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

**Visualization:**

- $I_{t-1}$: Previous Frame
- $I_t$: Current Frame
- $I_{s1}$: Scaled Frame 1
- $I_{s2}$: Scaled Frame 2

Encroachment is detected by examining a low resolution (160x120) fisheye video sequence for evidence of scene expansion. To the left, the smaller norm at $I_{s1}$ indicates expansion.

**Algorithm & Complexity:**

```
1: procedure ENCPHACMENTDETECTION($I_{t-1}$, $I_t$, $S$, $e$)
2: Let $\Delta_{bg} \leftarrow \mu(I_{t-1}, I_t)$ be baseline image change
3: if $\Delta_{bg} > e$ 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: $\Delta_s \leftarrow \mu(I_s, I_t)$
11: if $\Delta_s < \Delta_{bg}$ 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
```

Line 2 adds an $O(||I||)$ term, while the for loop at Line 7 adds $O(2||S||||I||)$ due to Lines 8 & 10. Together the total complexity is $O(2||S||||I||+||I||)$. A nice 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 things are fixed, the complexity is effectively constant.

In practice, this algorithm can be implemented efficiently. An unoptimized version written in python and OpenCV with $|S| = 2$ and $||I|| = 160 \times 120$ runs comfortably on a Raspberry Pi 3, as seen below.

**CPU time on a Raspberry Pi 3**

<table>
<thead>
<tr>
<th>PID USER</th>
<th>PR NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHE</th>
<th>CPU</th>
<th>MEM</th>
<th>TIME</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>9817 maeve-pi</td>
<td>28 0</td>
<td>23508</td>
<td>55136</td>
<td>26064</td>
<td>5308</td>
<td>0.2</td>
<td>2.45</td>
<td>python</td>
</tr>
<tr>
<td>9562 maeve-pi</td>
<td>28 0</td>
<td>95816</td>
<td>1482</td>
<td>12436</td>
<td>5</td>
<td>0.24</td>
<td>2737</td>
<td>republish</td>
</tr>
<tr>
<td>9816 maeve-pi</td>
<td>28 0</td>
<td>153398</td>
<td>18000</td>
<td>81568</td>
<td>1.7</td>
<td>0.2</td>
<td>0.85</td>
<td>raspi_cam_mode</td>
</tr>
<tr>
<td>9819 maeve-pi</td>
<td>28 0</td>
<td>74152</td>
<td>6688</td>
<td>6904</td>
<td>1</td>
<td>0.7</td>
<td>0.83</td>
<td>relay</td>
</tr>
<tr>
<td>9966 maeve-pi</td>
<td>28 0</td>
<td>7250</td>
<td>2772</td>
<td>2180</td>
<td>1.0</td>
<td>0.3</td>
<td>0.82</td>
<td>top</td>
</tr>
</tbody>
</table>

**Demo:**

A demonstration of the technique can be seen online by scanning the QR code or visiting the link below the QR code.

https://youtu.be/QDZJRk6QJZQ

Jeffrey Kane Johnson