


    m_szWriteBuffer.Empty();

    m_nPortNr = portnr;

    m_dwCommEvents = dwCommEvents;

    BOOL bResult = FALSE;
    char *szPort = new char[50];
    char *szBaud = new char[50];

    // now it critical!
    EnterCriticalSection(&m_csCommunicationSync);

    // if the port is already opened: close it
    if (m_hComm != NULL)
    {
        CloseHandle(m_hComm);
        m_hComm = NULL;
    }
```

```
     // prepare port strings
     sprintf(szPort, "COM%d", portnr);
     sprintf(szBaud, "baud=%d parity=%c data=%d stop=%d",
baud, parity, databits, stopbits);

     // get a handle to the port
     m_hComm = CreateFile(szPort,
     // communication port string (COMX)
                          GENERIC_READ |
GENERIC_WRITE, // read/write types
                          0,
               // comm devices must be opened with
exclusive access
                          NULL,
               // no security attributes
                          OPEN_EXISTING,
          // comm devices must use OPEN_EXISTING
                          FILE_FLAG_OVERLAPPED,
          // Async I/O
                          0);
          // template must be 0 for comm devices

     if (m_hComm == INVALID_HANDLE_VALUE)
     {
          // port not found
          delete [] szPort;
          delete [] szBaud;

          return FALSE;
     }

     // set the timeout values
     m_CommTimeouts.ReadIntervalTimeout = 1000;
     m_CommTimeouts.ReadTotalTimeoutMultiplier = 1000;
     m_CommTimeouts.ReadTotalTimeoutConstant = 1000;
     m_CommTimeouts.WriteTotalTimeoutMultiplier = 1000;
     m_CommTimeouts.WriteTotalTimeoutConstant = 1000;

     // configure
     if (SetCommTimeouts(m_hComm, &m_CommTimeouts))
     {
          if (SetCommMask(m_hComm, dwCommEvents))
          {
               if (GetCommState(m_hComm, &m_dcb))
```

```
                {
                    m_dcb.fRtsControl =
RTS_CONTROL_ENABLE;      // set RTS bit high!
                    if (BuildCommDCB(szBaud, &m_dcb))
                    {
                        if (SetCommState(m_hComm,
&m_dcb))
                            ; // normal operation...
continue
                        else

    ProcessErrorMessage("SetCommState()");
                    }
                    else

    ProcessErrorMessage("BuildCommDCB()");
                }
                else

    ProcessErrorMessage("GetCommState()");
            }
            else
                ProcessErrorMessage("SetCommMask()");
    }
    else
        ProcessErrorMessage("SetCommTimeouts()");

    delete [] szPort;
    delete [] szBaud;

    // flush the port
    PurgeComm(m_hComm, PURGE_RXCLEAR);

    // release critical section
    LeaveCriticalSection(&m_csCommunicationSync);

    TRACE("Initialisation for communicationport %d
completed.\nUse Startmonitor to communicate.\n", portnr);

    return TRUE;
}

//
//  The CommThread Function.
//
```

```
UINT CSerialPort::CommThread(LPVOID pParam)
{
     // Cast the void pointer passed to the thread back
to
     // a pointer of CSerialPort class
     CSerialPort *port = (CSerialPort*)pParam;

     // Set the status variable in the dialog class to
     // TRUE to indicate the thread is running.
     port->m_bThreadAlive = TRUE;

     // Misc. variables
     DWORD BytesTransfered = 0;
     DWORD Event = 0;
     DWORD CommEvent = 0;
     DWORD dwError = 0;
     COMSTAT comstat;
     BOOL  bResult = TRUE;

     // Clear comm buffers at startup
     if (port->m_hComm)        // check if the port is
opened
          PurgeComm(port->m_hComm, PURGE_RXCLEAR |
PURGE_TXCLEAR | PURGE_RXABORT | PURGE_TXABORT);

     // begin forever loop.  This loop will run as long
as the thread is alive.
     for (;;)
     {

          // Make a call to WaitCommEvent().  This call
will return immediatly
          // because our port was created as an async
port (FILE_FLAG_OVERLAPPED
          // and an m_OverlappedStructerlapped structure
specified).  This call will cause the
          // m_OverlappedStructerlapped element
m_OverlappedStruct.hEvent, which is part of the
m_hEventArray to
          // be placed in a non-signeled state if there
are no bytes available to be read,
          // or to a signeled state if there are bytes
available.  If this event handle
          // is set to the non-signeled state, it will be
set to signeled when a
```

```
        // character arrives at the port.

        // we do this for each port!

        bResult = WaitCommEvent(port->m_hComm, &Event,
&port->m_ov);

        if (!bResult)
        {
             // If WaitCommEvent() returns FALSE,
process the last error to determin
             // the reason..
             switch (dwError = GetLastError())
             {
             case ERROR_IO_PENDING:
                  {
                       // This is a normal return value
if there are no bytes
                       // to read at the port.
                       // Do nothing and continue
                       break;
                  }
             default:
                  {
                       // All other error codes
indicate a serious error has
                       // occured.  Process this error.
                       port-
>ProcessErrorMessage("WaitCommEvent()");
                       break;
                  }
             }
        }
        else
        {
             // If WaitCommEvent() returns TRUE, check
to be sure there are
             // actually bytes in the buffer to read.


             bResult = ClearCommError(port->m_hComm,
&dwError, &comstat);

             if (comstat.cbInQue == 0)
                  continue;
```

```
        }    // end if bResult

        // Main wait function.  This function will
normally block the thread
        // until one of nine events occur that require
action.
        Event = WaitForMultipleObjects(3, port-
>m_hEventArray, FALSE, INFINITE);

        switch (Event)
        {
        case 0:
             {
                  // Shutdown event.  This is event
zero so it will be
                  // the higest priority and be
serviced first.

                  port->m_bThreadAlive = FALSE;

                  // Kill this thread.  break is not
needed, but makes me feel better.
                  AfxEndThread(100);
                  break;
             }
        case 1:   // read event
             {
                  GetCommMask(port->m_hComm,
&CommEvent);
                  if (CommEvent & EV_CTS)
                       ::SendMessage(port->m_pOwner-
>m_hWnd, WM_COMM_CTS_DETECTED, (WPARAM) 0, (LPARAM) port-
>m_nPortNr);

                  if (CommEvent & EV_RXCHAR)
                       // Receive character event from
port.
                       ReceiveChar(port, comstat);

                  break;
             }
        case 2: // write event
             {
                  // Write character event from port
                  WriteChar(port);
```

```
                    break;
                }

            } // end switch

        } // close forever loop

        return 0;
}

//
// start comm watching
//
BOOL CSerialPort::StartMonitoring()
{
        if (!(m_Thread = AfxBeginThread(CommThread, this)))
            return FALSE;
        TRACE("Thread started\n");
        return TRUE;
}

//
// Restart the comm thread
//
BOOL CSerialPort::RestartMonitoring()
{
        TRACE("Thread resumed\n");
        m_Thread->ResumeThread();
        return TRUE;
}


//
// Suspend the comm thread
//
BOOL CSerialPort::StopMonitoring()
{
        TRACE("Thread suspended\n");
        m_Thread->SuspendThread();
        return TRUE;
}


//
// If there is a error, give the right message
```

```
//
void CSerialPort::ProcessErrorMessage(char* ErrorText)
{
     char *Temp = new char[200];

     LPVOID lpMsgBuf;

     FormatMessage(
          FORMAT_MESSAGE_ALLOCATE_BUFFER |
FORMAT_MESSAGE_FROM_SYSTEM,
          NULL,
          GetLastError(),
          MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT), //
Default language
          (LPTSTR) &lpMsgBuf,
          0,
          NULL
     );

     sprintf(Temp, "WARNING:  %s Failed with the
following error: \n%s\nPort: %d\n", (char*)ErrorText,
lpMsgBuf, m_nPortNr);
     MessageBox(NULL, Temp, "Application Error",
MB_ICONSTOP);

     LocalFree(lpMsgBuf);
     delete[] Temp;
}

//
// Write a character.
//
void CSerialPort::WriteChar(CSerialPort* port)
{
     BOOL bWrite = TRUE;
     BOOL bResult = TRUE;

     DWORD BytesSent = 0;

     LPTSTR ptrBuffer;
     DWORD bufferLen;

     ResetEvent(port->m_hWriteEvent);

     // Gain ownership of the critical section
```

```
	EnterCriticalSection(&port->m_csCommunicationSync);

	if (bWrite)
	{
		// Initailize variables
		port->m_ov.Offset = 0;
		port->m_ov.OffsetHigh = 0;

		// Clear buffer
		PurgeComm(port->m_hComm, PURGE_RXCLEAR);

		bufferLen = port->GetWriteBufferSize();
		ptrBuffer = port-
>m_szWriteBuffer.GetBuffer(bufferLen);

		bResult = WriteFile(port->m_hComm,
				// Handle to COMM Port
					ptrBuffer,
			// Pointer to message buffer in calling
finction
					bufferLen,	//
Length of message to send
					&BytesSent,
				// Where to store the
number of bytes sent
					&port->m_ov);
			// Overlapped structure

		// deal with any error codes
		if (!bResult)
		{
			DWORD dwError = GetLastError();
			switch (dwError)
			{
				case ERROR_IO_PENDING:
					{
						// continue to
GetOverlappedResults()
						BytesSent = 0;
						bWrite = FALSE;
						break;
					}
				default:
					{
						// all other error codes
```

```
                                port-
>ProcessErrorMessage("WriteFile()");
                           }
                }
          }
          else
          {
               LeaveCriticalSection(&port-
>m_csCommunicationSync);
          }
     } // end if(bWrite)

     if (!bWrite)
     {
          bWrite = TRUE;

          bResult = GetOverlappedResult(port->m_hComm, //
Handle to COMM port
                                                  &port-
>m_ov,          // Overlapped structure

&BytesSent,          // Stores number of bytes sent
                                                  TRUE);
          // Wait flag

          LeaveCriticalSection(&port-
>m_csCommunicationSync);

          // deal with the error code
          if (!bResult)
          {
               port-
>ProcessErrorMessage("GetOverlappedResults() in
WriteFile()");
          }
     } // end if (!bWrite)

     // Verify that the data size send equals what we
tried to send
     if (BytesSent != (DWORD)port-
>m_szWriteBuffer.GetLength())
     {
          TRACE("WARNING: WriteFile() error.. Bytes Sent:
%d; Message Length: %d\n", BytesSent, port-
>m_szWriteBuffer.GetLength());
```

```
    }
}

//
// Character received. Inform the owner
//
void CSerialPort::ReceiveChar(CSerialPort* port, COMSTAT
comstat)
{
    BOOL  bRead = TRUE;
    BOOL  bResult = TRUE;
    DWORD dwError = 0;
    DWORD BytesRead = 0;
    // unsigned char RXBuff;
    unsigned char RXBuff[2];

    for (;;)
    {
        // Gain ownership of the comm port critical
section.
        // This process guarantees no other part of
this program
        // is using the port object.

        EnterCriticalSection(&port-
>m_csCommunicationSync);

        // ClearCommError() will update the COMSTAT
structure and
        // clear any other errors.

        bResult = ClearCommError(port->m_hComm,
&dwError, &comstat);

        LeaveCriticalSection(&port-
>m_csCommunicationSync);



        if (comstat.cbInQue == 0)
        {
            // break out when all bytes have been read
            break;
        }
```

```
          EnterCriticalSection(&port-
>m_csCommunicationSync);

          if (bRead)
          {
               bResult = ReadFile(port->m_hComm,        //
Handle to COMM port
                                        &RXBuff,
          // RX Buffer Pointer
                                        2,
          // Read byte
                                        &BytesRead,
     // Stores number of bytes read
                                        &port->m_ov);
     // pointer to the m_ov structure
               // deal with the error code
               if (!bResult)
               {
                    switch (dwError = GetLastError())
                    {
                         case ERROR_IO_PENDING:
                              {
                                   // asynchronous i/o is
still in progress
                                   // Proceed on to
GetOverlappedResults();
                                   bRead = FALSE;
                                   break;
                              }
                         default:
                              {
                                   // Another error has
occured.  Process this error.
                                   port-
>ProcessErrorMessage("ReadFile()");
                                   break;
                              }
                    }
               }
               else
               {
                    // ReadFile() returned complete. It
is not necessary to call GetOverlappedResults()
                    bRead = TRUE;
               }
```

```
            }  // close if (bRead)

            if (!bRead)
            {
                  bRead = TRUE;
                  bResult = GetOverlappedResult(port-
>m_hComm, // Handle to COMM port

&port->m_ov,          // Overlapped structure

&BytesRead,           // Stores number of bytes read

TRUE);                // Wait flag

                  // deal with the error code
                  if (!bResult)
                  {
                        port-
>ProcessErrorMessage("GetOverlappedResults() in
ReadFile()");
                  }
            }  // close if (!bRead)

            LeaveCriticalSection(&port-
>m_csCommunicationSync);

            // notify parent that a byte was received
            ::SendMessage((port->m_pOwner)->m_hWnd,
WM_COMM_RXCHAR, (WPARAM) RXBuff, (LPARAM) port-
>m_nPortNr);
      } // end forever loop

}

//
// Write a string to the port
//
void CSerialPort::WriteToPort(char* string, DWORD length)
{
      LPTSTR buf;
      assert(m_hComm != 0);


      buf = m_szWriteBuffer.GetBuffer(length);
      for (DWORD i = 0; i < length; i++) {
```

```
            buf[i] = string[i];
        }
        m_szWriteBuffer.ReleaseBuffer();
        m_nWriteBufferSize = length;

        // set event for write
        SetEvent(m_hWriteEvent);
}

//
// Return the device control block
//
DCB CSerialPort::GetDCB()
{
        return m_dcb;
}

//
// Return the communication event masks
//
DWORD CSerialPort::GetCommEvents()
{
        return m_dwCommEvents;
}

//
// Return the output buffer size
//
DWORD CSerialPort::GetWriteBufferSize()
{
        return m_nWriteBufferSize;
}
```

APPENDIX B

MATHEMATICAL METHOD TO COMPUTE THE TRANSFORMATION BETWEEN LOCAL AND GLOBAL COORDINATE SYSTEMS

1.1 Unitary Transformation

In the following discussion, the superscript (g) indicates global coordinates and the superscript (l) indicates local coordinates. In general, the coordinates $p_{1,i}^{(l)}$, $p_{2,i}^{(l)}$, $p_{3,i}^{(l)}$ (i = 1-3) of three distinct markers in the local system will also be known in the global system, where they are called $p_{1,i}^{(g)}$, $p_{2,i}^{(g)}$, $p_{3,i}^{(g)}$. All coordinate systems considered here are right-handed.

Consider the triangle $P_1^{(l)}$, $P_2^{(l)}$, $P_3^{(l)}$ in the local coordinate system, which is formed by the three known markers (Figure 1). Let $p_1^{(l)}$, $p_2^{(l)}$, and $p_3^{(l)}$, denote the position vectors pointing from the origin of the local reference system to the central point of each marker. Note that lower-case bold letters are used here to denote vectors, and upper-case bold letters to denote matrices. The corresponding position vectors to the triangle $P_1^{(g)}$, $P_2^{(g)}$, $P_3^{(g)}$ in the global reference system are called $p_1^{(g)}$, $p_2^{(g)}$, and $p_3^{(g)}$. One may obtain the clearest perception of the rotations and translation involved in the coordinate

transformation between the two reference system by assuming that the origins and axes of both coordinate systems coincide, and that the vectors $p_1^{(l)}$, $p_2^{(l)}$, $p_3^{(l)}$

and $p_1^{(g)}$, $p_2^{(g)}$, $p_3^{(g)}$ represent two different marker sets. Then, the task to find a coordinate transformation between the two coordinate systems is identical to finding the transformation that maps the local marker set onto the global marker set.

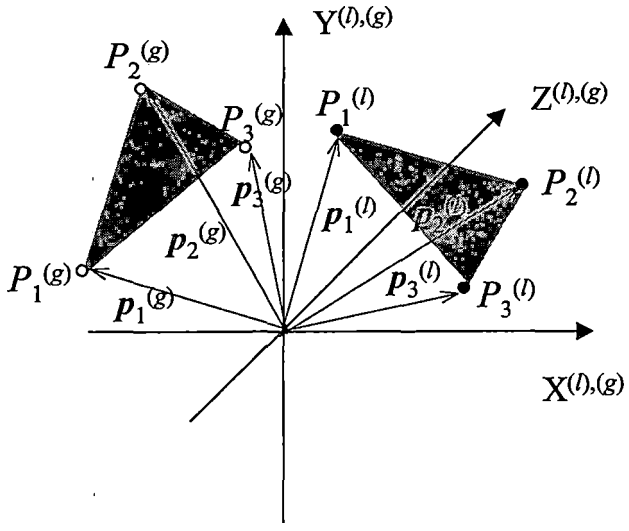


Figure 1. Conceptual view of the two marker sets in the local and global reference systems.

In general, the transformation equation, which maps corresponding l points onto g points, can be expressed as follows:

$$p_n^{(g)} = M_B \cdot M_A \cdot p_n^{(1)} + t \quad (n = 1 - 3)$$

where M_A and M_B are 3 x 3 matrices representing proper rotations. The matrix M_A corresponds to a rotation that makes the plane formed by the l marker set parallel to the plane formed by the g marker set. The matrix M_B corresponds to an "in-plane" rotation, which aligns corresponding triangle sides with respect to each other. After performing these two rotations on the l triangle, the vector t corrects for the residual translational difference between l points and corresponding g points.

1.1.1 Rotation of a Vector about a Non-collinear Vector

We now derive a useful equation for the matrix describing the rotation of a vector about another non-collinear vector. Consider a unit vector v, which we want to rotate around a unit vector o by an angle ϕ to form the vector v′. Note that the angle θ between v and o is given by $\cos(\theta) = v \cdot o$. We perform this rotation in a Cartesian coordinate system formed by the three orthogonal vectors o, $p = (v \times o)/\sin(\theta)$, and $q = [o \times (v \times o)]/\sin(\theta)$, where the factor $1/\sin(\theta)$ is required to assure unit length. The rotated vector v′ can then be expressed in terms of these

three unit vectors as follows:

$$v' = (v \cdot o)\, o + \sin(\theta)\, \sin(\phi)\, p + \sin(\theta)\, \cos(\phi)\, q$$

By substituting the expressions for p and q in terms of o and v, and by taking into account that

$o \times (v \times o) = v - o\,(v \cdot o)$, we find that

$$v' = v\, \cos(\phi) + o\,(v \cdot o)\,[1 - \cos(\phi)] + (v \times o)\, \sin(\phi)$$

This equation can also be expressed in matrix form as $v' = M\, v$, where the rotation matrix M is explicitly given by

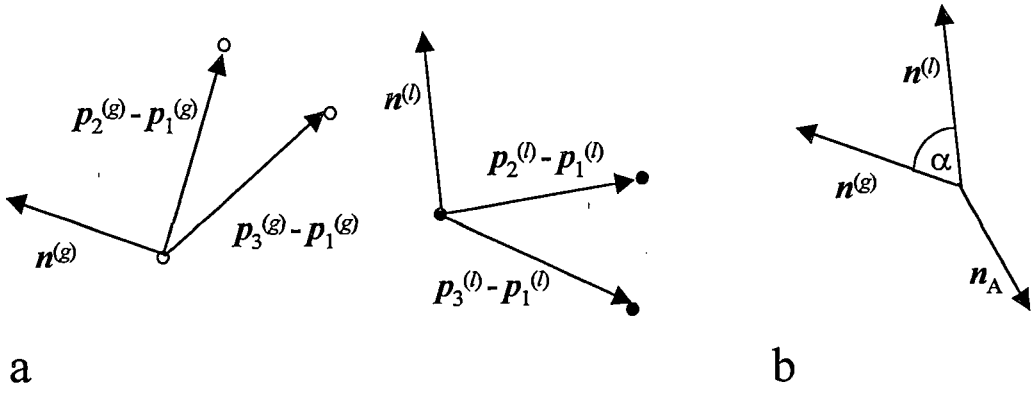
$$M = \begin{pmatrix} \cos(\Phi)+o_1^2(1-\cos(\Phi)) & o_3\sin(\Phi)+o_1o_2(1-\cos(\Phi)) & -o_2\sin(\Phi)+o_1o_3(1-\cos(\Phi)) \\ -o_3\sin(\Phi)+o_1o_2(1-\cos(\Phi)) & \cos(\Phi)+o_2^2(1-\cos(\Phi)) & -o_1\sin(\Phi)+o_2o_3(1-\cos(\Phi)) \\ o_2\sin(\Phi)+o_1o_3(1-\cos(\Phi)) & -o_1\sin(\Phi)+o_3o_2(1-\cos(\Phi)) & \cos(\Phi)+o_3^2(1-\cos(\Phi)) \end{pmatrix}$$

1.1.2 Derivation of the Matrix M_A

To find the mathematical expression for the matrix M_A, which transforms the l triangle into one that is coplanar with the g triangle, we first determine the unit normal vector of the l triangle, $n^{(l)}$, and the unit normal vector of the g triangle, $n^{(g)}$. The two unit vectors can be calculated by forming and normalizing the vector

products $(p_3^{(1)} - p_1^{(1)}) \times (p_2^{(1)} - p_1^{(1)})$ and $(p_3^{(g)} - p_1^{(g)}) \times (p_2^{(g)} - p_1^{(g)})$, respectively (Fig. 2a).

The matrix M_A corresponds to a rotation of the vector unit $n^{(1)}$ about the orthogonal vector $n_A = (n^{(1)} \times n^{(g)})$ by the angle α, where $\cos(\alpha) = n^{(1)} \cdot n^{(g)}$ (Fig. 2b).



a b

Figure 2 (a) Definition of the normal vectors $n^{(1)}$ and $n^{(g)}$, and (b) rotation performed by matrix M_A.

By normalizing the vector n_A to $o_A = n_A / \sin(\alpha)$, and by using the expression for the rotation matrix M derived above, we obtain the following expression for the matrix M_A:

$$M_A = \begin{pmatrix} \cos(\alpha)+o_{A1}^2(1-\cos(\alpha)) & n_{A3}+o_{A1}o_{A2}(1-\cos(\alpha)) & -n_{A2}+o_{A1}o_{A3}(1-\cos(\alpha)) \\ -n_{A3}+o_{A2}o_{A1}(1-\cos(\alpha)) & \cos(\alpha)+o_{A2}^2(1-\cos(\alpha)) & n_{A1}+o_{A2}o_{A3}(1-\cos(\alpha)) \\ n_{A2}+o_{A3}o_{A1}(1-\cos(\alpha)) & -n_{A1}+o_{A3}o_{A2}(1-\cos(\alpha)) & \cos(\alpha)+o_{A3}^2(1-\cos(\alpha)) \end{pmatrix}$$

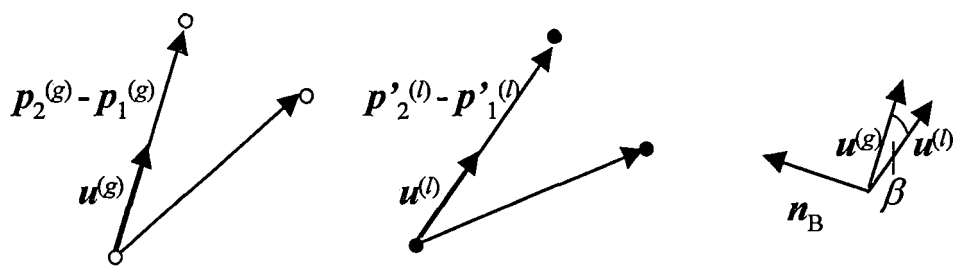
Note that in this expression the terms $o_{Ai}\sin(\alpha)$ have been replaced by n_{Ai} (i = 1-3).

1.1.3 Derivation of the Matrix M_B and the Vector t

Multiplication of the local position vectors $p_1^{(l)}$, $p_2^{(l)}$, and $p_3^{(l)}$ by matrix the M_A yields new vectors $p'_1{}^{(l)}$, $p'_2{}^{(l)}$, and $p'_3{}^{(l)}$ which form a triangle that is now coplanar with that formed by the global position vectors $p_1^{(g)}$, $p_2^{(g)}$, and $p_3^{(g)}$. To obtain the rotation matrix M_B, we normalize the triangle vectors $(p'_2{}^{(l)} - p'_1{}^{(l)})$, and $(p_2^{(g)} - p_1^{(g)})$, which yields the non-collinear unit vectors $u^{(l)}$ and $u^{(g)}$, respectively (Fig. 3a). The matrix that aligns unit vector $u^{(l)}$ with unit vector $u^{(g)}$ represents a rotation of the vector $u^{(l)}$ about the orthogonal vector $n_B = (u^{(l)} \times u^{(g)})$ by the angle β where $\cos(\beta) = (u^{(l)} \cdot u^{(g)})$ (Fig. 3b). By normalizing the vector n_B to $o_B = n_B / \sin(\beta)$ the matrix M_B can be expressed as

$$M_B = \begin{pmatrix} \cos(\beta)+o_{B1}^2(1-\cos(\beta)) & n_{B3}+o_{B1}o_{B2}(1-\cos(\beta)) & -n_{B2}+o_{B1}o_{B3}(1-\cos(\beta)) \\ -n_{B3}+o_{B2}o_{B1}(1-\cos(\beta)) & \cos(\beta)+o_{B2}^2(1-\cos(\beta)) & n_{B1}+o_{B2}o_{B3}(1-\cos(\beta)) \\ n_{B2}+o_{B3}o_{B1}(1-\cos(\beta)) & -n_{B1}+o_{B3}o_{B2}(1-\cos(\beta)) & \cos(\alpha)+o_{B3}^2(1-\cos(\beta)) \end{pmatrix}$$

Multiplication of the local position vectors $p'_1{}^{(l)}$, $p'_2{}^{(l)}$, and $p'_3{}^{(l)}$ by matrix M_B yields new vectors $p''_1{}^{(l)}$, $p''_2{}^{(l)}$, and $p''_3{}^{(l)}$, which makes the l triangle identical in orientation with respect to the g triangle. Finally we translate $p''_1{}^{(l)}$ into $p_1{}^{(g)}$ by adding the vector $t = p_1{}^{(g)} - p''_1{}^{(l)}$. If no systematic or random error is involved the triangles should now exactly superimpose.



a b

Fig. 3 (a) Definition of the normal vectors $u^{(l)}$ and $u^{(g)}$, and (b) rotation performed by matrix M_B.

The two rotations involved in the transformation can be combined into one rotation by calculating the matrix $M_{AB} = M_B \cdot M_A$. We then have

$$v^{(g)} = M_{AB} \cdot v^{(l)} + t$$

for transformations of any vector v from the local to the global coordinate system. Since the rotation matrix can be inverted, we can also transform in the opposite direction:

$$v^{(l)} = M^{-1}_{AB} \cdot (v^{(g)} - t)$$

This inverse transformation can be used to transform any vector from the global coordinate system into a local coordinate system.

1.2 Least Square Based Transformation

Least Square Problem Solution is the sum of Given is a matrix A in which the i^{th} column corresponds to the three coordinates of the i^{th} marker in the local reference system and another matrix B in which the i^{th} column corresponds to the three coordinates of the i^{th} marker in the global reference system. We search for a linear transformation, represented by a 3 × 3 matrix X, that transforms matrix A into matrix B:

$$AX = B \quad (1)$$

Provided data on at least three markers are available, in which case the matrices A and B are also 3 × 3 matrices, equation (1) can be solved exactly by multiplying both sides by the inverse matrix A^{-1}.

$$A^{-1} AX = A^{-1} B$$

$$X = A^{-1} B \quad (2)$$

In practice, however, the coordinate data have random additive errors, which means that only an approximate solution for the matrix X can be derived. The most likely solution in this case is derived from a least-squares regression procedure.

In particular, it can be shown that the LS-solution is given by

$$X = (A^T A)^{-1} A^T B \quad (3)$$

Note that in case of a exactly three markers, equation (3) becomes identical to equation (2).

1.2.1 Derivation of the Equation (3)

The least square solution for $Ax = b$ is given by;

$x_{ls} = (A^T A)^{-1} A^T b$

define $f : R^n$ as $f(x) = \| b - Ax \|^2 = \sum_{i=1}^{m} (b_i - a_i^T x)^2$ where

$$A = \begin{bmatrix} a_1^T \\ a_2^T \\ \vdots \\ a_m^T \end{bmatrix}$$

- f is a differentiable function of n variables
- LS solution x_{ls} is minimizer of f, i.e., characterized by

$$\Delta f(x_{ls}) = \begin{bmatrix} \frac{\partial f(x_{ls})}{\partial x_1} \\ \frac{\partial f(x_{ls})}{\partial x_2} \\ \vdots \\ \frac{\partial f(x_{ls})}{\partial x_3} \end{bmatrix} = 0$$

What are the partial derivatives of $f(x) = \sum_{i=1}^{m} (a_i^T x - b_i)^2$?

$$\frac{\partial f(x)}{\partial x_j} = \sum_{i=1}^{m} 2(a_i^T x - b_i) a_{ij}$$

gradient of f at x :

$$\Delta f(x) = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} \\ \frac{\partial f(x)}{\partial x_2} \\ \vdots \\ \frac{\partial f(x)}{\partial x_n} \end{bmatrix} = \sum_{i=1}^{m} 2(a_i^T x - b_i) \begin{bmatrix} a_{i1} \\ a_{i2} \\ \vdots \\ a_{in} \end{bmatrix}$$

$$= \sum_{i=1}^{m} 2(a_i^T x - b_i) a_i$$

$$= \sum_{i=1}^{m} 2(a_i a_i^T x - a_i b_i)$$

$$= 2(A^T A_x - A^T b)$$

hence $\Delta f(x) = 0 \Leftrightarrow A^T A_x = A^T b)$

$$x_{ls} = (A^T A_x)^{-1} A^T b$$

```
  /*    Coordinate    Transformation    using    Unitary
transformation

   Method.
*/

#include "stdafx.h"
#include "matlab.hpp"  // Matlab C++ Math Library
#include <conio.h>
#include "line.h"
#include <vector>
#include <fstream.h>
#include <iostream>
#include <string>

//Define a class so that It contains the transformation
matrix
class math{
public:
     Point p;
     double mabArray[9];
     double mtArray[3];
     void setValues(double x ,double y,double z);
     void addToGlobalVector(Point & p);
     void addToRawLocal(Point & p);
     void Transform();
     void TransformToLocal(Point & point);
     void TransformToGlobal(Point & point);

     std::vector<Point> global;
     std::vector<Point> rawLocal;


};

//Define a function that transforms from global to local
void math::TransformToLocal(Point & point){
     mwArray  temp1,temp2,temp3,temp4;
     Point p;
     double global[3];
     global[0] = point.getx();
     global[1] = point.gety();
```

```
    global[2] = point.getz();
    temp4 = mwArray(3,1,global);
    temp1=mwArray(3,3,mabArray);
    temp1 = inv(temp1);
    temp1=transpose(temp1);

    cout <<"MEAN MAB"<<temp1<<endl;

    temp2 =mwArray(3,1,mtArray);
    cout<<"mean t"<<temp2<<endl;
    temp3 = temp1 * (temp4-temp2);
    point.setx(temp3(1,1).Double());
    point.sety(temp3(2,1).Double());
    point.setz(temp3(3,1).Double());

}
//Define a function that transforms from local to global
void math::TransformToGlobal(Point & point){
    mwArray  temp1,temp2,temp3,temp4;
    Point p;
    double local[3];
    local[0] = point.getx();
    local[1] = point.gety();
    local[2] = point.getz();
    temp4 = mwArray(3,1,local);
    temp1=mwArray(3,3,mabArray);
    temp1=transpose(temp1);

    temp2 =mwArray(3,1,mtArray);
    temp3 = temp1 * temp4 + temp2;
    point.setx(temp3(1,1).Double());
    point.sety(temp3(2,1).Double());
    point.setz(temp3(3,1).Double());

}

void math::setValues(double x ,double y,double z) {
 p.setx(x);
    p.sety(y);
    p.setz(z);
}

//Add to global vector
void math::addToGlobalVector(Point & p) {
```

```
	global.push_back(p);
}

// Add to local vector
void math::addToRawLocal(Point & p) {
	rawLocal.push_back(p);
}

//Perform Transformation with 3 marker sets each time
void math::Transform() {
	int i=0;
	std::vector<Point>::iterator iter,iter2;
	iter = global.begin();
	iter2 = rawLocal.begin();
	mwArray MABTOTAL =0,ttotal=0;

	while(i<2) {
	mwArray l1,l2,l3,g1,g2,g3;
	double L1[3],L2[3],L3[3],G1[3],G2[3],G3[3];

	G1[0] = iter->getx();
	G1[1] = iter->gety();
	G1[2] = iter->getz();
	iter++;
	G2[0] = iter->getx();
	G2[1] = iter->gety();
	G2[2] = iter->getz();
	iter++;
	G3[0] = iter->getx();
	G3[1] = iter->gety();
	G3[2] = iter->getz();
	iter++;
	g1 = mwArray(3,1,G1);
	g2 = mwArray(3,1,G2);
	g3 = mwArray(3,1,G3);

	L1[0] = iter2->getx();
	L1[1] = iter2->gety();
	L1[2] = iter2->getz();
	iter2++;
	L2[0] = iter2->getx();
	L2[1] = iter2->gety();
	L2[2] = iter2->getz();
	iter2++;
	L3[0] = iter2->getx();
```

```
      L3[1] = iter2->gety();
      L3[2] = iter2->getz();
      l1 = mwArray(3,1,L1);
      l2 = mwArray(3,1,L2);
      l3 = mwArray(3,1,L3);
      iter2++;

      //mwArray is the Array structure defined by Matlab
      mwArray
nlocal,nglobal,nglobalunit,nlocalunit,G1test,G2test,G3tes
t;
      mwArray
nA,nB,oA,oB,temp1,temp2,temp3,L1prime,L2prime,unitglobal,
unitlocal;
      mwArray
MA,MA11,MA12,MA13,MA21,MA22,MA23,MA31,MA32,MA33;
      mwArray
MAB,t,MB,MB11,MB12,MB13,MB21,MB22,MB23,MB31,MB32,MB33;
      double     cos_alpha  =0,alpha   =0,cos_beta   =0,beta
=0,factor =0,factor2=0;

      nlocal =cross((l3-l1),(l2-l1));
      nglobal=cross((g3-g1),(g2-g1));
      nlocalunit=nlocal / norm(nlocal);
      nglobalunit=nglobal / norm(nglobal);
      temp1 = transpose(nlocal) * nglobal;
      temp2 = (norm(nlocal) * norm(nglobal));
      temp3 = temp1 / temp2 ;
      cos_alpha =temp3.Double();
      alpha = acos(cos_alpha);
      nA = cross(nlocalunit,nglobalunit);
       oA = nA / sin(alpha);
       factor = 1 -  cos_alpha ;

      MA11 =cos_alpha + oA(1,1)*oA(1,1) * factor;
      MA12=-nA(3,1)+oA(1,1)*oA(2,1)*factor;
      MA13=nA(2,1)+oA(1,1)*oA(3,1)*factor;
      MA21=nA(3,1)+oA(2,1)*oA(1,1)*factor;
      MA22=cos_alpha+oA(2,1)*oA(2,1)*factor;
      MA23=-nA(1,1)+oA(2,1)*oA(3,1)*factor;
      MA31=-nA(2,1)+oA(3,1)*oA(1,1)*factor;
      MA32=nA(1,1)+oA(3,1)*oA(2,1)*factor;
      MA33=cos_alpha+oA(3,1)*oA(3,1)*factor;
```

```
     //Place each matrix element into MA transformation
Matrix
     MA(1,1) = MA11;
     MA(1,2) = MA12;
     MA(1,3) = MA13;
     MA(2,1) = MA21;
     MA(2,2) = MA22;
     MA(2,3) = MA23;
     MA(3,1) = MA31;
     MA(3,2) = MA32;
     MA(3,3) = MA33;

     L1prime = MA * l1;
     L2prime = MA * l2;
     unitglobal = (g2-g1) / norm(g2-g1);
     unitlocal = (L2prime - L1prime) / norm(L2prime -
L1prime);
     temp2 = (transpose(unitlocal)* unitglobal ) /
(norm(unitlocal)*norm(unitglobal));
     cos_beta = temp2.Double();
     beta = acos(cos_beta);
     factor2 = 1 - cos_beta ;
     nB = cross(unitlocal,unitglobal);
     oB = nB / sin(beta);
     MB11=cos_beta+oB(1,1)*oB(1,1)*factor2;
     MB12=-nB(3,1)+oB(1,1)*oB(2,1)*factor2;
     MB13=nB(2,1)+oB(1,1)*oB(3,1)*factor2;
     MB22=cos_beta+oB(2,1)*oB(2,1)*factor2;
     MB23=-nB(1,1)+oB(2,1)*oB(3,1)*factor2;
     MB31=-nB(2,1)+oB(3,1)*oB(1,1)*factor2;
     MB32=nB(1,1)+oB(3,1)*oB(2,1)*factor2;
     MB33=cos_beta+oB(3,1)*oB(3,1)*factor2;
     MB21=nB(3,1)+oB(2,1)*oB(1,1)*factor2;

     //Place each matrix element into MB transformation
Matrix
     MB(1,1) = MB11;
     MB(1,2) = MB12;
     MB(1,3) = MB13;
     MB(2,1) = MB21;
     MB(2,2) = MB22;
     MB(2,3) = MB23;
     MB(3,1) = MB31;
     MB(3,2) = MB32;
     MB(3,3) = MB33;
```

```
      //MAB is the multiplication of MA and MB
      MAB = mtimes(MB,MA);

      //find translational matrix t
      t = g1 - MAB * l1 ;

      MABTOTAL = MAB + MABTOTAL;
      ttotal = t + ttotal;
      i++;
      }
      //Take the arithmetic Average for each matrix
element
      mabArray[0]=(MABTOTAL(1,1) / 2.0).Double();
      mabArray[1]=(MABTOTAL(1,2) / 2.0).Double();
      mabArray[2]=(MABTOTAL(1,3) / 2.0 ).Double();
      mabArray[3]=(MABTOTAL(2,1)/ 2.0 ).Double();
      mabArray[4]=(MABTOTAL(2,2)/ 2.0 ).Double();
      mabArray[5]=(MABTOTAL(2,3)/ 2.0 ).Double();
      mabArray[6]=(MABTOTAL(3,1)/ 2.0 ).Double();
      mabArray[7]=(MABTOTAL(3,2)/ 2.0 ).Double();
      mabArray[8]=(MABTOTAL(3,3)/ 2.0 ).Double();


      mtArray[0]= (ttotal(1,1)/ 2.0 ).Double();
      mtArray[1]= (ttotal(2,1)/ 2.0 ).Double();
      mtArray[2]= (ttotal(3,1)/ 2.0 ).Double();

}

int main(int argc, char* argv[])
{


      //Define six local coordinates measured by
      //Dimension Inspection Lab (DIL)
      Point      p1l(-9.666,-27.904,-187.742),p2l(-15.805,-
53.209,-187.508),
      p3l(-14.286,-122.789,-193.435),p4l(48.132,-230.938,-
193.283),
      p5l(60.618,-225.08,-186.971),p6l(124.087,-231.299,-
193.532);


      math m;
```

```
//Add DIL values to local vector
m.addToRawLocal(p1l);
m.addToRawLocal(p2l);
m.addToRawLocal(p3l);
m.addToRawLocal(p4l);
m.addToRawLocal(p5l);
m.addToRawLocal(p6l);

//Read the global (Vicon) coordinates from file
std::string filename;
std::cout <<"please enter the file name"<<"\n";
std::cin >> filename;
ifstream input;
ofstream output,output2,output3;
input.open(filename.c_str());
output.open("transform.txt");
output2.open("transformt.txt");
output3.open("output.txt");

double s;
std::vector<double>  vec;
std::vector<Point> vecPoint;
std::vector<Point>::iterator iter;

//place global coordinates into a vector
while(input >> s ) {
     vec.push_back(s);

}
for(int i=0; i<vec.size(); i++) {

     Point temp;
     temp.setx(vec[i++]);
     temp.sety(vec[i++]);
     temp.setz(vec[i]);
     m.addToGlobalVector(temp);
}
m.Transform();

m.TransformToGlobal(test);

output3 << test.getx()<<endl;
output3 << test.gety()<<endl;
output3 << test.getz()<<endl;
```

```
    for(i=0;i<9;i++) {
         output << m.mabArray[i]<<" ";
         if(i == 2 || i==5)
              output<<endl;
    }
    for(i=0;i<3;i++) {
         output2 << m.mtArray[i]<<" ";

    }

    _getch();
  return 0;
}
```

APPENDIX C

MATHEMATICAL METHOD TO COMPUTE THE SHORTEST DISTANCE

This document derives the mathematical formula for calculating the vector d.

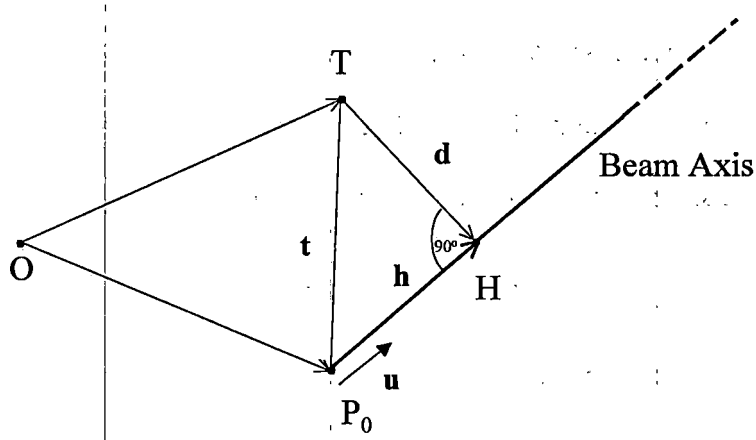


Figure 1. Geometry of the Problem

Figure 1 shows the geometry of the problem. Given are the origin O of the global coordinate system, the target point T, and the unit vector of the beam axis u. Furthermore, we know the vector t, which is defined as $t = P_0 - T$, where P_0 and T are the position vectors of the points P_0 and T from the origin O, respectively. Since vectors d and u are orthogonal, their inner product is zero:

$$d \cdot u = 0;$$

and, with $d = h + t$ and, we have

$$(h + t) \cdot u = 0$$

APPENDIX D

DISTANCE VERIFICATION ROUTINE

```
Double Marker_Array [6]  = {marker1, marker2, marker3,
marker4, marker5, marker6}
double markerDistance[6][6];
for(int i=0; i<6;i++) {
for(int j=0; j<6;j++) {
markerDistance[i][j] = distance(Marker_Array[i],
Marker_Array[j]);
output_file <<  markerDistance[i][j];  // output to file
if (j == 5)
output_file <<  endl;
}
}// end for

double distance ( Point &p1, Point &p2) {
double temp = 0;

// calculate distance between two points
temp = sqrt( pow(p1.getx() - p2.getx(),2) +

pow(p1.gety() - p2.gety(),2) +  pow(p1.getz() -
p2.getz(),2) );

return temp;

} //end distance function



//The third vector structure,contains 6 points in space
std::vector<std::vector<Point> > p(6);

Point temp[6];
for (j=0; j<= N; ++j)  {  // N: number of frames captured
by the camera
for (i=0; i<= 6; ++i) {
Read marker coordinates ;
temp[i].set(x);            // set x coordinate
temp[i].set(y);            // set y coordinate
temp[i].set(z);            // set z coordinate
} //end for

for (i=0; i<= 6; ++i) {
```

```
p[i].push_back(temp[i]);
} // end for; number of frames

double totalx = 0,avgx=0;
double totaly = 0,avgy=0;
double totalz = 0,avgz=0;

// Array that stores the points after the Arithmetic
Average
Point viconAverage[6];
for (i=0; i<= 6; ++i) {
std::vector<Point>::iterator iter;
for(iter = p[i].begin(); iter != p[i].end(); iter++) {
totalx += iter->getx();  //sum of x coordinates through N
frames
totaly += iter->gety();  //sum of y coordinates through N
frames
totalz += iter->getz();  //sum of z coordinates through N
frames
}

avgx = totalx / p[i].size();  //Arithmetic Average
avgy = totaly / p[i].size();
avgz = totalz / p[i].size();

//Since we have the average, erase all data and keep only
the average
p[i].erase(p[i].begin(),p[i].end());
Point temp;
temp.setx(avgx);
temp.sety(avgy);
temp.setz(avgz);
p[i].push_back(temp);
for(iter = p[i].begin(); iter != p[i].end(); iter++) {
viconAverage[i] = *iter;
}

totalx = 0;    //Set to zero for each marker
totaly = 0;
totalz = 0;

}// end for
```

```
//Distance of the markers among each other that is
obtained by the cameras
double viconDistance[6][6];
for(int i=0; i<6;i++) {
for(int j=0; j<6;j++) {
viconDistance[i][j] =
distance(viconAverage[i],viconAverage[j]);
output_file <<  viconDistance[i][j];  // output to file

if (j == 5)

output_file <<  endl;
}

}// end for
```

APPENDIX E

MATLAB IMAGE PROCESSING SEQUENCE

LEAST SQUARE PROBLEM SOLUTION FOR FITTING A CIRCLE

We want to find a circle that fits the given set of points best in a sense of least squares approximation. Let a circle be represented as

$$x^2+y^2+2Ax+2By+C=0$$

Then, the center of the circle is (-A,-B) and the radius is

$$r=\sqrt{A^2+B^2-C}$$

Assume this circle is used to approximate the given set of points p_i (i=1,2,...,n). Then, the squared error with respect to $p_i=(x_i,y_i)$ is $(x_i^2+y_i^2+2Ax_i+2By_i+C)^2$. Accordingly, the total squared error is given by

$$\phi=\sum (x_i^2+y_i^2+2Ax_i+2By_i+C)^2$$

We thus want to find A,B,C such that □ is minimized, which is equivalent to solving the following system of linear equations:

$$\frac{\partial \phi}{\partial A}, \frac{\partial \phi}{\partial B}, \frac{\partial \phi}{\partial C},$$

Explicitly, we need to solve

$$2\Sigma x_i^2\ A+2\Sigma x_i y_i\ B+\Sigma x_i\ C+\Sigma (x_i^2+y_i^2)\, x_i=0$$

$$2\Sigma x_i y_i\ A+2\Sigma y_i^2\ B+\Sigma y_i\ C+\Sigma (x_i^2+y_i^2)\, y_i=0$$

$$2\Sigma x_i\, A+2\Sigma y_i\, B+n\ C+\Sigma(x_i^2+y_i^2)=0$$

MATLAB SEQUNCE

```
RGB = imread('tape.png');
imshow(RGB);

I = rgb2gray(RGB);
threshold = graythresh(I);
BW = im2bw(I,threshold);
imshow(BW)
connectivity = 8;
num_points    = 180;
contour = bwtraceboundary(BW, [row, col], 'N',
connectivity, num_points);

imshow(RGB);
hold on;

plot(contour(:,2),contour(:,1),'g','LineWidth',2);


/*Rewrite basic equation for a circle:

(x-xc)^2 + (y-yc)^2 = radius^2,  where (xc,yc) is the

center  in terms of parameters a, b, c as
x^2 + y^2 + a*x + b*y + c = 0,
where a = -2*xc, b = -2*yc, and
c = xc^2 + yc^2 - radius^2 */
```

```
Solve for parameters a, b, c, and use them to calculate
the radius.

x = contour(:,2);
y = contour(:,1);

// solve for parameters a, b, and c in the least-squares
//sense by

//using the backslash operator
abc=[x y ones(length(x),1)]\[-(x.^2+y.^2)];
a = abc(1); b = abc(2); c = abc(3);

// calculate the location of the center and the radius

xc = -a/2;
yc = -b/2;
radius  =  sqrt((xc^2+yc^2)-c)

// display the calculated center

plot(xc,yc,'yx','LineWidth',2);


// plot the entire circle
theta = 0:0.01:2*pi;


// use parametric representation of the circle to obtain
coordinates  of points on the circle

Xfit = radius*cos(theta) + xc;
Yfit = radius*sin(theta) + yc;
```

APPENDIX F

SOFTWARE REQUIREMENT SPECIFICATIONS

1. Introduction

    1.1 Purpose

    1.2 Scope

    1.3 Definitions, acronyms, and abbreviations

    1.4 References

    1.5 Overview

2. Overall Description

    2.1 Product Perspective

        2.1.1 System Interfaces

        2.1.2 User Interfaces

        2.1.3 Hardware Interfaces

        2.1.4 Software Interfaces

        2.1.5 Communications Interfaces

        2.1.6 Memory Constraints

        2.1.7 Operations

        2.1.8 Site Adaptation Requirements

    2.2 Product Functions

        2.3 User Characteristics

        2.4 Constraints

        2.5 Assumptions and Dependencies

        2.6 Apportioning of Requirements

3. Specific Requirements

3.1 External Interfaces Requirements

3.2 Functions

3.3 Performance Requirements

3.5 Design Constraints

3.6 Software System Attributes

3.6.1 Reliability

3.6.2 Availability

3.6.4 Maintainability

1. Introduction:

1.1 Purpose

This document defines the requirements and design layout of a software product for the Sequential Alignment and Position Verification System For Functional Proton Radiosurgery. This document is produced with the intention of aiding software developers who are willing to work on SAPVS. The document and the software product will serve the following types of users:

- Software Developers
- Professors who will review this thesis academically

- Decision makers who will decide on the future of SAPVS

1.2 Scope

The software product defined in this document is intended to document design and implementation of the SAPVS software.

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
Matlab	A tool developed by Mathworks Inc. for software development and engineering purposes.
Vicon	The company that manufactures vicon cameras
Algorithm	A set of steps defined such that when they are followed in a specific order, a task is accomplished.
C++	An object oriented language developed by Biarne

	Stroustrup. C++ is a superset of the C programming language.
OOD	Object Oriented Design
OS	Operating System

1.4 References

IEEE Std 830-1993 IEEE Recommended Practice of Software Requirements Specification

IEEE Std 830-1998 IEEE Recommended Practice of Software Requirements Specification-Annex A

Classical And Object-Oriented Software Engineering, Stephen R. Schach

http://web.csusb.edu/public/class/cs455_1/winter99/documentation/algorithma99 /srs.html

http://web.csusb.edu/public/class/cs455_1/winter2001/2001/docs/prototype7/srs7v1.html

1.5 Overview

The remainder of this document defines the functions and specific requirements of SAPVS a format consistent with the IEEE Std 830-1998 SRS format

2. Overall Description

2.1 Product Perspective

2.1.1 System Interfaces

This product is coded in Microsoft Visual C++ and therefore it requires Microsoft Windows platforms.

2.1.2 User Interfaces

This program is invoked through a Windows Dialogue.

2.1.3 Hardware Interfaces

This project will not directly implement any hardware interfaces. All interfacing to I/O devices will be provided by the operating system.

2.1.4 Software Interfaces

Not applicable

2.1.5 Communication Interfaces

This project is to implement serial communication with the positioner table through RS-232 communication interface.

2.1.6 Memory Constraints

Requires a Random Access Memory which is sufficient to run an operating system.

2.1.7 Site Adaptation requirements

This program doesn't require any software for adaptation.

2.2 Product Functions

A serial communication module, a verification module for the distance of the markers among each other, a Coordinate Transformation Module, a module for finding the shortest distance, a Matlab Sequence for the image processing phase.

2.3 User Characteristics

The intended users of this program are the software developers that have a good understanding in software applications and no elementary instructions will be needed for these users to run or activate this product.

2.4 Constraints

The development of the product is constrained by the funding opportunities.

2.5 Assumptions and Dependencies

The assumptions is made that this products performs at an acceptable level.

3. Specific Requirements

3.1 External Interfaces Requirement

Please see section 2.1.5

3.2 Modules

3.2.1 Serial Communication

Serial communication on Windows NT operating system is implemented through RS-232 communication interface.

3.2.2 Verification Module

To verify whether the distances among markers are accurate enough for alignment purposes.

3.2.3 Coordinate Transformation

The transformation needed from the global (Vicon) coordinate system to local coordinate system.

3.2.4 Finding the shortest distance

For alignment purposes, it is best to find the shortest distance to align the target point with the proton beam.

3.2.5 Image Processing

The matlab sequence is to find the distance between the circles which represents the degree of error in alignment accuracy.

3.3 Performance Requirements

This program should work on one terminal, handle one file, and be supported by one user.

3.4 Design Constraints

The followings limit the usability of the program:

- Windows platform limits the portability.
- Matlab compiler and C++ Math Library is needed for operation.

3.5 Software System Attributes

This program is designed by incremental model, constructed step by step. Therefore, it is easy to maintain and debug, flexible to add new functions on it.

3.5.1 Reliability

This program will execute on the stated supported platforms described in section 2.

3.5.2 Availability

The executable software, documentation, and source code will be available at Loma Linda University Medical Center Research Laboratory Server.

3.5.3 Maintainability

Once the software is released, maintenance will not be needed until the release of the next prototype release.

REFERENCES

1. Weaver, K., et al, "A Ct-Based Computerized Treatment Planning System For I-125 Stereotacfic Brain Implants", Int J. Radiation Oncology Biology Physics Vol.18, pp. 445-454

2. Schulte, R., et al, "Analysis of Head Motion Prior and During Proton Beam Therapy", Int J of Radiation Biology Oncology, Vol. 47, No. 4, June 2000, pp. 1105-1110

3. Schulte, R., et al, "Stereotactic Radiosurgery--the Role of Charged Particles", Acta Oncology, Vol. 38, No. 2, June 1999, pp. 165-169

4. Arun, K., et al, " Least-square fitting of two 3D point sets", IEEE Transactions Pattern Analysis and Machine Intelligence Vol.9, No.5, June 1987, pp.698-700

5. Feldmann, S., and Patrick, L., "A Least Squares Approach to Reduce Stable Discrete Linear Systems Preserving Their Stability", Linear Algebra and its Applications, Vol.381, April 2004, pp.141-163

6. Beliakov, G., "Least Squares Splines with Free Knots: Global Optimization Approach", Applied Mathematics and Computation, Vol.149, June 2004, pp.783-798

7. Ford, R., et al, "Effects of Coordinate Transformation and Differencing Schemes on Matrix Structure and Convergence History", Applied Mathematics and Computation, Vol.77, June 1996, pp.53-66

8. Bigun, J., "Pattern Recognition in Images by Symmetries and Coordinate Transformations", Computer Vision and Image Understanding, Vol.68, No.3, December 1997, pp.290-307

9. Hartmann, G., et al, "Quality Assurance Program on Stereotactic Radiosurgery", Springer-Verlag Publishing Company, 1995, pp.5-25

10. Boyd, S., Least squares and least norm in Matlab
Bosch, DA., "Stereotactic Techniques in Clinical Radiosurgery", Springer Verlag Publishing Company, 1986, pp.45-50

11. Rugh, W., Linear System Theory, Prentice Hall, Inc, 1996, pp.159-169

12. Russ, J., The Image Processing Handbook, CRC Press Publishing Company, 1999, pp.371-374

13. Hogg, R., and Craig, A., Introduction to Mathematical Statistics, Fifth Edition, Prentice Hall, Inc, 1995, pp.471-477

14. Chatterjee S., and Price B., Regression Analysis by Example, Second Edition, John Wiley and Sons, Inc, 1991, pp.123-126

15. Clarke, G., and Cooke D., A Basic Course in Statistics, Fourth Edition, Arnold Publishing Company, 1998, pp.54-59

16. Beers, Y., Introduction to the Theory of Error, Addison-Wesley Publishing Company, Second Edition, 1962, pp.7-10

17. Weisberg, S., Applied Linear Regression, Second Edition, John Wiley and Sons, Inc, 1985, pp

18. Sonka M., et al, Image Processing, Analysis and Machine Vision, Chapman and Hall Computing Publishing Company, 1993, pp.113-118

19. Efford N., Digital Image Processing, Addison-Wesley Publishing Company, 2000, pp.250-255

20. Jahne, B., Digital Image Processing, Springer-Verlag Publishing Company, 1991, pp. 193-195

21. Murata, P., et al, "Final Report to Loma Linda University Medical Center", May 2001, pp.2-5