Application of a Generalized Regressive Motion algorithm to mobile, vision-based robots

AMRIT RAU
KRZYSZTOF CHALUPKA
Dr. PIETRO PERONA

Abstract
We explored the viability of Generalized Regressive Motion (GRM) algorithms in the context of mobile robotic navigation. Initially, an extension to the basic GRM algorithm was explored to enable the application of GRM to a vision-based robotic system. A prototype vehicular testing platform was constructed to evaluate the performance of our algorithm. We found that a GRM algorithm onboard a mobile robot is an extremely low-cost and viable means of real-time collision avoidance.

Introduction
In simulation, generalized regressive motion (GRM) has been shown to be an effective visual cue to collision as well as possibly an inherent biological collision avoidance scheme in fruit flies by Chalupka et al. However the value of GRM in mobile robotic path planning and collision avoidance has generally been relegated in favor of simpler but less effective algorithms such as looming and simple regressive motion.

By modifying the original GRM algorithm of Chalupka et al., we were able to generalize and apply a GRM algorithm to a simple Android-based robotic testbed. Our results suggest that GRM algorithms are a lightweight, viable option for vehicular navigation and collision avoidance because of their ability to detect the proximity of not only stationary but also moving objects.

Our testing platform running the GRM algorithm scored highly in both safety and mobility performance tests, suggesting that GRM can be applied in the real world to areas such as rapid, real-time collision avoidance.

Methods
Our investigation consisted of two components: the GRM optical flow algorithm and the mobile robotic testbed.

GRM optical flow algorithm
Chalupka, et al. defined GRM as:

- Rightward (clockwise) optical flow in the left visual hemifield and/or
- Leftward (counterclockwise) optical flow in the right visual hemifield and/or
- Any optical flow in the extent of the contralateral visual angle (CVA, defined as the “angular extent of the eye’s field of view on the nasal side”)

If for any feature point’s x-position on the image the GRM \( \| \mathbf{x} \| \) is greater than a specified threshold velocity \( T_{GRM} \), a collision flag is thrown.

Radial vector projection
In order to account for impending collisions in three dimensions, we generalized the existing bidirectional algorithm to all optical flow vectors on the field of view by projecting all optical flow vectors onto a radius, then comparing the radially projected vectors’ magnitude (representing velocity towards the vehicle’s future trajectory) to a desired threshold. This took into account motion flow vectors in all directions that tended towards the center of the image, not just those with significant horizontal components.

Because of this, the CVA is also projected radially and becomes the contralateral visual solid angle, or \( CV\Omega \). Any optical flow within this solid angle over the GRM threshold is considered GRM.

Taking into account both the \( x \)- and \( y \)-components of the image optical flow vectors, we can calculate GRM as \( \left\| \frac{\partial x}{\partial t} \mathbf{i} + \frac{\partial y}{\partial t} \mathbf{j} \right\| \), where \( \mathbf{i} \) and \( \mathbf{j} \) are unit vectors in the \( x \)- and \( y \)-directions respectively.

Figure 1 Our radial vector projection implementation of a GRM algorithm.
Mobile robotic testbed
Our mobile robotic testbed consisted of two components: an Android phone to run the vision algorithm and a LEGO Mindstorms NXT-based drivetrain.

Vision (Android)
An Android application was written to compute GRM utilizing Lucas-Kanade sparse optical flow estimation built in to the OpenCV computer vision library\(^1\). To improve performance, the video input was downsampled by convolving each frame with a 3x3 Gaussian kernel prior to running the optical flow algorithm. The application was run on a Motorola Moto E cell phone, which was mounted on the robot. The cell phone camera’s solid angle of view was 0.295\(\pi\) sr.

![Figure 2](image)
Figure 2 Our GRM algorithm applied to film of a street. Optical flow vectors are drawn in green with blue tails, while the red circle represents the CV\(\Omega\).

Drivetrain & Microprocessor (LEGO NXT)
A treaded vehicular platform was constructed with parts from the LEGO Mindstorms NXT kit, utilizing two motors in a tank drive configuration. A simple control scheme, written using the LabView graphical programming interface, utilized binary light sensor input to throw a collision flag on board the robot.

Communication
To maximize processing speed of the GRM algorithm, we did not use Bluetooth to communicate between the two components of the drivetrain. Rather, when GRM was detected, the phone screen changed from black to white. A LEGO Mindstorms light sensor sent this binary value to the microprocessor, which followed the specified control scheme to determine the next course of action.

Procedures
We tested the viability of our GRM algorithm in a series of experiments, investigating collisions with stationary obstacles, progressively moving obstacles, and regressively moving obstacles\(^2\).

![Figure 2](image)

A virtual obstacle was projected onto a screen in the robot’s path. The obstacle was a 0.105\(\pi\)sr red square against a white background. The GRM threshold was 30 px\(\cdot\)frame\(^{-1}\), and the CV\(\Omega\) was 0.038\(\pi\) sr.

For the first set of experiments, the robot was stationary at a point 0.60 m away from the screen. If the phone detected GRM, it beeped. A stationary obstacle, a progressively moving obstacle, and a regressively moving obstacle were projected 10 times each. The moving obstacles had azimuthal velocities of 0.605\(\pi\) rad/s. This tested the phone-based implementation of the algorithm. If the phone beeped in response to no motion or progressive motion, or the phone did not beep in response to regressive motion, then the algorithm was faulty. If the phone did in fact beep for regressive motion, then the algorithm was implemented correctly.

For the next set of experiments, the robot approached the screen at 0.14 m/s starting from a point 0.60 m away. If the phone detected GRM, a collision flag was thrown on the robot, which stopped. The same obstacles were tested 10 times each. This tested whether the system was still effective when in motion. If the robot stopped before colliding with the screen, it was assumed that the robot detected GRM and thus successfully avoided the collision.

A true positive (TP) represented a situation in which the robot stopped as a result of a GRM cue. A true negative (TN) represented a situation in which the robot did not stop when no GRM cues were present. A false positive (FP) represented a situation in which the robot stopped despite a lack of GRM cues, and a false negative (FN) represented a situation in which the robot failed to stop in spite of GRM cues.

These labels (TP, TN, FP, and FN) help us quantify the robot’s performance by safety and mobility as shown below.

\[
safety = \frac{TP}{TP + FN}
\]

\[
mobility = \frac{TP}{TP + FP}
\]

In the real world, a high safety score indicates few collisions, while a high mobility score indicates few

---

\(^1\) See [http://opencv.org](http://opencv.org).

\(^2\) A movie with experiment footage will be made available at [http://vision.caltech.edu](http://vision.caltech.edu).
unnecessary stops. With any such robotic system with a goal of practical application, maximal safety is crucial and a high indicator of suitability to the real world. However, the ultimate goal is of course to maximize both safety and mobility.

**Results**

As in Chalupka, et al. 2014, we score the overall performance of the system by safety and mobility. Our system achieved 100% safety, with 73% mobility.

<table>
<thead>
<tr>
<th>OBSTACLE</th>
<th>Stationary</th>
<th>Progressive</th>
<th>Regressive</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROBOT</td>
<td>Stationary</td>
<td>100% TN</td>
<td>20% TN, 80% FP</td>
</tr>
<tr>
<td>0.14 m/s</td>
<td>70% TP, 30% FP</td>
<td>80% TN, 20% FP</td>
<td>90% TP, 10% FP</td>
</tr>
</tbody>
</table>

**Discussion**

On the whole, our algorithm performed as expected. It was highly successful in detecting generalized regressive motion both inside and outside the CVΩ. In terms of safety and mobility, it was found to be extremely promising. In a simulated experiment involving looming, which did not take into account mechanical and sensor noise, a looming algorithm only achieved 25% mobility at 100% safety.

Although most of our results were as expected, the progressive obstacle/stationary robot test returned false positives for the most part, negatively impacting the overall mobility score. Visual inspection of the optical flow vectors indicated that false GRM readings were being triggered at the moment the obstacle’s image left the camera’s field of view. The spurious optical flow vectors that triggered the GRM appeared in the bottom row of pixels captured by the camera in the opposite direction of the actual motion flow. This suggests a bug in the OpenCV implementation of the Lucas-Kanade algorithm. If so, this could easily be solved client-side by disregarding the border pixels of each frame when computing optical flow.

In addition, there were many issues with our testbed. Due to the small size of the vehicle, the camera shook vigorously when moving, resulting in false GRM readings of magnitudes up to 20 px-frame$^{-1}$ (two-thirds of our threshold) in the most extreme cases. In addition, another major obstacle to real-world application is the need to fine-tune the system configuration to account for slight changes in the environment and lighting.

As we increase $T_{GRM}$, we expect our safety score to decrease but our mobility score to increase. In contrast, if we decrease $T_{GRM}$, we expect safety to increase but mobility to decrease. However, the computational time-complexity of our algorithm increases with the square of $T_{GRM}$. We suspect that our relatively high $T_{GRM}$ value of 30 px-frame$^{-1}$ significantly slowed the Android phone (with ~400 MB of usable RAM) camera’s framerate to approximately 3 fps. This did not appear to be a major issue at low speeds on the small scale, but clearly more computing power is necessary for higher speeds.

Our results suggest that GRM has great potential as a low-cost assistive tool for autonomous vehicle navigation. Provided that the framerate of the video capture device is sufficiently high, GRM can detect not only static but also dynamic collisions in the real world.

**Further research**

One area of further research would be to investigate the viability of GRM vs. looming on robotic platforms to confirm that GRM is superior to looming in mobile robots.

In the interest of making the algorithm more robust, versatile and applicable in three dimensions, we suggest that the GRM optical flow threshold should be on a sliding scale to take into account the ground motion.

---

$3$ As $T_{GRM}$ increases, the Lucas-Kanade search radius for the nearest pixel with the closest intensity $l$ increases linearly, and the search area increases with the square of the search radius.
vector of the vehicle. In future work, we hope to test this definition on a three-dimensional testing platform.

The three-dimensional case has many applications in aeronautics and space technologies -- a simple GRM system could be implemented as a last line of defense against collisions with uncooperative, tumbling objects in orbit, or could act as a low cost, real-time cue to collision for low-flying aerial drones in urban areas. Testing the 3-d case could not only prove valuable in the field of robotics, but could also prove valuable in the field of biology. Chalupka et al. suggested that fruit flies (Drosophila) utilize GRM cues to avoid collisions while walking, but a 3-d testing platform could potentially simulate and analyze aerial, vision-based Drosophila interactions of higher complexity.

References


