

Supplementary Material

It's Moving! A Probabilistic Model for Causal Motion Segmentation in Moving Camera Videos

Pia Bideau, Erik Learned-Miller

College of Information and Computer Sciences,
University of Massachusetts, Amherst
{pbideau,elm}@cs.umass.edu

Overview

Our supplementary material contains:

- a review of the BMS-26 data set and a detailed overview about excluded video sequences,
- additional details about the *Camouflaged Animals Data Set*,
- a description of our modified Bruss and Horn error [1], a fundamental part of our motion segmentation initialization.

1 Comparability of the motion segmentation data sets

A lot of different databases have been created to provide a common benchmark for motion segmentation. Often a clear definition of what people intend to segment when they provide a ground truth is missing. This results in many inconsistent segmentations, which makes it hard to compare against other motion segmentation methods. In our paper we give a clear definition of motion segmentation

- (I) Every pixel is given one of **two labels**: static background or moving objects.
- (II) If only part of an object is moving, the **entire object** should be segmented.
- (III) **All freely moving objects** should be segmented, but nothing else.
- (IV) Stationary objects are not segmented, even when they moved before or will move in future. We consider segmentation of previously moving objects to be tracking.

In the following subsection we give detailed information about all videos that do not correspond to our understanding of motion segmentation.











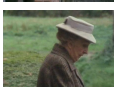



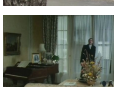

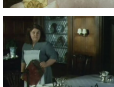



video	ground truth		I	II	III	IV	comment
cars9			5	✓	✓	✗	white van is segmented before it starts to move
people2			3	✓	✗	✓	third person in the bg is not segmented
tennis			4	✓	✗	✓	tennis ball is not segmented
marple2			5	✓	✗	✓	static fg building is segmented
marple6			3	✓	✗	✗	bike in the bg is not segmented
marple7			5	✓	✗	✗	static hedge is segmented
marple10			5	✓	✗	✓	static wall is segmented
marple11			2	✓	✓	✗	man is segmented before he starts to move
marple12			4	✓	✓	✗	pot is segmented, which was moved before
marple13			4	✓	✓	✗	chair is segmented, which was moved before

Table 1: **Ground truth of BMS-26** video sequences we excluded for evaluation due to mislabeled regions in ground truth.

1.1 Berkeley Motion Segmentation database (BMS-26)

To satisfy the first criterion of motion segmentation we converted the given ground truth into a binary segmentation, removing the provided motion labels. If all four criteria are satisfied, we used the video for comparison. The effect of the mislabeled ground truth varies a lot. The difference between a correct ground truth of *marple2* and the provided ground truth for example is enormous, whereas the difference of a correct ground truth of *tennis* would be almost not noticeable. Trying to be as objective as possible we excluded all videos where one of our four criteria of motion definition is violated, independently of the size

of the mislabeled region. The following table shows the sequences we excluded for evaluation.

1.2 Camouflaged Animals Data Set

Our new data set includes nine short video sequences extracted from YouTube videos and an accompanying ground truth. Table 1 shows the link to the original YouTube video, the exact time where our chosen video sequence starts within the YouTube video, and the number of frames the sequence contains. For our motion segmentation algorithm, we converted the video sequence to an image sequence in the png format using the VideoReader function of Matlab. Each sequence contains hand-labeled ground truth of moving objects in every fifth frame.

Video	Link	Start	Frames
chameleon	[2]	02:28.20	218
frog	[3]	00:05.38	31
glow-worm beetle	[4]	06:02.13	104
snail	[4]	06:07.02	84
scorpion1	[4]	02:09.00	105
scorpion2	[4]	02:25.04	61
scorpion3	[4]	02:27.48	76
scorpion4	[4]	00:06.11	80
stickinsect	[5]	00:05.68	80

2 Modified Bruss and Horn Error

As described in the main text, we introduced a modification to the error function of the Bruss and Horn algorithm that we call the modified Bruss and Horn (MBH) error. We first give some basic background on perspective projection, describe the Bruss and Horn error function and a particular issue that makes it problematic in the context of motion segmentation, and then describe our modification to the algorithm.

Basics of perspective projection. Given a particular set of translation parameters (U, V, W) (and assuming no camera rotation), the direction of optical flow of the background can be predicted for each point (x, y) in the image via the perspective projection equations. Let a point P have 3-D coordinates (X, Y, Z) . Then the image coordinates (x, y) for this point are:

$$x = \frac{Xf}{Z} \quad \text{and} \quad y = \frac{Yf}{Z}, \quad (1)$$

where f is the camera's focal length. A translational camera motion (U, V, W) yields in a pixel displacement \mathbf{v} in the image.

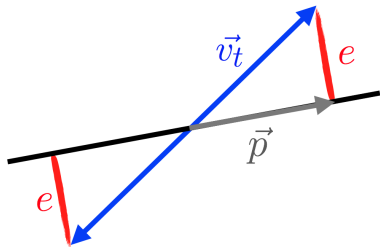
$$v_x = \frac{W \cdot x - U \cdot f}{Z} \quad \text{and} \quad v_y = \frac{W \cdot y - V \cdot f}{Z}. \quad (2)$$

The *direction* of the motion, given by

$$\arctan(W \cdot y - V \cdot f, W \cdot x - U \cdot f), \quad (3)$$

is then a function of the original image position (x, y) , the direction of motion (U, V, W) and the focal length f , and has no dependence on the depth Z of the point.

Fig. 1: **Bruss and Horn error.** Let \mathbf{p} be a vector in the direction of preferred motion with respect to a motion hypothesis (U, V, W) . The Bruss and Horn error assigned to a translational flow vector \mathbf{v}_t is then the distance of its projection onto \mathbf{p} . However, this *same* error would be assigned to a vector $-\mathbf{v}_t$ pointing in the opposite direction, which should have much lower compatibility with the motion hypothesis.



The Bruss and Horn Error Function. The point of the Bruss and Horn algorithm (translation-only case) is to find the motion direction parameters (U, V, W) that are as compatible as possible with the observed optical flow vectors. Let \mathbf{p} be a vector in the direction of the flow expected from a motion (U, V, W) (see Figure 1). Then the Bruss and Horn error for the observed flow vector \mathbf{v}_t is the distance of the projection of \mathbf{v}_t onto \mathbf{p} , shown by the red segment e on the right side of the figure.

The problem with this error function is that this distance is small not only for vectors which are close to the preferred direction, but also for vectors that are in a direction *opposite* the preferred direction. That is, *observed optical flow vectors that point in exactly the wrong direction with respect to a motion (U, V, W) get a small error* in the Bruss and Horn algorithm. In particular, the error assigned to a vector \mathbf{v}_t is the same as the error assigned to a vector $-\mathbf{v}_t$ in the opposite direction (See Figure 1).

Because the Bruss and Horn algorithm is intended for motion estimation in scenarios where there is only a single moving object (the background), such motions in the opposite direction to the preferred motion are not common, and thus, this “problem” we’ve identified has little impact. However, in the motion segmentation setting, where flows of objects may be in opposite directions, this can make the flow of a separately moving object (like a car), look as though it is compatible with the background. We address this problem by introducing a modified version of the error.

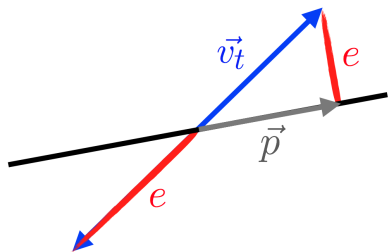
The modified Bruss and Horn error. As stated above, the Bruss and Horn error is the distance of the projection of an optical flow vector onto the vector \mathbf{p} representing the preferred direction of flow according to a translational

motion (U, V, W) . This can be written simply as

$$e_{BH}(\mathbf{v}_t, \mathbf{p}) = \|\mathbf{v}_t\| \cdot |\sin(\angle(\mathbf{v}_t, \mathbf{p}))|.$$

This error function has the appropriate behavior when the observed optical flow is within 90 degrees of the expected flow direction, i.e., when $\mathbf{v}_t \cdot \mathbf{p} \geq 0$. However, when the observed flow points *away* from the preferred direction, we assign an error equal to the magnitude of the entire vector, rather than its projection, since no component of this vector represents a “valid direction” with respect to (U, V, W) . This results in the modified Bruss and Horn error (see Figure 2):

Fig. 2: Modified Bruss and Horn error. When an observed translation vector \mathbf{v}_t is within 90 degrees of the preferred direction, its error is computed in the same manner as the traditional Bruss and Horn error (right side of figure). However, when the observed vector is more than 90 degrees from the preferred direction, its error is computed as its full magnitude, rather than the distance of projection (left side of figure). This new error function keeps objects moving in opposite directions from being confused with each other.



$$e_{MBH} = \begin{cases} \|\mathbf{v}_t\|, & \text{if } \mathbf{v}_t \cdot \mathbf{p} < 0 \\ \|\mathbf{v}_t\| \cdot |\sin(\angle(\mathbf{v}_t, \mathbf{p}))|, & \text{otherwise.} \end{cases}$$

This error has the desired behavior of penalizing flows in the opposite direction to the expected flow.

References

1. Bruss, A.R., Horn, B.K.: Passive navigation. *Computer Vision, Graphics, and Image Processing* **21**(1) (1983) 3–20
2. : Most amazing camouflage (part 1 of 3). <http://www.youtube.com/watch?v=Wc5wMX61FZ8> (2014) [Online; accessed 1-March-2015].
3. : Most amazing camouflage (part 2 of 3). <https://www.youtube.com/watch?v=yoG1P4new04> (2014) [Online; accessed 1-March-2015].
4. : Most amazing camouflage (part 3 of 3). https://www.youtube.com/watch?v=GnEkBJnS_FU (2014) [Online; accessed 1-March-2015].
5. : Phasmid - incredible insect camouflage. <https://www.youtube.com/watch?v=adufPBDNCKo> (2014) [Online; accessed 1-March-2015].