Self-Consistency and MDL: A Paradigm for Evaluating Point–Correspondence Algorithms, and Its Application to Detecting Changes in Surface Elevation

Yvan G. Leclerc, Q.-Tuan Luong and P. Fua *
Artificial Intelligence Center, SRI International, Menlo Park, CA
{leclerc,luong}@ai.sri.com
EPFL, Lausanne, Switzerland
pascal.fua@epfl.ch

Revised Submission VISI351-OOF

Abstract

The self-consistency methodology is a new paradigm for evaluating certain vision algorithms without relying extensively on ground truth. We demonstrate its effectiveness in the case of point–correspondence algorithms and use our approach to predict their accuracy.

For point–correspondence algorithms, our methodology consists in applying independently the algorithm to subsets of images obtained by varying the camera geometry while keeping 3-D object geometry constant. Matches that should correspond to the same surface element in 3-D are collected to create statistics that are then used as a measure of the accuracy and reliability of the algorithm. These statistics can then be used to predict the accuracy and reliability of the algorithm applied to new images of new scenes.

An effective representation for these statistics is a scatter diagram along two dimensions: A normalized distance and a matching score. The normalized distance make the statistics invariant to camera geometry, while the matching score allows us to predict the accuracy of individual matches. We introduce a new matching score based on Minimum Description Length (MDL) theory, which is shown to be a better predictor of the quality of a match than the traditional Sum of Squared Distance (SSD) score.

We demonstrate the potential of our methodology in two different application areas. First, we compare different point–correspondence algorithms, matching scores, and window sizes. Second, we detect changes in terrain elevation between 3-D terrain models reconstructed from two sets of images taken at a different time.

We finish by discussing the application of self-consistency to other vision problems.

1 Introduction

Our visual system has a truly remarkable property: given a static natural scene, the perceptual inferences it makes from one viewpoint are almost always consistent with the inferences it makes from a different viewpoint. We call this property self-consistency, with respect to natural scenes.

*This work was sponsored in part by the Defense Advanced Research Projects Agency under contract F33615-97-C-1023 monitored by Wright Laboratory. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, the United States Government, or SRI International.
Note that saying that our visual system is self-consistent is not the same thing as saying that it makes complete and accurate inferences about the world for a given viewpoint. Indeed, this would be impossible because there is never enough information from a single viewpoint to make all possible correct inferences for the visible part of the world. However, if such an impossible system did exist, it would necessarily be self-consistent.

One of the ultimate goals of our research is to design computer vision algorithms that also have the property of being self-consistent, with respect to given classes of scenes and imaging conditions.

As a first step towards this ultimate goal, we have developed the self-consistency methodology for empirically estimating the self-consistency of a computer vision algorithm given many images of many scenes (within a given class of scenes imaged within a given class of imaging conditions). When estimated from a sufficient number of images and scenes, these self-consistency statistics can be used to predict the behavior of the algorithm when it is applied to new images of new scenes.

The specific goals of this paper are to: 1. characterize the overall accuracy of a point-correspondence algorithm, 2. characterize the accuracy of individual matches produced by the algorithm, and 3. use this characterization to predict the accuracy of the algorithm and individual matches in new images.

To satisfy these specific goals, we present a specific instance of the self-consistency methodology developed for point-correspondence, or "stereo," algorithms, which constitute one of the most important classes of computer vision algorithms. Section 8 describes a formalization of our methodology that can be applied to some, but not all, computer vision algorithms.

Note that one way of characterizing the overall accuracy of a point-correspondence algorithm is in terms of the error of matches compared to "ground truth." If sufficient quantities of accurate ground truth were available, estimating the statistics of the errors over many image pairs of many scenes would be relatively easy. Unfortunately, acquiring ground truth for many scenes is an expensive and problematic proposition at best. Our methodology, on the other hand, automatically estimates the related self-consistency statistics from the matches of many image pairs of many scenes. The only additional information we need is the precise estimation of the camera parameters or projection matrices and, optionally, of their covariances.

Assuming that bias has been removed, a task which can be done with a minimal amount of ground truth data, the self-consistency statistics provide a global characterization of the accuracy of a stereo algorithm. Furthermore, the statistics can be used to predict the accuracy of the algorithm when applied to new images of new scenes.

To predict the accuracy of individual matches, we associate a number, or "score," with each match, which is meant to be a reliable predictor of its accuracy. Although there has been much work in analyzing the predictive power of such scores as Cross Correlation (CC) and Sum of Squared Distance (SSD), these theoretical predictions can only be derived for very simple scene geometries and imaging conditions (typically, images of planar textured Lambertian surfaces corrupted by white noise). Furthermore, these analyses say nothing about the probability of a match being an outlier. The self-consistency methodology allows us to empirically measure the predictive power of a score with respect to a given algorithm, class of scenes, and class of imaging conditions. Thereafter, these self-consistency statistics conditioned on the score can be used to predict the accuracy of individual matches in new images of new scenes.

The three next sections introduce the components of the self-consistency methodology applied to stereo or multiple-image point-correspondence algorithms.

In Sec. 2, we introduce the basic methodology and the self-consistency statistics. The statistics are represented in several ways to illustrate different aspects of the experiments. The first important element in estimating the statistics is a normalization of match pair triangulation dis-

---

1For brevity, when we use "self-consistent" or "self-consistency" in the remainder of this paper, we mean "self-consistent (cy) within a given class of scenes imaged within a given class of imaging conditions."

2For brevity in the remainder of the paper, statements about the prediction of algorithms refer to images within the given class of scenes and imaging conditions.
stances that make the statistics invariant to camera geometry and choice of coordinate systems (Sec. 3). The second important element is conditionalization of the statistics with respect to a score so that the accuracy of individual matches can be characterized and predicted (Sec. 2.3).

In Sec. 4, we use our methodology to empirically compare two stereo algorithms, compare three scores, including an original MDL-based score described in the appendix, and measure the performance of a stereo algorithm as its window size changes.

In Sec. 5 we apply the self-consistency methodology to the problem of detecting change in the elevation of terrain given two sets of images taken at a different time. The last three sections place self-consistency in a broader perspective. In Sec. 6, we compare our approach to previous work on measuring uncertainty and detecting change. In Sec. 7, we show how the self-consistency methodology can be used for several vision algorithms beyond stereo. We conclude in Sec. 8 and point to interesting possibilities opened by the new formalism.

Self-consistency and the resultant methodology are very simple ideas that have powerful consequences beyond the applications described in this paper. The methodology can be used to compare algorithms, compare scoring functions, evaluate the performance of an algorithm across different classes of scenes, tune algorithm parameters, and so forth. We hope that others will be able to use this methodology in new and exciting ways.

2 Self-Consistency of Stereo Algorithms

2.1 The basic idea

The self-consistency methodology is a recipe for doing experiments. It requires that the experimenter take care in preparing the data and following the stipulations. Otherwise, the results will be meaningless.

We start the experiment with a fixed collection of images taken at exactly the same time or, equivalently, a collection of images of a static scene taken over time. Each image has a unique index and associated projection matrix and, optionally, projection covariances, which are stipulated to be correct. This projection matrix describes the projective linear relationship between the 3-D coordinates of a point in a common coordinate system, and its projection in the image. Although the methodology is easier to apply when the common coordinate system is Euclidean, the minimal requirement is that the set of projection matrices use a common projective coordinate system. This could be obtained from point correspondences using projective bundle adjustment [20, 28] and does not require camera calibration.

We then apply a stereo algorithm independently to all pairs of images in this collection. A stereo algorithm takes as input a pair of images of a stationary scene, and their associated projection matrices $P_1$, $P_2$. The output of the algorithm is a set of matches $m_{ij}$, where each match is a pair of 2-D points, one per image, that are stipulated to lie on corresponding epipolar lines. Each match is an assertion by the stereo algorithm that the 2-D points represent the 2-D projections of a single point in the world. Optionally, the stereo algorithm can associate a score $s_j$ with each match $m_{ij}$. A low value of the score is an assertion by the stereo algorithm that the match is certain, and a high value is an assertion that the match is uncertain. For each image pair processed, the image indices, match coordinates, and score are reported in a match file.

We now search the match files for pairs of matches that have a common point in one image. If the stereo algorithm provides the covariances of the matches, we can derive the probability that a pair of matches has a common point, and take that into account in the self-consistency statistics below by thresholding or weighting. Otherwise, we assume a nominal 1 pixel accuracy for the precision of localization.

---

³For brevity, we describe the methodology for "stereo" algorithms only, but the methodology is applicable to (and had been implemented for) point-correspondence algorithms that take image $n$-tuples as input and produce matches with up to $n$ points as output.

⁴A measure of the localization of the match, as opposed to the score defined above which is a measure of the confidence of the match.
Figure 1: Two matches that have a common point in one image should correspond to the same point in the world.

For example, if a match is derived from images 1 and 2, another match is derived from images 1 and 3, and these two matches have a common point in image 1, then this is an implicit assertion by the stereo algorithm that these two matches are the 2-D projections of a single point in the world, as illustrated in Fig. 1. This pair of matches, or common-point match pair, should be self-consistent because they should correspond to the same point in the world.

Given a common-point match pair we now compute the distance between their triangulations, after normalizing for the camera configurations (see Sec. 3). The histogram of these normalized differences, computed over all common-point match pairs, is our estimate of the self-consistency statistics.

Self-consistency is a necessary, but not sufficient, condition for a computer vision algorithm to be correct. That is, it is in principle possible for a computer vision algorithm to be self-consistent over many scenes but to be severely biased or entirely wrong. However, having never seen this in practice, we conjecture that this cannot be the case for non-trivial algorithms. If bias can be ruled out, then the self-consistency statistics are a measure of the accuracy of an algorithm—one which requires no “ground truth.” In practice, bias can be ruled out using either manual inspection of a few matches over a few scenes, or comparison of a few matches against ground truth.

Note that estimating the self-consistency statistics from common-point match pairs is quite different than applying the trinocular stereo constraint to these match pairs [31, 3]. In the former we are estimating statistics from many images of many scenes that can be used for many purposes, while in the latter we are filtering out bad matches using three images.

2.2 An Example of the Self-Consistency Statistics

To illustrate the estimation and use of the self-consistency statistics, we first apply the above methodology to the output of a simple stereo algorithm [10]. The algorithm first rectifies the input pair of images and then searches for $7 \times 7$ windows along scan lines that maximize a normalized cross-correlation metric. Sub-pixel accuracy is achieved by fitting a quadratic to the metric evaluated at the pixel and its two adjacent neighbors. The algorithm first computes the match by comparing the left image against the right and then comparing the right image against the left. Matches that are not consistent between the two searches are eliminated. Note that this is a way of using a weaker form of self-consistency as a filter, much like the trinocular stereo constraint. Once found, the coordinates of the match points are unrectified and reported
in the match files.

The stereo algorithm was applied to all pairs of five aerial images of bare terrain, one of which is illustrated in the top row of Figure 2(a). These images are actually small windows (up to 900 pixels on a side) from much larger images (about 9000 pixels on a side) for which precise ground control and bundle adjustment were applied to get accurate camera parameters and projection matrices.

Because the scene consists of bare, relatively smooth, terrain with little vegetation, we would expect the stereo algorithm described above to perform well. This expectation is confirmed anecdotally by visually inspecting the matches.

However, we can get a quantitative estimate for the accuracy of the algorithm for this scene by computing the self-consistency statistics of the output of the algorithm applied to the ten images pairs in this collection. Figure 2(b) shows two representations of the statistics. The solid curve is the probability density function (PDF), the probability that the normalized distance equals \( x \). It is useful for seeing the mode and the general shape of the distribution. The dashed curve that increases monotonically is the cumulative distribution function (CDF), the integral of the PDF, which is the probability that the normalized distance is less than \( x \). It is useful for seeing the median of the distribution (the point where the curve reaches 0.5) or the fraction of match pairs with normalized distances exceeding some value.

In this example, the PDF and CDF show that the mode is about 0.5 normalized distance units, about 95% of the normalized distances between match pairs are below 1, and that about 2% of the match pairs have normalized distances above 10. In short, the stereo algorithm has done pretty well for this image set.

In the bottom row of Figure 2 we see the self-consistency statistics for the same algorithm applied to all pairs of five aerial images of a tree canopy. Such scenes are notoriously difficult for stereo algorithms. Visual inspection of the output of the stereo algorithm confirms that most matches are quite wrong.

This can be quantified using the self-consistency statistics shown in Figure 2(b). Here we see that, although the mode of the distribution is still about 0.5 normalized distance units, only 10% of the matches have a normalized distance less than 1, and only 42% of the matches have a normalized distance less than 10. In short, the stereo algorithm has done very poorly.

Note that we have been able to make quantitative statements about tens of thousands of matches between 20 image pairs with no manual intervention beyond the initial bundle adjustment. Furthermore, the bundle adjustment has been done on the larger images, so that many other experiments can be automatically carried out involving millions of matches over thousands of image windows.

Finally, note that the distributions illustrated above are not well modeled using Gaussian distributions because of the predominance of outliers, especially in the tree canopy example. This is why we have chosen to estimate and use the non-parametric distribution rather than use its variance as a summary.

2.3 Conditionalization

To evaluate the accuracy of individual matches and thereafter predict their accuracy in new images, we let the stereo algorithm associate a score with each match. The idea is that this score is the algorithm’s best estimate of some measure of the match’s accuracy. We can then estimate the self-consistency statistics conditioned on the score. This is best visualized using a self-consistency scatter diagram.

Examples of such a diagram are shown in Figure 2(c). Here we use the original score based on MDL theory that we introduce in the appendix, because we have discovered that traditional scores like SSD are ambiguous and so are not good predictors of match accuracy in general. Every point in the scatter diagram represents a common-point match pair. The \( x \) coordinate of the point is the larger of the scores of the two matches, and the \( y \) coordinate is the normalized distance between the match triangulations.
The top scatter diagram corresponds to the rough terrain and shows that match pairs with a score below 0 have relatively small normalized distances. As the score increases, the normalized distances increase roughly monotonically, at least until, above a score of 2 or so, there are too few points to be statistically valid. Thus, roughly speaking, the score seems to have segregated good matches—those with scores below 0, which have normalized distance below 2—from bad matches/outliers—those with scores above 0, which have normalized distances as large as 10-20.

Although we have shown the scatter diagram for a single-valued score, it is possible to use a multi-valued score vector. For example, the first element of the score vector could be the score for the best match along the epipolar line, and the second element could be the score for the second-best match. We expect this to be useful in scenes with repeating structures. For example if the first two elements are “good” scores, then it’s likely that there is a repeating element along the epipolar line, and so the match shouldn’t be trusted. We will explore the use of such multi-valued scores in a future paper.

**Aggregating statistics across experiments** Ideally, the self-consistency statistics should be estimated using all possible variations of viewpoint and camera parameters, within some class of variations, over all possible scenes, within some class of scenes. However, we can approximate the statistics using some small number of images of a scene, and aggregate the statistics over as many scenes as possible. As the number of scenes increases, the statistics hopefully converge towards a certain overall shape.

Aggregating statistics across experiments is particularly easy when using the scatter diagram representation: Simply take the union of the points from each of the scatter diagrams, since the PDF and CDF can be derived from the scatter diagram. For example, Fig. 7(MDL) shows the union of scatter diagrams from seventeen rural scenes, including the two in Figure 2.

Note that for negative values of the score, the diagram looks quite similar to the top diagram in Fig. 2(c), while for positive values, it looks quite similar to the bottom diagram in Fig. 2(c), though a little more filled in. This implies that the first two scenes together represented much
of the expected variation for rural scenes (at least for this dataset), and that the score is a good predictor of self-consistency across all of these scenes. It also implies that only a few scenes were required to reach convergence with this class of scenes and this score.

However, it is entirely possible that the statistics would not converge if we tried to aggregate them over all possible types of scenes. This would indicate a limitation of the scoring mechanism. There are two ways to cope with this problem. The first, and least desirable, is to restrict the class of scenes considered. The second, and most desirable, is to find a score that can work over larger classes of scenes.

Preliminary experiments with the datasets described in Sec. 5 (involving a few dozen rural and urban scenes) indicate that the MDL-based score converges independently for the two classes of scenes. Nonetheless, the MDL-based score is clearly not perfect, since the statistics from the rural and urban scenes are clearly quite different. We are hoping that the multi-valued score vector we mentioned earlier will work not only for both classes of scenes, but for indoor scenes as well. We will explore this in future work.

Once we have aggregated the statistics for a class of scenes and imaging conditions, and it appears that they have converged, then we can use them to represent the statistics of new images of a new scene within the same class. That is, we can predict the expected variation in reconstruction when we have only a single image pair of a new scene within the same class and processed by the same algorithm and score.

2.4 A Few Variations

Triangulations and reprojections As discussed previously, to apply the self-consistency approach to a set of images, all we need is the set of projection matrices in a common projective coordinate system. The Euclidean distance, which is used to compare two triangulations, is not invariant to the choice of projective coordinates, but this dependence can often be reduced by using the normalization described in Sec. 3.

Furthermore, a method to cancel the dependence on the choice of projective coordinates, is to compute the difference between the reprojections instead of the triangulations. For two matches that have a common point in image 1, one match derived from images 1 and 2, and one derived from images 1 and 3, we can triangulate the first match, and reproject it into image 3. We can then compare the image coordinates of the reprojected point against the coordinate of the second match in image 3. Similarly for image 1. The dependence on the arbitrary projective basis used in 3D is eliminated because the projections are independent of this basis, unlike the triangulations. This, however, does not cancel the dependence on the relative geometry of the cameras, which is dealt with in Sec. 3. An alternative would be to explicitly handle covariance propagation, through the use of the differentiation chain rule to compute the required Jacobians.

More details and results using those distributions were reported in [15]. In the calibrated case, we favor the comparison of the triangulated points over the comparison of the reprojected points because it is more symmetric and computationally simpler, but in the case of uncalibrated cameras, the reprojection method would be preferable.

Image centered vs object-centered techniques So far the description was for stereo methods which produce point matches. The above procedure can also be used for object-centered surface reconstruction techniques such as [11]. In this case, one needs to compare 3-D surface reconstructions derived from different image sets of a static scene. Again, this can be done either in image space or in Euclidean space. In the first case, one can simply derive a dense set of matches by casting a ray through every pixel of, say, image 1 into the world. If the ray intersects the reconstructed surface, then reproject it into all of the images used to derived the surface. This will provide a dense set of matches that can be used as above. The second case requires identifying points on the surface that should be the same. One example of this will be used in Sec. 5: the surface is expressed as \( z = f(x, y) \). The \((x, y)\) coordinates are fixed, and only the \( z \) coordinate varies. This is useful for deriving terrain elevation models from aerial images,
where the ground (x,y) coordinates are fixed, but the elevation is unknown. In this case, instead of finding matches which have a common 2-D point, we find matches that triangulate to 3-D points which have the same (x,y) coordinates. The histogram of the appropriately normalized differences between the coordinates for such matches is what we call the common-x-y-coordinate self-consistency statistics.

**Alternatives measures of deviation** Another distribution that one could compute using the same data files would involve using all the matches in a common-point match set, rather than just pairs of matches. For example, one might use the deviation of the triangulations from the mean of all triangulations within a set. This is problematic for several reasons.

First, there are often outliers within a set, making the mean triangulation less than useful. One might mitigate this by using a robust estimation of the mean. But this depends on various more or less arbitrary parameters of the robust estimator that could change the overall distribution.

Second, and perhaps more importantly, we see no way to extend the normalization used to eliminate the dependence on camera configurations, described in Sec. 3, to the case of multiple matches.

Third, we see no way of using the above variants of the self-consistency distribution for change detection.

The first and third problems can be alleviated when we have a large enough number of matches for any given point to perform robust statistics. This, however, is not often the case and the second problem remains.

### 3 Projection Normalization

#### 3.1 The Mahalanobis distance

Assuming that the contribution of each individual match to the statistics is the same ignores many imaging factors like the geometric configuration of the cameras and their resolution, or the distance of the 3D point from the cameras, as illustrated in Fig. 3. There is a simple way to take into account all of these factors, applying a normalization which makes the statistics invariant to these imaging factors. In addition, this mechanism makes it possible to take into account the uncertainty in camera parameters, by including them into the observation parameters.

We assume that the observation error due to image noise and digitalization effects is Gaussian. This makes it possible to compute the covariance of the reconstruction given the covariance of the observations. Let us consider two reconstructed estimates of a 3-D point, \( M_1 \) and \( M_2 \) to be compared, and their computed covariance matrices \( \Lambda_1 \) and \( \Lambda_2 \). We weight the squared Euclidean distance between \( M_1 \) and \( M_2 \) by the sum of their covariances. This yields the squared Mahalanobis distance:

\[
d^2 = (M_1 - M_2)^T (\Lambda_1 + \Lambda_2)^{-1} (M_1 - M_2) .
\]

#### 3.2 Determining the reconstruction and reprojection covariances

If the measurements are modeled by the random vector \( x \), of mean \( x_0 \) and of covariance \( \Lambda_x \), then the vector \( y = f(x) \) is a random vector of mean is \( f(x_0) \) and, up to the first order, covariance \( J_f(x_0) \Lambda_x J_f(x_0)^T \), where \( J_f(x_0) \) is the Jacobian of \( f \), at the point \( x_0 \).

In order to determine the 3-D distribution error in reconstruction, the vector \( x \) is defined by concatenating the 2-D coordinates of each point of the match, i.e \( [x_1, y_1, x_2, y_2, \ldots, x_n, y_n] \) and the result of the function is the 3-D coordinates \( X, Y, Z \) of the point \( M \) reconstructed from the match, in the least-squares sense. The key is that \( M \) is expressed by a closed-form formula of the form \( M = (L^T L)^{-1} L^T b \), where \( L \) and \( b \) are a matrix and vector which depend on the projection matrices and coordinates of the points in the match. This makes it possible
Figure 3: The 3-D statistics of differences depend on the geometric configurations of the cameras. It is necessary to normalize this effect out.

to obtain the derivatives of $M$ with respect to the $2n$ measurements $w_i, i = 1 \ldots n, w = x, y$. We also assume that the errors at each pixel are independent, uniform, and isotropic. The covariance matrix $A_x$ is then diagonal, therefore each element of $A_M$ can be computed as a sum of independent terms for each image.

The above calculations are exact when the mapping between the vector of coordinates of $m_i$ and $M$ (resp. $m'_j$ and $M'$) is linear, since it is only in that case that the distribution of $M$ and $M'$ is Gaussian. The reconstruction operation is exactly linear only when the projection matrices are affine. However, the linear approximation is expected to remain reasonable under normal viewing conditions, and to break down only when the projection matrices are in configurations with strong perspective.

### 3.3 Simulations

In order to gain insight into the nature of the normalized self-consistency distributions, we investigate the case when the noise in point localization is Gaussian. We assume in this section that the 3D points are independent. In practice, this might not be the case, depending on how these 3D points are obtained. However, the main purpose of this section is to show with a particular instanciated numerical model how the normalization makes the distribution less sensitive to the varying geometry, and it is to be expected that a similar result would be obtained with a slightly different model for the 3D points.

We first derive the analytical model for the self-consistency distribution under those hypotheses. We then show, using monte-carlo experiments that, provided that the geometrical normalization described in Sec.3 is used, the experimental self-consistency distributions fit this model quite well when perspective effects are not strong. A consequence of this result is that under the hypothesis that the error localization of the features in the images is Gaussian, the self-consistency distribution could be used to recover exactly the accuracy distribution.

**Modeling the Gaussian self-consistency distributions.** Assuming that the 3D points are independent, the squared Mahalanobis distance in 3D follows a chi-square distri-
bution with three degrees of freedom:

\[ d^2 \sim \chi^2_3 = \frac{1}{\sqrt{2\pi}} \sqrt{x e^{-x/2}} \]

In our model, the Mahalanobis distance is computed between \( M, M' \), reconstructions in 3D, which are obtained from matches \( m_i, m'_j \) of which coordinates are assumed to be Gaussian, zero-mean and with standard deviation \( \sigma \). If \( M, M' \) are obtained from the coordinates \( m_i, m'_j \) with a linear transformation \( A, A' \), then the covariances are \( \sigma^2 A A'^T \), \( \sigma^2 A' A'^T \). The Mahalanobis distance follows the distribution:

\[ d \sim \sqrt{\chi^2_3} = \frac{1}{\sqrt{2\pi}} \sqrt{2/\pi} e^{-x^2/2\sigma^2} \] (1)

Using the Mahalanobis distance, the self-consistency distributions should be statistically independent of the 3D points and projection matrices. Of course, if we were just using the Euclidean distance, there would be no reason to expect such an independence.

**Comparison of the normalized and unnormalized distributions** To explore the domain of validity of the first-order approximation to the covariance, we have considered three methods to generate random projection matrices:

1. General projection matrices are picked randomly.
2. Projection matrices are obtained by perturbing a fixed, realistic, close to affine matrix. Entries of this matrix are each varied randomly within 500\% of the initial value.
3. Affine projection matrices are picked randomly.

Each experiment in a set consisted of picking random 3D points, random projection matrices according to the configuration previously described, projecting them, adding random Gaussian noise to the matches, and computing the self-consistency distributions by labelling the matches so that they are perfect.

To illustrate the invariance of the distribution that we can obtain using the normalization, we performed experiments where we computed both the normalized version and the unnormalized version of the self-consistency. As can be seen in Fig. 4, using the normalization reduced dramatically the spread of the self-consistency curves found within each experiment in a set. In particular, in the two last configurations, the resulting spread was very small, which indicates that the geometrical normalization was successful at achieving invariance with respect to 3D points and projection matrices.

**Comparison of the simulated and theoretical distributions** Using the Mahalanobis distance, we then averaged the density curves within each set of experiments, and tried to fit the model described in Eq. 1 to the resulting curves, for six different values of the standard deviation, \( \sigma = 0.5, 1, 1.5, 2, 2.5, 3 \). As illustrated in Fig. 5, the model describes the average self-consistency curves very well when the projection matrices are affine, as expected from the theory, but also when they are obtained by perturbing a fixed matrix. When the projection matrices are picked totally at random, the model does not describe the curves very well, but the different self-consistency curves corresponding to each noise level are still distinguishable.

**Comparison of this model with experimental results** As mentioned earlier, the model used in this section is valid for independent 3D points. If the 3D points are reconstructed using one common point in one image, there would be a common ray and the variation would be only along this common ray. In that situation, with only one degree of freedom, the squared distance \( d^2 \) would follow a \( \chi^2_1 \) distribution instead of the \( \chi^2_3 \) distribution used in those simulations. As a consequence, the \( \chi^2_1 \) distribution indeed describes better some experimental results with real images presented later, because a common point was used.
4 Experiments in measuring the self-consistency of stereo algorithms

4.1 Comparing Two Algorithms

The experiments described here and in the following section are based on the application of stereo algorithms to seventeen scenes, each comprising five images, for a total of 85 images and 170 image pairs. At the highest resolution, each image is a window of about 900 pixels on a side from images of about 9000 pixels on a side. Some of the experiments were done on Gaussian-reduced versions of the images. These images were controlled and bundle-adjusted to
provide accurate camera parameters. They have a ground resolution of approximately 15cm. A single self-consistency distribution for each algorithm was created by merging the scatter data for that algorithm across all seventeen scenes.

For practical purposes such as algorithm comparison and change detection, the information in the scatter diagram can be summarized by a series of curves, called s% significance level curves \( f(s, x) \). For a given value of the score \( x \), the s% significance level is the largest normalized difference for which s% of the common-point matches lie below: \( P(x < f(s, x)) \leq s \). Examples of significance level curves can be seen in Fig. 6.

In the top of Fig. 6 we see several representations of the common-xy-coordinate self-consistency distribution described in Sec. 2.2 obtained by applying the simple point-by-point stereo algorithm [10]. The bar graph of Fig. 6(b) is the histogram of the normalized difference in the coordinate of the triangulation of all common-xy-coordinate match pairs. One can see from this histogram that the mode of the differences is smaller than 1 normalized unit. The curve is the integral of this graph, or the cumulative distribution function. One can see from this that about 90% of the match pairs have normalized differences below 2 units. Each point in Figure 6(a) corresponds to a common-xy-coordinate match pair. The x-coordinate of the point in the diagram is the larger of the scores for the two matches, and the y-coordinate is the normalized difference between their triangulated coordinates. The curve in Fig. 6(a) is the 99% significance level, that is, it is the largest normalized z difference for which 99% of the common-point matches with a given score lie below. Note that significance level increases as the score increases, indicating that matches with larger scores are less self-consistent than matches with a lower score, an indication of the quality of our MDL score. The drop that we observe for positive values of the score is due to the fact that there are only few common-xy-coordinate match pairs with positive scores, so that calculations done with those values are not statistically meaningful.

In the bottom of Fig. 6 we see the common-xy-coordinate self-consistency distribution for the deformable mesh algorithm [11] applied to the same images. Note that it is significantly more self-consistent than the distribution for the stereo algorithm. This is as expected, since the deformable mesh algorithm was specifically designed to provide highly accurate reconstructions of terrain.

Comparing these two graphs shows some interesting differences between the two algorithms. The deformable mesh algorithm has a much greater proportion of matches with distances below 0.25. This is not unexpected since the strength of the deformable meshes is its ability to do very precise matching between images. However, the algorithm can get stuck in local minima. Self-consistency now allows us to quantify how often this happens.

But this comparison also illustrates that one must be very careful when comparing algorithms or assessing the accuracy of a given algorithm. The distributions we get are very much dependent on the scenes being used, as would also be the case if we were comparing the algorithms against ground truth—the “gold standard” for assessing the accuracy of a stereo algorithm. In general, the distributions will be most useful if they are derived from a well-defined class of scenes, such as the rural ones used in this Section. The shape of the graphs can potentially be different when using different kinds of scenes, such as the urban ones presented in the following sections. It might also be necessary to restrict the imaging conditions such as resolution or lighting as well, depending on the algorithm. Only then can the distribution be used to predict the accuracy of the algorithm when applied to images of similar scenes.

### 4.2 Comparing Three Scoring Functions

To eliminate the dependency on scene content, we propose to use a score associated with each match, as described in Sec. 2.3. We saw scatter diagrams in Figure 2(c) that illustrated how a scoring function might be used to segregate matches according to their expected self-consistency.

In this section we will compare three scoring functions, the one based on Minimum Description Length Theory introduced in the appendix, the traditional sum-of-squared-differences (SSD) score, and the SSD score normalized by the localization covariance (SSD/GRAD score)
Figure 6: Experimental comparison of two algorithms: point-by-point stereo algorithm (top) vs deformable mesh algorithm (bottom). (a),(c): Scatter diagrams for 170 image pairs of rural scenes. The superimposed curve is the 99% significance level, which is the largest normalized $z$ difference for which 99% of the common-point matches with a given score lie below. (b),(d): The histogram of the normalized $z$ differences for all common-point matches. The superimposed curve is the cumulative distribution of the histogram.

[9]. All scores were computed using the same matches computed by our deformable mesh algorithm applied to all image pairs of the seventeen scenes mentioned above. The scatter diagrams for all of the areas were then merged together to produce the scatter diagrams show in Figure 7.

The MDL score has the very nice property that the confidence interval defined earlier rises monotonically with the score, at least until there is a paucity of data, when then score is greater than 2. It also has a broad range of scores, those below zero, for which the normalized distances are below 1, with far fewer outliers than the other scores.

Because the MDL score depends on the logarithm of the standard deviations, for a fair comparison, we plotted both the SSD/GRAD and GRAD scores on a log scale. They exhibit similar behaviors, but have many more outliers, making them less useful as predictors for self-consistency.

4.3 Comparing Window Size

One of the common parameters in a traditional stereo algorithm is the window size. Figure 8 presents one image from six urban scenes, where each scene comprised four images. Figure 9 shows the merged scatter diagrams (a) and global self-consistency distributions (b) for all six scenes, for three window sizes ($7 \times 7$, $15 \times 15$, and $29 \times 29$). Some of the observations to note
from these experiments are as follows.

First, note that the scatter diagram for the $7 \times 7$ window of this class of scenes has many more outliers for scores below -1 than were found in the scatter diagram for the terrain scenes. This is reflected in the global self-consistency distribution in (b), where one can see that about 10% of matches have normalized distances greater than 6. The reason for this is that this type of scene has significant amounts of repeating structure along epipolar lines. Consequently, a score based only on the quality of fit between two windows (such as the MDL-based score) will fail on occasion. A better score would include a measure of the uniqueness of a match along the epipolar line as a second component. We are currently exploring this.

Second, note that the number of outliers in both the scatter diagram and the self-consistency distributions decreases as window size decreases. Thus, large window sizes (in this case) produce more self-consistent results. But it also produces fewer points. This is probably because this stereo algorithm uses left-right/right-left equality as a form of self-consistency filter.

We have also visually examined the matches as a function of window size. When we restrict ourselves to matches with scores below -1, we observe that matches become sparser as window size increases. Furthermore, it appears that the matches are more accurate with larger window sizes. This is quite different from the results of Faugeras et al. [8]. There they found that, in general, matches became denser but less accurate as window size increased. We believe that this is because an MDL score below -1 keeps only those matches for which the scene surface is approximately fronto-parallel within the extent of the window, which is a situation in which larger window sizes increases accuracy. This is borne out by our visual observations of the matches. On the other hand, this result is basically in line with the results of Szeliski and Zabih [27, 29], who show that prediction error decreases with window size.

![Figure 7: Scatter diagrams for three different scores.](image)

![Figure 8: Three of six urban scenes used for the window comparisons. Each scene contained 4 images.](image)
Figure 9: Comparing three window sizes. (a) The combined self-consistency distributions of six urban scenes for window sizes 7 × 7, 15 × 15, and 29 × 29. (b) The scatter diagrams for the MDL score for these urban scenes.

5 Experiments in Detecting Height Changes

5.1 The principle

So far, we have assumed that all the images have been taken at exactly the same time, or equivalently, that the scene is static.

One application of the self-consistency distribution is detecting height changes in a scene over time. Given two collections of images of a scene taken at two points in time, we can compare matches (from different times) that belong to the same surface element to see if the difference in triangulated coordinates exceeds some significance level. This gives a mechanism for distinguishing changes which are significant from changes which are due to modeling uncertainty.

We would like to be able to compare a reconstruction of a scene created with as few as two images of the scene taken at one time against a reconstruction of the same scene created with as few as two images taken at a different time. There is not enough data in these conditions to compute the self-consistency distribution. However, using sets of images of similar scenes, the conditionalization method described in Sec. 2.3 based on the MDL score lets us can summarize self-consistency, and therefore expected variation for individual surface elements. As it was found in Sec. 4.2, the MDL score has a stronger correlation with self-consistency than other scores we have examined, and its use is essential for the proposed change detection method to work.

Although our methodology for change detection is quite general, the experiments we conducted are based on two simplifying assumptions. First, we use a specific class of objects: terrain (both rural and urban) viewed from above using aerial imagery. We take advantage of the special nature of terrain to simplify the problem. The 3-D shape is modeled as a single-valued function, whose value represents elevation above the ground plane. Second, we assume that all the camera parameters for all the images are known in a common coordinate system, which we obtain by bundle adjustment over all the images. Together, these two assumptions reduce the problem of detecting changes in 3-D shape to that of finding point-by-point significant differences in scalar values.

15
5.2 The algorithm

In the first stage, we run the stereo algorithm on a large number of subsets of images of the same class as those in which we want to perform change detection. We use a bucketing method to find all the common-xy-coordinate matches (pairs of matches for which the 3-D reconstruction has the same $(x,y)$ value within a threshold). Each such pair is accumulated in a scatter diagram (see Fig. 6) in which the $x$-coordinate of is the larger of the scores for the two matches, and the $y$-coordinate is the normalized difference between their triangulated coordinates. We then extract the significance level curves for the values of significance $s\%$ (or in other words, percent confidence in the significance of the change) which we plan to use. For a given value of the score (the $x$-axis), this is the normalized difference below which $s\%$ of the common-xy-coordinate match pairs with that score lie.

In the second stage, we use the pre-computed significance level curves to judge whether a pair of matches derived from images taken at different times is significantly different. We find, using the same technique as before, the common-xy-coordinate matches where each match originates from a different instant, compute the larger of their scores and the normalized difference between triangulated $z$ coordinates. If, for that score, this distance is above the significance level $s\%$ then the pair of matches is deemed to be a difference significant with confidence $s\%$.

All this assumed that the registration of all images between the two instants is perfect. However, often there is a small registration error in the $z$-coordinates. For example, in the rural scenes examples, we have often found an error of about half a meter. Since the changes that we try to detect are of a comparable magnitude, the registration error would cause a severe bias in the results. To solve this problem, once the common-xy-coordinate matches are found, we estimate the registration error between each pair of views taken at a different instant as the offset in $z$ which minimizes the median of the squared differences between points in each of the views. Because the median is a robust operator, this will give an accurate estimation of the registration error as long as a majority of points didn’t have a change in $z$ coordinates.

5.3 Experimental results

Rural scenes

In Fig. 10 we show the changes detected in one of the rural scenes mentioned above, using the deformable mesh algorithm. In Fig. 10 we see one of 5 images of the scene taken in 1995. The dark diagonal in the bottom right corner of the image is a dried creek bed. In Fig. 10 we see one of 5 images of the same area taken in 1998. The dried creek bed has been filled in with dirt, creating a change in elevation of about 1 meter. We applied the deformable mesh algorithm to one pair of images taken in 1995. We then compared this to the deformable mesh derived from one pair of images taken in 1998. Vertices that were deemed to be significantly different (above the 99\% level of the self-consistency distribution of Fig. 6(c)), are overlaid as white cross on the image in Fig. 10 (c), which is a magnified view of the dried creek bed of Fig. 10 (a). We have also applied our algorithm to forested areas of the same rural scene. Although the normalized differences in $z$-coordinates is sometimes much larger (10 meters), no changes were deemed significant. Indeed, it is known that the mesh algorithm performs poorly on images of tree canopies, so that reconstruction noise could account for the differences.

Urban scenes In Fig. 11 we show the changes (significant differences in $z$) detected in one of the urban scenes mentioned above, but this time using the stereo algorithm with $15 \times 15$ windows, for which the distribution was shown in Fig. 9. In Fig. 11 (a), we see one of 4 images taken at time 1. Note the new building near the center of the image. In Fig. 11 (b) we see one of the images taken at time 2. In Fig. 11 (c) we see the significant differences between the matches derived from a single pair of images taken at time 1 and the matches derived from a single pair of images taken at time 2, for a significance level of 99.99\%. In Fig. 11 (d) we have merged the significant differences between each pair of images at time 1 and each pair of images at time 2.
Figure 10: Changes detected in one of the 17 rural scenes. (a) One of the 5 images of a rural site taken in 1995. Note the dried creek bed in the white box in the bottom right corner of the image. (b) One of 5 images of the same area taken in 1998. Note that the dried creek bed has been filled in with dirt, causing a change in elevation of about 1 meter. (c) The white crosses are located where significant differences have been found using the self-consistency distribution of Fig. 6(c) between the deformable-mesh model created with one pair of 1995 image and a model created using one pair of 1998 images.

Note that virtually all differences are at the location of the new building. For comparison, we show what would happen if we simply thresholded the normalized difference in triangulated z coordinates. In Fig. 12 (a) we show the differences between a single pair of images at time 1 and a single pair of images at time 2, for a threshold of 3 units. In Fig. 12 (b) we show the differences for a threshold of 6 units, which is the average difference found in Fig. 11. This value of the threshold is the highest one for which no correct changes are missed, yet it is seen that many incorrect changes are still detected. In Fig. 12 7(c) we see the union of the differences for all image pairs.

In Fig. 13 and Fig. 14 we see the results of change detection for two other urban scenes, one with a new building, the other without significant changes.

Figure 11: The first urban scene. (a) Image at time 1. (b) Image at time 2. Note the new buildings near the center of the image. (c) The significant differences between matches derived from one pair of images taken at time 1 and matches derived from one pair of images taken at time 2. (d) The union of all significant differences found between matches derived from all pairs of images taken at time 1 and all pairs of images taken at time 2.
Figure 12: For comparison, a simple thresholding of the normalized difference in $z$ coordinates. (a) Matches with normalized $z$ differences $> 3$ units, for the same matches as in Fig. 11. (b) Same as (a), for $z$ differences $> 6$ units (which is the average difference found in Fig. 11. (c) The union of all differences for all pairs of images, as in Fig. 11 (d).

Figure 13: A second urban scene. (a) One of the 4 images taken at time 1. (b) One of the 3 images taken at time 2. Note the changed building near the center. (c) The union of all significant differences found between matches derived from all pairs of images taken at time 1 and all pairs of images taken at time 2.

6 Comparison with Previous work

6.1 Other measures of uncertainty

Existing work on estimating uncertainty without ground truth falls into three categories: analytical, statistical, and empirical approaches.

The analytical approaches are based on the idea of error propagation [32]. When the output is obtained by optimizing a certain criterion (like a correlation measure), the shape of the optimization curve [9, 19, 14] or surface [1] provides estimates of the covariance through the second-order derivatives. These approaches make it possible to compare the uncertainty of different outputs given by the same algorithm. However, it is problematic to use them to compare different algorithms.

Statistical approaches make it possible to compute the covariance given only one data sample and a black-box version of an algorithm, by repeated runs of the algorithm, and application of the law of large numbers [7].
Both of the above approaches characterize the performance of a given output only in terms of its expected variation with respect to additive white noise. In [30], the accuracy was characterized as a function of image resolution. The bootstrap methodology [6] goes further, since it makes it possible to characterize the accuracy of a given output with respect to IID noise of unknown distribution. Even if such an approach could be applied to the multiple image correspondence problem, it would characterize the performance with respect to IID sensor noise. Although this is useful for some applications, for other applications it is necessary to estimate the expected accuracy and reliability of the algorithms as viewpoint, scene domain, or other imaging conditions are varied. This is the problem we seek to address with the self-consistency methodology.

Our methodology falls into the realm of empirical approaches. See [24], for a good overview of such approaches.

Szeliski [27] has recently proposed prediction error to characterize the performance of stereo and motion algorithms. Prediction error is the difference between a third real image of a scene and a synthetic image produced from the disparities and known camera parameters of the three images. This approach is especially useful when the primary use of stereo is for view interpolation, since the metric they propose directly measures how well the algorithm has interpolated a view compared to a real image of that same view. In particular, their approach does not necessarily penalize a stereo algorithm for errors in constant-intensity regions, at least for certain viewpoints. Our approach, on the other hand, attempts to characterize self-consistency for all points. Furthermore, our approach attempts to remove the effects of camera configuration in computing the measure over many observations and scenes.

Szeliski and Zabih have recently applied this approach to comparing stereo algorithms [27, 29]. A comprehensive comparison of our two methodologies applied to the same algorithms and same datasets should yield interesting insights into these two approaches.

An important item to note about our methodology is that the projection matrices for all of the images are provided and assumed to be correct (within their covariances). Thus, we assume that a match produced by the stereo algorithm always lies on the epipolar lines of the images. Consequently, a measure of how far matches lie from the epipolar line, is not relevant.

6.2 Change detection

Change detection is an important task in computer vision that has been addressed early at the image intensity level [21, 18] where it is still a topic of interest [23] with many other papers in between. However, comparing intensity values is not very effective because such changes
don't necessarily reflect actual changes in shape, but could be caused by changes in viewing and illumination conditions or even in reflectance (e.g., seasonal changes). Although it has been attempted, this is not easy to take them into account at this level. For man-made objects such as buildings, higher-level comparisons have been proposed, based on feature organization [25], and 3-D models [4, 13]. These specialized approaches are the most successful, but are not applicable to more general objects like natural terrain. A few of the ideas needed for general change detection in shape are found in other areas of computer vision. In work on tracking (see for instance [5] which deals with small bodies in a natural environment), statistics have been computed during a learning phase and then used to differentiate between significant and insignificant changes. Of course, the problem is simplified by the fact that the camera is stationary, whereas we want to deal with various viewpoints. In mobile robotics (see for instance [2]), the problem of fusing several general 3-D maps to maintain representations of the environment of a robot has been cast in a statistically rigorous framework taking into account uncertainties. However, the specific issue of change detection has not been addressed there.

7 Extending the Self-Consistency Methodology

So far we have studied the application of the self-consistency methodology to stereo algorithms. In this section, we discuss how we can extend this to other computer vision algorithms. We start by proposing a formalization of computer vision algorithms that we believe is applicable to many, but certainly not all, such algorithms. We then write the self-consistency distribution in terms of this formalization. Finally, we describe the application of our methodology to stereo algorithms based on this formalization and discuss two other algorithms that can probably be formalized in this way (though we have not yet implemented or experimented with these other algorithms).

7.1 A Formalization of Self-Consistency

Some computer vision algorithms can be formalized as a function that takes an observation Ω of a world W as input and produces a set of hypotheses H about the world as output:

\[ H(h_1, h_2, \ldots, h_n) = F(Ω, W). \]

An observation Ω is one or more images of the world taken at the same time, perhaps accompanied by meta-data, such as the time the image(s) was acquired, the internal and external camera parameters, and their covariances.

A hypothesis h nominally refers to some aspect or element of the world (as opposed to some aspect of the observation), and it nominally estimates some attribute of the element it refers to. We formalize this with the following set of functions that depend on both F and Ω:

1. \( Ref(h) \), the referent of the hypothesis h (i.e., which element in the world that the hypothesis refers to).
2. \( R(h, h') = \text{Prob}(Ref(h) = Ref(h')) \), an estimate of the probability that two hypotheses h and h', (computed from two observations of the same world), refer to the same object or process in the world.
3. \( Att(h) \), an estimate of some well-defined attribute of the referent.
4. \( Acc(h) \), an estimate of the accuracy distribution of \( Att(h) \). When this is well-modeled by a normal distribution, it can be represented implicitly by its covariance, \( Cov(h) \).
5. \( Score(h) \), an estimate of the confidence that \( Att(h) \) is correct.

Intuitively, we can state that two hypotheses h and h', derived from observations Ω and Ω' of a static world W, are consistent with each other if they both refer to the same object in the world and the difference in their estimated attributes is small relative to their accuracies,
or if they do not refer to the same object. When the accuracy is well modeled by a normal distribution, the consistency of two hypotheses, \( C(h, h') \), can be written as

\[
C(h, h') = R(h, h')(\text{Att}(h) - \text{Att}(h'))^T (\text{Cov}(h) + \text{Cov}(h'))^{-1} (\text{Att}(h) - \text{Att}(h'))^T
\]

Note that the second term on the right is the Mahalanobis distance between the attributes, which we refer to as the normalized distance between attributes throughout this paper.

Given the above, we can measure the self-consistency of an algorithm as the histogram of \( C(h, h') \) over all pairs of hypotheses in \( H = F(\Omega(W)) \) and \( H' = F(\Omega(W)) \), over all observations over all suitable static worlds \( W \). We call this distribution of \( C(h, h') \) the self-consistency distribution of the computer vision algorithm \( F \) over the worlds \( W \). To simplify the exposition below, we compute this distribution only for pairs \( h \) and \( h' \) for which \( R(h, h') \approx 1 \). We will discuss the utility of the full distribution in future work.

### 7.2 Applications to Self-Consistency of Stereo Algorithms

To help the reader’s intuition, in this section we detail how the previous abstract self-consistency formalism applies to stereo algorithms. Stereo algorithms attempt to reconstruct 3D surface geometry from multiple images, assuming that the camera geometry is known.

The hypothesis \( h \) produced by a traditional stereo algorithm is a pair of image coordinates \((x_0, x_1)\) in each of two images, \((I_0, I_1)\). In its simplest form, a stereo match hypothesis \( h \) asserts that the closest opaque surface element along the optic ray through \( x_0 \) is the same as the closest opaque surface element along the optic ray through \( x_1 \). That is, the referent of \( h \), \( \text{Ref}(h) \), is the closest opaque surface element along the optic rays through both \( x_0 \) and \( x_1 \).

Consequently, as the camera geometry is varied, two stereo hypotheses should have the same referent if their image coordinates are the same in one image. Self-consistency, in this case, is a measure of how often (and to what extent) this assertion is true.

The above observation can be used to write the following set of associated functions for a stereo algorithm. We assume that all matches are accurate to within some nominal accuracy, \( \sigma \), in pixels (typically \( \sigma = 1 \)). This can be extended to include the full covariance of the match coordinates.

1. \( \text{Ref}(h) \), The closest opaque surface element visible along the optic rays through the match points.
2. \( R(h, h') = 1 \) if \( h \) and \( h' \) have the same coordinate, within \( \sigma \), in one image; 0 otherwise.
3. \( \text{Att}(h) \), The triangulated, or projective, 3-D coordinates of the surface element.
4. \( \text{Acc}(h) \), The covariance of \( \text{Att}(h) \), given that the match coordinates are \( N(x_0, \sigma) \) and \( N(x_0, \sigma) \) random variables.
5. \( \text{Score}(h) \), A measure such as normalized cross-correlation or coding loss.

### 7.3 Applying Self-consistency to other domains

**Shape-From-Shading**  Shape-from-shading algorithms [12] attempt to reconstruct the surface orientation from a single image, assuming that the light source direction is known.

The hypothesis produced by a shade from shading algorithm is a pair of image coordinates and 3-D unit vector \((x, n)\). This asserts that the surface orientation at the closest opaque surface element along the optic ray through \( x \) is the vector \( n \). As the light source direction is varied, two shape-from-shading hypotheses should have the same surface orientation if their image coordinates are the same in one image.

**Line-drawing interpretation**  Line-drawing interpretation algorithms [26, 17] attempt to build a wire-frame 3-D model from a line-drawing sketch of a polyhedral object viewed from a certain direction.
The hypothesis produced by a wire-frame reconstruction algorithm is a segment label and a reconstructed 3-D segment given by the 3-D coordinates of its end points \((\lambda, [A, B])\). This asserts that an orthographic projection of the 3D segment \([A, B]\) yields the 2D segment of label \(\lambda\). Corresponding segments across different views are supposed to have the same label, therefore when the direction of projection is varied, two line-drawing interpretation hypotheses should yield the same 3D segment if they have the same label. This criterion was used to assess the results of two different algorithms in [17].

8 Conclusion and Perspectives

We have introduced a general formalization of a perceptual observation called self-consistency. We have proposed a methodology based on this formalization as a means of estimating the accuracy and reliability of point–correspondence algorithms, comparing different stereo algorithms, comparing different scoring functions, comparing window sizes, and detecting change over time. We have presented a detailed prescription for applying this methodology to multiple-image point–correspondence algorithms, without any need for ground truth or camera calibration, and have demonstrated its utility in several experiments.

The self-consistency distribution is a very simple idea that has powerful consequences. It can be used to compare algorithms and scoring functions, evaluate the performance of an algorithm across different classes of scenes, tune algorithm parameters such as window size, reliably detect changes in a scene, and so forth. All of this can be done at little manual cost beyond the precise estimation of the camera parameters and perhaps manual inspection of the output of the algorithm on a few images to identify systematic biases.

Finally, we believe that the extended self-consistency formalism envisaged in Sec. 7.1, which examines the self-consistency of an algorithm across independent experimental trials of different “viewpoints” or imaging parameters of a static scene, can be used to assess the accuracy and reliability of algorithms dealing with a range of computer vision problems.

Once we can measure the self-consistency of an algorithm, and we observe that this measure remains reasonably constant over many scenes, or at least for certain subsets, then we can be reasonably confident that the algorithm will be self-consistent over new scenes. More importantly, such algorithms are also likely to exhibit the self-consistency property of the human visual system: given a single view of a new scene, such an algorithm is likely to produce inferences that would be self-consistent with other views of the scene should they become available later. Thus, internal redundancy as measured by self-consistency is an excellent and easy-to-use predictor of algorithm performance that can substitute for “ground-truth” when it is not available, as is often the case. Similarly, it could be used to learn the parameters of an algorithm that lead to self-consistency over a wide range of scenes without the need for external training data.

Appendix: The MDL score

A.1 Principle of the coding loss

A classic score used in stereo is the sum of squared differences (SSD). The score is supposed to be small for “good” matches, and high for “bad” matches. The problem with the SSD measure is that it’s ambiguous. That is, a low SSD measure can occur not only when the match is correct, as it should, but also when the match is incorrect and the terrain is spatially uniform. In aerial images, this happens in the cases of unmarked parking lots, flat sandy areas, or uniformly colored planar roofs. Intuitively, then, we want an image-matching measure that is low only when the match between the predicted and observed pixel values is close and the pixel values form a sufficiently complex pattern that it is unlikely to be matched elsewhere.

We have developed a measure that satisfies this intuitive requirement reasonably well. We call this measure the coding loss. The coding loss is based on Minimum Description Length (MDL)
<table>
<thead>
<tr>
<th>Good match</th>
<th>Bad match</th>
<th>Little structure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Model</strong></td>
<td><img src="image" alt="Model" /></td>
<td><img src="image" alt="Model" /></td>
</tr>
<tr>
<td><strong>Differences</strong></td>
<td><img src="image" alt="Differences" /></td>
<td><img src="image" alt="Differences" /></td>
</tr>
<tr>
<td><strong>Encoding Cost</strong></td>
<td>$C + 2\epsilon$</td>
<td>$3C$</td>
</tr>
<tr>
<td><strong>Individual images</strong></td>
<td><img src="image" alt="Individual images" /></td>
<td><img src="image" alt="Individual images" /></td>
</tr>
<tr>
<td><strong>Encoding Cost</strong></td>
<td>$2C$</td>
<td>$2C$</td>
</tr>
</tbody>
</table>

**MDL Score**

$$(C + 2\epsilon) - (2C) = -C + 2\epsilon$$

$$(3C) - (2C) = +C$$

$$(3\epsilon) - (2\epsilon) = +\epsilon$$

---

Figure A.1: The MDL measure for a correct match with large pixel variation, incorrect match with large pixel variation, and small pixel value variation.

In MDL theory [22], quantized observations of a random process are encoded using a model of that process. This model is typically divided into two components: A parameterized predictor function $M(z)$ and the residuals taken to be the differences between the observations and the values predicted by $M$. The residuals are typically encoded using an i.i.d. noise model [16]. MDL is a methodology for computing the parameters $z$ that yield the optimal code length for this model and for a given encoding scheme. This optimal code length is the minimum number of bits required to encode the parameters $z$ and the corresponding residuals, such that he resulting encoding is a loss-less encoding of the observations.

In our case, the process we are observing is the object surface. The observations are the pixel values in all of the images covered by a given surface element, which we call a facet. The parameters $z$ are the coordinates and albedos of the facet. The parameterized predictor function $M(z)$ for the facet predicts the covered image pixel values given the facet parameters and the fixed camera parameters. The coding loss is the difference between the code length of the image pixels using the facet model and the code length using independent noise models. To see why this might be a good measure, consider the following cases, illustrated by Fig. A.1:

1. **The facet is correct and the pixel value variation is large.** Here, we expect the residuals to be small, costing, say, $\epsilon$ bits to encode, and the material property samples to vary about as much as the image pixels, costing, say, $C$ bits to encode. Thus, the cost of encoding the image pixels using the facet model will be about $C + n\epsilon$ bits, while the cost of encoding the image pixels in the $n$ images will be about $nC$ bits. Thus, the coding loss is expected to be a large negative value: $C + n\epsilon - nC = (1 - n)C + n\epsilon$ bits.

2. **The facet is incorrect and the pixel value variation is large.** Here, we expect the material property samples as well as the residuals to vary about as much as the image pixel values. Thus, the material property samples and the residuals for a given image will each cost about $C$ bits to encode. Thus, the cost of encoding the image pixels using the facet model will be about $(n + 1)C$ bits, while the cost of encoding the pixel values without the facet model will be about $nC$ bits. Thus, the coding loss is expected to be a large positive value: $(n + 1)C - nC = C$ bits.

3. **The pixel value variation is small.** Here, we expect the cost of encoding the property samples and the residuals to be about the same and small, say $\epsilon$ bits. Similarly, the cost
of encoding the pixel values in a given image will be about $\epsilon$ bits. Thus, the coding loss is expected to be a small positive value: $(n + 1)\epsilon - n\epsilon = \epsilon$ bits.

Note that cases 1 and 3 above are expected to have significantly different values for the coding loss, whereas they are expected to have about the same value for the SSD measure, namely $\epsilon$. Thus, we would expect the coding loss to be more effective in distinguishing between correct and incorrect facet parameters. As discussed in Sec. 4.2, this is what we have observed experimentally. There are many ways of encoding both the facet parameters and the residuals. The choice of encoding schemes can have a significant effect on the overall code length and, more importantly, the values of the parameters that minimize the code length. To date we have used a very simple encoding scheme, which basically assumes that the parameters and residuals are well encoded using an i.i.d. white noise model. Even though this is clearly not an optimal encoding scheme, the results are very good. Better encoding schemes could significantly improve the results.

### A.2 Analytical details

Given $N$ images, let $M$ be the number of pixels in the correlation window and let $g^{i}_{j}$ be the image gray level of the $i^{th}$ pixel observed in image $j$. For image $j$, the number of bits required to describe these gray levels as IID white noise can be approximated by:

$$C_{j} = M (\log \sigma_{j} + c)$$  \hspace{1cm} (A.1)

where $\sigma_{j}$ is the measured variance of the $g^{i}_{1 \leq i \leq N}$ and $c(1/2) \log(2\pi e)$.

Alternatively, these gray levels can be expressed in terms of the mean gray level $\overline{g}$ across images and the deviations $g^{i}_{j} - \overline{g}$ from this average in each individual image. The cost of describing the means, can be approximated by

$$\overline{C} = M(\log \overline{\sigma} + c)$$  \hspace{1cm} (A.2)

where $\overline{\sigma}$ is the measured variance of the mean gray levels. Similarly the coding length of describing deviations from the mean is given by

$$C^{d}_{j} = M (\log \sigma^{d}_{j} + c)$$  \hspace{1cm} (A.3)

where $\sigma^{d}_{j}$ is the measured variance of those deviations in image $j$. Note that, because we describe the mean across the images, we need only describe $N - 1$ of the $C^{d}_{j}$. The description of the $N$th one is implicit.

The MDL score is the difference between these two coding lengths, normalized by the number of samples, that is

$$Loss = \overline{C} + \sum_{1 \leq j \leq N - 1} C^{d}_{j} - \sum_{1 \leq j \leq N} C_{j}.$$  \hspace{1cm} (A.4)

When there is a good match between images, the $g^{i}_{1 \leq j \leq N}$ have a small variance. Consequently the $C^{d}_{j}$ should be small, $\overline{C}$ should be approximately equal to any of the $C_{j}$ and $Loss$ should be negative. However, $C_{j}$ can only be strongly negative if these costs are large enough, that is, if there is enough texture for a reliable match.

### References


