Recovering Motion Fields: An Evaluation of Eight Optical Flow Algorithms

B. Galvin, B. McCane, K. Novins, D. Mason and S. Mills
Computer Science Department
University of Otago, New Zealand.
bgalvin@atlas.otago.ac.nz

Abstract

Evaluating the performance of optical flow algorithms has been difficult because of the lack of ground-truth data sets for complex scenes. We describe a simple modification to a ray tracer that allows us to generate ground-truth motion fields for scenes of arbitrary complexity. The resulting flow maps are used to assist in the comparison of eight optical flow algorithms using three complex, synthetic scenes. Our study found that a modified version of Lucas and Kanade's algorithm has superior performance but produces sparse flow maps. Proesmans et al.'s algorithm performs slightly worse, on average, but produces a very dense depth map.

1 Introduction

Seventeen years have passed since Horn and Schunck published their influential paper on the calculation of optical flow [4]. Since then, a substantial amount of research has been devoted to finding ways to calculate optical flow more efficiently and more accurately. Optical flow extraction has been proposed as a preprocessing step for many high level vision algorithms.

Despite the volumes of research on this topic, few attempts have been made to evaluate the performance of these algorithms on complex image sequences. This is at least partly due to the difficulties in obtaining ground truth motion fields for complex scenes.

This paper builds on the the research of Barron et al. [1]. They conducted an empirical analysis of nine optical flow algorithms. The algorithms were tested on five synthetic image sequences where ground truth motion fields were available and four real image sequences for which no ground-truth motion field was available. In the four real image sequences, no objective evaluation of performance was available. Four of the synthetic sequences were constructed from simple 2D image operations, for example, translating a simple pattern. In these cases the ground-truth motion field was trivially available, however it was not known how relevant these results where to real image sequences. The final synthetic scene was more complex. It shows a fly-through of the Yosemite national park, however there is little occlusion, and none of the complex lighting effects that occur in real scenes.

We introduce a simple modification that allows a ray tracer to generate ground-truth motion fields for arbitrarily complex scenes. The source code for the modified ray tracer

is available at http://heffalump.otago.ac.nz/∼masondr/of/. We use the resulting ground-truth motion fields to evaluate the performance of eight optical flow algorithms, six of which were also tested by Barron et al. The three test sequences and their corresponding ground-truth motion fields are also available.

2 What is Ground-Truth Optical Flow?

Horn and Schunck define optical flow as follows:

> The optical flow is a velocity field in the image which transforms one image into the next image in a sequence. As such it is not uniquely determined ... The motion field, on the other hand, is a purely geometric concept, without any ambiguity - it is the projection into the image of three-dimensional motion vectors [5].

In practice, specular effects, shadows, insufficient texturing, and occlusion make the unambiguous recovery of the true motion field impossible. If these effects are present to sufficient degree, the optical flow vectors produced can be uncorrelated to the ground-truth motion field.

It would be possible to evaluate algorithm performance according to the definition above by measuring how well the optical flow field transforms one image into the next. However this is inadequate for 3D structure recovery. Horn and Schunck state: "One endeavors to recover an optical flow that is close to the motion field" [5]. A more appropriate measure is then the correlation between the calculated optical flow and the true motion field. It should be noted that this measure is better suited to applications which attempt to extract 3 dimensional structure, rather than image coding applications.

3 Generating the Ground-Truth Motion Field

The ground-truth motion fields were generated by modifying a public license ray tracer, Mirage [2]. Mirage objects can be animated by specifying affine transformations which describe their rigid motion from one frame to the next.

The motion field is generated during the ray tracing of two consecutive frames. The entire process is illustrated in Figure 1. Rendering of the first frame begins by casting rays from the camera into the scene. Whenever a ray strikes an object, the 3D position of the intersection, the object, and the corresponding pixel position (x, y) are recorded. Before the second frame is rendered, Mirage applies a transformation to each object. In our modified version of Mirage, the transformation applied to each object is also applied to each recorded ray-object intersection.

Once this has been completed, rays are cast from each new intersection point to the camera. Simple geometry yields the new pixel position (x', y'). A simple subtraction then yields the ground-truth motion vector.

4 Error Metrics

Trying to summarize the performance of over a million flow vectors with a single number is a difficult task. It is further complicated by attempting to define a general, non-

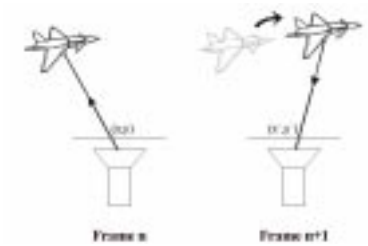


Figure 1: Geometry for calculating the the ground-truth motion field.

application specific measure. We felt simplicity was important for any error metric, even if it was at the expense of more subtle characteristics.

Our first error metric is the difference in angle between the correct and estimated flow vectors. We define

$$\text{Angular Error} = \cos^{-1}(\hat{c} \cdot \hat{e}), \tag{1}$$

where c is the correct motion vector, e is the estimate optical flow vector, and $\hat{}$ denotes vector normalization.

The second metric is the average magnitude of the vector difference between the correct and estimated flow vectors for each pixel. This can be pictured as the magnitude of the "correction image" that would have to be added to the calculated flow to get the correct flow:

$$\text{Magnitude of Difference} = \|c - e\|. \tag{2}$$

One of the more significant problems encountered in the calculation of optical flow is the aperture problem. In order to measure how effectively each algorithm compensates for this problem, we use an additional metric:

$$\text{Error Normal to Gradient} = \|(c - e) \cdot \hat{g}^{\perp}\|, \tag{3}$$

where $g = (\frac{\partial I}{\partial x}(x, y), \frac{\partial I}{\partial y}(x, y))$ and $\hat{g}^{\perp}$ denotes the vector perpendicular to $\hat{g}$. A large error normal to the image gradient would suggest the algorithm does not compensate adequately for the aperture problem.

5 Optical Flow Algorithms

Many of the algorithms reviewed in this paper were also reviewed and implemented by Barron et al. [1], and are available via anonymous ftp from `ftp.csd.uwo.ca` in the directory `/pub/vision`. Due to space restrictions we will only describe the algorithms not mentioned in Barron et al.'s paper. Instead, the reader is referred to their paper for a more detailed description, or to the original source for a complete description.

5.1 Algorithms Implemented by Barron et al.

5.1.1 Anandan

In addition to testing the original algorithm, we also tested the performance of this algorithm when a threshold was used to determine which flow vectors were calculated. Thresholding occurred when the minimum confidence level was less than 0.01.

5.1.2 Horn and Schunck

We have tested 4 versions of this algorithm. As well as Horn and Schunck's original algorithm, we tested the performance of this algorithm when a threshold based on the magnitude of the image gradient was used: $(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2 < 2.5^2$. The original formulation of this algorithm uses a two-point central difference approximation to image derivatives. We also evaluated performance when a four-point central difference approximation was used, and when this was used in conjunction with thresholding.

5.1.3 Lucas & Kanade

Four versions of this algorithm were tested. We have tested the original algorithm, and a version which uses a thresholding technique developed by Simoncelli et al. . The thresholding technique was based on a Bayesian interpretation of the original algorithm. Simoncelli et al. have used this interpretation to derive a slightly different way of solving the underlying equations. We have also evaluated this algorithm by itself, and when used in conjunction with the Bayesian thresholding technique.

5.1.4 Nagel

We have tested the original algorithm, and a version which used a threshold based on the magnitude of the image gradient. Thresholding occurs when $(\frac{\partial I}{\partial x})^2 + (\frac{\partial I}{\partial y})^2 < 2.5^2$.

5.1.5 Singh

This algorithm consists of 2 stages. The first stage uses a region matching technique to determine flow. The second stage propagates the flow to neighbouring pixels. Results are reported for both stages.

5.1.6 Uras et al.

We have tested the original algorithm, as well as a version which uses a threshold based on the determinant of the Hessian of $I(x, y, t)$. Thresholding occurs where $det(H) \leq 1.0$.

5.2 Algorithms not Implemented by Barron et al.

5.2.1 Camus [3]

This is a simple area matching algorithm that uses an exhaustive search over a small neighbourhood, across the previous n frames. For each pixel position it calculates a value

which indicates how likely it is that the pixel has moved into that spatio-temporal location. A typical matching function is

$$\text{match value} = \sum_{i,j} \|I(x+i, y+j, t) - I(x+\delta x+i, y+\delta y+j, t+\delta t)\|. \quad (4)$$

Once the optimum match value has been found, the flow is calculated as $\frac{1}{\delta t}(\delta x, \delta y)$. Two versions of this algorithm were tested. The first uses only 3 frames for each calculation, the second uses 15. Both algorithms used a 3x3 matching neighbourhood.

5.2.2 Proesmans et al. [6]

Proesmans et al. present a method similar to that of Horn and Schunck. The main differences are that a matching process is incorporated into the constraint equation and a method for dealing with discontinuous flows is introduced.

In order to evaluate the current flow estimate, Proesmans et al. look at the correspondences suggested by the current flow estimate. For each point, (x, y), in the first image the estimated optical flow, (u, v), can be used to find a corresponding point, $(x+u, y+v)$, in the second image. If the flow estimate is good then the brightness at these two points should be similar. If the brightnesses are not the same then the flow estimate is moved along the image gradients in order to correct for the difference.

A process known as *anisotropic diffusion* is used to smooth the flows. This means that the local averages of the flow components are not simple averages but are weighted by a consistency map. Unlike the normal smoothing process, this can maintain discontinuities in the flow but still retains a high level of continuity in local regions.

Our implementation used a four level image pyramid to account for large flows, and fifty relaxation iterations. The smoothness constraint, λ, was set to 30.

6 Results

In this section we refer to each algorithm by the name of the author(s), followed by a *T* if the algorithm used a threshold to restrict which flow vectors were calculated, or an *M* if we are referring to a modified version of the algorithm. See Section 5 for a brief description of any thresholding criteria or modifications. We have used the same parameters as Barron et al. unless stated otherwise.

Three test sequences were used of varying complexity. The first sequence shows a simple textured sphere rotating about its own axis. A similar scene was used by Horn & Schunck [4]. A typical frame from this sequence is shown in Figure 2(a). A summary of algorithm performance on this scene is displayed in Figure 3. Some example flow maps taken from a small region of this scene are displayed in Figure 7. An office scene is the subject of the second image sequence. The camera moves in the direction of its viewing axis, at a constant velocity. Specular highlights and transparency are deliberately included to simulate a real scene as much as possible. A typical frame is shown in Figure 2(b). Algorithm performances are summarized in Figure 4. The final sequence is set on a simulated busy street corner. The camera follows a car as it turns into a side-street. A typical frame is shown in Figure 2(c). Algorithm performances are displayed in Figure 5.

Up to this point, the image sequences have been noiseless (although slight aliasing artifacts are introduced by the ray tracing process). To better simulate real data, the algorithms were re-tested on the office sequence after a small amount of noise was added

(a) Rotating Sphere Scene (b) Office Scene (c) Street Scene

Figure 2: Typical scenes from the test sequences. All image sequences were rendered at 200x200 resolution in 256 level gray-scale. The rotating sphere and office sequence are 60 frames long. The street sequence is 150 frames long.

to each image in the sequence. The noise was taken from a Gaussian distribution with standard deviation 3, and was almost undetectable visually. Figure 6(a) shows how this noise affected the Average Magnitude of Difference error metric.

The algorithms vary widely in execution times. The execution times required to extract a single flow map from a 200x200x256 gray level image sequence are displayed in Figure 6(b). Obviously these timings are implementation dependent and should only be used as rough guidelines. Timings are for a Pentium Pro 180.

7 Discussion

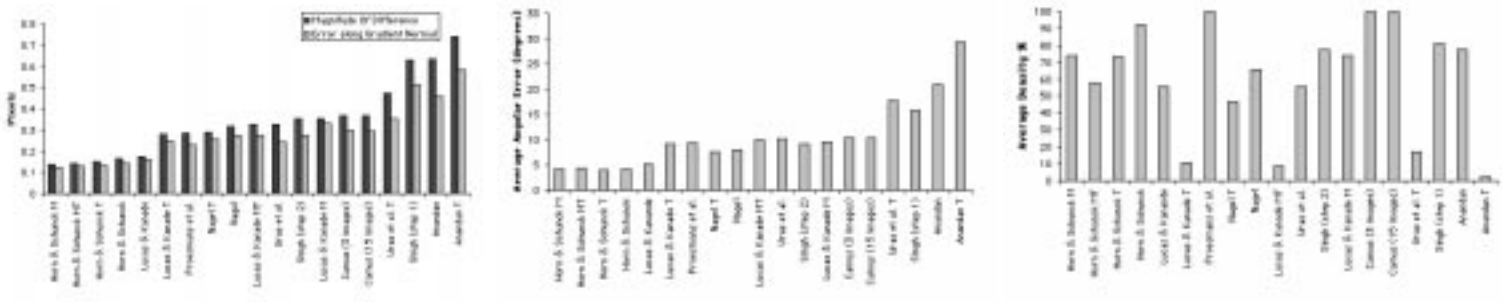
Overall, the algorithm based on Lucas and Kanade when used with a threshold had the best performance. It consistently produces accurate depth maps, has a low computational cost, and good noise tolerance. On the down side, it produces quite sparse depth maps. Barron et al. also concluded that this algorithm had the best performance.

Proesmans et al.'s algorithm is the only algorithm with consistent, accurate performance that produces a flow vector for each pixel. It provides reasonably accurate, consistent results, with good noise tolerance at moderate computational cost.

Horn and Schunck's algorithm also performs well provided the input images were smoothed, and accurate approximations to the derivative were used. When these modifications were used, the average error was reduced by 2.5 times, and the error caused by the addition of noise to the image sequence decreases by almost 5 times as illustrated in Figure 6(a). These modifications were at the expense of a modest 17% increase in computation time. Thresholding the resulting flow maps based on image gradient alone produced a negligible reduction in average error but significantly reduced the density of the flow maps. Many of the other optical flow algorithms also suffer from having no thresholding method.

The second order differential method of Uras et al. performed very well on the two complex scenes, but quite poorly on the rotating sphere scene. It has poor noise tolerance although this was significantly improved with thresholding.

In general, the derivative based techniques proved superior to region matching. In the rotating sphere scene, the ratio of Error Along Normal to Magnitude of Difference

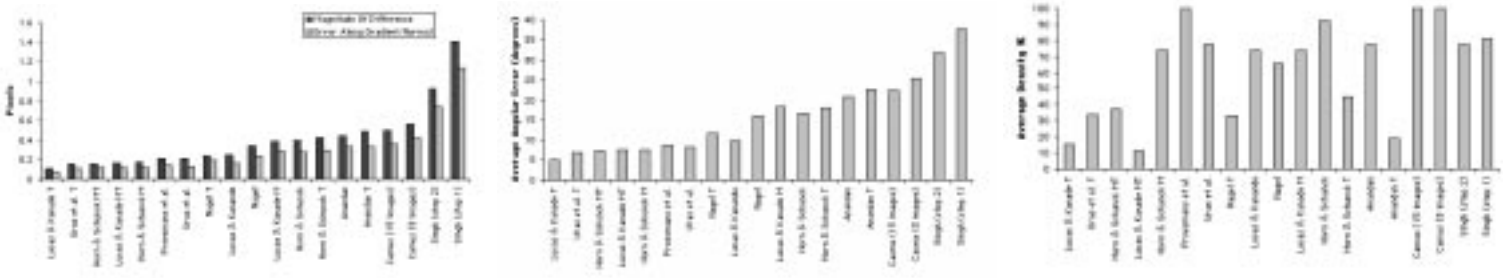


(a) Average Magnitude of Difference Error and Error Along Gradient Direction

(b) Average Absolute Angular Error

(c) Average Density

Figure 3: Results for the rotating sphere scene.

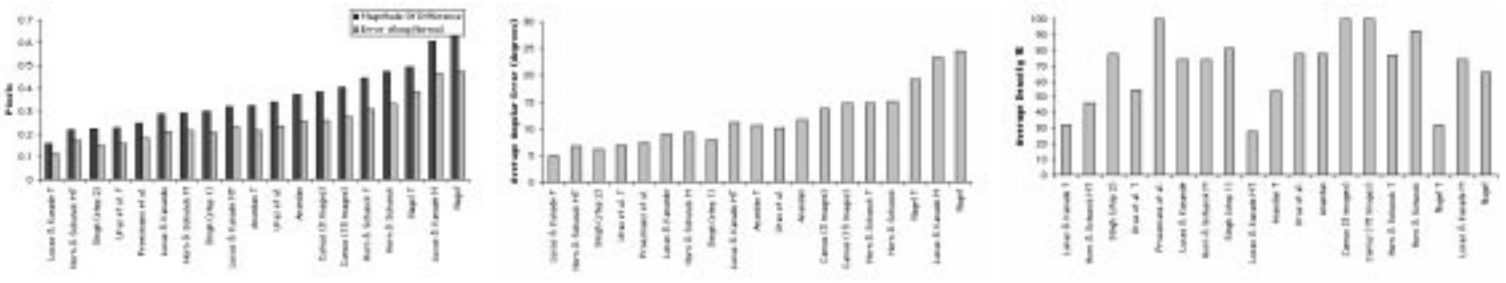


(a) Average Magnitude of Difference

(b) Average Absolute Angular Error

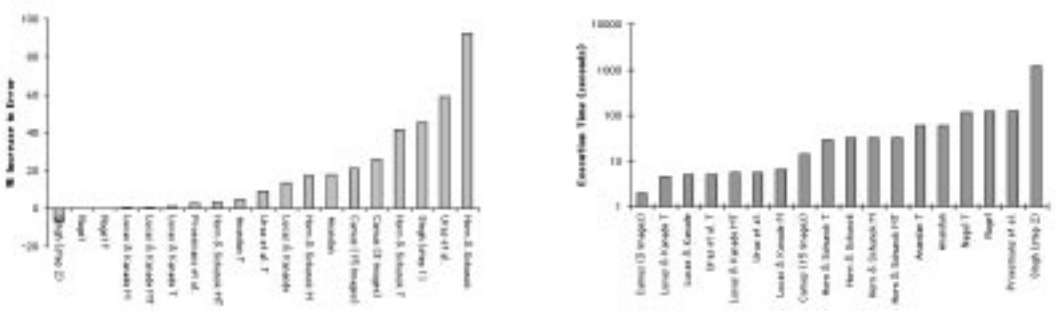
(c) Average Density

Figure 4: Results for the office scene.



(a) Average Magnitude of Difference (b) Average Absolute Angular Error (c) Average Density

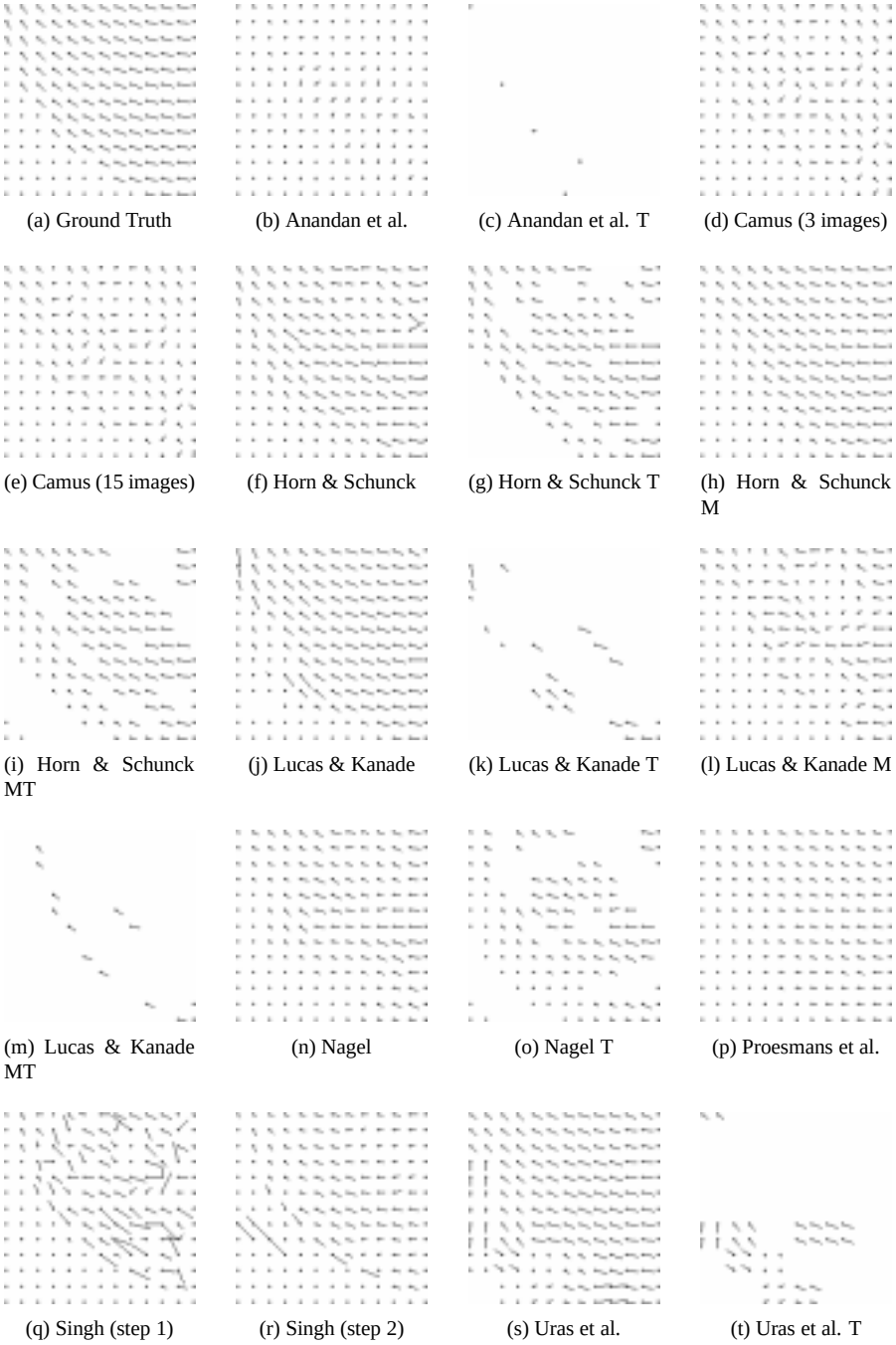
Figure 5: Results for the street scene.



(a) Percentage increase in the Magnitude of Difference error metric, as a result of adding noise to the image sequence. The noise had a Gaussian distribution with $\sigma = 3$.

(b) Time (in seconds) required to extract a single flow map from a 200x200x256 gray level image sequence. Note the log scale.

Figure 6: Noise and Timing Results.



(a) Ground Truth (b) Anandan et al. (c) Anandan et al. T (d) Camus (3 images)

(e) Camus (15 images) (f) Horn & Schunck (g) Horn & Schunck T (h) Horn & Schunck M

(i) Horn & Schunck MT (j) Lucas & Kanade (k) Lucas & Kanade T (l) Lucas & Kanade M

(m) Lucas & Kanade MT (n) Nagel (o) Nagel T (p) Proesmans et al.

(q) Singh (step 1) (r) Singh (step 2) (s) Uras et al. (t) Uras et al. T

Figure 7: Some example flow maps. These are taken from a small section of the rotating sphere scene.

was significantly higher in all derivative based algorithms (with the exception of Uras et al.'s algorithm). This indicates that derivative based algorithms are more susceptible to the aperture problem in this type of scene. However, in the more complex scenes this difference was not as apparent, indicating that perhaps the aperture problem is not as significant in practice as in theory.

8 Conclusion

We have developed a method for producing ground-truth optical flow from synthetic scenes of arbitrary complexity. This has allowed us to objectively evaluate the performance of eight optical flow algorithms on complex scenes. We have also used this technique to investigate the effect of image noise on algorithm performance. We concluded that Lucas and Kanade's algorithm had the best overall performance. This is consistent with the results of Barron et al. [1]. Proesmans et al.'s algorithm also performed well and has the additional advantage of producing one flow vector for each pixel. Horn and Schunck's algorithm also had excellent results, as long as suitable approximations to the derivative were used.

We have identified a number of issues with the error metrics used in this study which warrant further investigation. First, the size of the error depends on the magnitude of the flow. This is not consistent with our intuitive notion of what makes 'good' optical flow. Second, averaging the individual errors means that a large error in a small region is equivalent to small errors over a large region. Arguably, we would like the case with many small errors to produce a lower error metric. Thirdly, the current method of comparing a dense depth map with a sparse one is particularly unfair to the dense maps, since a simple average is taken. Creating an error metric which satisfies all of these constraints would inevitably be subjective to some degree. Application oriented error metrics may provide a more promising approach.

Ideally, an algorithm's performance should be evaluated by comparing its output with ground truth. Unfortunately, determining the ground truth motion field for complex, real scenes is an intractable problem. The use of complex, synthetic scenes brings us closer to this ideal, and is a step towards better evaluation of optical flow algorithms.

References

[1] J. Barron, D. Fleet, S. Beauchemin, and T. Burkitt. Performance of optical flow techniques. *Proc. of the IEEE on Computer Vision and Pattern Recognition*, pages 236–242, 1992.

[2] C. Butcher. Mirage. A ray tracing package, available from http://atlas.otago.ac.nz:800/students/butcher_cc/pages/mirage.html.

[3] T. Camus. Real-time quantized optical flow. *The Journal of Real-Time Imaging (special issue on Real-Time Motion Analysis)*, 3:71–86, 1997.

[4] B. K. P. Horn and B. G. Schunck. Determining optical flow. AI Memo 572, Massachusetts Institute of Technology, 1980.

[5] B. K. P. Horn and B.G. Schunck. Determining optical flow: a retrospective. *Artificial Intelligence*, 59:81–87, 1993.

[6] M. Proesmans, L. Van Gool, E. Pauwels, and A. Oosterlinck. Determination of optical flow and its discontinuities using non-linear diffusion. In *3rd European Conference on Computer Vision, ECCV'94*, volume 2, pages 295–304, 1994.