Prediction Error as a Quality Metric for Motion and Stereo

Richard Szeliski
Vision Technology Group, Microsoft Research
One Microsoft Way, Redmond, WA 98052-6399

Abstract

This paper presents a new methodology for evaluating the quality of motion estimation and stereo correspondence algorithms. Motivated by applications such as novel view generation and motion-compensated compression, we suggest that the ability to predict *new views or frames is a natural metric for evaluating such algorithms. Our new metric has several advantages over comparing algorithm outputs to true motions or depths. First of all, it does not require the knowledge of ground truth data, which may be difficult or laborious to obtain. Second, it more closely matches the ultimate requirements of the application, which are typically tolerant of errors in uniform color regions, but very sensitive to isolated pixel errors or disocclusion errors. In the paper, we develop a number of error metrics based on this paradigm, including forward and inverse prediction errors, residual motion error, and local motion-compensated prediction error. We show results on a number of widely used motion and stereo sequences, many of which do not have associated ground truth data.*

1 Introduction

The ability to quantitatively evaluate the performance of competing algorithms is important in many branches of computer science, e.g., speech recognition, information retrieval, and machine learning. Quantitative evaluation allows us to measure progress in our field and motivates us to develop better algorithms. It allows us to carefully analyze algorithm characteristics and to improve overall performance by focusing on sub-components. It allows us to ensure that algorithm performance is not unduly sensitive to the setting of "magic parameters". Furthermore, it enables us to design or tailor algorithms for specific applications, by tuning these algorithms to problem-dependent cost or fidelity metrics and to sample data sets.

Unfortunately, computer vision does not have a very strong tradition of quantitative evaluation. In part, this is unavoidable, since computer vision encompasses some very ambitious goals such as general scene understanding. For more specific problems, such as *edge detection*, there is a tendency to evaluate the quality of results by visual inspection, which leads to subjective decision by experimenters or readers (but see, e.g., [10] for an attempt to quantify performance).

Two of the most widely studied problems in computer vision are motion estimation (sometimes called *optic flow*) and stereo correspondence. The goal in both problems is to place pixels in two or more images into correspondence so as to extract a dense (per-pixel) low-level description of the scene.

The availability of quantitative results for motion estimation improved a few years ago with the publication of Barron *et al.*'s comparative paper on optic flow estimation [5]. Since then, most motion estimation papers publish at least one quantitative result based on the Barron *et al.* data set. In stereo correspondence, it is still relatively rare to see quantitative evaluation (but see, e.g., [8]). Most papers that include comparative results tend to just publish depth maps for two or more algorithms, and leave it to the reader to gauge their relative quality.

One reason for this situation is that it is relatively difficult to get accurate ground truth results, when what is required is the exact motion vector or depth estimate at *each* pixel. (Some attempt has been made to provide ground truth at a *sparse* set of pixels [7], but as we argue below, this does not reflect the typical requirements of a lot of modern applications.) In motion estimation, the most widely used "realistic" motion data set (with ground truth) is the *Yosemite* data set, which is actually computer-generated, based on a texture-mapped digital terrain model. A less widely used data set [36] is a relatively simple scene made of textured marble blocks whose motion was labeled by hand. In stereo matching, a recent hand-labeled data set [33] is starting to be used, but the accuracy of the data is only to the nearest integer disparity level. Some synthetic test sequences have been developed for stereo matching [19, 17], but comparative quantitative results have not been reported.

A second problem with using ground truth data for evaluating motion or stereo algorithms is that it is unclear why overall root-mean square (RMS) deviation from ground truth should be a good predictor of a motion or stereo algorithm's utility. Is it, for example, necessary to be equally accurate in low-texture areas, where motion or stereo is difficult, as it is in textured areas? Are regions near discontinuities more or less important in evaluating algorithms?

In this paper, we propose an alternative to evaluating correspondence algorithms on the basis of their error from ground truth. Our suggestion is to measure how accurately the motion or depth estimates (combined with the original image) can *predict* the appearance of an image from a *novel* view or at a future time, i.e., the appearance of an image that was *not* used to compute the motion or depth estimate. We want to emphasize that this is *not* the same as motion-compensated prediction in video coding. Instead of being given two images and being asked to predict the second from the first (which is not that hard to do, given a general flow field), we are asked to predict novel views, not used in the matching.

Our approach has two advantages over measuring ground truth error. First, it is much easier to acquire datasets for which algorithms can be evaluated in this manner. For example, given any collection of three or more images, motion or stereo can be computed on a subset (two or more) of the images, and tested on the remaining images. (Another possibility would be to measure the *self-consistency* of estimates computed using different pairs of images [26].)

Second, prediction error is a useful indicator of the expected quality and utility of motion or stereo algorithm when used in newer application areas such as virtual reality, image-based rendering, and video editing or special effects [23, 6]. Examples of such applications include view interpolation [11] and view morphing [38], frame-rate conversion [35], and de-interlacing [13].[1] Many of these applications require the estimates to be accurate at *every* pixel (or sometimes even fractions of a pixel [44, 2]), since even small errors show up as "halos" or scintillating pixels.

The idea of partitioning data into *training* and *testing* subsets is common in many areas of computer science such as speech recognition. It is also a commonly used technique in statistics, where it is called *cross-validation* [12, 46], and can be used to recover unknown internal parameters and to prevent overfitting. Unlike many of these tasks, which involve classifying the inputs or producing a concise description, we propose a complete image prediction problem, where a large amount of data (several images) is used to estimate a large number of unknowns (the motion estimates), which in turn are used to predict another large set of data (the test images).

The remainder of this paper is structured as follows. We begin in Section 2 with a brief review of stereo and motion representations and estimation. In Section 3 we introduce our novel family of view prediction metrics. In Sections 4 and 5, we propose a number of novel view generation algorithms and a set of potential quality (cost) metrics. In Section 6 we use some simple experiments to demonstrate these metrics, and then show some results on standard data sets (with and without ground truth) using our new metrics. We close with a discussion of our results and some ideas for future work.

[1] We haven't included video compression as an application, since this also requires coding the motion itself, and hence would add additional cost terms to our quality metric.

2 Motion and stereo representations

Dense stereo or motion estimation can be formulated as follows. Given two or more images, find a per-pixel *correspondence* that matches each pixel in a given *reference image* to corresponding pixels in the other images. In general 2-D motion estimation, motion vectors can be in any direction. (If more than two images are used, it is common to assume that motion vectors are constant in time, i.e., that there is no acceleration.) In stereo matching, an *epipolar geometry* is usually computed ahead of time, which restricts the search for matching pixels to a 1-D *epipolar line* for each pixel. (Stereo matching naturally generalizes to multiple frames, without any restrictions on camera placement or motion, by associating a separate epipolar geometry (camera matrix) with each image.) Descriptions of commonly used stereo and motion estimation algorithms can be found in computer vision textbooks [3, 20, 34, 15] and in a number of survey articles [4, 1, 14, 9, 5, 32].

A single depth or motion map associated with a chosen reference image is just one possible representation for shape or motion. Single-valued depth/motion maps do not normally capture the full information available in a sequence of images, e.g., regions not visible in the reference image (due to occlusion or cropping to the image boundary) are not represented, and cannot be reproduced. For this reason, a number of alternative representation have be proposed in recent years.

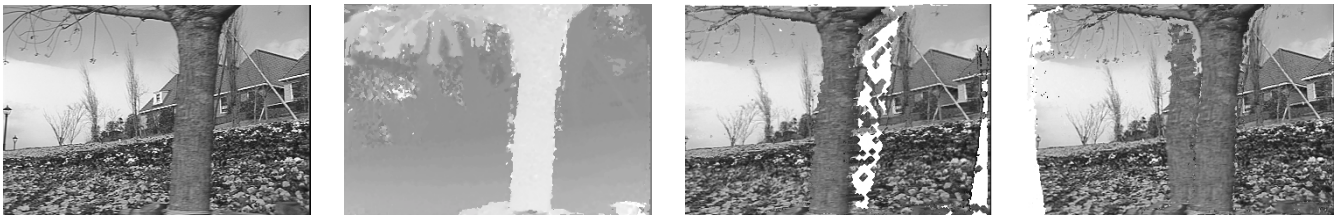
One possibility is to estimate more than one depth value per pixel. Such a multivalued representation can be thought of as a *volumetric* description of the scene (under some projective resampling of three dimensions). Stereo algorithms based on this representation have been developed [39, 44, 24], as well as novel image-based rendering algorithms [40]. Another possibility is to represent the scene as a collection of potentially overlapping *layers* [47, 48, 2]. Three-dimensional surface models are another possibility [18, 16]. Finally, motion or depth maps can be associated with more than just one image [43], and then used for novel view generation using image-based rendering [11] or bi-directional motion interpolation [27].

3 Prediction error as a quality metric

As we mentioned in the introduction, a traditional way to evaluate the quality of such algorithms is to measure the deviation in motion or depth estimates from *ground truth* motion or depth [5, 36, 33, 8]. Since such data sets are hard to come by, we propose using the input images themselves as both "training" and "test" data.

Given three or more images in a motion or stereo data set, we select a proper subset as input to our estimation algorithm (the "training" phase). We then predict the appearance of the remaining images (Section 4), and compute the difference between the predicted and actual images using some error metric (Section 5).

In general, we expect to see different kinds of errors for



(a) (b) (c) (d)

Figure 1. Flower garden sequence: (a) original frame 2, (b) estimated depth map, (c) frame 2 extrapolated (forward warped) to frame 10, (d) frame 10 inverse warped to frame 2. Note the different behavior of the occluded region behind the tree.

interpolating a novel view or frame which lies *between* the images chosen as input vs. *extrapolating* a view or frame *beyond* the set of input images. The first kind of error is more indicative of the expected performance of an algorithm in view interpolation, image-based rendering, and motion-compensated frame rate conversion applications, since it is in general dangerous to extrapolate from a given set of images (e.g., there may be disoccluded regions where no data exists).[2] On the other hand, extrapolation error will behave more like *ground truth* error metrics, since errors in uniform color regions will eventually contaminate extrapolated views, while they may not show up in interpolated views.

4 Novel view generation

Given an image and its associated depth or motion map, how do we generate a novel view or frame in order to compare it against a real image? In general, there is no single right or optimal answer, and the answer is highly dependent on the representation chosen. In fact, this question lies at the heart of current research into image-based rendering [40].

One possibility is to use texture-mapping hardware (or software) to paint the novel image as a collection of teeny-tiny triangles (two or four per input pixel or quad of pixels). Given a depth map, a 3D surface can be created (optionally mapping from projective depths to Euclidean coordinates), and then rendered from a novel viewpoint, using the original image as a texture map [30, 42]. Given a 2-D motion map, the reference image can again be drawn using a texture-mapped computer graphics rendering algorithm, with the triangles drawn at their displaced locations. However, in this case, the actual order of rendering may affect the final results, since there is no third dimension to correctly resolve visibility issues. This is an endemic problem with general 2-D motion interpolation (or extrapolation), and cannot be solved without making layering information explicit [47]. For better visual quality (less aliasing), bilinear or higher order texture interpolation should be used.

A second possibility is to use a forward-mapping or *splatting* algorithm [49]. Here, each pixel is painted at the location where it would land in the novel image (in the case of depth maps, a back-to-front ordering can be enforced [31]). Overwriting the nearest pixel results in a poor quality rendering (a lot of aliasing and gaps). Therefore, most splatting algorithms use a soft kernel that increments several adjacent pixels. This kind of operation may require a post-processing stage to renormalize colors, or may result in order-dependent artifacts [40].

A third possibility is to use a two-pass algorithm, where a depth map or flow map is first created for the novel view [40]. Since depth or motion maps tend to vary smoothly (at least, away from discontinuities), a nearest neighbor (single pixel) splat can often be used in this stage, followed by single-pixel gap filling. The new depth or motion map can then be used to perform an *inverse* warping algorithm, i.e., to find the interpolated pixel value in the reference image that corresponds to each (non-empty) novel pixel. Figure 1c shows an image produced with this two-pass algorithm. Note how the regions behind the tree show up as gaps with "empty" pixels (no values).

In general, all of these forward mapping algorithms have to deal with a number of thorny issues. One of these issues is the question of discontinuities: when a break occurs in a motion or depth map, do we interpolate across the break, or leave the disoccluded area blank? In the latter case, do we fill with some background color, or flag these pixels as special and adjust the error metric accordingly? And how does an algorithm estimate the discontinuities in the first place? Another issue is the appropriate resampling/interpolation function. We know that bilinear interpolation of intensities/color is better than nearest neighbor, but how much better are the results with higher order interpolators? Do we still want to use higher order filters near image edges and discontinuities?

An alternative to these forward mapping algorithms is to directly perform an inverse warping of the actual novel view, i.e., to resample the new image so that it conforms (as best as possible) to the reference image. While this breaks the spirit of our train and test (prediction) paradigm, it can still give us useful quality metrics, as we will show in the experimental section of this paper. Figure 1d shows an example of inverse warping. Note how the pixels in regions that become occluded

[2] In fact, Seitz [37] proves that for a photoconsistent reconstruction [24], interpolated views *will* be rendered correctly, except for regions partially or fully occluded in the input views.

in later frames are painted with colors from the occluding foreground object (the tree).

So far, we have only discussed novel view generation algorithms for the simplest case of a single reference image with an associated depth or motion map, as this is the case we focus on in this paper. Other motion or shape estimation representations and algorithms entail their own associated view generation algorithms. For example, a volumetric data set can be visualized using a back-to-front warping and compositing algorithm [25, 44]. Layered representations can be rendered by first warping (rendering) each layer separately, and then compositing the resulting images in back-to-front order (which enables a better prediction of mixed foreground/background pixel values [2]). 3-D surface models can be rendered using texture-mapping algorithms. Multiple sets of colored depth or motion maps can be rendered using image-based rendering techniques that blend rendered images while preserving visibility relationships.

The quality estimates produced by the error metrics discussed in the next section depend heavily on the choice (and quality) of the prediction or resampling algorithms. This can be mitigated somewhat by fixing the rendering algorithm while varying the estimation algorithm or its parameters. To a certain extent, however, viewing the estimation and rendering algorithms as part of a complete system whose end-to-end performance is being optimized may be unavoidable and probably even desirable.

5 Error metrics

Once we have synthesized the appearance of predicted images, how do we measure their fidelity? The simplest method is to compute the root mean square (RMS) difference between the two images, expressed in gray levels. However, depending on the acquisition hardware, individual images may be differently exposed. It is therefore prudent to compute a global bias and gain (additive and linear) correction to apply to one of the two images before measuring RMS error. (Ideally, we would like to use perceptually-based error metrics [28], but the state of research in this field is still not very advanced. This is likely to be an important area of future research.)

Even once we have compensated for exposure effects, we still often find that the error is far from being identically distributed Gaussian noise. In fact, the magnitude of the brightness error is strongly correlated with the amount of local intensity variation. Simoncelli *et al.* [41] explain that this can arise because the gradient estimates used in most flow estimation techniques are themselves noisy. Other sources of error include sub-pixel shifts in the digitization process [30], mis-estimation of the epipolar geometry which can result in *vertical parallax*, and general re-sampling (interpolation) errors due to poor quality (e.g., bilinear) filters or aliasing during image acquisition.

Given that these problems are endemic, would it be fairer to down-weight the squared error at each pixel by some term that includes the local intensity gradient [41]? If we do this, then are we still getting good estimates of the perceptual quality of an estimation / rendering combination?

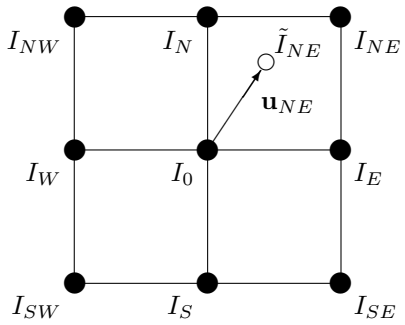


Figure 2. Residual flow computation and value compensation.

5.1 Residual motion estimation and compensation

A better option is to estimate the *residual flow* [22], which is the per-pixel estimate of flow required to register the predicted and actual novel image. We can then compute the RMS residual flow magnitude as well as the RMS intensity error after residual motion compensation, in order to get a feel for how much of the prediction error is due to small mis-registrations.

In our current implementation, since we do not want regions with little texture variation to contribute to the residual flow measure, we compute residual flow as follows (see Figure 2). For each pixel I_0 in the image to be corrected, we examine each of the four quads of pixels surrounding I_0. We use the nearest two neighbors to form a linear approximation to the interpolated intensity, e.g.,

$$\tilde{I}_{NE} = I_0 + (I_E - I_0)u_{NE} + (I_N - I_0)v_{NE}. \quad (1)$$

We find the residual flow vector $\mathbf{u}_{NE} = (u_{NE}, v_{NE})$ that minimizes

$$\|\tilde{I}_{NE} - I_1\|^2 + \lambda\|\mathbf{u}_{NE}\|^2, \quad (2)$$

where I_1 is the corresponding pixel in the image we are correcting against, and the second term suppresses the computation of spurious residual flow vectors in region with little texture or color variation (see [41] for a probabilistic justification).[3]

We clip the four candidate flow vectors to lie within the quad being examined (which means that many of the vectors get clipped to $(0, 0)$), and compute a better interpolated color value $\tilde{I}$ using *bilinear* interpolation. We then pick the residual

[3] Minimizing the above cost is similar to a one-pixel version of the Lucas-Kanade flow estimation algorithm [29], and essentially computes normal flow. For all of our experiments, we set $\lambda = 16 \cdot (\#bands)$, where $\#bands$ is the number of bands (3 for color images, 1 for monochrome).

flow vector with the lowest $\|\tilde{I} - I_1\|$ value (or $(0,0)$ if none of these errors beats $\|I_0 - I_1\|$), and set the corrected pixel value to $\tilde{I}$. Some examples of estimated residual flows and corrected images are given in the next Section.

5.2 Outliers and invisible pixels

Two other issues that arise in computing error statistics are outliers (robust statistical estimates), and invisible pixels (i.e., pixels for which there are no corresponding pixels in other images). Traditional statistical measure such as the standard deviation of the intensity errors can be heavily affected by outliers, which may be caused by occlusions, variation of brightness with viewpoint, and residual motion errors. In order to get a more robust set of error statistics, we compute a robust measure of the standard deviation using $\sigma = 1.4826\ \text{med}|I_1 - I_0|$ [21]. We also compute the percentage of outliers, i.e., the number of pixels for which $|I_1 - I_0| > 3\sigma$. These statistics are reported along with the more traditional RMS (root mean square) error.

In order to compensate for pixels that are invisible in other images (e.g., pixels whose correct motion has carried them outside the image frame), we modify our forward and inverse warping algorithms to flag pixels as *invisible* when their source is outside the image (inverse warping), or when no pixels map to a given pixel after gap filling (forward warping). These pixels do not participate in the computation of RMS or robust statistics, but their percentage is reported in the experimental section.

6 Experiments

Because of space limitations, we only have room to present a few simple experiments to demonstrate the behavior of our error metrics. We are currently undertaking a more comprehensive set of comparative experiments, focusing in particular on dense two-frame stereo matching [45].

6.1 Synthetic flow error

To get a sense of the behavior of our error metrics, we first generated a synthetic motion field by taking an image from the flower garden video (Figure 1), and using $(u, v) = (k/16, k/8), k = 0 \ldots 4$ as motion field estimates (the true motion is $(0,0)$). Figure 3a shows a variety of error metrics as a function of frame number k (increasingly erroneous flow estimates). The raw RMS error increases linearly as a function of k, as does the robust estimate of σ. Figure 3c shows the uncompensated difference image for $k = 4$, and Figures 3d–e show the horizontal and vertical components of the residual flow estimates. Notice how there is little flow in untextured areas, but that the flow is in general quite noisy. (Remember that no area-based averaging is used to compute these flows—they are used solely to reduce the difference error. It may be fruitful to compute residual flows over larger areas, as is done in [22]) After compensation, the RMS error (and robust error)

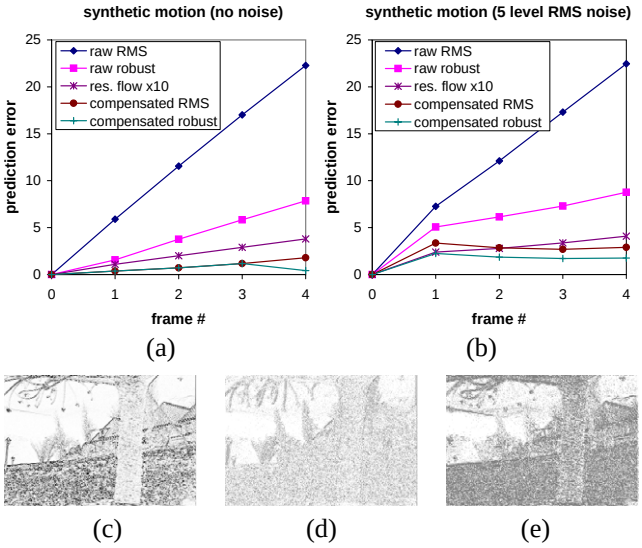


Figure 3. Synthetic motion errors: (a) no noise, (b) $\sigma = 5$. (c) uncompensated difference image, (d–e) horizontal and vertical residual flow estimates.

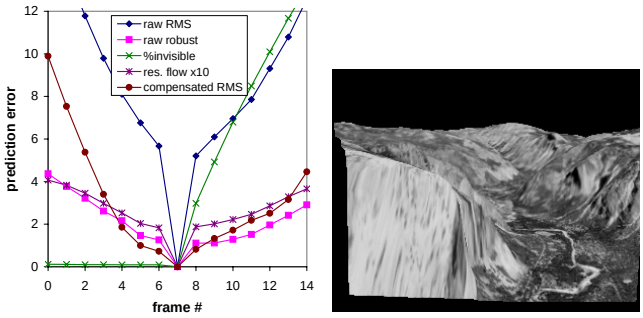
are both very small (imperceptible in the difference image, which is not shown).

To generate these plots, we used inverse warping, optionally followed by residual flow estimation and image compensation. The image being compensated was always the *original* and not the *warped* image. This ensures that the two images compared have been resampled the same number of times. If we interchange the sense of which image is compensated, we find that there is very little improvement in the RMS error after compensation. On a smoother image (e.g., the SineC sequence used in [5]), this asymmetry goes away. However, for realistic images, it is very important to compare apples to apples, i.e., to try to ensure that resampling errors are similar between images being compared.

If we add synthetic noise to the images in the sequence, the overall noise estimates go up. Figure 3b shows the error plots when Gaussian noise with $\sigma = 5$ was added. While the uncompensated difference does not change much (which indicates that motion error dominates), the compensated difference is now much higher, with a standard deviation of about 2.5 (which indicates that residual motion compensation is removing about half of the imaging noise).

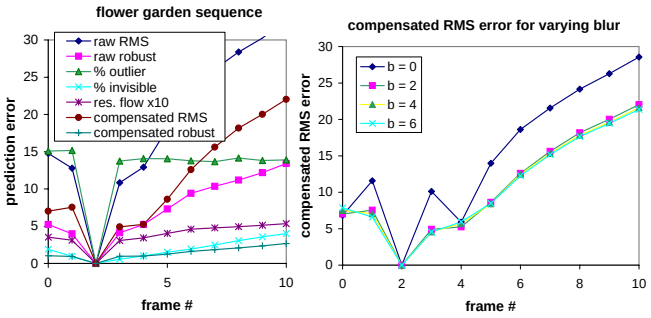
6.2 Sequences with ground truth

To demonstrate the behavior of our metrics when ground truth motion is known, we chose the Yosemite sequence. (Table 1 and Figure 6 list some more image sequences for which ground truth motion or disparity is known.) Using the ground truth flows for the middle frame (frame 7), we computed the raw RMS and robust error metrics, the residual flow, and the compensated RMS error metric. Figure 4a shows these plots, along with the percentage of invisible pixels. As the sequence



(a) (b)

Figure 4. Yosemite sequence: (a) plots of prediction errors; (b) frame 14 inverse warped to frame 7 (note the missing pixels along the lower left edge).



(a) (b)

Figure 5. Flower garden sequence: (a) plot of prediction errors based on motion estimated from frames 0, 2, and 4; (b) plots error for various amount of blurring

progresses, the lower left section of the image moves out of view, so that more an more pixels become invisible during backward warping (Figure 4b).

The plots show that the error continuously increases as we move away from the reference frame. By looking at the frames inverse warped toward the central frame, we have observed that the ground truth motion does not accurately predict the appearance of frames far from the center. In fact, because the motion is looming, the flows have a significant acceleration, which shows up as a "swinging" motion on the foreground object. Using a rigid motion model (with known disparities) should alleviate a lot of this problem.

6.3 Sequences without ground truth

Figure 5a shows the error metrics applied to a disparity field computed from frames 0, 2, and 4 of the flower garden sequence (Figure 1). The stereo algorithm used was a simple plane sweep algorithm, which is functionally equivalent to a robustified sum-of-squared-differences (SSD) algorithm. Instead of aggregating information over a square window, iterative convolution with a binomial kernel was used before the min-picking stage.

Since the motion was computed on the first three even frames, we might expect the prediction error to be lower at these frames than at other frames. On the other hand, there is a tendency for prediction error to increase systematically away from the reference frame, both due to errors in motion estimation, and other effects such as disocclusions, sampling artifacts, and photometric effects. Figure 5a shows that both of these effects are indeed present. The raw RMS and robust errors increase monotonically away from the reference frame, while the compensated error is slightly higher at the two interpolated frames (1 and 3) than at the two frames used in the stereo computation (2 and 4).

To see whether cross-validation could be used to automatically tune one of the stereo algorithm parameters, we re-ran our algorithm with a varying number of blurring steps after the initial per-pixel cost computation. Figure 5b shows the compensated RMS error for various choices of b (the number of blurring iterations). Not using any blurring results in significantly worse estimates, whereas there is not much difference between the other choices. However, observe that if we were given just four frames from the sequence (e.g., frames 0, 1, 2, and 4), we could use the prediction error for frame 1 to tune b for the three-frame stereo algorithm being run on the other three frames. Figure 5b shows that this would give us a good choice of b, since the interpolation error at frame 1 is a good predictor of the relative prediction errors at other frames. We are currently applying this kind of analysis to other stereo algorithm parameters, such as window size and the amount of regularization [45].

7 Discussion and Conclusions

In this paper, we have introduced prediction error as a novel quality measure for evaluating and designing motion and stereo correspondence algorithms. Prediction error is much easier to obtain than ground truth data for multi-image sequences. It also more closely matches modern requirements for motion and stereo algorithms. We have shown how raw prediction error depends on the interaction between motion errors and local intensity gradient structure, and suggested a means (residual flow compensation) for mitigating this factor.

Quantitative evaluation is essential if we are to continue making progress in the development of motion and stereo algorithms. Not only does it enable us to judge when an advance is truly worthwhile, but it also enables us to dissect existing algorithms and approaches to learn which components are responsible for their good (or bad) behavior, and to determine how sensitive they are to internal parameters. It also holds out the hope that algorithms can themselves discover (*learn?*) internal parameter values, using generalized cross-validation to see whether their outputs accurately predict images that have intentionally been held back.

The work presented in this paper is just a first step towards what we hope will become an accepted framework for research in this area. Our models to date do not incorporate any notion

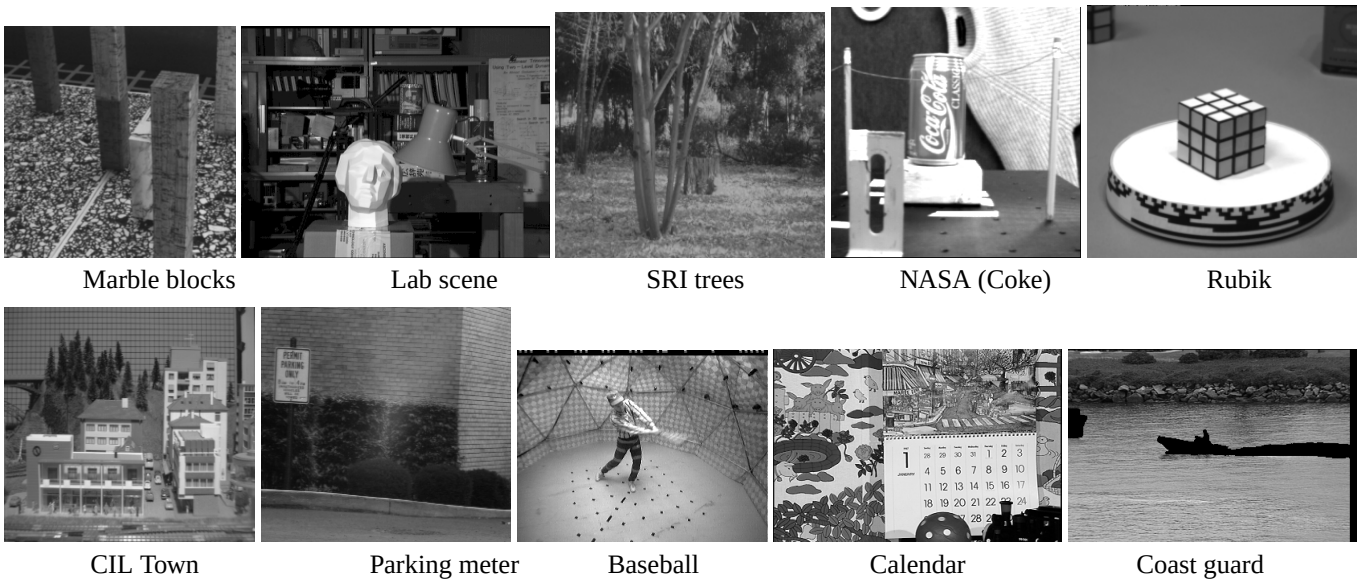


Figure 6. Images from some commonly used multi-image sequences

of perceptual quality or perceptual similarity. Adding such models should prove to be very usuful, although the state of research in this area is still not very advanced. An interesting question this bring up is: is it better for an algorithm to *look good* (i.e., to have no visible artifacts), or to *be good*, i.e., to accurately predict the geometric and photometric appearance of a scene.

Other variants on prediction error should also be explored. For instance, given a 3D reconstruction of a scene from stereo, it should be possible to excise foreground elements or insert mid-ground elements. Algorithms that mis-classify pixels near object boundaries will produce "halos" when such element insertion is performed. It should be possible to structure the acquisition process to add and remove scene elements. It would also be interesting to compare our new methodology with the self-consistency metric proposed in [26].

One question that comes up when considering prediction error as a quality metric is: *why not just directly optimize the 3D scene description with respect to how closely it matches the input images?* While this approach is at the heart of many newer stereo algorithms, it fails to take into account that algorithms will often happily overfit their input data, unless some data is held back to "keep them honest".

In current work, we are starting to apply our methodology to a number of stereo algorithms, in order to evaluate their quality, refine their behavior, and shed more light on desirable properties of quality metrics [45]. It is our hope that our approach will lead to a marked increase in the quantitative evaluation of low-level vision algorithms. Rather than providing yet another mechanism for researchers to bag "bragging rights" about their latest results, we hope that this will help increase our general understanding of the nature and behavior of low-level motion and structure estimation algorithms.

Name	Source	# img.	motion	grnd. truth
Yosemite	Quam [5][1,2]	15	rigid[6]	flow/Z
Marble block	Otte [36][3]	31	rigid[7]	flow
Lab scene	Nakam. [33]	25	rigid	Z
SRI Trees	Bolles [5][1]	21	rigid	-
NASA	NASA [5][1]	37	rigid	-
Rubik	Szeliski [5][1]	21	non-rig.	-
CIL town	Matth. [30][4]	60	rigid	-
Park. meter	CMU[4]	25	rigid	-
Baseball	Kanade [23][5]	51	rigid[8]	-
Flower grdn	MPEG-4	150	rigid	-
Calendar	MPEG-4	300	non-rig.	-
Coast guard	MPEG-4	300	non-rig.	-

1. ftp://csd.uwo.ca/pub/vision/TESTDATA
2. http://www.parc.xerox.com/spl/members/black
3. http://i21www.ira.uka.de/image_sequences
4. http://www.ius.cs.cmu.edu/idb
5. http://www.cs.cmu.edu/afs/cs/project/VirtualizedR/www
6. clouds are non-rigid; 7. one block moves non-rigidly; 8. 51 *movies*.

Table 1. List of some commonly used multi-image sequences

References

[1] J. K. Aggarwal and N. Nandhakumar. On the computation of motion from sequences of images—a review. *Proceedings of the IEEE*, 76(8):917–935, Aug. 1988.

[2] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. In *CVPR'98*, pp. 434–441, June 1998.

[3] D. H. Ballard and C. M. Brown. *Computer Vision*. Prentice-Hall, Englewood Cliffs, New Jersey, 1982.

[4] S. T. Barnard and M. A. Fischler. Computational stereo. *Computing Surveys*, 14(4):553–572, Dec. 1982.

[5] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance

of optical flow techniques. *Intl. J. Comp. Vis.*, 12(1):43–77, Jan. 1994.

[6] L. Blonde *et al.* A virtual studio for live broadcasting: The Mona Lisa project. *IEEE Multimedia*, 3(2):18–29, 1996.

[7] R. C. Bolles, H. H. Baker, and M. J. Hannah. The JISCT stereo evaluation. In *Image Under. Work.*, pp. 263–274, 1993.

[8] Y. Boykov, O. Veksler, and R. Zabih. A variable window approach to early vision. *IEEE Trans. Patt. Anal. Mach. Intell.*, 20(12):1283–1294, Dec. 1998.

[9] L. G. Brown. A survey of image registration techniques. *Computing Surveys*, 24(4):325–376, Dec. 1992.

[10] J. Canny. A computational approach to edge detection. *IEEE Trans. Patt. Anal. Mach. Intell.*, PAMI-8(6):679–698, Nov. 1986.

[11] S. Chen and L. Williams. View interpolation for image synthesis. *SIGGRAPH'93*, pp. 279–288, Aug. 1993.

[12] P. Craven and G. Wahba. Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. *Numerische Mathematik*, 31:377–403, 1979.

[13] G. de Hann and E. B. Beller. Deinterlacing—an overview. *Proceedings of the IEEE*, 86(9):1839–1857, Sept. 1998.

[14] U. R. Dhond and J. K. Aggarwal. Structure from stereo—a review. *IEEE Trans. Sys. Man Cyber.*, 19(6):1489–1510, Nov./Dec. 1989.

[15] O. Faugeras. *Three-dimensional computer vision: A geometric viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.

[16] O. Faugeras and R. Keriven. Variational principles, surface evolution, pdes, level set methods, and the stereo problem. *IEEE Trans. Im. Proc.*, 7(3):335–344, Mar. 1998.

[17] T. Frohlinghaus and J. M. Buhmann. Regularizing phase-based stereo. In *ICPR'96*, volume A, pp. 451–455, Aug. 1996.

[18] P. Fua and Y. G. Leclerc. Object-centered surface reconstruction: Combining multi-image stereo and shading. *Intl. J. Comp. Vis.*, 16:35–56, 1995.

[19] W. Hoff and N. Ahuja. Surfaces from stereo: integrating feature matching, disparity estimation, and contour detection. *IEEE Trans. Patt. Anal. Mach. Intell.*, 11(2):121–136, Feb. 1989.

[20] B. K. P. Horn. *Robot Vision*. MIT Press, Cambridge, Massachusetts, 1986.

[21] P. J. Huber. *Robust Statistics*. John Wiley & Sons, New York, New York, 1981.

[22] M. Irani, B. Rousso, and S. Peleg. Detecting and tracking multiple moving objects using temporal integration. In *ECCV'92*, pp. 282–287, May 1992.

[23] T. Kanade, P. W. Rander, and P. J. Narayanan. Virtualized reality: constructing virtual worlds from real scenes. *IEEE MultiMedia Magazine*, 1(1):34–47, Jan.-Mar. 1997.

[24] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. In *ICCV'99*, Sept. 1999.

[25] P. Lacroute and M. Levoy. Fast volume rendering using a shear-warp factorization of the viewing transformation. *SIGGRAPH'94*, pp. 451–457, July 1994.

[26] Y. G. Leclerc, Q.-T. Luong, and P. Fua. Self-consistency: A novel approach to characterizing the accuracy and reliability of point correspondence algorithms. In *DARPA Image Understanding Workshop*, Nov. 1998.

[27] D. Le Gall. MPEG: A video compression standard for multimedia applications. *Comm. of the ACM*, 34(4):44–58, Apr. 1991.

[28] J. Lubin. A human vision system model for objective picture quality measurements. *IEE Conference Publications*, 1(447):498–503, 1997.

[29] B. D. Lucas and T. Kanade. An iterative image registration technique with an application in stereo vision. In *IJCAI-81*, pp. 674–679, 1981.

[30] L. H. Matthies, R. Szeliski, and T. Kanade. Kalman filter-based algorithms for estimating depth from image sequences. *Intl. J. Comp. Vis.*, 3:209–236, 1989.

[31] L. McMillan. A list-priority rendering algorithm for redisplaying projected surfaces. Technical Report 95-005, University of North Carolina, 1995.

[32] A. Mitiche and P. Bouthemy. Computation and analysis of image motion: A synopsis of current problems and methods. *Intl. J. Comp. Vis.*, 19:29–55, 1996.

[33] Y. Nakamura, T. Matsuura, K. Satoh, and Y. Ohta. Occlusion detectable stereo - occlusion patterns in camera matrix. In *CVPR'96*, pp. 371–378, June 1996.

[34] V. S. Nalwa. *A Guided Tour of Computer Vision*. Addison-Wesley, Reading, MA, 1993.

[35] O. A. Ojo and G. de Haan. Robust motion-compensated video upconversion. *IEEE Trans. Cons. Elec.*, 43(4):1045–1056, Nov. 1997.

[36] M. Otte and H.-H. Nagel. Optical flow estimation: advances and comparisons. In *ECCV'94*, volume 1, pp. 51–60, May 1994.

[37] S. M. Seitz and C. M. Dyer. Toward image-based scene representation using view morphing. In *ICPR'96*, volume A, pp. 84–89, Aug. 1996.

[38] S. M. Seitz and C. M. Dyer. View morphing. In *SIGGRAPH'96*, pp. 21–30, Aug. 1996.

[39] S. M. Seitz and C. M. Dyer. Photorealistic scene reconstrcution by space coloring. In *CVPR'97*, pp. 1067–1073, June 1997.

[40] J. Shade, S. Gortler, L.-W. He, and R. Szeliski. Layered depth images. In *SIGGRAPH'98*, pp. 231–242, July 1998.

[41] E. P. Simoncelli, E. H. Adelson, and D. J. Heeger. Probability distributions of optic flow. In *CVPR'91*, pp. 310–315, June 1991.

[42] R. Szeliski. Video mosaics for virtual environments. *IEEE Computer Graphics and Applications*, pp. 22–30, Mar. 1996.

[43] R. Szeliski. A multi-view approach to motion and stereo. In *CVPR'99*, pp. 157–163, June 1999.

[44] R. Szeliski and P. Golland. Stereo matching with transparency and matting. In *ICCV'98*, pp. 517–524, Jan. 1998.

[45] R. Szeliski and R. Zabih. An experimental comparison of stereo algorithms. In preparation, 1999.

[46] G. Wahba. Bayesian "confidence intervals" for the cross-validated smoothing spline. *J. Roy. Stat. Soc.*, B 45(1):133–150, 1983.

[47] J. Y. A. Wang and E. H. Adelson. Representing moving images with layers. *IEEE Trans. Im. Proc.*, 3(5):625–638, Sept. 1994.

[48] Y. Weiss. Smoothness in layers: Motion segmentation using nonparametric mixture estimation. In *CVPR'97*, pp. 520–526, June 1997.

[49] G. Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, California, 1990.