FAST OBSTACLE DETECTION USING TARGETED OPTICAL FLOW

Nasim Sepehri Boroujeni¹, S. Ali Etemad¹, Anthony Whitehead²

¹ Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
² School of Information Technology, Carleton University, Ottawa, Canada

ABSTRACT

This paper presents a new method for obstacle detection using optical flow. The method employs a highly efficient and accurate adaptive motion detection algorithm for determining the regions in the image which are more likely to contain obstacles. These regions then have optical flow performed on them. We call this method targeted optical flow. Targeted optical flow performs significantly faster compared to regular optical flow. We employ two types of optical flow to demonstrate the performance and speed increase of the proposed system. Finally, k-means clustering is employed for obstacle reconstruction. The system is designed for color videos for better performance. Several benchmark and recorded sequences have been used for testing the system.

Index Terms— Obstacle detection, optical flow, motion estimation, clustering.

1. INTRODUCTION

Obstacle detection (OD), in general, is the process of distinguishing and locating blockages in the path of a moving vehicle or robot. A robust OD system would eventually take over the role of the eyes and brain of an operator for controlling the vehicle. As the applications of automated and unmanned systems have grown in recent years, the need for precise and robust guidance and navigation systems has increased as well. Many different applications can be mentioned for OD systems such as those in [1,2]. Automated cars, unmanned aerial vehicles (UAV), automated industrial systems, and visually impaired aid systems among many others are different examples of such systems.

There are various factors in need of consideration when performing OD using optical (passive) sensors:

i. vague, inconsistent, and complex backgrounds [2]
ii. imperfect input images due to low quality cameras and high frequency vibrations of the camera [3]
iii. the necessity yet difficulty of real time processing [2]
iv. presence of obstacles in varying shapes, colors, and sizes [4]

Due to the complexities and constraints caused by the mentioned issues, different strategies need to be carried out based on the application at hand. Using stereo vision provides the application with the benefit of depth information. In many cases however, especially for the more simple and practical robots and vehicles, stereo vision is not utilized. Therefore, this research employs a single color video camera as input.

In this paper, backgrounds of scenes and objects are first segmented through an adaptive motion detection method. Optical flow (OF) is then computed for further analysis of the scene in order to detect the obstacles. The operation is carried out on each of the color channels individually and the results are blended together. These steps form what we refer to as targeted OF. Two types of OF, Lucas–Kanade (LK) and Horn–Schunck (HS) form the basis for comparison. Upon computing the flow, k-means clustering is used for reconstructing the general shape of the obstacles. The entire system is tested on scenes from two benchmark sequences along with newly recorded videos. The speed increase of the system is analyzed in detail.

2. TARGETED OPTICAL FLOW

OF contains information about the relative motion between a camera and objects. This information can be used for detecting obstacles in a scene. There are several methods for estimating OF. These methods are classified into three main groups: correlation-based [3], differential-based [5], and block-based [6]. Differential-based techniques perform better when dealing with textured backgrounds. In this paper the LK and HS methods, which are differential-based, are employed.

The LK [7] method calculates the velocity of each pixel in an image frame based on the following set of equations.

\[
\frac{\partial I}{\partial x} + \frac{\partial I}{\partial y} + \frac{\partial I}{\partial t} = 0
\]  \hspace{1cm} (1)

\[
\frac{\partial I}{\partial x} u + \frac{\partial I}{\partial y} v + \frac{\partial I}{\partial t} = 0
\]  \hspace{1cm} (2)

\[
\begin{bmatrix}
I_{x1} & I_{y1} \\
I_{x2} & I_{y2} \\
\vdots & \vdots \\
I_{xN} & I_{yN}
\end{bmatrix}
\begin{bmatrix}
u \\
v
\end{bmatrix}
= \begin{bmatrix}
-I_{t1} \\
-I_{t2} \\
\vdots \\
-I_{tN}
\end{bmatrix}
\]  \hspace{1cm} (3)

In Eq. 1, \(I(x,y,t)\) represents pixel intensity at location \((x,y)\) at time \(t\). In Eq. 2 \(u\) and \(v\) are horizontal and vertical velocities (OF vectors) of pixel \((x,y)\) at \(t\) and in Eq. 3, \(I_{x}, I_{y}\)
and \( I \) illustrate partial derivatives of the image with respect to \( x, y, \) and \( t \).

In the HS [8] method the equality constraint (Eq. 1) in OF estimation is replaced with a minimization problem. In other words, the goal is to find the OF vectors \((u, v)\) such that the energy function which is described by Eq. 4 is minimized.

\[
\iint \left[ I_x (u, v) + I_y + I_t \right]^2 + \alpha^2 \left[ \|I_x\|^2 + \|I_y\|^2 \right] dxdy \tag{4}
\]

In Eq. 4, \( \alpha^2 \) is a weighting factor and \( \|I_x\| = \sqrt{u_x^2 + u_y^2} \) and \( \|I_y\| = \sqrt{v_x^2 + v_y^2} \) where \( u_x, u_y, v_x, \) and \( v_y \) are spatial derivatives of velocities. Iterative methods are used to minimize the above equation and find the velocity vector of each pixel in the images.

\[
\begin{align*}
I_x &= \frac{\alpha I_x + \beta I_y}{\alpha^2 + I_x^2 + I_y^2} \\
I_y &= \frac{\beta I_x - \alpha I_y}{\alpha^2 + I_x^2 + I_y^2}
\end{align*}
\tag{5}
\]

where

\[ \beta = \frac{I_x I_y^* + I_y I_x^* + I_t}{\alpha^2 + I_x^2 + I_y^2} \tag{6} \]

In the above equations, \( n \) is the iteration number, and \( \overline{I} \) and \( \overline{V} \) are the average velocities of pixels that are in the neighborhood of pixel \( i \).

Computing OF for all of the pixels in an image within a sequence is very time consuming. We were able to overcome this problem by computing the flow vectors for pixels that are strong candidates for motion as opposed to all pixels in the image. This is done using a precise and adaptive motion analysis technique. The method is a hybrid technique which maintains the benefits of both temporal differentiation and background subtraction. In this method, a pixel in the image at \( t = n \) is subtracted from the corresponding pixel of the images at \( t = n - 1 \) and \( t = n - 2 \). If both results surpass an adaptive threshold, that particular pixel is considered as a moving one; otherwise it is labeled as stationary. This is presented by Eqs. 7 and 8.

\[
\begin{align*}
I_x (i, j) - I_x (i, j) &> T_x (i, j) \\
I_y (i, j) - I_y (i, j) &> T_y (i, j)
\end{align*}
\tag{7, 8}
\]

In this technique, which was employed by Collins et al. [9], the background \( B_n \) and threshold \( T_n \) are updated in each stage based on the Eqs. 9 and 10 where \( S \) is the set of stationary pixels, \( D \) is the set of moving pixels, and \( \lambda \) is a time constant related to the speed of moving vehicle.

While the presented method maintains the benefits of temporal and background differentiation, it does not show any of their major setbacks such as high sensitivity to motion and existences of residues and ghosts in the resulting output.

\[
\begin{align*}
B_{n+1}(i, j) &= \frac{\lambda B_n(i, j) + (1 - \lambda) I_x (i, j)}{\lambda B_n(i, j) + (1 - \lambda) I_y (i, j)}, & (i, j) \in S \\
T_{n+1}(i, j) &= \frac{\lambda T_n(i, j) + (1 - \lambda) I_x (i, j) - T_n(i, j)}{\lambda T_n(i, j) + (1 - \lambda) I_y (i, j) - T_n(i, j)}, & (i, j) \in D
\end{align*}
\tag{9}
\]

Once the background pixels are segmented out, LK and HS flow vectors are computed for the set \( D \) of object pixels. This, as illustrated in section 4, reduces the run-time significantly. We also compare the results obtained from these two OF methods based on run time and performance.

In order to increase the precision of the algorithm, all the steps of the presented method are applied on the three R, G, and B color channels independently. The OF vectors of the three channels are analyzed separately and blended based on Eq. 11.

\[
V = \sqrt{\sum_{k=r,g,b} \left| v_k \right|^2 / |B|} + \sqrt{\sum_{k=r,g,b} \left| v_k \right|^2 / |S|}
\tag{11}
\]

The results of the system described above is then used to detect and reconstruct the obstacles in the scene.

3. OBSTACLE RECONSTRUCTION

Different parts of objects maintain varying intensity profiles depending on the texture, perspective, and lighting of the scene. Therefore it is quite natural for some sections of the object to be falsely labeled as parts of the background while, in fact, they are parts of obstacles. This issue is further intensified when targeted OF is employed and as a result, some useful vectors have not been included.

In order to compensate for these issues, we have employed \( k \)-means clustering to reconstruct the obstacles once a portion of them have been detected by the system.

\( k \)-means clustering [10] classifies \( n \) points into a predefined number of groups (\( k \)). Each object is assigned to a group according to its proximity with the center point of that group. When the number of clusters is determined, \( k \) clusters are formed randomly with their centroids defined. Next, each point is assigned to the nearest center point. In the third step, the center points are updated and again classification is carried out. This procedure continues until meeting a specific convergence criterion. Eqs. 12 and 13 present the formal definition of \( k \)-means clustering used in this research.

\[
C_i(t) = \left\{ x_j : \left| x_j - \mu_i(t) \right| \leq \left| x_j - \mu_j(t) \right| \right\}
\tag{12}
\]

\[
\text{for all } i = 1, \ldots, k \text{ and } j = 1, \ldots, n
\]

\[
\mu_i(t+1) = \frac{1}{c_i(t)} \sum_{x_j \in C_i(t)} x_j
\tag{13}
\]

In Eq. 12 \( C_i(t) \) represents the \( i \)th cluster at iteration \( t \) while \( x_j \) is the sample being placed in one of the clusters. \( \mu_i \) denotes the centroid of the cluster, calculated by Eq. 13. The number of clusters in the scene is selected based on the complexity of the environment. Any cluster containing an average flow vector magnitude per pixel above a threshold \( t \) is labeled an obstacle.
4. EXPERIMENTAL RESULTS

The proposed algorithm was developed and implemented in MATLAB and on a system with 4.0 GB of RAM and a CPU with a speed of 2.40 GHz. Several moving camera sequences were used for testing the proposed system. The moving vehicle was constructed by attaching a single lens webcam to a wheeled cart. The webcam had a frame grab rate of 20 per second and spatial resolution of 240 × 320 pixels. Several objects were placed in the path of the cart as obstacles. Furthermore, benchmark test sequences of crowded and dynamic scenes, Bahnhof and Jelmoli [11], were used to test the performance.

Figure 1 illustrates snapshots of the original test sequences along with the outputs from the background detection technique in three color channels. The colored pixels represent foreground sections of the scene. Furthermore, it is illustrated that while the different channels show similarities, they are different in some sections.

The outputs from this step are employed to compute targeted OF (again separately on each channel). Figure 2 presents the original OF vectors computed using HS and LK techniques. Moreover, the figure illustrates targeted OF using HS and LK. It is evident that while the scenes are less populated with flow vectors, the critical moving pixels are detected using OF. Moreover, the size of the flow vectors indicates the amount of motion for each pixel. Finally, k-means is utilized and the objects are reconstructed. The objects in the third and sixth column of Figure 2 are illustrated in red. This OD output, especially if combined and interpreted with some type of depth information, can be extremely accurate and informative. Figure 3 illustrates the results of OD using plain OF. Comparing the results with targeted OF, it is evident that while the main obstacles are detected in both, more background pixels are falsely labeled as obstacles in plain OF.

Making use of targeted OF significantly increases the speed of the system compared to standard plain OF. The runtime is decreased to approximately 1 second per frame. Table 1 illustrates the speed increase of the system. The values are calculated for 100 frames. This increase measured on the 4 different sequences varies between 79.3% to 87.5% which is significant when used for real time vehicles and robots. Moreover, employing targeted OF instead of plain OF benefits the system by excluding background pixels from objects being labeled as obstacles and thus making the system more robust.

<table>
<thead>
<tr>
<th></th>
<th>Bahnhof</th>
<th>Jelmoli</th>
<th>Hallway 1</th>
<th>Hallway 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horn-Schunck</td>
<td>81.1%</td>
<td>79.3%</td>
<td>87.2%</td>
<td>86.7%</td>
</tr>
<tr>
<td>Lucas-Kanade</td>
<td>82.3%</td>
<td>81.5%</td>
<td>87.5%</td>
<td>87.3%</td>
</tr>
</tbody>
</table>

5. CONCLUSION

This paper presents a highly efficient and accurate method for detecting obstacles in both crowded and simple scenes. The system was based on acquiring single camera color images and with no depth information present. An adaptive technique for background detection was initially employed and the foreground sections of the scene were further analyzed using two types of optical flow, Horn-Schunck and Lucas-Kanade. These steps were carried out on each color channel and the outcomes were blended together.
Two benchmark sequences, Bahnhof and Jelmoli, were used for testing the system. Moreover, several sequences were obtained from a hallway with obstacles placed in the path of the moving cart mounted with a camera. The results show very accurate and robust extraction of obstacles while a speed increase of above 80% is accomplished over using standard optical flow methods alone. The results indicate that targeted optical flow excludes background outliers thus making it more suitable for obstacle detection.

6. REFERENCES


