DEPTH DETECTION OF TARGETS IN A MONOCULAR IMAGE SEQUENCE

Hong Guo and Yi Lu
Department of Electrical and Computer Engineering
The University of Michigan-Dearborn
Dearborn, MI 48128-1491

Susanta Sarka
U. S. Army Tank-Automotive Command
ATTN: AMSTA-O
Warren, MI 48397-5000

Abstract
This paper describes an approach for recovering structure of a moving target from a monocular image sequence. Within this paper, we assume the camera is stationary. We first use a motion detection algorithm to detect moving targets based on four heuristics derived from the properties of moving vehicles, maximum velocity, small velocity changes, coherent, and continuous motion. The second algorithm then estimates the distance of the moving targets using an over-constrained approach. We will show a proof-of-concept example from synthetic data. We have applied the approach to monocular image sequences captured by a moving camera to recover the 3D structure of stationary targets such as trees, telephone pole, etc. The experimental results on a monocular image sequence captured in an outdoor environment are also presented.

1. Introduction
The detection of targets and the estimation of the distance of the target in outdoor scenes has a number of important applications such as video surveillance, battlefield surveillance, obstacle detection, etc. Target detection is generally accomplished by discrimination between target and background using locally computed images features such as texture, edges, etc. Most effort in R&D community has been in the development of computational algorithms using signal analysis methods [1,2]. As a result, these algorithms only work in narrowly defined environment, and usually do not give good predictive results when they are applied to targets highly resolved in complex background scenes[1, 3, 4]. Recovering 3D structure of objects in a dynamic scene usually consists of two computational steps matching corresponding points/regions and computing depth from the motion vectors [5]. This is still an active research area and many research issues remain open for investigation [5,6].

This paper first describes an algorithm for detecting moving targets from a monocular image sequence captured by a stationary camera. The algorithm detects the moving targets based on four heuristics derived from the properties of moving vehicles, maximum velocity, small velocity changes, coherent, and continuous motion. The moving target detection algorithm outputs the location of every moving target in two consecutive images in the monocular sequence. Then a depth estimation algorithm takes the output from the moving target detection algorithm and estimates the distance of the moving targets.
using an over-constrained approach. We have applied the algorithm to monocular image sequences captured by a moving camera to recover the 3D structure of stationary targets such as trees, people, telephone pole, etc. The experimental results are also presented.

2. Detection of moving targets

In order to compute the depth of an object in an image sequence, it is necessary to establish, either explicitly or implicitly, certain type of correspondences between images in the sequence. There are two issues involved in the correspondence problem, the matching element and matching methods. The matching elements, or tokens, can be color, intensity or object features such as edges and corners. In this algorithm we use image intensity as matching elements. There are three major approaches to estimating correspondence: Fourier method, matching, and method of difference measurement[5]. The approach we use is matching image segments combined with heuristics.

In an outdoor environment, moving target tracking is difficult because there are many objects that can move either randomly or with certain directions. For example, clouds in the sky, weeds or grass moving by the wind, telephone lines or electrical lines, etc. Pure data driven approaches will not be sufficient to solve this problem. We need to develop heuristic knowledge that can identify moving vehicles from other moving objects. The following four heuristics are used in the algorithm:

- **Maximum velocity.** In general, we can estimate the maximum velocity of a moving target in the scene, which limits the moving distance of each pixel. This maximum velocity is used to define a search window. Figure 1 illustrates this concept, the bigger square represents the search window, and the smaller one represents the matching window to be used in finding the matching target in a successive frame.

- **Small velocity changes.** For rigid body targets such as vehicles, they are constrained to travel at small angle changes. Therefore it is reasonable to assume that the velocity change of a vehicle is always within the ranges of \(-90 < \theta < 90\), or \(90 < \theta < 270\) if it is moving in horizontal direction, \(0 < \theta < 180\), or \(180 < \theta < 360\) if it is moving towards or away from the camera.

- **Coherent motion.** Rigid body targets are spatially coherent objects, therefore, a target appear in successive images as regions of points sharing a “common motion.”

- **Continuous motion.** Moving targets have similar motion vectors across several image frames. Therefore we can track vehicles based on the motion vectors obtained through the previous frames.

The algorithm for computing moving targets has the following computational steps:

**Step 1 Motion estimation**

Motion estimation has been well studied in the computer vision field. One fundamental approach is to match one portion of an image \(I_t\) at time \(t\), with each portion of the successive image \(I_{t+1}\) at \(t+1\), using image features such as edges, contour or intensity. If image features are used as matching elements, the outcome of the matching very much depends on the accuracy of the image feature extraction algorithms, for examples, edge detection, contour extraction, etc. This algorithm uses
image intensity as matching feature. The algorithm uses the matching window to compute the similarity between the two portions in \( I_t \) and \( I_{t+1} \). The search window is used to limit the search region for the possible location of a matching pixel in the image frame \( I_{t+1} \). According to the maximum velocity heuristic, a pixel \((x', y')\) in \( I_t(x, y) \) has a maximum translation in image \( I_{t+1}(x, y) \). Specifically, the new location for \((x', y')\) is \((x' + u(x', y'), y' + v(x', y'))\), where \(|u(x', y')| \leq \text{search_window_size} \) and \(|v(x', y')| \leq \text{search_window_size} \). From the heuristic knowledge of small velocity changes, we know that the motion direction change of a vehicle cannot be more than 90 degrees. Within the effective search window, the new location of \((x', y')\) is found by using the following maximum likelihood function:

\[
\Phi(u, v) = \sum_{i} \sum_{j} \left( I_t(x_i + p, y_j + q) - I_{t+1}(x_i + p, y_j + q) \right)^2
\]

where \(p\) and \(q\) should be within the matching window. The new location of \((x_i, y_j)\) within image \( I_{t+1}(x, y) \) is at \((x'_i, y'_j)\) where \(x' = x + u\), \(y' = y + v\), \(-p \leq u \leq p\) and \(-q \leq u \leq q\) and \(\Phi(u, v) < \Phi(ui, vi)\), where \((ui, vi)\) is within the search window. The motion vector for \((x'_i, y'_j)\) is \((dx, dy)\), where \(dx = x'_i - x_i\), \(dy = y'_j - y_j\). The concept of effective window not only reduces the computational time but also provides more accurate search results.

**Step 2. Moving vehicle detection**

From the motion estimation at Step 1, we obtain motion vectors at every image pixel. Pixels have nonzero motion vectors are candidates for moving objects. As we indicated earlier that in an outdoor environment, many objects other than vehicles can be moving. However, since vehicle motion is a rigid body motion, which means every point on the vehicle must have the same motion vector, many other moving objects do not have this property, for example, grass, or electrical wires. Therefore we have applied the coherent motion heuristics to eliminate the non-vehicle objects. We used following two rules to eliminate non-vehicle and moving objects.

**Rule 1: Applying a proximity filter**

We divided the image space evenly into 16 orientation zones. If two motion vectors are in the same orientation zone, they are considered as in the same orientation. The proximity filter rule states that if the base pixel \((x, y)\) in \( I_{t+1} \) has \(\alpha\) number of neighbors whose motion vectors \((dx, dy)\) are within the same orientation zone as the base pixel’s motion vector, then keep the point, otherwise eliminate the point. The parameter \(\alpha\) can be set by the user based on either a priori knowledge on the direction of the moving vehicle or on the moving vehicle size. If we know the moving direction of the vehicle, we can determine the orientation zone of vehicle points. The most conservative way is to set \(\alpha = 1\), which requires at least one of the eight neighbors of the base pixel to have the similar motion direction. If the base point has \(\alpha\) neighbors that have the motion vectors in its orientation zone, then the base point is considered as the possible vehicle point and therefore is kept, otherwise, this base point is discarded.

**Rule 2: Eliminating negligible motion pixels**

This rule intends to eliminate the random motion vectors. Two techniques are used. The first technique simply removes the motion vectors whose magnitudes are very small. This is implemented by setting a minimum motion threshold, MIN_threshold. A point is considered as a moving vehicle pixel if the magnitude of its motion vector is greater than a preset threshold, MIN_threshold, which can be derived from knowledge of vehicle speed. The second technique uses three directional filters, a horizontal pass filter, a vertical pass filter, and a horizontal-or-vertical pass filter. The directional filters attempt to remove all motion vectors that are not primarily horizontal or primarily vertical. The horizontal pass filter requires the displacement at the \(x\) direction to be larger than the displacement at the \(y\) direction, which means that the object is moving more along the \(x\) direction. Similarly
the vertical filter requires the displacement at the y direction to be larger than the displacement at the x direction, which means the object is moving more along the y direction. The horizontal filter requires

\[
|\text{displacement}[i,j].x| - |\text{displacement}[i,j].y| \leq \text{XY}_\text{LIM} \text{ AND } (|\text{displacement}[i,j].x| - |\text{displacement}[i,j].y| > 0),
\]

the vertical filter requires

\[
(\text{displacement}[i,j].y - |\text{displacement}[i,j].x|) \leq \text{XY}_\text{LIM} \text{ AND } (\text{displacement}[i,j].y - |\text{displacement}[i,j].x|) > 0),
\]

the horizontal-or-vertical filter requires

\[
| |\text{displacement}[i,j].x| - |\text{displacement}[i,j].y| | \leq \text{XY}_\text{LIM},
\]

where \(\text{displacement}[i,j].x\) and \(\text{displacement}[i,j].y\) represent the displacement of the current pixel \((i,j)\) at direction x and y respectively, and \(\text{XY}_\text{LIM}\) is a threshold which can be estimated based on the knowledge of the vehicle velocity. This rule is based on the assumption that the orientation of a moving vehicle is primarily either horizontal or vertical if the image sampling rate is high.

**Step 3 Tracking moving vehicle in an image sequence**

After the first two frames, \(I_0\) and \(I_1\), we obtain a set of moving regions using the methods described in the above two steps. For the subsequent image frames, the algorithm tracks these regions and compares the last computed motion vectors with the current motion vectors to further eliminate the non-vehicle objects. Assume the current image frame is \(I_t\), \(t > 1\). As we stated before that we partition the image space into 16 different orientation zones. A pixel \((x, y)\) in \(I_t\), which has current motion vector \((dx, dy)\), belongs to a moving vehicle if the neighboring pixels covered by the matching window have more than \(\text{num} \_ \text{th}\) motion vectors in the same orientation zone as \((dx, dy)\). If this condition is true, then all the pixels within the matching window are marked as belonging to the same moving vehicle. This technique is effective in eliminating objects with random motion such as grass.

**Step 4. Generating regions of moving vehicles**

After obtaining individual motion vectors and eliminate the non-vehicle pixels, we group the pixels that have nonzero motion vectors and are spatially connected in the image to form regions that are possible moving vehicles. A threshold can be set to filter out small regions that are impossible to be considered as the target vehicle. The grouping is implemented by generating a binary image in which a pixel is set to 0 if its corresponding motion vector is 0, otherwise is 1. A connected component algorithm can be applied to the binary image to obtain the regions of moving vehicles.

**Step 5. Computing the plausible motion vector for each moving vehicle.**

Due to noise and background interference, the regions representing moving vehicles may contain the non-zero motion vectors in different orientations. What is the vehicle motion orientation? Since we assume coherent and rigid body motion, motion vectors of the same target should be identical. However due to various types of noise, the non-zero motion vectors inside the same bounding box may vary. To solve this problem, we sort the motion vectors inside each bounding box into different bins. The motion vector inside the largest bin is considered as the motion vectors of the target.

Figure 2 shows the results generated by algorithm at every step of tracking in an image sequence. Figure 2(a) shows the results achieved from the first two images in the sequence, (b) shows the results achieved from the third image, and (c) shows the results achieved from the fourth image in the sequence. The results generated by this algorithm include the locations and the most plausible motion vector for each vehicle. The output is then send to the following algorithm.
for computing the distance of each moving target.

![Image](image1.png)

![Image](image2.png)

![Image](image3.png)

**Figure 2. An example of target detection in a monocular image sequence.**

3. Detection of depth of moving targets

This algorithm inputs target locations in the image sequence and computes the relative depth of the targets. Let us assume we have N moving target points and each point can be written as \( (u_n, v_n, u_n, v_n) \) \( (n=1, \ldots, N) \), where \( (u_n, v_n) \) is the target pixel inside the image and \( (u_n, v_n) \) is the 2D motion vector of the target pixel \( (u_n, v_n) \). Assume \( (x_n, y_n, z_n) \) is the 3D location of \( (u_n, v_n) \) and is unknown. The task for the algorithm is to compute the 3D motion vector of the target \( x, y, z \). According to the well-known formula developed in the computer vision research community, the over-constrained system subject can be solved by

\[
A \begin{bmatrix} x \\ y \\ z \end{bmatrix} = 0
\]

with the constraint \( x^2 + y^2 + z^2 = k^2 \), where

\[
A = \begin{bmatrix} v_1 f & -u_1 f & u_1 v_1 - u_1 v_1 \\ \vdots & \ddots & \vdots \\ v_N f & -u_N f & u_N v_N - u_N v_N \end{bmatrix}
\]

and \( f \) is a constant denoting the focal length of the camera. The solution is given by the eigenvector of \( A'A \) that has the smallest eigen value. The eigen vectors and eigen values of \( A'A \) are obtained by using any well-known numerical algorithm[7]. Let the eigenvector corresponding to the smallest eigen value be \( (dx, dy, dz) \), which is common motion vector of the moving target. Then the depth can be obtained from the following equation.

\[
z_n = \frac{u_n (f \dot{x} - u_n \dot{z}) + v_n (f \dot{y} - v_n \dot{z})}{u_n^2 + v_n^2}.
\]

4. Experiments and conclusion

We have tested the depth computation algorithm on a set of synthetic data. The data used in the experiment was produced by the calculation of 3-D projection. Four points were used, \((35,25), (36,25), (37,25)\) and \((36,26)\). To produce the velocity of points on an image plane, a velocity vector of points is assumed as \( (dx, dy, dz) = (10,20,1) \) and focal length \( f=1 \).

Without losing generality, the \( z \) coordinates of every point were set as 10. If time \( dt \) equals to 1, the new positions of projection of point \( (x,y,z) \) is calculated as below:

\[
\begin{bmatrix} u_n \\ v_n \end{bmatrix} = \frac{f}{Z + Z^* dt} \begin{bmatrix} x + x^* dt \\ y + y^* dt \end{bmatrix} = \frac{f}{Z + Z} \begin{bmatrix} x + x \\ y + y \end{bmatrix}
\]

From this formula, we get the 4 point optical field data:
Use the formula above, we can find that eigen values are $4.46 \times 10^{-7}$, $0.010222$ and $7282.47$. The corresponding eigen vectors are $[0.4387, 0.8975, 0.045]^T$, $[0.8985, -0.437539, -0.034]^T$ and $[0.01106, -0.055, 0.998]^T$ respectively. Go back to the constraint stated above we have $k = \sqrt{x^2 + y^2 + z^2} = 22.38$. The final solution should be $k*[0.4387, 0.8975, 0.045]^T = [9.82, 20.09, 1.01]^T$. The Z coordinates are calculated using the formula above to get $z_n=11.16$, $n=1,2,3,4$, which are very close to the true $z=10$.

We have applied the algorithms to the problem of finding target distance from a moving camera mounted on a vehicle. Since the camera that captures the scene is mounted on a moving vehicle and the targets are stationary, all the heuristics derived from physics of vehicles can be used in finding the correspondence of target regions. The image sequence shown here was captured at 10 frames/sec as the vehicle moves at 15mph. The view angle of the camera is 14 degrees, focal length is 0.025m and the image size is 700x240. Figure 3 shows the first two images in the sequence. In this image sequence we have two targets of interest, a person and a tree, each is outlined by a rectangle. The true distances of the person and the tree to the camera are 166 and 350 feet, respectively. For each target we use three points to compute the distance. The correspondence of the points for the person is shown in Table 1 and for the tree is shown in Table 2.

### Table 1. The corresponding points of target “person” in the two image frames.

<table>
<thead>
<tr>
<th>(x,y) of Image 1</th>
<th>(x,y) of Image 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(172,141)</td>
<td>(153,156)</td>
</tr>
<tr>
<td>(165,152)</td>
<td>(145,168)</td>
</tr>
<tr>
<td>(163,173)</td>
<td>(142,192)</td>
</tr>
</tbody>
</table>

Considering the vehicle speed (15mph=21.87feet/s) and the actual focal length of the camera, we have, for the “person,” velocity vector : $k*[dx, dy, dz]= [-0.000612, 0.005271, -6.669998]^T$. For the corresponding pair (141.0000,172.0000) and (156.0000,153.0000) we obtain $z= 67.4774m$; for (152.0000,165.0000), (168.0000,145.0000) we obtain $z= 66.9904m$; and for (173.0000,163.0000), (192.0000,142.0000) $z= 64.16m$.

The average of the $z$ value of the person is $66.21m = 217feet$.

Similarly the absolute distance, i.e. the Z coordinates of the pixels on the “Tree” are 92.8161m, 101.2869m and 90.5538m. The average Z value is 94.88m = 311feet. The error margin are 11% and 30% on tree and person respectively. The errors are largely due to the nature of the monocular image sequence, rounding and conversion errors.

Currently we are conducting an in-depth study on the problem of recovering depth from a monocular image sequence. Our study shows that computing depth from monocular image sequences can have a rather large error margin in absolute depth. However relative depth of targets computed from monocular image sequences are quite accurate.

### Table 2. The corresponding points of target “tree” in the two image frames.

<table>
<thead>
<tr>
<th>(x,y) of Image 1</th>
<th>(x,y) of Image 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>(474,43)</td>
<td>(489,50)</td>
</tr>
<tr>
<td>(415,70)</td>
<td>(425,78)</td>
</tr>
<tr>
<td>(469,163)</td>
<td>(484,179)</td>
</tr>
</tbody>
</table>

5. Reference:

(a) Frame 1 in the image sequence.

(b) Frame 2 in the image sequence.

Figure 3. A monocular image sequence captured by a moving camera mounted on a vehicle.