Alignment and Tracking using Graphics Hardware

William M. Wells III[‡] [*] and **Michael Halle** [‡] [†] and **Ron Kikinis** [‡] and **Paul Viola**[*]

Abstract

This paper explores an optimized variant of mutual-information based alignment that uses graphics hardware to accelerate tracking of a 3D object in a video scene. It is designed for fast operation on computers that are equipped with graphics hardware.

The algorithms described use a histogram-based method for computing the mutual information between object and scene. The gradient of mutual information is ascended in an effort to estimate the pose parameters of the object within the scene. This method has linear cost in the number of pixels and the number of histogram bins.

The computer graphics techniques used to implement the registration algorithm are described. The general approach is to map the problem of computing three-dimensional scene information onto readily available computer graphics accelerator hardware. This increases the algorithm's performance significantly.

Initial experiments are described where the experimental system is used to track moderate-speed manual motions of the test object in a high-contrast scene.

1 Introduction

This paper explores a variant of mutual-information based registration that uses graphics hardware to accelerate tracking of an object in a video scene.

Previous work [1] [2] [3] [4] has demonstrated that mutual information is an effective metric for the registration of signals of differing modalities. The approach makes few assumptions about the nature of the imaging process. As a result the metric is quite general and can be used with a wide variety of sensors. Mutual information has been used to align 3D models to images [1] [2], align 2D view-based representations to images [1] [2] and to register medical images of differing modalities [3] [4]. The technique makes many of the difficult problems of model comparison substantially easier, including accommodation of the vagaries of illumination and reflectance.

This paper focuses on the alignment of 3D objects with 2D images, and develops an implementation of the alignment by maximization of mutual information method that is designed for fast operation on computers that are equipped with graphics hardware acceleration.

When used for 3D-2D alignment, or for tracking moving 3D objects in a video stream, the method repeatedly adjusts the pose of a 3D model in order to maximize agreement with the video images, as measured by mutual information. The mutual information measure is used to compare surface normals from the model (at the current pose) with corresponding intensities from video. A part of this computation, the prediction of model depths and surface patch normals at the current pose, may be efficiently computed by using graphics hardware (see Section 2.4).

Typical graphics accelerators are designed to produce complete rendered images. While efficient on a per-pixel basis, the generation of a rendered image typically takes some time, and it becomes important make use of all of the rendered pixels, if possible.

The research reported in [1] [2] [3] used an iterative algorithm to perform a local search in pose space for a maximum of mutual information. This algorithm, called *EMMA*, used small sparse samples randomly drawn from the model to estimate the mutual information.

In order to take full advantage of graphics hardware acceleration, we were motivated to find an algorithm that generates a model pose update based on all of the pixels in a rendered image. The method described here uses dense histograms of the joint data (image and model at the current pose) to estimate both the probability density of the joint data, and its derivative. These estimates are used to construct an estimator of the derivative of the joint information with respect to the pose parameters, which is in turn used in a gradient ascent search in pose space. The estimators have linear cost in the number of pixels and the number of histogram bins, while the EMMA method has quadratic cost in the number of

[*]Massachusetts Institute of Technology, Artificial Intelligence Laboratory, 545 Technology Square, Cambridge, MA 02139

[†]Massachusetts Institute of Technology, Media Laboratory

[‡]Harvard Medical School and Brigham and Women's Hospital, Department of Radiology, 75 Francis Street, Boston, MA 02115

joint data samples used.

The system is based on the following iteration (also see Figure 1):

Repeat:

- Standard graphics hardware is used to create three-channel pseudo-color renderings of the test object that encode the x- and y- components of the surface normal, as well as depth.
- Pixels from the renderings are compared to the corresponding pixels from the incoming video stream.
- Based on this comparison, the derivative of mutual information with respect to the pose parameters is calculated, and the pose of the 3D model is updated.

The experimental system is able to track moderate-speed manual motions of the test object in a high-contrast scene.

2 Description of Method

In this section we review the basic formulation of alignment by maximization of mutual information, and describe a histogram-based method for computing the mutual information between object and scene, and its derivative with respect to pose parameters. The method has linear cost in the number of pixels and the number of histogram bins.

2.1 Alignment by Maximization of Mutual Information

Alignment by maximization of mutual information has been described in detail elsewhere [1] [2] [3] [4]. A detailed description appears in Viola's thesis [2]. We seek an estimate of the transformation $\hat{T}$ that aligns the model u and image v by maximizing their mutual information over the transformations T,

$$\hat{T} = \arg\max_T I(T)$$

where

$$I(T) = I(u(x),\ v(T(x))) \ . \tag{1}$$

In this application, x is a random variable that ranges over pixels that correspond to visible surface patches of the model at the current pose. In this application v consists of the components of the model surface normal that are perpendicular to the current viewing direction. T represents the current pose of the model, it is a transformation that projects model surface patch coordinates to image pixel locations.

Mutual information is defined in terms of entropy in the following way: [1]

$$I(u(x),\ v(T(x))) \equiv$$
$$H(u(x)) + H(v(T(x))) - H(u(x), v(T(x))) \ . \tag{2}$$

$H(\cdot)$ is the entropy of a random variable, and can be interpreted as a measure of uncertainty, variability, or complexity.

[1] We follow the definitional conventions of Papoulis [5] for mutual information and entropy.

Information has three components. The first term on the right in Equation 2 is the entropy in the model. This component is not used in the present application. The second term is the entropy of the part of the image into which the model projects. It encourages transformations that project u into complex parts of v. The third term, the (negative) joint entropy of u and v, takes on large values if u and v are functionally related. It encourages transformations where u explains v well. Together the last two terms identify transformations that find complexity and explain it well.

2.2 Estimating Entropy and its Derivative

In this section we develop a simple mechanism for estimating the derivative of the entropy of a random variable with respect to some parameter that the random variable is dependent on. This method will be used in later sections to evaluate the derivative (with respect to pose parameters) of the mutual information between a 3D model and a 2D image.

The entropy of a random variable may be defined in the following way,

$$H(z) \equiv E_z(-\ln p(z)) \ .$$

We may approximate the entropy of a random variable as follows,

$$\hat{H}(z) \equiv -\frac{1}{N} \sum_{z_i \in A} \ln \hat{p}(z_i, B) \ , \tag{3}$$

where A is a sample of observations of z, and $\hat{p}$ is a histogram approximator of the probability density on z computed from a second sample of observations, B. The derivative of $\hat{H}(z)$ can be computed as follows:

$$\frac{d\hat{H}(z)}{ds} \approx -\frac{d}{ds}\frac{1}{N} \sum_{z_i \in A} ln(\hat{p}(z_i, B)) \tag{4}$$

$$= -\frac{1}{N} \sum_{z_i \in A} \frac{\frac{d}{ds}\hat{p}(z_i, B)}{\hat{p}(z_i, B)} \tag{5}$$

$$= -\frac{1}{N} \sum_{z_i \in A} \frac{\frac{d}{dz_i}\hat{p}(z_i, B)\frac{dz_i}{ds} + \frac{d}{dB}\hat{p}(z_i, B)\frac{dB}{ds}}{\hat{p}(z_i, B)} \tag{6}$$

The numerator of the derivative has two components. The first, $\frac{d}{dz_i}\hat{p}(z_i, B)\frac{dz_i}{ds}$, is the change in entropy that results from changes in z_i. The second, $\frac{d}{dB}\hat{p}(z_i, B)\frac{dB}{ds}$ is a measure of the change in the histogram density estimate that results from changes in the sample B. For histograms computed from a finite sample, a differential change in s will move a vanishingly small number of sample points from one bin to another. As a result, we will assume that the second term is zero. This is in contrast to work described in [1] [2] [3] where a different formulation of the *complete* entropy derivative is used.

When used in the context of entropy minimization, Equation 6 may be understood as encouraging movement of data points from areas of low density to areas of higher density. This results in a distribution which is more peaked, and has lower entropy.

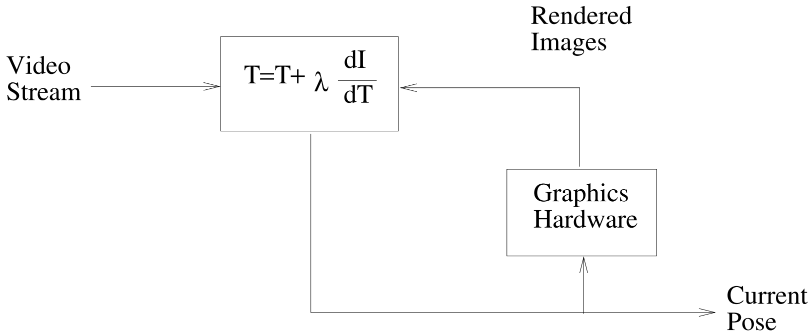

Figure 1: Information Flow in Mutual-Information Based Object Tracking Algorithm

To use this estimate for the derivative of entropy we will need estimates of the probability density on z, and its derivative. In the experiments reported below, we have used histograms for density estimation, and smoothed finite differences of the histograms for estimating derivatives of the density. The histogram affords computational efficiency at the expense of some accuracy. While histogram based approaches for alignment using mutual information have appeared [4], to our knowledge they have not been used in derivative-based optimization methods, but rather in non-derivative methods, such as Powell's method. In preliminary experiments, we have found the gradient-ascent approach to be substantially faster than Powell's method. Finally, this seems to be the first use of histogram entropy estimates for visual 3D-2D alignment.

2.3 Maximization of Mutual Information

The approach described in the previous section may be used to maximize the mutual information of the model and image (Equation 2).

In order to seek a maximum of the mutual information, we calculate an approximation to its derivative with respect to the pose of the model,

$$\frac{d}{dT}I(T) = \sum_{v_i \in A} \frac{\partial \hat{I}}{\partial v_i}\frac{dv_i}{dT} \quad .$$

We use this form because v_i is the part of the joint data that is primarily dependent on the pose of the object. A are the intensities of the image pixels that correspond to visible surface patches of the model (at the current pose)

Since mutual information is defined in terms of entropy, we may use the entropy derivative estimator of Section 2.2 to calculate the derivative of mutual information with respect to the data.

The term $\frac{d}{dT}(v_i)$ will generally involve gradients of the image intensities, and the derivative of transformed coordinates with respect to the transformation. In the simple case that T is a linear operator, we obtain the following outer product expression,

$$\frac{d}{dT}v(T(x_i)) = \nabla v(T(x_i))x_i^T \quad .$$

We seek a local maximum of the mutual information by using a gradient ascent. Steps are repeatedly taken that are proportional to the approximation of the derivative of the mutual information with respect to the transformation:

$$T \quad \leftarrow \quad T + \lambda \frac{dI}{dT} \quad . \tag{7}$$

The parameter λ is a step size parameter – if it is too small, convergence will be slow, if it is too large, the iteration may diverge. The above procedure is repeated a fixed number of times or until convergence is detected.

2.4 Graphics Computations

In this section, we describe the computer graphics techniques we have used to implement the registration algorithm. The general approach is to map the problem of computing three-dimensional scene information into a computer graphics framework. This mapping allows the use of readily available computer graphic accelerator hardware to increase the algorithm's performance. Computer graphics hardware transforms geometric three-dimensional model information into a two-dimensional pixel matrix. Conventionally, this matrix represents an image that is to some approximation photo-realistic and designed for viewing by a human observer. For our purposes, the rendering process produces a pixel array that contains information about the model's orientation and distance from the computer graphics camera. The discrete pixel representation of this data is a natural and useful one because each pixel can be mapped to a corresponding pixel in the incoming video image.

The computer graphics rendering process is broken up into two passes, one that computes surface orientation and one that computes depth. The rendering scheme encodes the orientation of the object into the green and blue components of each pixel, while depth is encoded in the pixel's red component. We will consider these two rendering passes separately.

2.4.1 Calculating Surface Orientation

The registration algorithm requires knowledge of the orientation of the visible surfaces of the object with respect to the camera. This orientation is represented as the x and y components of the normal vector at every point on the surface. Because the coordinate axes of the these normals are referenced to the camera, all front-facing surfaces have a positive z value. Thus, an (x, y) pair uniquely represents the orientation of a visible part of a surface. We encode the x component of the surface normal into the blue component of the image, and the y component into the green.

To perform this orientation rendering, we note that Gouraud lighting [6] of a purely diffuse surface using a directional light source performs an operation that is similar to a dot product of the light source direction and the surface orientation at each vertex of the object, except that the result is clamped to prevent negative values. For example, a sphere lit from the top would be brightest at its north pole and fade to black at its equator, with its entire southern hemisphere being black. (This operation is only exact at the location of the vertices of the model. Between vertices, Gouraud shading linearly interpolates vertex values, which is close enough to the real surface normals for our purposes.)

If negative pixel values were allowed, a more complete approximation to a dot product could be obtained by using, in addition to the first directional light source, an opposing directional light source of negative intensity. In the example above, the "negative light" from such a source would be strongest at the south pole, and taper off to zero at the equator. The sum of the effect of the two sources would be a dot-product shaded sphere. In fact, negative pixel values are not needed if an ambient light source (or an emissive surface material) is introduced to bias each pixel value to one-half full intensity. In such a scheme, each pixel value contains the value $d' = (d+1)/2$, where d is the dot product approximation.

Dot product rendering depends on the implementation of the physically unrealizable concept of a negative light source. Fortunately, several graphics libraries or accelerators implement negative light sources. For example, the OpenGL [7] specification specifically allows light source intensities to vary from -1 to 1. The manual for XGL, Sun Microsystem's [8] proprietary graphics API, states that light sources are clamped to a value between 0 and 1. However, two of Sun's fastest graphics accelerator systems, the ZX/TZX and the FFB, both violate the specification and (fortunately) permit negative light sources.

This ability to render dot products can be used to perform the orientation rendering pass using a total of five light sources. Four of these sources are directional. A blue light source shines in the positive x direction, while a negative blue light shines in the negative x direction. Similarly, a green positive/negative light pair shine along the y axis. The remaining light is a half-intensity blue-green ambient bias source. Using this lighting environment, all pixels are encoded in the desired fashion.

2.4.2 Calculating Depth

To analyze the effect of in-plane rotations, the registration algorithm requires a depth map of the scene. Almost all computer graphics hardware uses a z- or depth-buffer to perform hidden surface elimination, so this buffer could be used directly. However, the z-buffer is usually 24 or 32 bits wide, a much higher resolution than the registration algorithm needs. In addition, many computer graphics subsystems are very slow at reading pixels back from the framebuffer, so the large data size of the z-buffer would, in general, significantly limit rendering performance.

Instead of using the z-buffer depth map, we instead render the object using specialized shading parameters that encode a depth map as intensity. Both OpenGL and XGL APIs implement depth cueing or "fog". Depth cueing causes the object's intensity to interpolate from its calculated shaded color to a given constant color, which in this case is black. A depth map can be formed by setting the object's shading characteristics to be ambient or emissive only, with no diffuse or specular components. The parts of the object nearest the camera are relatively unattenuated and have a high intensity. Parts of the object further away have a proportionally lower intensity. In our implementation, this intensity value forms the red component of each pixel of the image.

Because the two rendering passes require different light sources, material parameters, and depth cueing configurations, they must be performed sequentially by rendering twice all the geometric primitives of the model. The passes are kept from interfering with each other by setting a global pixel plane mask to limit the effect of subsequent rendering operations. Once both passes have been rendered, the pixel array can be read back from the graphics subsystem in a single operation. For sufficiently small models and particular graphics hardware, this pixel transfer can be by far the slowest part of the computer graphics process.

3 Experiments

3.1 Initial Experiment

A prototype of the method described above was implemented without using graphics hardware and used to test the basic computational approach.

Figures 2 and 3 show an initial and final configuration of the algorithm. The initial pose includes a misorientation by an out-of-plane rotation. The final configuration was obtained in about 20 iterations. A 3D model of the subject's head, along with surface normals, was derived from a Cyberware scan of the subject. It is rendered on the right to illustrate the current pose of the model.

The joint data histogram was constructed with 32 by 32 buckets, and an image intensity histogram contained 32 buckets. The stepsizes λ were set in the following way, for in-plane translation: 1000.0, for depth translation: 3000.0, for all rotations: .05 .

3.2 Real-Time Tracking Experiment using Graphics Hardware

3.2.1 Intensity-Seeking Component

An intensity-seeking component was incorporated into the experiment in order to make the tracker very robust while tracking a light colored object against a darker background.

The iteration of Equation 7 was altered in the following way,

$$T \leftarrow T + \lambda \frac{dI}{dT} + \mu \frac{dL}{dT} \quad , \tag{8}$$

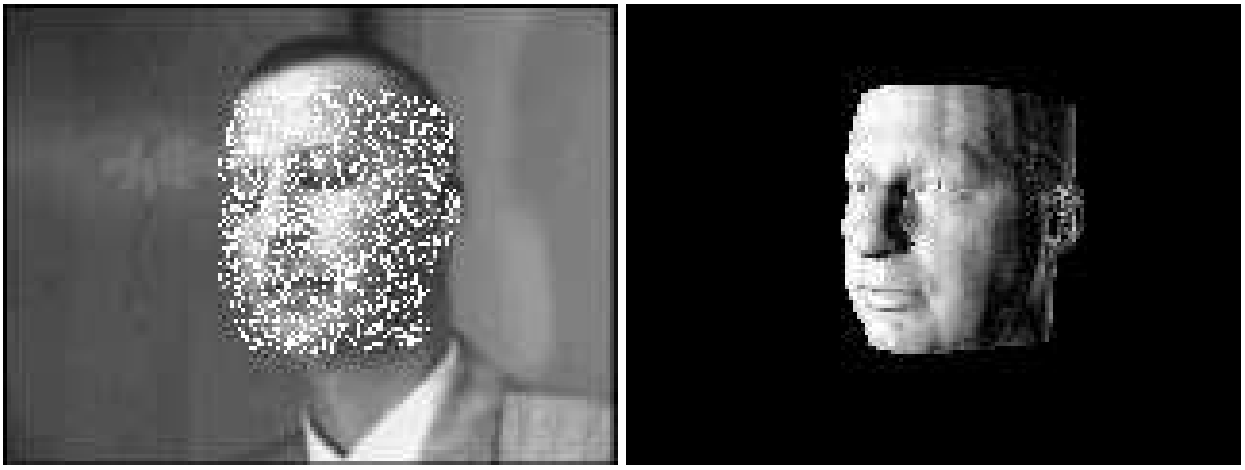

Figure 2: Video Image and Rendering of Initial Configuration of Model. The Pose of the Model is Illustrated by White dots in the Upper Image, and by the Rendering in the Lower Image.

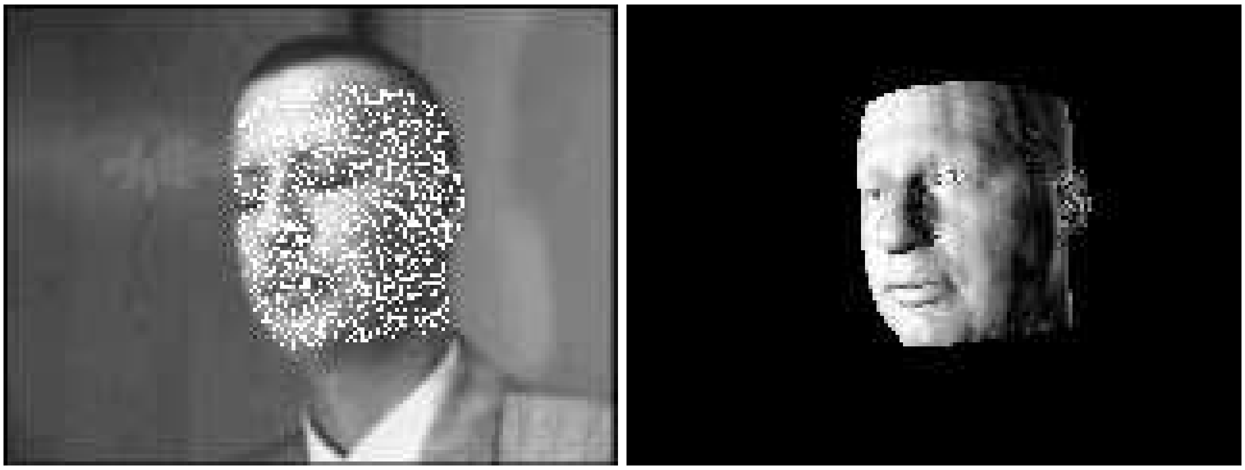

Figure 3: Video Image and Rendering of Final Configuration of Model.

where L is the total image brightness covered by the model,

$$L \equiv \sum_{v_i \in A} v_i \quad ,$$

where A is intensities of the image pixels that correspond to visible surface patches of the model (at the current pose).

This iteration may be viewed as a gradient maximizer of $\lambda I + \mu L$. This objective function is a linear combination of the mutual information and the total image brightness covered by the model at the current pose. This extension to the objective function increases the capture range of the tracker under under the conditions mentioned above.

Equation 8 may be written as

$$T \leftarrow T + \sum_{v_i \in A} \left[\lambda \widehat{\frac{dI}{dv_i}} + \mu \right] \frac{d}{dT}(v_i) \quad ,$$

since $\frac{dL}{dv_i} = 1$.

3.2.2 Tracking Experiment

The above iteration was used in an experiment that used graphics hardware (Sun [8] FFB board), as described in Section 2.4, to calculate surface normals and depths.

Figure 4 show the initial and final configuration of the tracker.

This convergence required about 20 iterations. The system, in an initial implementation on a sun sparc ultra performed about four iterations per second, and was able to track moderate-speed manual manipulations of the test object, a plastic skull.

The graphical model of the skull was derived from a computed tomography (CT) scan of the skull. The scan was smoothed and converted to a triangle model by using the marching cubes iso-surface algorithm [9], and was further processed by a triangle reduction algorithm. The resulting model contained about 4000 triangles.

The parameters used in this experiment were similar to the previous experiment, with μ kept comparatively small.

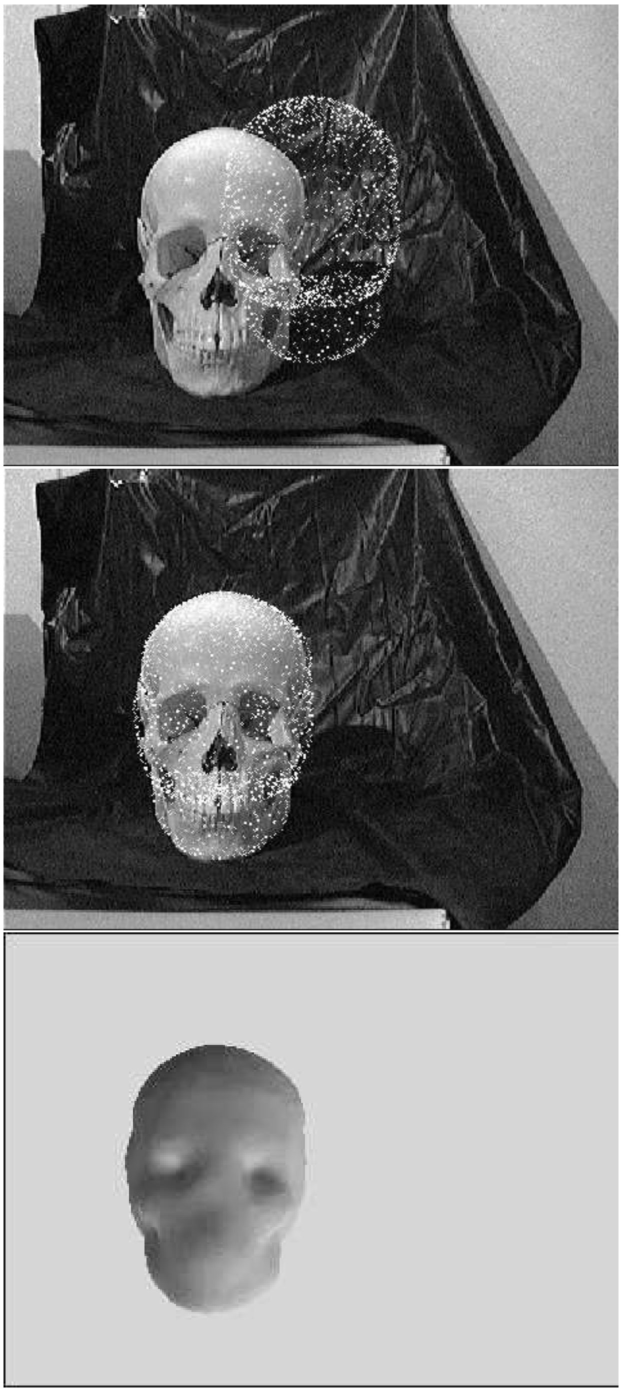

Figure 4: Video Image and dot Overlay Illustrating an Initial Configuration of the Model, and a Final Configuration of the Model. A Rendering of the Smoothed Graphical Model as is Compared to the Incoming Video Stream, at the Final Pose.

4 Acknowledgements

We thank the following sources for their support of this research: USAF ASSERT program, Parent Grant#:F49620-93-1-0263 (Viola), Howard Hughes Medical Institute (Viola), ARPA IU program via ONR #:N00014-94-01-0994 (Wells) and AFOSR #F49620-93-1-0604 (Wells).

We thank Bill Lorensen for generating the reduced polygon model of the skull.

References

[1] P.A. Viola and W.M. Wells III. Alignment by maximization of mutual information. In *Proceedings of the 5th International Conference on Computer Vision*, 1995.

[2] Paul A. Viola. *Alignment by Maximization of Mutual Information.* PhD thesis, Massachusetts Institute of Technology, 1995.

[3] W. Wells, P. Viola, and R. Kikinis. Multi-Modal Volume Registration by Maximization of Mutual Information. In *Second International Symposium on Medical Robotics and Computer Assisted Surgery*, 1995.

[4] A. Collignon et al. Automated multi-modality image registration based on information theory. In *Proceedings of the Information Processing in Medical Imaging Conference*, pages 263–274. Kluwer Academic Publishers, 1995.

[5] Athanasios Papoulis. *Probability, Random Variables, and Stochastic Processes.* McGraw-Hill, Inc., third edition, 1991.

[6] D Hearn and MP Baker. *Computer Graphics.* Prentice-Hall, 1986.

[7] OpenGL Architecture Review Board. *OpenGL Reference Manual.* Addison Wesley, 1992.

[8] Sun Microsystems Inc., Mountain View, CA.

[9] H. Cline, W. Lorensen, S. Ludke, C. Crawford, and B. Teeter. Two Algorithms for the Three-Dimensional Reconstruction of Tomograms. *Med. Phys.*, (15):320 – 327, 1988.