January 2000

Image Registration Using Mutual Information

Geoffrey Egnal
University of Pennsylvania

Kostas Daniilidis
University of Pennsylvania, kostas@cis.upenn.edu

Follow this and additional works at: http://repository.upenn.edu/cis_reports

Recommended Citation

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-00-05.

This paper is posted at ScholarlyCommons. http://repository.upenn.edu/cis_reports/117
For more information, please contact libraryrepository@pobox.upenn.edu.
Image Registration Using Mutual Information

Abstract
Almost all imaging systems require some form of registration. A few examples are aligning medical images for diagnosis, matching stereo images to recover shape, and comparing facial images in a database to recognize people. Given the difficulty of registering images taken at different times, using different sensors, from different positions, registration algorithms come in different shapes and sizes.

Recently, a new type of solution to the registration problem has emerged, based on information theory. In particular, the mutual information similarity metric has been used to register multi-modal medical images. Mutual information compares the statistical dependence between the two images. Unlike many other registration techniques, mutual information makes few a priori assumptions about the surface properties of the object or the imaging process, making it adaptable to changes in lighting and changes between sensors. The method can be applied to larger dimensional registration and many other imaging situations.

In this report, we compare two approaches taken towards the implementation of rigid 2D mutual information image registration. We look further at algorithm speedup and noise reduction efforts. A full background is provided.

Comments
University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-00-05.
Image Registration Using Mutual Information

Submitted by Geoffrey Egnal
Department of Computer and Information Science
University of Pennsylvania

To
Kostas Daniilidis (Chairperson)
James Gee
Max Mintz

For the
Second Written Preliminary Examination
December 16, 1999
Abstract

Almost all imaging systems require some form of registration. A few examples are aligning medical images for diagnosis, matching stereo images to recover shape, and comparing facial images in a database to recognize people. Given the difficulty of registering images taken at different times, using different sensors, from different positions, registration algorithms come in many different shapes and sizes.

Recently, a new type of solution to the registration problem has emerged, based on information theory. In particular, the mutual information similarity metric has been used to register multi-modal medical images. Mutual information compares the statistical dependence between the two images. Unlike many other registration techniques, mutual information makes few a priori assumptions about the surface properties of the object or the imaging process, making it adaptable to changes in lighting and changes between sensors. The method can be applied to larger dimensional registration and many other imaging situations.

In this report, we compare two approaches taken towards the implementation of rigid 2D mutual information image registration. We look further at algorithm speedup and noise reduction efforts. A full background is provided.

1 Introduction

Many imaging applications require registration. Image registration involves finding a geometric transformation mapping one of the images to the other, usually for easy comparison. Given the wide range of imaging applications, there are many different types of registration algorithms. Recently, a new type of solution to the registration problem has emerged, based on information theory in the form of a mutual information similarity metric. Viola and Wells [19] and Collignon, Maes, Delaere, Vandermeulen, Suetens and Marchal [4] developed the method independently in 1995. Mutual information compares the statistical dependence between two images. The metric works from the pixel values and makes few assumptions about the surface properties of the object or the imaging process. As a result, mutual information is robust with respect to changes in lighting and even imaging modality.

Multi-modal medical imaging was one of the original motivations for the mutual information registration systems. In many applications, a medical practitioner wants to view multiple sensor images to locate abnormalities and decide upon a treatment plan. For instance, in radiotherapy the initial detection is often based on one of the functional scans, such as Single Photon Emission Computed Tomography (SPECT) or Positron Emission Tomography (PET). Subsequently, the outlining of the tumor is based on the Magnetic Resonance scan (MR), and the dosage is based on a Computed Tomography (CT) scan.

Multi-modal images present a difficult problem for registration methods (see Figure 1.) Firstly, the size of the scanned region differs between the images. A CT scan is harmful to the patient, and so the area scanned is often smaller than that of the MR scan. The various sensors also measure different aspects of the brain. For example, SPECT and PET measure functional aspects of the brain, while MR and CT measure anatomical aspects. An example of a functional aspect of the brain would be blood flow. Within the anatomical group, only MR measures brain structure, while bone shows up more strongly in the CT image. These differences have proven prohibitively difficult for other registration algorithms to overcome in the past.

![CT](image1.png) ![MR](image2.png) ![PET](image3.png) ![SPECT](image4.png)

**Figure 1:** Many medical applications need multiple images for accurate diagnosis. The task of registering such different images is difficult because they provide different kinds of information. Mutual information has served well in registering such images. (Images from ISL, Utrecht)

One accurate method to register such images implants ‘bone markers’ in the patient’s skull. Since these markers show up in all scans and are of known position, registration becomes much easier. However, ‘patient unfriendliness’
and the labor intensive process of placing the markers make such methods less attractive. Mutual information, as a fully automatic method, registers such images in less time with more patient comfort. Although many consider the bone-implanted markers as the ‘gold standard’, a recent study has shown that mutual information obtains reasonably similar accuracy.[20]

The applications of mutual information registration systems have extended far beyond simple image registration. Already, people aim to integrate medical scans for better diagnosis and visualization. Model alignment could also aid in object or face recognition, bringing a myriad of applications. In any situation where two sensors (of any type) model the same scene, mutual information could be helpful in the registration process.

The main contribution of the current paper is to:

- Compare the mutual information registration algorithms of Collignon et al. and Viola and Wells
- Compare correlation-based and mutual information-based registration
- Compare the various density estimation, interpolation and maximization methods employed by Collignon and Viola
- Review multi-resolution implementations of the algorithm
- Demonstrate basic concepts using a one-dimensional C implementation

For the remainder of the paper we will abbreviate [19] as Viola and [4] as Collignon.

1.1 Algorithm Mechanics

The registration algorithms of Collignon and Viola follow a normal registration framework, or script. In the hope of providing context, we give an outline of the methods. The body of this report fills in the details of each algorithm and compares them.

Viola’s algorithm begins with two images: a model \( u \) and an image \( v \). He uniformly picks random points \( x \) in \( u \) and calls that sample \( u(x) \). Viola does not specify from where to sample: windows within the image or the complete image itself. Starting with an hypothesized transformation \( T \), he picks points corresponding to \( u(x) \) in \( v \), called \( v(T(x)) \), using trilinear interpolation. In order to test whether he has found the optimal transformation, the algorithm tests whether the mutual information is at a maximum using gradient descent. It takes the derivative of the mutual information (defined later) of the gray values of these points with respect to the \( T \) under consideration using the Parzen window density approximation. If the derivative is close to zero, the algorithm stops. Otherwise, the gradient descent procedure outputs a new transformation \( T \) to test. The algorithm continues until it times out or comes within a tolerance of finding the transformation that leads to the maximal mutual information between \( u(x) \) and \( v(T(x)) \).

Collignon’s paper follows a similar form, but differs in the details. It interpolates to \( v(T(x)) \) using trilinear and also partial volume interpolation (defined later) He maximizes the mutual information using Powell’s algorithm, which calculates mutual information from a histogram estimate of the density. The overall framework is the same.

There are many ways to classify registration algorithms. A compact method is proposed by Brown [3], where algorithms are classified on the basis of:

1. feature space - which information in the images is used for matching, i.e. pixel intensity or selected feature points
2. search space - the class of transformations that is assumed to be capable of aligning the images, e.g. affine or projective warps
3. search strategy - how to choose the next transformation in the search for the optimal transformation to register the images, e.g. hill climbing or dynamic programming
4. similarity metric - how the search strategy measures the relative merit for each candidate transformation, e.g. correlation or mutual information
Mutual information is a new similarity metric. The items of Brown's list are not orthogonal; often the methods' components are chosen in correlated clusters. In the case of mutual information, the formulation of the problem can allow for an expedient search strategy. Both Viola and Collignon choose to search pixel space using affine 2D warps.

1.2 Previous Work

The origin of mutual information is largely credited to the 1948 work by Shannon [16]. Since then, there have been many uses of mutual information, including basic statistics, communication theory, and complexity analysis. It is only recently that we have used this measure to register images.

Viola and Wells [18][19] and Collignon et al. [4] developed the method independently in 1995. The differences between their implementations are reviewed in this paper. Since then, many in the medical community have developed interest in these methods, especially its ability to register multi-modal images. A group at Utrecht, under Max Viergever [12][13] has delved into various aspects of mutual information. Additionally, Philippe Thévenaz and Michael Unser, from Swiss Federal Institute of Technology, have published an improved multi-resolution method for mutual information registration. Many hospitals and groups are now using and testing the new method (see http://www.ai.mit.edu/~viola/MITRefs.html for a short list.) Sebastian Gilles provides an on-line tutorial of Viola's method at Oxford [7]. Mutual information has featured favorably in comparison studies for image registration [20]. In the last four years, the topic has caught much attention.

Mutual information represents one of many statistical image registration techniques. Brown provides an overview of some of the general algorithms [3]. Although this paper does not purport to cover these topics, one may refer to such articles as Leventon and Grimson [10] or Hill, Studholme and Hawkes [8] for additional information on other joint density registration techniques.

The background for each of the maximization and density estimation algorithms we discuss is given in the appropriate sections.

2 Statistical Background

2.1 Preliminaries

Mutual Information is a statistical term, and many of the implementation details require an understanding of basic probability. As such, this chapter provides a rudimentary background to the terms used later in the paper. Many of the definitions given here are brief and rely on previous reader knowledge. For a more complete and rigorous treatment, the reader can refer to more dedicated works, such as [9][2].

Probability describes random events, such as the roll of a die, the numbers chosen in tomorrow's lottery, or where a baseball will land on an open field. All of these events are unknown before they happen. Here we develop a formal language to describe such events, involving three items:

- Sample space - A set \( \Omega \) consisting of all (conceptually) possible outcomes.
- Outcomes - atomic sample points and realizations \( \omega \in \Omega \).
- Events - Subsets of \( \Omega \) for which probability is defined.

The family of events is a \( \sigma \)-algebra, denoted \( \mathcal{F} \). A \( \sigma \)-algebra is a family of subsets of a set \( \Omega \) containing \( \Omega \) and closed under countable union, intersection, and complementation.

**Definition 1 (Probability)** Let \((\Omega, \mathcal{F})\) be a sample space and a \( \sigma \)-algebra of events. A probability function on \((\Omega, \mathcal{F})\) is a function \( P : \mathcal{F} \to \mathbb{R} \) such that:

1. \( P(A) \geq 0 \quad \forall A \in \mathcal{F} \)
2. \( P(\Omega) = 1 \)
3. Whenever \( A_1, A_2, ... \) are (pairwise) disjoint sets in \( \mathcal{F} \), \( P(\sum_{n=1}^{\infty} A_n) = \sum_{n=1}^{\infty} P(A_n) \)

One can think of the probability as the long-run frequency of occurrence of \( A \) when the experiment is replicated. For example, if a fair 6-sided die were rolled, \( \Omega = \{1, 2, 3, 4, 5, 6\}, \mathcal{F} = \text{Powerset}(\Omega) \) and \( P \) (any single element of \( \Omega \)) = 1/6.

At this point, we know technically what a probability function is. We would like a general, compact method to describe probability functions. Each probability function can be specified by a cumulative distribution function:
Definition 2 (Cumulative Distribution Function) The cumulative distribution function (CDF) of $P$ is the function $F_P : \mathbb{R} \to [0, 1]$ defined by $F_P(t) \triangleq P((\infty, t])$.

With the CDF, we can succinctly specify many different probability functions, describing a variety of situations. For instance, a Uniform distribution function over $[0, 1]$ looks like:

$$F_P(t) = \begin{cases} 0 & t < 0 \\ t & 0 \leq t \leq 1 \\ 1 & t > 1 \end{cases} ,$$

The probability function is called discrete, mixed or continuous, depending on the status of its distribution function. If the probability is continuous, there is is often a more intuitive way to describe probabilities than the CDF:

Definition 3 (Probability Density Function) The probability function $P$ on $\mathbb{R}$ is considered absolutely continuous if there exists a positive function $f$ on $\mathbb{R}$, the probability density function of $P$ (PDF), such that for every interval $(a, b)$, $P((a, b]) = \int_a^b f(t)dt$.

The density function can also be defined for discrete probabilities using measure theory (not covered here.) Using previous definitions, we can see that $\int_{-\infty}^{\infty} f(s)ds = 1$ and that $F(t) = \int_{-\infty}^t f(s)ds$ for $t \in \mathbb{R}$. The PDF gives the probability point mass at any point $x \in \mathbb{R}$. A famous example of a density function is the Normal Density Function, abbreviated $N(\mu, \sigma^2)$:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

A related concept, conditional probability, allows us to account for the knowledge gained when a previous event occurs. For instance, knowledge that a die was loaded so that the number two came up often would affect the probability of each number on the die coming up. The conditional probability will be important in this paper because it reflects how much information one RV has about another, a central concept in image alignment. Note that the conditional probability is not a RV because once the conditioning event has occurred, the conditional probability is defined deterministically by the probability function. Before the conditioning event occurs, its potential outcome is random, making the final conditional probability another RV.

Definition 4 (Conditional Probability) Let $A$ and $B$ be events. Provided that $P(A) > 0$, the conditional probability of $B$ given $A$ is $P(B|A) \triangleq P(B \cap A)/P(A)$

Often, one wishes to look at functions of events, rather than the events themselves. For example, if a coin were tossed 10 times, we would like to examine the number of heads, and not the head/tail sequence. For this reason, we examine random variables of the outcomes of an experiment. More formally,

Definition 5 (Random Variable) A random variable (RV) is a function $X : \Omega \to \mathbb{R}$ such that $X^{-1}(B) \in \mathcal{F}$ for all $B$ in the Borel $\sigma$-algebra, which is created by taking all open intervals on the Real line. Associated with every random variable $X$ is a probability on $\mathbb{R}$, $P_X(B) = P(X^{-1}(B))$ and a support set $S = \text{Closure}\{x \in \mathbb{R} \text{ such that } f_X(x) \neq 0 \}$.

This concept extends to multiple dimensions. Let $X = (X_1, \ldots, X_d)$ be a random $d$-vector. The distribution of $X$ is the probability of $P_X(B) \triangleq P\{X \in B\}$ on $\mathbb{R}^d$. The joint distribution function of $X_i$ is the function $F_X : \mathbb{R}^d \to [0, 1]$ given by $F_X(t_1, \ldots, t_d) = P\{X_1 \leq t_1, \ldots, X_d \leq t_d\}$.

Related to the concept of conditional probability is the concept of independence. If RV $A$ has no probabilistic interaction with RV $B$ (i.e., there is no information about $A$ in $B$ and vice versa), the two RVs are independent. For instance, two rolls of a die are usually considered to be independent. More formally,

Definition 6 (Independent Random Variable) Random variables $X_1, \ldots, X_n$ are independent if $P\{X_1 \in B_1, \ldots, X_n \in B_n\} = \prod_{i=1}^n P\{X_i \in B_i\}$. 

4
Although a RV allows us to answer many questions about the data at hand, we seek to summarize our knowledge of a PDF in a compact form. One such concept is called expectation of a RV.

**Definition 7 (Expectation)** The following are computational formulae for the expectation of a function $g(X)$ of the RV $X$:

$$
E_X[g(X)] = \sum_{x_i \in \Omega_X} g(x_i)P(X = x_i) \quad \text{Discrete } X
$$

$$
E_X[g(X)] = \int_{-\infty}^{\infty} g(t)f_X(t)dt \quad \text{Continuous } X
$$

For a mixed RV that is sometimes continuous and sometimes discrete, the expectation can be expressed as a linear combination of the two above formulae.

Note that expectation is not found in the data. It is a function of the CDF / PDF only. We can find good estimates of $E_X[X]$ that are based on the data. Bias, or $E_X[g(X)] - g(X)$, provides one way to measure how ‘good’ an estimate is. If $\{x_n\}$ are a set of $N$ trials of $X$, the sample mean $\frac{1}{N}\sum_{i=1}^{N} x_i$ gives an unbiased estimate of $E_X[X]$. If the $x_i$ are iid with finite first moment, then the law of large numbers says that the sample mean equals the expectation for infinitely large samples. A function of the data, like the sample mean, that deterministically estimates behavior about a RV is called a *statistic*. Note that the sample mean is a random variable, while $E_X[X]$ is not. Variance is another summary statistic of a RV:

**Definition 8 (Variance)** If $E_X[X^2] < \infty$, then

$$
\text{Variance}(X) = E_X[(X - E_X[X])^2].
$$

Variance measures how far a random variable deviates from its mean in size or likelihood.

In this report, we consider data in images. In some sense, the data is not at all random; we examine pixel values after they have been taken. However, we do not base our registration calculations off all the data points. We subsample the data in a random way, making the image data random. In this light, the data is a stochastic process, or an indexed family of RVs. The stochastic process has an unspecified probability function. The assumption permits us to use mutual information in our registration algorithms.

### 2.2 Entropy

#### 2.2.1 Discrete Entropy

Entropy measures the randomness of a random variable. The probability associated with a RV does not indicate how random it is. What does ‘highly random’ mean? It means the average probability over the support set for a RV is low. (See Figure 2) A practical example: a low-odds lottery has a higher randomness than a coin-toss. Entropy gives information about randomness of a RV, allowing us to compare different RVs properties. Another way to think of entropy for an image is as the ‘crispness’ of the probability distribution of an image.

Physics, and in particular thermodynamics, provided the original definitions of entropy. Since then, the physical interpretation has widened to a statistical interpretation. Shannon is largely credited with starting the shift in 1948. Since then, entropy has been used in communication theory, complexity theory, investment allocation, and many other areas.

Perhaps the simplest definition of entropy ($H(X)$) is an expectation. We begin with the classical definition of $H(X)$ in the discrete case. For more detail, see [5].

**Definition 9 (Entropy)**

$$
H(X) \triangleq -E_X[\log(P(X))] = -\sum_{x_i \in \Omega_X} \log(P(X = x_i))P(X = x_i)
$$

Since it is based on the expectation and probability associated with $X$, entropy does not depend on the actual values taken by the RV $X$. Since the PDF for $X$ is often unknown, we make use of the data to estimate a likely PDF from which to calculate entropy. This estimate is known as *empirical entropy*.

Just as we extended the definition of a RV to multiple dimensions, we can extend the definition of entropy to a pair of RVs for both joint and conditional entropy. Conditional entropy reflects how knowledge of RV $Y$ reduces the uncertainty of $X$. In other words, the more that $X$ depends on $Y$, the lower the conditional entropy.
Definition 10 (Joint Entropy)

\[ H(X, Y) \triangleq -E_{X,Y}[\log(P(X, Y))] = - \sum_{(x, y) \in X \times Y} \log(P(X = x, Y = y))P(X = x, Y = y) \]

Definition 11 (Conditional Entropy)

\[ H(X|Y) \triangleq -E_{X,Y}[\log(P(X \mid Y))] = - \sum_{(x, y) \in X \times Y} \log(P(X = x \mid Y = y))P(X = x, Y = y) \]

These two concepts are related by the simple formula:

\[ H(X, Y) = H(X) + H(Y|X) \]

2.2.2 Differential Entropy

We can extend the concept of entropy of a discrete RV to that of a continuous RV. The result is called differential entropy. The two are closely related, but do have their differences.

Definition 12 (Differential Entropy)

\[ H(X) \triangleq -E_X[\log(P(X))] = - \int S \log(f_X(t)) f_X(t) \, dt. \]

Where \( S \) is the support set and \( f \) is the PDF associated with RV \( X \).

We can similarly extend the definition for joint and conditional entropy from the discrete case to the continuous case.

Example 13 (Normal Distribution) If \( X \sim N(\mu, \sigma^2) \), we can calculate the differential entropy of \( X \) (See
\[ H(X) = - \int_S f_X(t) \ln(f_X(t)) dt \]
\[ = - \int_S \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-u)^2}{2\sigma^2}} \left[ -\frac{(x-u)^2}{2\sigma^2} - \ln(\sqrt{2\pi\sigma^2}) \right] \]
\[ = 0 + \frac{1}{2} \ln(2\pi\sigma^2) \]
\[ = \frac{1}{2} \ln(2\pi\sigma^2) \]
\[ = \frac{1}{2} \ln(2\pi\sigma^2) \]

Unfortunately, differential entropy does have a few differences from discrete entropy. The major difference is that the differential entropy can be negative because \( f_X(x) \) can take on values greater than 1.

### 2.3 Mutual Information

Mutual information measures the amount of information that one variable contains about another RV.

**Definition 14 (Mutual Information)**

\[ I(X, Y) \triangleq E_{X,Y} \left[ \log \left( \frac{P(X, Y)}{P(X)P(Y)} \right) \right] = H(X) + H(Y) - H(X, Y) \]

Mutual information satisfies the following properties (for both discrete and differential entropy, assuming the differential entropy is well behaved.)

1. **Symmetry:** \( I(X, Y) = I(Y, X) \)
2. **Independence:** \( I(X, Y) = 0 \leftrightarrow (X \text{ and } Y \text{ are independent}) \)
3. **Boundedness:** \( 0 \leq I(X, Y) \leq \min(I(X, X), I(Y, Y)) \)
4. **Invariance:** If \( T \) is a \( 1 \to 1 \) mapping, \( I(X, T(X)) = I(X, X) \). More generally, if \( S(X) \) is a sufficient statistic for \( X \), then \( I(X, S(X)) = I(X, X) \).

The fourth point deserves special attention: it helps to justify the performance of mutual information as a similarity metric. When we say a 1-1 mapping between images, we mean that the gray values of the signal \( v \) can be expressed as a 1-1 mapping of the gray values of the signal \( u \). Recall that \( g \) is a 1-1 mapping if and only if \( x \neq y \to g(x) \neq g(y) \) for \( x, y \in \text{Domain}(g) \).

We can rewrite the definition in terms of conditional entropy:

\[ I(X, Y) = H(X) - H(X | Y) \]

This makes mutual information the reduction in the uncertainty of \( X \) due to the knowledge of \( Y \). This interpretation explains the mutual information intuitively. We now know that mutual information is lower-bounded at 0 when \( X \) and \( Y \) are completely independent. Also, we have an upper bound with 1-1 mappings, as indicated in property 4. Notice that \( I(X, X) = H(X) \), which is the reason entropy is sometimes called self-information.

### 3 Alignment

The aim of a registration algorithm is to put two signals into geometric alignment. Let’s say we are given two signals of form or space \( u(x) \) and \( v(y) \), where \( u(x) \) is the model and \( v(y) \) is the image of the model. The two signals can be completely different in form and even coordinate system. For example, we can attempt to align 3D to 2D data.

In the case of image registration, there are two components of the transformation that relate \( u(x) \) and \( v(y) \). The first is a physical displacement, called \( T \), representing pose transformation. \( T \) can have many dimensions.
For instance, it can map 2D to 2D, or 3D to 3D in an affine or projective sense. The choice depends on the images
the user wants to register. Here, we will talk about an affine 2D warp. The second component is an imaging
function \( F(u(x), q) \), which determines the actual value at the point \( v(T(x)) \). The \( q \) argument to \( F \) is a collection
of all the exogenous influences, such as lighting. The model then looks like:

\[
v(T(x)) = F(u(x), q) + \eta,
\]

where \( \eta \) represents noise in the imaging process. Note that \( F \) is the identity when the two images are from the
same sensor. At that point, the registration process reduces to a displacement transformation estimation problem.

The problem of alignment is to find this transformation. This is a difficult problem because \( F \) can be difficult
to model, and \( T \) may be of high dimensionality.

In this section, we will compare two similarity metrics: correlation and mutual information. The following
sections compare Collignon and Viola's implementations of the mutual information metric.

3.1 Correlation-based Alignment

Cross-correlation is a basic statistical similarity metric used in many registration methods. It is generally
useful for images from the same sensor which are misaligned by small rigid or affine transformations. The metric,
as its name suggests, incorporates the idea behind correlation. Before we define correlation, we need to define
covariance, a concept closely related to variance.

**Definition 15 (Covariance)** If \( E_X[|X|^2] < \infty \) and \( E_Y[|Y|^2] < \infty \), then

\[
\text{Covariance}(X, Y) = E_{X,Y}[(X - E[X])(Y - E[Y])] = E_{X,Y}[XY] - E_X[X]E_Y[Y].
\]

**Definition 16 (Correlation)** If \( \text{Variance}(X) > 0 \) and \( \text{Variance}(Y) > 0 \), then

\[
\text{Correlation}(X, Y) = \left( \frac{\text{Cov}(X, Y)}{\text{Var}(X)\text{Var}(Y)} \right)^\frac{1}{2}.
\]

The correlation of \( X \) and \( Y \) measures the linear association between the RVs. It is possible for \( X \) and \( Y \) to be
dependent but uncorrelated because they are not linearly associated. However, if \( X \) and \( Y \) are independent, then
they are uncorrelated.

How does this translate into a similarity metric? Although the correlation similarity metric has many forms,
a simple definition gives the general flavor [3].

**Definition 17 (Correlation Similarity Metric)**

\[
C(a) = \left( \frac{E[u(x)]E[v(T(x))] - E[u(x)v(T(x))]}{\text{Var}(u(x))\text{Var}(v(T(x)))} \right)^\frac{1}{2}
\]

Where the expectations and variances are sample estimates of the true expectations and variances.

Notice that the similarity metric is the negative of the correlation. If \( u \) matches \( v \) at a transformation \( T \),
\( C(a) \) will have peak cross-correlation. So, by comparing \( C \) across possible transformations \( T \), we can measure
the similarity of \( u \) and \( v \) under various \( T \). The denominator in the definition serves to normalize the covariance
since individual image intensity would affect the correlation, making general image registration difficult.

To decide which \( T \) is best, the registration algorithm usually has to separately recompute \( C \) for all different
rotations, translations, and scalings of interest. Without efficient search routines, the computation becomes
expensive. Over the years, many people have developed optimizations to speed up correlation optimization.
Searching in a smaller space of features, such as edges, speeds up the correlation search considerably.

The real limitation of correlation-based measurements is their inability to deal with dissimilar images. The
metric defined above is optimized for identical matches, and not for a match with a transformed element, for
example by \( F \). If we knew \( F \), we could calculate the correlation between \( v(T(x)) \) and \( F(u(x)) \). However, this is
usually not the case. Sometimes, correlation works adequately when assuming that \( F \) is the identity. However,
for multi-modal images this assumption leads correlation horribly astray.
One solution is to standardize the signal $u$ before applying correlation:

$$
\hat{u} = \frac{u(x) - E[u(x)]}{\text{Var}(u(x))^{1/2}}
$$

This process makes the $\hat{u}$ invariant to multiplicative and additive changes to the original signal. However, this improvement covers only a slightly larger class of functions. Correlation still has difficulty dealing with unknown functional relationships between $u$ and $v$.

### 3.2 Entropy-based Alignment

An alternative similarity metric to correlation is mutual information. As defined previously, the metric measures the reduction in uncertainty of the model $u$ due to the knowledge gained by the image $v$. Mutual information does not explicitly model the transformation $T$; it relies on a $u(x)$ and $v(T(x))$ having high mutual information when $T$ represents the aligning transformation.

Why is mutual information at a maximum when the two signals are aligned? When the signal is of the same object from the same sensor, or $T$ is the identity, the answer is easier to see. We know that $I(X, X)$ or self information, is an upper bound on mutual information, and so we can see that when the signal is aligned with itself mutual information is at a maximum.

However, when the images are from different sensors, the answer is harder to see. Why would the subject object, viewed under a different sensor, look anything like itself when aligned? If a sensor delivers information that could help predict the other sensor’s value, then there should be some relationship between the images. The fourth property of mutual information (if $T$ is a sufficient statistic for $X$ or 1-1 mapping from $X$, $I(X, T(X)) = I(X, X)$), tells us that mutual information will peak at alignment. See Figures 3 and 4 for an example of how a 1-1 transformation does not affect the alignment procedure. So, the closer the images are to having a 1-1 mapping between them, the better mutual information will serve. The basic assumption of mutual information is that the sensors are viewing the same object and represent the data in a related way, where one image could help predict the other. The multi-modal medical images presented in the introduction all fit within these limits.

In order to test how different the signals can be, we have built a one-dimensional simulation and passed various one-dimensional signals through it. The simulation uses the Parzen windowing density estimation to calculate mutual information and takes about 20 seconds to run 60 different translation guesses on a signal of 126 sample points using a shared Sun server (GRIP). We have tested two instances when the image is a 1-1 mapping of the model, one of when the image has zero-mean Gaussian noise added to the model, and two of when the image is a non-function of the model. See Figures 3 through 7.

![Figure 3: The mutual information metric is as effective in aligning a 1-1 mapping of the image as the original image. Notice that the joint entropy is largely responsible for the alignment. The mutual information includes the individual entropies to ensure maximal overlap between the two signals; otherwise, minimizing joint entropy in more complicated images can lead to registration of the backgrounds of distinctive signals (minimizing individual entropies as well.) In this figure, the compared signals of 126 samples over $[0, 2\pi]$ of $u = (10 \sin(i) - 10 \cos(i \ast 3 + 5 \sin(i \ast 10) - 5 \cos(i \ast 20))$ and its negative.](image)

The joint entropy has a similar interpretation to mutual information. Both metrics reflect on how much information one signal has about another. The reason that researchers choose mutual information over joint entropy for image registration is to avoid favoring a transformation that forces the images so far apart that only background is contained in the region of overlap (which would give minimum joint entropy [13]). Considering the formula for joint entropy, $H(X, Y) = H(X) + H(Y | X)$, the joint entropy can be minimized at two points:

- $X$ is the identity
- $Y$ is the identity
when $H(X)$ is small, or when $Y$ is dependent on $X$. However, maximizing the mutual information $I(X; Y) = H(X) - H(X|Y)$ attempts to maximize $H(X)$ while at the same time limiting $H(X|Y)$. The maximal entropy of $H(X)$ ensures that the final region of overlap between the two images contains most of the images, including the complex parts that increase individual entropy.

3.3 Comparison

Mutual information has advantages over correlation. For instance, it can accurately register multi-modal images. Mutual information is able to measure alignment of signals where one can help predict the other, while correlation fails to accurately compare multi-modal signals where one signal would have different values to the other when aligned. However, if one knows how to transform one of these signals to the other, it is simple to apply that transformation and then perform correlation on the normalized signals. At that point, correlation works well.

Correlation’s direct comparison of the two images’ pixels lies at the heart of the difference between the two methods. Mutual information takes three basic units: the entropies within each image and the joint entropy. The first two are completely non-comparative measures. The joint entropy does compare the images, but only by using a frequency-based probability measure. Mutual information measures the ability of the model to predict the image, rather than making a pixel-by-pixel based comparison.

Mutual information does need some form of similarity between the images that it registers. We have explored some of these limits above. However, it does not need as much similarity as correlation.

4 Implementation: Density Estimation

In order to estimate the mutual information, one has to know the probability density of the RV at hand. Unfortunately, the density function of MR / CT / PET images is not easily predicted. For this reason, we need to estimate the density function that gave rise to the data. Collignon and Viola chose two different approaches to the problem, motivated by their need for speed and their desire for a continuous density approximation.

The problem of density estimation is to take a sequence of independent identically distributed random variables $X_1, X_2, \ldots, X_n$ with common probability density function $f(x)$ and then to estimate $f(x)$. In the case that the random variables do not have a continuous probability function, the density estimation problem becomes that.
Figure 6: In this example, we tried to approximate multi-modal images using two non-linear functions of the original signal. For the 126 point used in this Figure, we failed to register the two functions of the original signal \( u \) in Figure 4: \( \sin\left(\frac{1}{8}x\right) \) and \( u_i^2 \). Despite Viola's success with this combination, our results demonstrate a limit to the matching ability of mutual information.

Figure 7: When we increase our sample size to 5000 points, our ability to register images using mutual information improves. We use the same functions to transform the images as in Figure 6, but obtain better results. The situation in this figure better models that of a real medical image, where we would have thousands of sample points.

of estimating the probability of a random variable, where the random variable is assumed to share a common probability function with the sequence of random variables as above. For a more detailed discussion, see the dedicated textbook by Scott [15].

One solution to density estimation is to make strong assumptions about the density we are trying to estimate. We could assume the data is from a certain family, or that the data can be parametrized. This allows us to simplify the estimation process. For instance, if we believe the data is from a Normal density, then the estimation process is simple: find the sample mean and variance, and create an appropriate Normal density function. Since the Normal density is completely specified by the mean and variance, this is a quick process. However, this solution has problems. In many cases, the data is not Normally distributed. More generally, it is hard to parametrize the data to any particular distribution because the data may not be from that distribution. In the case of MR and CT scans, we do not know the form of the density function. We need a more general approach. The two approaches that Viola and Collignon use are both non-parametric; they do not make assumptions about the shape of the density function beforehand.

4.1 Histogramming

The simplest method of density estimation is histogramming. In this method, one creates bins and counts the frequency of sample points in each bin, normalized so that the histogram frequencies all add to one. More formally, we choose a series \( \{t_k \in \mathbb{R} \mid -\infty < k < \infty \} \) and let \( B_k = [t_k, t_{k+1}) \) denote the \( k \)th bin. The histogram has fixed bin width \( h \) if \( t_{k+1} - t_k = h \) \( \forall k \); \( h \) is often called the smoothing parameter. The density histogram uses building blocks of height \( 1/(nh) \) so that the area of each block is \( 1/n \) (where \( n \) is the number of sample points.) If \( a_k \) denotes the bin count of the \( k \)th bin, then the histogram is:

\[
\hat{f}(x) = \frac{a_k}{nh} = \frac{1}{nh} \sum_{i=1}^{n} I_{[t_k, t_{k+1})}(x_i) \text{ for } x \in B_k.
\]

If we see that the bin counts are Binomial random variables with \( v_k \sim \text{Bin}(n, p_k) \), where \( p_k = \int_{B_k} f(z)dz \). The
probability of a Binomial RV X is

\[ p(X = x) = \binom{n}{k} p^k (1-p)^{n-k}. \]

Then

\[ \text{Var}(\hat{f}(x)) = \frac{\text{Var}(a_k)}{(nh)^2} = \frac{p_k(1-p_k)}{nh^2} \]

and

\[ \text{Bias}(\hat{f}(x)) = E(\hat{f}(x)) - f(x) = \frac{1}{nh} E(a_k) - f(x) = \frac{p_k}{h} - f(x). \]

The bins largely determine the histogram's accuracy. If the bins are too wide, the variance of the histogram is low, but the bias is high. If the bins are too narrow, the variance increases, but the bias is small. In the limit, when \( h = 0 \), the histogram is precisely the empirical probability density function, with infinite variance. See Figure 8 for illustration. The crucial task is estimating the bin size that balances the variance and bias.

There are a number of algorithms for determining the correct bin size. Many take an initial look at the data and then decide after the fact how large the bins should be. Unfortunately, this approach takes time. Other approaches try to predict the bin size based on prior knowledge or assumptions. For example, a simple rule for the number of bins is Sturges' rule, developed in 1926 by Herbert Sturges. The rule assumes that an optimally constructed histogram with scaled Normal data can be modeled by a binomial distribution, \( B(n, p = 0.5) \). If the histogram centers on the points \( i = 0, 1, \ldots, k - 1 \) and we choose the \( i \)th bin to be the Binomial coefficient \( \binom{k-1}{i} \), then the total sample size is

\[ n = \sum_{i=0}^{k-1} \binom{k-1}{i} = (1 + 1)^{k-1} = 2^{k-1}. \]

The optimal number of bins is then \( k = 1 + \log_2 n \). Even though Sturges' rule technically gives the optimal number of bins, in practice, Sturges' rule is used for the bin width where the sample range is divided by the optimal number of bins. Of course, this assumes the data is Normal. There are other estimates for bin size that make fewer assumptions. Collignon does not specify how he chooses his bin size.

![Figure 8: The bin width \( h \) of the histogram affects the overall appearance of the histogram. A small \( h \) increases variance, but also increases accuracy. A large \( h \) decreases variance, but increases bias.](image)

One assumption that both Viola and Collignon make is that the sequence \( X_1, X_2, \ldots, X_n \) are independent and identically distributed RVs. Since both methods independently and uniformly sample the images to collect their model points \( u(x) \), they meet this assumption.

The advantages of the histogram are that it takes a short time to make the histogram and calculate probabilities. If the bins have been set beforehand, the histogram can be built in time linear to the number of sample points. The calculation of \( \hat{f} \) then takes merely one lookup. Another advantage of the histogram is that it makes few assumptions about the density it is trying to model.
Figure 9: The chart shows the Parzen density estimate at multiple Gaussian smoothing kernel variances versus an actual \( \mathcal{N}(0, 1) \) curve. As the kernel widens, the estimate degrades. In this example, we used 200 random samples from a \( \mathcal{N}(0, 1) \) density in our approximation. The kernel variances were 0.5, 1, 2, and 3

4.2 Parzen Window Density Estimation
Viola uses the Parzen window density estimate. The idea behind this scheme is similar to that behind histogramming, but we get better smoothness at the expense of more calculation. For the original description of the concept, see Parzen’s 1962 paper [11]. For a more broad description of this method, Duda and Hart’s textbook is helpful [6].

The Parzen window density estimation assumes that it is modelling a continuous probability density function \( f(x) \). It starts with a kernel function \( K \) and gives \( f^*(x, A) \) an approximation to the density \( f(x) \) at a point \( x \) given a sample \( A \) of RVs \( X_1, \ldots, X_n \).

\[
f^*(x, A) = \frac{1}{n} \sum_{X_i \in A} K(x - X_i).
\]

Essentially, this scheme interpolates between the sample points, with each sample point contributing to the estimate in accordance with its distance from \( x \). The kernel function is often called a window function, or smoothing function. Most commonly, the kernel is unimodal, symmetric about the origin, and vanishes quickly. Viola employs a Normal density function of small variance \( \psi \) as his kernel.

Example 18 (Parzen Window Density Estimation) If our sample \( A = \{1, 5, 6, 9\} \) and our kernel \( K = \mathcal{N}(0, \psi) \), then

\[
f^*(2, A) = \frac{1}{4\sqrt{2\pi}\psi} e^{-\frac{(2-1)^2}{2\psi^2}} + e^{-\frac{(2-5)^2}{2\psi^2}} + e^{-\frac{(2-6)^2}{2\psi^2}} + e^{-\frac{(2-9)^2}{2\psi^2}} = 0.0012852 \text{ for } \psi = 0.09.
\]

Clearly, the choice of \( \psi \) affects the density estimate dramatically. If it is too large, the estimate will have coarse resolution. If \( \psi \) is too narrow, the estimate will suffer from too much variability. See Figure 9. So, \( \psi \) approximates the bin width of the histogram.

Viola did a short study on the sensitivity of entropy estimates to the changes in the Parzen kernel. He found that for variances around the optimal variance, the entropy estimate was relatively insensitive to changes in the kernel’s variance. However, his algorithm spends a lot of time optimizing his variance estimate. Also, he only tests out his hypothesis that variance is unimportant to entropy at minima on one sample. We repeated his study for a small number of other samples, and our results agree with his. Entropy is relatively insensitive to small changes in the variance when the variance is close to a minimum. Both of these studies are far from conclusive as they only include a few sample sets. Further study needs to be done.

We would like our density estimate to approach reality. As the sample size of \( A \) increases, it can be shown that the Parzen estimate \( f^* \) is asymptotically unbiased. We give slightly stronger assumptions than necessary because the methods presented do not need the extra freedom of weaker assumptions. The proof requires a few assumptions on the kernel \( K \) and the unknown density function \( f \) [11]:

13
1. $f$ must be continuous ($C^0$) at $x$.

2. $\int_{-\infty}^{\infty} K(\xi) d\xi = 1$

3. $\int_{-\infty}^{\infty} |K(\xi)| d\xi < \infty$

4. $\sup_{\xi} |K(\xi)| < \infty$

5. $\lim_{\xi \to \infty} |\xi K(\xi)| = 0$

We say a function is in $C^2$ if all derivatives of the function of orders up to 2 exist and are continuous. $f \in C^0$ implies that $f$ is continuous. With these assumptions, the Parzen estimate $f^*$ will asymptotically approach the actual $f$.

Since Viola uses a Gaussian kernel, he satisfies the assumptions on $K$, so his density estimate is asymptotically unbiased.

Differentiability is another advantage of using a Gaussian kernel. Since the approximation is a sum of Gaussians, which are in $C^\infty$, the resulting $f^*$ is also in $C^\infty$. The gradient descent method that Viola uses critically depends on this assumption (see below.)

When using the Parzen method, we must be careful that the samples used to represent the density must be different than the samples used to calculate the density approximation at $x$. If we use the same sample to calculate the density and represent the density, as $\psi$ goes to 0, the probability will tend to oo, bringing the entropy estimate becomes arbitrarily large. If instead, we make sure to use different samples to calculate the density and represent the density, then the chances of exact overlap at each point are minimal. Then, as $\psi$ goes to 0, the approximated probability will tend to 0, bringing the entropy estimate closer to oo. The algorithm that Viola uses to find the optimal $\psi$ sees this increase in entropy and avoids the degenerate solution of $\psi = 0$.

4.3 Comparison

The first item of comparison between the histogramming method and the Parzen method is that they are quite similar. In fact, if we set the kernel to a unit square pulse, each term in the density estimation sample will contribute to the sum once. At this point, the density estimator behaves much like the traditional histogramming method. So, the Parzen window density estimation method includes the histogramming method.

When we compare the two methods on speed, we see that histogramming is fast. For example, take 100 points. If we conservatively use an extremely crude model of computation where a single calculation includes the complex action of looking up a Normal density value, the time consuming task of adding an array of numbers, and the extremely simple task of looking up a value in an array, we will underestimate how fast histogramming is. We would like to estimate a probability distribution. The histogram would take 1 step to bin each point and total the bins, assuming we could easily preset the range of the bins. After that, it would take 1 step to look up each point's probability, bringing the total to roughly 200 calculations. For Parzen window density estimation, if we conservatively chose 40 points to represent the sample and 60 points for which to calculate probability, then it would take no time to set up the density. However, it would take 40 smoothing function calls for each of the 60 points for which we would like to estimate the probability, bringing the total to roughly 2,400 calculations. Although these calculations are rough, the idea is that Parzen windowing is more expensive.

Parzen windowing provides a smoother solution to the problem. At the bin boundaries, the histogram will deliver a discontinuous probability. The Parzen estimation scheme gives a density that has infinitely many derivatives. Even with the smoothness, Parzen windowing can be prone to error. The variance of the kernel has a large impact on the accuracy of the density estimate (see above.) In addition to providing a smooth estimate, the Parzen method assumes it is approximating a smooth density. This assumption may not be true. In this case, the Parzen estimate will not converge asymptotically to an accurate estimate. The extent of the damage depends directly on the extent of the discontinuity of the density.

Which method is best? There is no clear answer. For accuracy, Parzen windowing provides a more continuous density estimate, more closely hugging the real density function if the real density is continuous. The downside is that the scheme uses greater computational resources. For current medical applications, this expense matters little, but future applications may require faster registration.
5 Implementation: Interpolation Artifacts

The alignment algorithms compare the model and image signals, \( v(T(x)) \) and \( u(x) \), using a transformation \( T \). In many cases, the signals are not continuous, but rather discrete sequences of sample points. When \( T \) maps \( u(x) \) to a place in between two points of \( v \), the algorithm uses interpolation to make a guess at the value of \( v(T(x)) \). Collignon and Viola use two different interpolation methods, each having a different effect on the entropy estimate. Piezas, Mainz, and Viergever compare these methods in detail [13].

We study interpolation because it affects the results of the entropy maximization. Although the numerical result of the interpolated entropy calculation approaches what it should be without interpolation, the shape of the resulting curve often changes significantly. Interpolation could shift the peak of the mutual information curve erroneously; it could even create a local minimum at the true peak.

5.1 The Two Methods

The first method we discuss is called trilinear interpolation, or in the two dimensional case bilinear interpolation. Viola chooses this method, and in some sense, it is the most intuitive method. An example for a two dimensional image is presented in Figure 10. In most cases, the transformed point \( v(T(x)) \) does not coincide with signal values of \( v \), and so requires interpolation. For a two dimensional image, the transformed value falls between four actual values in the image. Bilinear interpolation estimation forms \( v^*(T(x)) \) using the weighted average of the four nearest neighbors where the weighting depends on the relative position of \( v(T(x)) \). If we are using histogramming, the histogram entry corresponding to \( (u(x), v^*(T(x))) \) is incremented by one. If we are using the Parzen windowing scheme, then the data point \( (u(x), v^*(T(x))) \) is entered into the sample from which to estimate the density.

\[
\begin{align*}
& v(x_1) & v(x_2) \\
& v(x_3) & v(T(x)) \quad v(x_4) \\
\end{align*}
\]

**Figure 10:** Bilinear interpolation gives the value at the point \( v(T(x)) \) using the formula \( v^*(T(x)) = w_1 G(v(x_1)) + w_2 G(v(x_2)) + w_3 G(v(x_3)) + w_4 G(v(x_4)) \), where the \( x_i \) are the four nearest neighbors of \( v(T(x)) \) in \( v \), the weights \( w_i \) are the surface areas indicated in the image, and \( G(z) \) denotes the intensity at pixel \( z \). Bilinear interpolation then uses \( (u(x), v^*(T(x))) \) in the density estimation sample. Partial volume interpolation uses the same four nearest neighbors of \( v(T(x)) \), but instead of using the one value \( (u(x), v^*(T(x))) \) in the density estimation sample, it increments the four values \( (u(x), v(T(x))) \) in the density estimation sample.

Collignon formulated the second method, called partial volume interpolation. In his formulation using histograms, he updates several histogram entries for \( v(T(x)) \); for each nearest neighbor \( x_i \), the joint histogram entry \( (u(x), v(T(x))) \) is incremented by the weight \( w_i \), where \( w_i \) is calculated as in bilinear interpolation. No new, and therefore false, intensity values are formed. The process also keeps the changes to the histogram relatively slow at each step.

5.2 Problems and Solutions

The two interpolation methods have different effects. Partial volume interpolation increases the joint entropy. The reasoning is interesting. Partial volume updates multiple entries in the joint density function, creating dispersion resulting in a larger joint entropy value. Subsequently, the mutual information decreases.

The effects of bilinear interpolation on mutual information are more varied (see Figure 11.) On one hand, if the image has relatively few gray values, the interpolation process creates new averaged gray values. These new values further disperse the histogram (spread out the Parzen density), and decrease joint entropy. On the other hand, if the image has many gray values, the interpolation process averages the diverse values into fewer values, reducing joint entropy and increasing mutual information. Real images have noise, which introduces many new values and creates the latter situation.

Interpolation does not harm higher resolution images as much. Because we have more detailed knowledge of the signal, we are interpolating less. When we use a multi-resolution approach, the harmful influence of interpolation grows. We are required to interpolate between coarse values of the image, introducing more artefact patterns.
One might argue that these artefacts are small. However, Pluim has shown that in clinical images, even errors that may seem correct to a human observer pose serious problems. Interpolation artefacts hurt accuracy by masking sub-voxel accuracy. On the other hand, Collignon finds over 2mm in error when registering a CAT scan with a MR image. Thus, the error for multi-modal registration is large for many reasons. Interpolation artifact plays only a minor role in the accuracy error when using high resolution imagery. In contrast, interpolation has a large effect on the maximization strategy. Both interpolation methods shown create many artificial local maxima in the mutual information curve, potentially throwing off a solution. Collignon acknowledges that trilinear interpolation would find a false maximum far away from the real solution. Partial volume interpolation smoothes out the mutual information curve, providing an easier function for maximization.

Pluim points out a method to smooth out the interpolation artefacts: ensuring that the voxel sizes are different. In this way, grid alignment is impossible and artefact from interpolation will uniformly apply to the image, making the artefact less noticeable. A specific way to accomplish this task is to resize each image slightly so that the voxel sizes are different. In the paper, Pluim rescales the MRI in-plane voxel size from 0.9768 mm to 0.9842 mm. The resulting mutual information curve appears much smoother. Resampling the images provides a less practical way to achieve the rescaling of voxel size. The smoother curve has fewer local maxima to foil the maximization algorithms.

A consequence of this analysis is that multi-modal registration between sensors with different voxel sizes will suffer less from interpolation artefacts. Neither Collignon nor Viola explicitly rescale their images, and so, the different scales used in their multi-modal registration examples may have saved them from their interpolation method’s worst case scenarios.

6 Implementation: Maximization

As a part of the registration algorithm, we need to find a ‘best match’ $u(T(x))$ for the subsample in $u$. In our case, this involves finding the maximum mutual information between the two subsamples. The simplest approach would waste incredible resources by calculating the mutual information between $u(x)$ and $(u(T(x)))$ for each possible transformation $T$. Both Collignon and Viola used cheaper methods. Collignon used Powell’s method, which does not require explicit derivative calculations and searches along lines for the best transformation. Viola uses the Parzen window estimate form to calculate a derivative estimate of $df$. Using this derivative estimate, he employs hill-climbing to find the maximum mutual information.

6.1 Powell’s Method

Collignon chooses to make fewer assumptions about the gradient of his entropy. He uses Powell’s direction set method with Brent’s one dimensional optimizer (see [14] for detail.)

Powell’s method to maximize a multi-dimensional function by breaking the transformation up and searching along one-dimensional lines. For instance, for a six dimensional transformation $T$, Powell’s method would find a set of directions to search and perform six one dimensional line optimizations. The set of one-dimensional lines, or directions, is known as a ‘direction set.’ At the end of the search along the first direction, the method searches along the second, starting at the end point of the first search. The method cycles through the whole set
of directions as many times as necessary, or desired, to converge to a solution.

Naturally, the method would like to find a ‘good’ set of directions, where ‘good’ implies two conditions: 1) the direction takes the method along a steep path, finding maximal points more quickly and 2) the maximization along the direction is not spoiled by maximization in subsequent directions. The second condition, more commonly called the search for ‘conjugate directions’, can be made more explicit. The directions \( u \) and \( v \) are conjugate when \( u \cdot A \cdot v = 0 \), with \( A \) as the second derivative of a point \( P \).

The method takes a long time to converge if the direction set is bad. For example, if the surface we are searching has a narrow valley at an angle to the the direction vectors, the algorithm will take tiny steps in each direction, converging slowly. This degenerate condition happens when the second derivative is much larger in magnitude in some directions than others. To avoid this situation, the algorithm adapts its direction set during operation. Over time, continually adjusting the direction set can lead to directions that are similar. In this case, we would only be searching a subspace of the functional range. One version of Powell’s method avoids this solution by discarding the direction of greatest increase at each iteration. This seems paradoxical since this direction was just so helpful. However, this direction will be substantially included in the next direction and keeping both the new and the greatest increase directions around will only contribute to the growing similarity of the directions.

Algorithmically, the method starts with the set of directions \( u_i \) set to the basis vectors, for \( i = 1, \ldots, N \). Then, the algorithm cycles through the following steps until either it times out or converges:

1. Save the starting position \( P_0 \);
2. For \( i = 1, \ldots, N \), move \( P_{i-1} \) to the maximum along direction \( u_i \) and call this point \( P_i \);
3. Set \( u_{\text{new}} := P_n - P_0 \);
4. Replace the direction \( u_i \) along which the largest increase was achieved by \( u_{\text{new}} \);
5. Move from \( P_n \) along direction \( u_{\text{new}} \) to the maximum in that direction.

For some situations, we keep the old set of directions in the next iteration. Basically, the situations happen when the new candidate direction \( P_n - P_0 \) seems bad or when the old direction set still appears promising. We measure these quantities by looking at \( P_n - P_0 \) and the gradient.

Powell’s method does not specify the one-dimensional maximizer. Collignon chooses to use Brent’s method for one-dimensional maximization. This method assumes that the maximum has already been bracketed. The search to bracket the maximum assumes that for a function \( f \), \( a \) and \( b \) contain the maximum at \( b \) when \( f(a) \) and \( f(c) \) are both less than \( f(b) \). From an initial point \( a \), the bracketing algorithm moves toward the highest peak and sets \( b = a + \delta \), and \( c = b + 1.61803(b-a) \) following the golden section search ratio. At each step, the bracketing algorithm proceeds only if \( f(c) \geq f(b) \) by fitting a parabola to the three points. The algorithm shifts points around the maximum of this fitted parabola, keeping \( f(a) \leq f(b) \), and iterates until \( f(c) < f(b) \).

Brent’s method may now continue. The method keeps track of six function points (not necessarily distinct) \( a, b, u, v, w \) and \( x, x \) represents the point with the maximal function value found to date; \( w \) represents the second greatest function value, \( v \) is the previous value of \( w \), and \( u \) is the point at which the function was evaluated most recently. \( a \) and \( b \) bracket the maximum. At each stage, the algorithm interpolates a parabola around points \( x, v \) and \( w \). We require the step to (i) fall within \( (a, b) \) and (ii) move from \( x \) to a value less than half of the movement during the step before last. The second criterion ensures convergence. The method evaluates the function at the maximum of the parabola to get \( u \). The method then updates the variables \( v \) and \( w \) and continues a specified number of iterations or until it achieves convergence.

One potential advantage of Powell’s algorithm is that it only requires function calls to evaluate the maximum. It does not require potentially expensive derivative calculations, which may not be feasible for the histogram approximation used to calculate the mutual information curve. Thevenaz and Unser [17] found that a single iteration of a derivative-based mutual information maximization could take up to twice as long as a single iteration of Powell’s algorithm.

Unfortunately, the overall inefficiency of the algorithm makes derivative-based methods faster, despite Powell’s fast single iteration speed. The parabolic approximation’s inaccuracy makes Powell’s algorithm need extra time to converge; the bracketing costs several function estimations; and the order of the line optimizations in the direction set can have a large effect on the computation time. Although a single iteration for Powell’s algorithm
may be faster than derivative methods, over the long term, these inefficiencies double the time needed compared to mutual information derivative methods [17].

The algorithm will find the maximum it has bracketed. However, the maximum might be a local maximum, not even close to a global maximum. The bracketing algorithm depends on three points. The low number of points allows the possibility for the algorithm to catch a noisy point for \( a \) or \( b \), thereby bracketing a local maximum far away from the true maximum.

**6.2 Stochastic Gradient Descent**

Viola uses an approximation to gradient descent to find the maximum mutual information. His hill-climbing routine relies on the derivative of the mutual information in the image, which in turn relies on the derivative of the empirical entropy, an estimate of the true entropy.

First, he derives the form for the derivative of the entropy. Normally, the derivative of entropy is hard to compute. In this case, the Parzen approximation to the density yields a surprisingly tractable formula for the derivative of entropy. The Parzen estimate of the entropy is

\[
H^*(Y) = \frac{1}{N_b} \sum_{y_b \in b} \log(P^*(y_b, a))
\]

Where \( H^* \) is the approximation of the entropy, \( P^* \) is the Parzen density approximation, \( a \) is the sample used to approximate the density, and \( b \) is the sample of size \( N_b \) used to calculate the empirical entropy.

Let the RV \( v(T(x)) \) be a function of the parameters \( T \) and the RV \( u(x) \) for a random sample \( x \) in the model. If we abbreviate \( v_i = v(T(x_i)) \), then the derivative of this entropy estimate can be expressed as

\[
\frac{d}{dT} H^*(v(T(x))) = -\frac{1}{N_b} \sum_{y_b \in b} \sum_{v_a \in a} \frac{\partial g(v_b - v_a)}{\partial v} \psi^{-1}(v_b - v_a)
\]

where the Parzen estimate kernel is Gaussian.

After differentiating the Gaussian,

\[
\frac{d}{dT} H^*(v(T(x))) = -\frac{1}{N_b} \sum_{y_b \in b} \sum_{v_a \in a} g(v_b - v_a)(v_b - v_a)^T \psi^{-1} \frac{\partial g}{\partial v}(v_b - v_a)
\]

This can be compactly rewritten as,

\[
\frac{d}{dT} H^*(v(T(x))) = -\frac{1}{N_b} \sum_{v_b \in b} \sum_{v_a \in a} W_v(v_b, v_a)(v_b - v_a)^T \psi^{-1} \frac{\partial}{\partial v}(v_b - v_a)
\]

if we set

\[
W_v(v_b, v_a) = \frac{g(v_b - v_a)}{\sum_{v_b \in b} g(v_b - v_a)}
\]

The corresponding formula for the derivative of mutual information is,

\[
\frac{d}{dT} (I(u(x), v(T(x))) = H(u(x)) + H(v(T(x))) - H(u(x), v(T(x)))
\]

Since, \( H(u(x)) \) does not depend on \( T \),

\[
\frac{d}{dT} (I(u(x), v(T(x))) = \frac{d}{dT} H(v(T(x))) - \frac{d}{dT} H(u(x), v(T(x)))
\]

Using the formula for the derivative of entropy, derived above, and assuming that \( \psi^{-1} = \text{DiagonalMatrix}(\psi^{-1}_u, \psi^{-1}_v) \),

\[
\frac{d}{dT} = \sum_{x_i \in B} \sum_{x_j \in A} (w_i - w_j)^T [W_v(v_i, v_j)\psi^{-1}_v - W_w(v_i, v_j)\psi^{-1}_w] \frac{d}{dT}(w_i - w_j)
\]
Where \( w_t = [u(x_t), v(T(x_t))]^T \).

Usually, taking the derivative of entropy is difficult. The general form is much more complex. As an example:

\[
\frac{dH(T)}{dT} \approx \frac{1}{N} \sum_{y_b \in b} \log(P(y_b; \alpha))
\]

\[
= \frac{1}{N} \sum_{y_b \in b} \frac{dP(y_b; \alpha)}{P(y_b; \alpha)}
\]

\[
= \frac{1}{N} \sum_{y_b \in b} \frac{dP(y_b; \alpha) \log + dP(y_b; \alpha) \log}{P(y_b; \alpha)}
\]

The second term of the final numerator is hard to calculate; it represents the change in density estimate when \( \alpha \) changes. In the Parzen scheme, the two numerator terms merge and are easy to calculate. In other frameworks, this may not happen. The general form of the derivative of entropy presents difficulty for most algorithms.

Armed with this derivative, Viola uses a standard hill-climbing routine to find a local maximum of the mutual information. The algorithm updates the estimate for the registration transformation \( T \) at each stage using the formula:

\[
T_{n+1} = T_n + \lambda \frac{dT}{dT}
\]

The algorithm iterates a fixed number of times or until it detects convergence. The parameter \( \lambda \) is called the ‘learning rate’; it usually decreases as the procedure runs. A small \( \lambda \) slows down convergence as the algorithm takes ‘baby steps.’ A high \( \lambda \) forces the descent to make large bounds, potentially overstepping the maximum and not converging.

The algorithm differs from regular gradient descent in that it uses an estimate for the derivative rather than the actual derivative. Since Viola uses a randomized method, we should examine under which conditions we expect to achieve the same results as normal gradient descent. Viola gives three summary conditions for convergence:

- The gradient estimate is unbiased;
- The parameter update rate converges to 0;
- The error surface is quadratic in the parameters.

Viola shows that his derivative estimate is slightly biased, but can be corrected to be less biased. To do this, he bounds his estimate between the true \( E[P(X)] \) and \( E[\log(P(X))] \). He then devises a method to choose between estimates of the two bounds and shows that he has been successful in using this method. However, he does not show that the term is unbiased.

For his second criterion, he uses a series of \( \lambda \)s that almost converge to 0, which works for the tolerances of his computing system. For his third criterion, he shows that the third assumption can be reworded. All he needs is that the algorithm enter the ‘basin of attraction’ of the maximum infinitely often. A ‘basin of attraction’ of an optimum is a set of points from which the true gradient descent will converge to the same optimum. Since we have a finite number of optima and an infinite limit on our search time, he shows that the algorithm must be in one of the basins infinitely often.

We believe the term ‘stochastic approximation’ is a misnomer for the method that Viola chooses. A standard definition of stochastic approximation comes from Albert and Gardner [1]. Let \( \{Y_n : n = 1, 2, \ldots\} \) be a stochastic process whose mean-value sequence is a member of a family of known sequences, \( E[Y_n] = F_n(\theta) \), where \( \theta \) is a vector parameter which must be estimated. The stochastic approximation to \( \theta \) is given by:

\[
t_{n+1} = t_n + a[Y_n - F_n(t_n)],
\]

where \( t_{n+1} \) is the estimate of \( \theta \) based upon the first \( n \) observations and \( \{a_n\} \) is a suitably chosen sequence of ‘smoothing vectors.’ Notice that the update depends on the prediction error \( Y_n - F_n(t_n) \). For this reason, these schemes are often called ‘differential correction’ procedures. Viola’s updated derivative does not depend on the error of a previous prediction of the derivative of mutual information; it comes only from the current sample. For this reason, traditional stochastic approximation proofs do not apply to Viola’s method. Because Viola does not use differential correction, Viola’s assumptions bear no resemblance to the assumptions Albert and Gardner use.
to prove convergence for stochastic approximations. Viola’s method will approximate a maximum because it gives a relatively unbiased gradient estimator and uses gradient descent. However, this is not technically stochastic approximation.

One danger of hill-climbing gradient descent schemes is that we may climb the wrong hill. In other words, we find a local maximum, believe it’s a global maximum because the derivative is 0, and we stop. The problem is that the derivative is 0 at all maxima, and we are using it to look for the largest maximum. Viola claims that the noise introduced by the sampling allows his stochastic gradient-descent to effectively skip over narrow local maxima. He claims alignment techniques often give rise to these narrow maxima because of false matches between high frequencies. The question that remains is how wide does a local maximum have to be to halt the hill climbing method? Viola does not offer a detailed analysis of how his algorithm avoids such peaks. As seen in this paper, the mutual information algorithm produces many local maxima (see Figure 3.) We have no way to predict whether the maximization scheme would find the true peak.

In sum, the dependability and accuracy of algorithm have not been specified fully. Nevertheless, Viola shows that the algorithm works in practice as a good approximation to gradient descent.

6.3 Comparison

The two maximization methods are quite different. Powell’s method makes few smoothness assumptions about the function for which it finds the maximum, As a result, the algorithm often takes more time to compute than gradient descent. This judgement is case specific; if the derivative calculation takes much longer to compute than a function call, Powell’s algorithm will be faster. Viola’s approximation algorithm relies on a derivative estimate to compute each iteration. Although each iteration of Viola’s gradient descent is slower, the overall efficiency of the algorithm makes it faster than Powell’s method.

In terms of accuracy, both methods only search for local maxima. Thus, they can easily mistake the forest for the trees. Viola claims that the noise inherent in his probabilistic approximation masks the local maxima, enabling the algorithm to find the true maximum. However, he does not show the limits of this assertion. The algorithm might still find local maxima just big enough to break through the noise. Collignon does not address this issue, except during a brief comparison of interpolation methods. As a result, the methods may require multiple runs at different initialization points to guarantee convergence to a global optimum.

7 Implementation: Multiresolution

The algorithms of both Collignon and Viola are slower than need be. A few authors have tried to speed up the mutual information registration using multiresolution [12][17]. The results have shown that multiresolution approaches can give more accuracy at higher speed.

The basic idea behind multiresolution is to first solve the problem (find the transformation maximizing mutual information) at a coarse scale where the image has fewer data points. The solution propagates to initialize the next step, which finds an optimal transformation in a larger, more refined image. The process iterates until it reaches the finest possible scale. The images of different levels of resolution form a “pyramid”, where the top level is the smallest, coarsest image and the bottom level is the largest, finest image.

The method that samples images to form the pyramid can have a large effect on computation. For example, creating a separate image for each level of the pyramid can be quite expensive. On the other hand, simply sampling the full image at appropriate points at runtime can save time. Pluim claims the sampling method can increase accuracy over pyramid building; however the mechanism is unclear. Sample points should approximate the pyramid-building process perfectly. Additionally, without smoothing, the sampled images can have artifacts. Pluim noticed artificial maxima when using a simple sampling scheme. There are implementation details, but they can have a significant effect on the outcome of the algorithm.

Multiresolution approaches strengthen the robustness of the optimization algorithm because it eliminates local optima at coarser scales. In other words, there are fewer details at coarser scales, so there are fewer local maxima to trap optimization algorithms. Also, the smoothing operations we use to subsample the images eliminates local maxima. So, the optimization algorithms will have fewer false maxima to avoid, making them more likely to find the right maximum. The better estimates from the coarser scales propagate to the finer scales. So, the algorithm should stay close to the global maximum even while traveling coarse to fine. Thevenaz and Unser found that their optimizer would get lost in local maxima without the use of multiresolution.

The multiresolution approach speeds up the process of registration. In order for this to happen, the total time taken must be less than the time taken at the finest level without a multi-resolution strategy. This only happens
when the optimizer from a coarser scale delivers a good estimate to the next step, and the next step capitalizes on the situation by quickly finding an optimum. Powell's algorithm is a bad algorithm in this respect because it searches all directions in its direction set at least once regardless of how close the initial value is. Gradient
based approaches take more advantage of the potential speedup from a close initial configuration. Thevenaz and Unser found that total time for registration went down about 90% with extreme improvement in accuracy using a four-level pyramid in their gradient-based optimization approach. Plum found speedups of about three times when calculating using a pyramid of levels 1/4, 1/3, 1/2, and 1 of the original image. One hypothesis for the difference is the wastefulness of Powell's algorithm in this setting.

As mentioned previously, interpolation becomes extremely important in a multiresolution framework. At a coarse level, the steps made by the optimizer correspond to large steps in the finest image. At coarser levels, when we calculate \( v(T(x)) \) from our \( u(x) \), we are interpolating between farther and farther actual points. Incremental accuracy in the interpolation method helps more and more.

8 Discussion

Mutual information has become a viable solution for image registration. The two papers compared in this paper present different implementations of the mutual information scheme. The authors had different motivations. Collignon was only trying to register 2D medical images, and more specifically, multi-modal medical images. Viola was looking for a general solution to the alignment problem in any dimension. Unlike Viola, Collignon used off-the-shelf maximization schemes and did not try to develop derivative maximization schemes. The derivative maximization required a smooth entropy curve, and so Viola used Parzen density estimation. Collignon was free to use histogramming, which is faster but does not give a continuously differentiable curve. Viola used classical trilinear interpolation, while Collignon developed a newer interpolation method which he found to give fewer false maxima. In the end, the algorithms have reasonably similar performance.

<table>
<thead>
<tr>
<th>Algorithm Component</th>
<th>Viola and Wells</th>
<th>Collignon et al.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature Space</td>
<td>Pixel Values</td>
<td>Pixel Values</td>
</tr>
<tr>
<td>Search Space</td>
<td>Affine Transformations 3D→2D, 2D→2D</td>
<td>Affine Transformations 2D→2D</td>
</tr>
<tr>
<td>Search Strategy</td>
<td>Probabilistic Gradient Descent</td>
<td>Powell's Method</td>
</tr>
<tr>
<td>Similarity Metric</td>
<td>Mutual Information</td>
<td>Mutual Information</td>
</tr>
<tr>
<td>Interpolation Method</td>
<td>Trilinear Interpolation</td>
<td>Partial Volume Interpolation and Trilinear Interpolation</td>
</tr>
<tr>
<td>Density Estimation Method</td>
<td>Parzen Windows</td>
<td>Histogramming</td>
</tr>
<tr>
<td>Multi-Resolution</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>

Figure 12: The chart summarizes the similarities and differences between Viola and Collignon’s papers. See individual sections for in depth discussion and comparison of each component.

In measuring the performance, Viola and Collignon are slightly unclear about their methods. For MR to MR, both authors transform an image to create a model. Thus, they know the actual transformation \( T \) and can easily compare the resulting transformation for error. Both algorithms achieve subpixel accuracy. For MR to CT registration, Viola does not give error measurements. Collignon gives a number (2 mm), but does not say how he computed it. If a human operator is considered ground truth, then the error measurement variability might be much greater than the error itself. The multi-modal error measurement method needs clarification. There is a study favorably comparing two implementations of Collignon's mutual information method with the bone implant method, but there are no known studies comparing Viola's derivative-based method to other methods yet.

In summary, the mutual information method has much promise. Like any similarity metric, mutual information has difficulty coping with extremely different signals. However, it has already been used to register images with which other registration algorithms have had problems, and it has performed successfully. Work to improve the speed and accuracy of the algorithms continues.

References


