Encroachment Detection with Monocular Vision for Small, Low-cost, Compute-constrained Platforms

Jeffrey Kane Johnson[1]

***Abstract*— A computationally efficient monocular encroachment detection technique is presented, and a proof of concept is implemented on a low-cost mobile robot platform. This is an extended version of an abstract submitted to IROS 2017.**

I. INTRODUCTION

One of the primary functional requirements of mobile robots is that they be capable of detecting encroaching objects in order to avoid collision. Many methods and sensors for this task exist, but when there are severe computational or cost constraints, existing methods and sensors may not be suitable. To address such cases, a novel monocular *Encroachment Detection* technique is developed that has limited computational complexity and requires only a single monocular camera as sensor input.

II. ENCROACHMENT DETECTION

The Encroachment Detection problem, defined below, is related to the general object detection and tracking problems.

Problem 1. Let *encroachment* refer to the reduction in minimum proximity between two or more objects in a workspace $\mathscr{W}$ beyond desired limits as measured by a metric $\mu(\cdot,\cdot)$. Assume A receives some observation input O_t of $\mathscr{W}$ over time. Let $\mathscr{A}$ be the set of agents that does not include A. For a sequence of observations $O_i,\ldots,O_t$, how can A estimate when $\min_{A_j\in\mathscr{A}}\mu(A,A_j)$ violates some threshold?

Note that a solution to this problem does not require explicit information about objects or their tracks. This can be exploited to define efficient approximating functions.

A. Background

The approach taken here draws on a wealth of related works in monocular collision avoidance approaches, such as [1], [2], [3], [4], but is most closely related to [5] & [6]. The intuition of these approaches is that the optical flow derived from a sequence of monocular stills provides sufficient information to compute *time-to-contact* (TTC), which informs an agent about the degree to which it is being encroached upon.

B. Assumptions

In order to define a proof-of-concept solution, several simplifying assumptions are made about the target domain:

1) Camera frame is fixed in the direction of motion
2) Only head-on encroachment is of interest
3) Objects fill the field of view as they near

[1]Jeffrey Kane Johnson received his PhD from Indiana University and is principal of Maeve Automation, Mountain View, CA 94043
contact@maeveautomation.com

Fig. 1: Top: The Donkey Car encroaches on an object in its field of view. Bottom: The Donkey Car stopped at the point encroachment is detected. Video of this scenario at: https://youtu.be/QDZJRk6OJZQ

C. Approach

The approach approximates TTC by finding the rate of dilation of the scene in the field of view. However, rather than computing TTC explicitly, this approach detects when the rate of apparent expansion of the scene in the field of view violates a threshold. The rate of expansion roughly approximates the dominant optical flow in the scene, the divergence of which can be used to compute a term that is proportional to the TTC [7].

The solution is implemented on the Donkey Car [8] mobile platform with libraries available from [9]. In the approach a hybrid proportional velocity control law moves the car away from encroachment, similar to the approach proposed in [10]. To compute the control term, a hierarchy of dilated versions of the *previous* image frame are computed at various scale factors. An image space metric $\mu(\cdot,\cdot)$ is used to compare each scaled image with the current image to see whether one of the scaled images is nearer the current image than the previous image. When any of the scaled versions is nearer, the scene is interpreted as undergoing expansion and detection of encroachment is triggered. Under Assumption 3, this approach becomes more accurate as proximity decreases. The detection trigger can acts as an error term the controller to modulate agent speed.

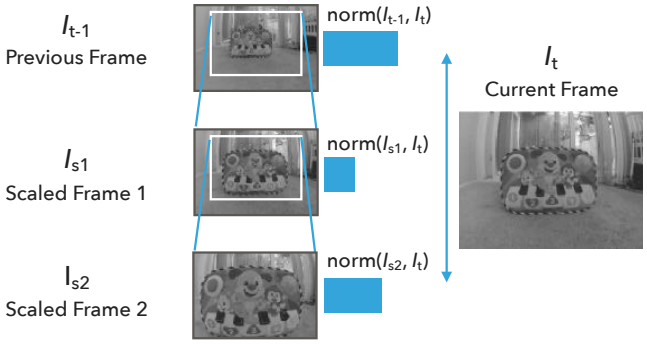

Fig. 2: The current frame (right) is compared to the previous image frame and two scaled versions of if (left column). When a scaled image is nearer in image space than the original, encroachment is detected. In this figure, the first scaled image I_{s_1} is nearer in image space, so a detection fires. The method can work despite low resolution, 160×120 pixels, and unrectified images.

```
  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 9817 maeve-pi  20   0  213640  55136  26064 S  50.8  6.2   2:45.57 python
 9562 maeve-pi  20   0   95816  14852  12056 S   6.9  1.7   0:24.37 republish
 9816 maeve-pi  20   0  153108  10500   8868 S   1.7  1.2   0:05.91 raspicam_node
 9819 maeve-pi  20   0   74152   6608   6004 S   1.0  0.7   0:03.91 relay
 9966 maeve-pi  20   0    7200   2712   2180 R   1.0  0.3   0:02.48 top
```

Fig. 3: Process list of encroachment detection running on a Raspberry Pi 3 during the trials depicted in Figure 1. The python process running the library consumes 50% CPU.

Algorithm 1 This algorithm addresses Problem 1. For previous and current images I_{t-1} and I_t, and scale set S, compute a scale pyramid from I_{t-1} according to S and match I_t to it using an L_1 matrix norm $\mu(\cdot)$. Let ε be a noise threshold.

1: **procedure** ENCROACHMENTDETECTION($I_{t-1}, I_t, S, \varepsilon$)
2: Let $\Delta_{\text{bg}} \leftarrow \mu(I_{t-1}, I_t)$ be baseline image change
3: **if** $\Delta_{\text{bg}} > \varepsilon$ **then**
4: Image change too great detect reliably
5: **return** False
6: **end if**
7: **for** $s \in S$ **do**
8: Let I_s be I_{t-1} expanded about its center by s
9: Crop I_s to the dimensions of I_{t-1}
10: $\Delta_I \leftarrow \mu(I_s, I_t)$
11: **if** $\Delta_I < \Delta_{\text{bg}}$ **then**
12: I_s is "closer" to I_t than I_{t-1}, this indicates
13: that the scene is undergoing expansion
14: **return** True
15: **end if**
16: **end for**
17: No expansion detected
18: **return** False
19: **end procedure**

D. Complexity

In Algorithm 1, Line 2 adds an $O(|I|)$ term, while the for loop at Line 7 adds $O(2|S||I|)$ due to Lines 8 & 10. Thus, the total complexity of Algorithm 1 is $O(2|S||I|+|I|)$.

A desirable property of this approach is that the complexity is only sensitive to the number of scale factors and the size of the images. Therefore, in uses where both of these are fixed, the complexity is effectively $O(C)$ for a constant factor $C = 2|S||I|+|I|$.

III. Conclusion

Although unoptimized and written in Python, Figure 3 shows the process running the encroachment detection algorithm only consuming half of available compute resources of the onboard Raspberry Pi 3. Further, because the complexity of the algorithm is effectively constant, it is guaranteed to never consume more.

This approach is simple, computationally efficient, and easy to tune, even when used with distorted, webcam-quality images (see Figure 1). This approach can also be straightforwardly extended to interpret other types of relative motion by using other image transforms.

References

[1] D. Coombs, M. Herman, T. Hong, and M. Nashman, "Real-time obstacle avoidance using central flow divergence and peripheral flow," in *ICCV*, 1995, pp. 276–283.

[2] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis, 13th Scandinavian Conference, SCIA 2003, Halmstad, Sweden, June 29 - July 2, Proceedings*, 2003, pp. 363–370.

[3] R. C. Nelson and Y. Aloimonos, "Obstacle avoidance using flow field divergence," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 10, pp. 1102–1106, 1989.

[4] S. J. Pundlik, E. Peli, and G. Luo, "Time to collision and collision risk estimation from local scale and motion," in *Advances in Visual Computing - 7th International Symposium, ISVC 2011, Las Vegas, NV, USA, September 26-28, 2011. Proceedings, Part I*, 2011, pp. 728–737.

[5] T. Mori and S. Scherer, "First results in detecting and avoiding frontal obstacles from a monocular camera for micro unmanned aerial vehicles," in *2013 IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, May 6-10*, 2013, pp. 1750–1757.

[6] J. Byrne and C. J. Taylor, "Expansion segmentation for visual collision detection and estimation," in *IEEE International Conference on Robotics and Automation, Kobe, Japan, May 12-17*, 2009, pp. 875–882.

[7] T. Camus, D. Coombs, M. Herman, and T. Hong, "Real-time single-workstation obstacle avoidance using only wide-field flow divergence," in *13th International Conference on Pattern Recognition, ICPR 1996, Vienna, Austria, 25-19 August, 1996*, 1996, pp. 323–330.

[8] W. Roscoe, *Donkey Car: An opensource DIY self driving platform for small scale cars*. [Online]. Available: http://www.donkeycar.com

[9] J. K. Johnson, "Maeve automation development libraries," https://bitbucket.org/maeveautomation/maeve_automation_core, 2017.

[10] D. N. Lee, "A theory of visual control of braking based on information about time-to-collision." *Perception*, vol. 5, no. 4, pp. 437–459, 1976.