Optical Flow Based Object Detection and Avoidance

Fall 2018

Image Compression, Packet Video, and Video Processing

Christina Paolicelli

Introduction

In this report the current state of optical flow for robotic obstacle detection and avoidance is discussed and a optical flow methodology using dense measurements was developed as a proof of concept implementation.

Background

Optical Flow

Optical Flow is defined as the pattern of apparent motion of objects, surfaces and edges in a visual scene caused by the relative motion between an observer and a scene. Consider a pixel *I(x,y,t)* in the first frame the pixel moves by the distance *(dx, dy)* in the next frame after having taken *dt* time, since these pixels are the same and intensity does not change we can say $I(x, y, t) = I(x + dx,\ y + dy,\ t + dt)$. Taking the taylor series approximation of right-hand side of the equation, remove the common terms and divide by *dt* to get the Optical Flow equation, $f_x u + f_y v + f_t = 0$, where $f_x\ and f_y$ are image gradients and f_t is the gradient along time. *u and v* are two unknowns which represent the velocity vector of the pixels which is what the desired output of optical flow is, however the equation cannot be solved for two unknowns so several methods have been developed to approach this problem of computing optical flow vectors. In this project the Farneback method is used, however common methods exist including the Luca-Kanade method.

State of Field

Across the field of robotics, optical flow methods are frequently used for navigation and object avoidance through a variety of methods, most commonly some form of time to contact calculations. Optical flow has, however, been found, even with robust optical flow systems which provide range information on par with sonar information and in some frames on par with laser information, to function better as part of a full vision system. (School of Information Technology and Electrical Engineering, University of Queensland, 2018) With all the downsides of optical flow and the need for additional vision technology for robust control optical flow is still used widely across the industry for a number of reasons. It is quite common for quadcopters and smaller robots as there is no specialized hardware needed other than a camera which most these systems already have, the lack of specialized technology makes optical flow a low cost, lightweight option for obtaining depth information where the industry standard is a laser range finder a comparatively expensive choice.

A survey on Visual Navigation for Mobile Robots (Bonin-Font, Ortiz and Oliver, 2008) indicates that most robotic approaches to using optical flow for navigation use either the Horn and Shunk or Lucas and Kanade methods, the Lucas and Kanade method, while not what was used in this design, is available as a pre-designed function in the OpenCV python library. Another common approach of optical flow algorithms is to use biomimicry to emulate insect flight as many insects including bees use a form of optical flow for navigation. (Iida, 2003)

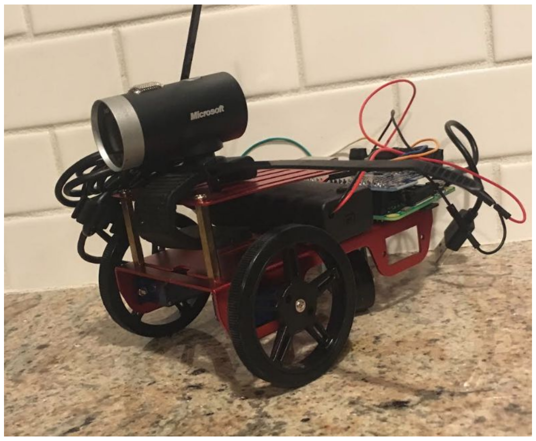
In order to optimize computation time of optical flow it is common to first extract features of the image such as corners or edges and only compute the motion vectors on those points. A downside of the extraction of prominent features is that irregularities on the floor or walls can be interpreted as edges or corners and therefore wrongly considered obstacles causing errors during navigation.

A particularly interesting, related, application of optical flow is in the autonomous car industry. Because Optical flow not only has the ability to detect obstacles but also to track them, this is an area of optical flow which also has a lot of work being done in it the technology is promising for this application as other cars or pedestrians can be tracked rather than having to be identified every frame. (Menze and Geiger, 2018)

Approach

Hardware Platform

The tested robot was a two wheel drive Adafruit Mini Robot Rover Chassis driven by a Raspberry PI 3 with a Motor Hat. A generic USB webcam was used to obtain images to process optical flow on. OpenCV was used to take the images and perform optical flow with the library's built in functionality. A separate machine running a SSH client was used to command and control the robot.



Dense Flow

The OpenCV method used to compute the optical flow motion vectors is based on "Two-Frame Motion Estimation Based on Polynomial Expansion" by Gunner Farneback (Computer Vision

Laboratory, Linkoping University, 2018). The benefit of this approach is that it computes the flow for all points in the frame. This process is resource intensive and time consuming which may have issues if this methodology is applied to actual applications, however, for this implementation which has a relatively low quality camera and is being tested in poor lighting and non optimal conditions this method is most likely to provide sufficient data to allow the tests to be successful.

This approach works by approximating neighborhoods of both frames by using quadratic polynomials. The method observes how an exact polynomial transforms under translation as a method to estimate displacement fields.

Balance Algorithm

A number of different methods were analyzed for a simple approach to optical flow object avoidance which could be implemented on the robotic platform, it was decided to implement the balance algorithm from (Souhila and Karim, 2007). The idea behind this approach is that when the robot is translating closer objects give rise to faster motion than farther objects. To use this in a practical way the image is split into a right and left half and the motion vectors are summed on both halves, because flow will be greater when the objects are closer the robot turns away from the greater flow. The equation behind this is presented below.

$$\Delta(F_L - F_R) = \frac{\sum \|w_L\| - \sum \|w_R\|}{\sum \|w_L\| + \sum \|w_R\|}$$

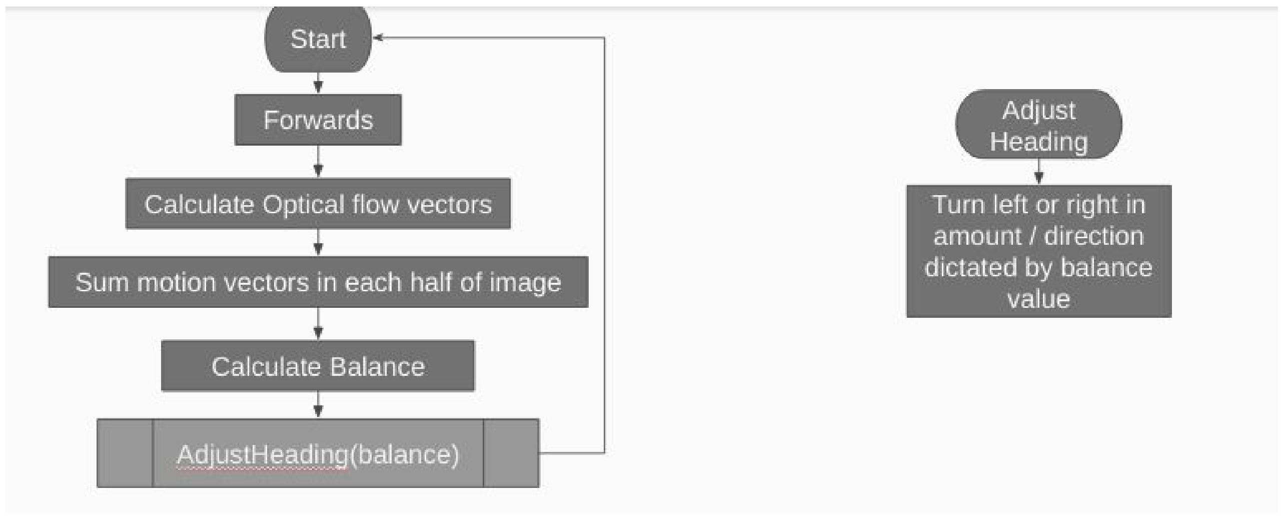
Where

$F_L \equiv Force\ on\ Left\ side\ of\ the\ robot$

$F_R \equiv Force\ on\ Right\ side\ of\ the\ robot$

$\sum \|w\| \equiv sum\ of\ the\ magnitude\ of\ optical\ flow\ vectors\ on\ one\ side\ of\ the\ robot's\ heading$

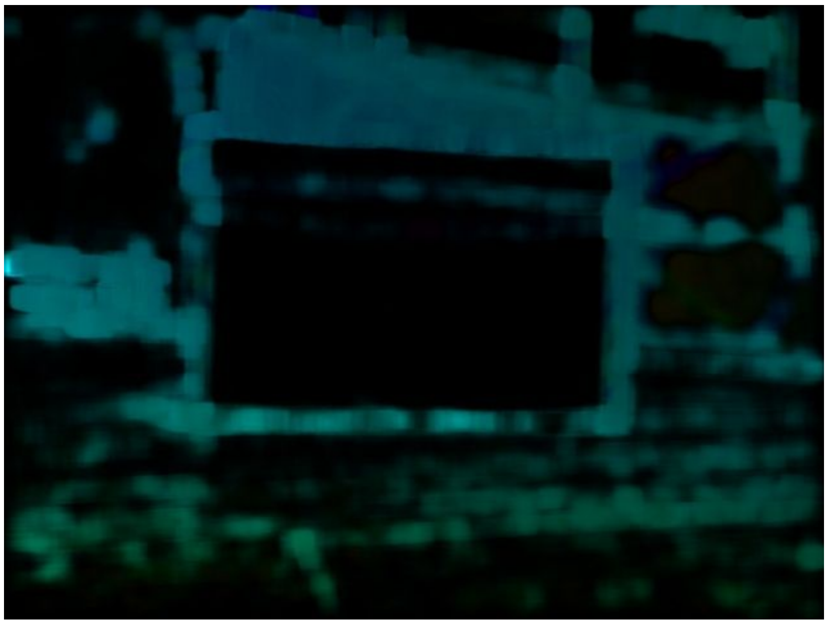
Software Overview


Start
Forwards
Calculate Optical flow vectors
Sum motion vectors in each half of image
Calculate Balance
AdjustHeading(balance)
Adjust Heading
Turn left or right in amount / direction dictated by balance value

The software implementation of this methodology is relatively simple. The robot has an initial image, frame 1, moves a short distance forwards and takes a second image, frame 2, and computes the optical flow vectors between the two frame. These optical flow vectors are summed on each side of the image and the balance algorithm is calculated and passed to the adjust heading function which turns the robot right or left by the amount dictated by the calculated balance value.

Testing

The algorithm was tested in a fairly empty room using a cardboard box as the obstacle. No empirical data was produced, as it was difficult to replicate conditions well enough that empirical outputs were meaningful, but the robot successfully avoided the box as long as it was given space for at least one or two turns. As a logging output the raw frames taken by the camera to be used for optical flow calculations were saved along with depth images generated from the optical flow calculations. In the image below the box can be clearly seen in the center of the depth image, showing that the algorithm detects the obstacles.



Conclusion

Challenges

A challenge with this approach was the computational intensity of the algorithm. The robot is very slow, there is a noticeable pause between moves where computation occurs. While this delay is fine for the proof of concept that was the goal of this project it would likely be an issue in any real world deployment of this system. There are however several ways to mitigate this issue. A better processor than a raspberry pi could be used. While raspberry pis are fantastic for prototyping they are not particularly powerful. Prior to using optical flow edges and corners could have been detected and the optical flow would only have to be applied to those points - this decreases the points at which optical flow vectors are calculated and therefore the overall computational intensity.

The software for implementing optical flow using the OpenCV library was relatively straight forwards, future work on this project could move to an feature extraction method and tune some of the gains in the system but overall the robot was successful from a software perspective. The hardware on the other hand presented some challenges, at the update time frame of the project I was unsure if I was going to be able to get the hardware working, at the time I was using the raspberry pi to directly drive motor phases not though a hat, I was considering a simulation approach, however without updating images from true movement simulating object avoidance is a difficult problem. At this point the hardware was changed and a it was decided to use a motorhat, this significantly simplified the motor control and made the image processing part of this project successful. As is almost always the case with implementing a software project on hardware the hardware proved to be more complicated to work on than the software.

As much of the work in the field alludes to optical flow by itself is not the best method of robotic navigation, however as this project shows it can be used to avoid obstacles and the field has concluded that optical flow is a valuable an important tool in the overall scheme of autonomous robot and more generally vehicle navigation due to its low cost ability to provide depth information when compared to other industry standards such as sonar and laser.

References

Industries, A. (2018). *Adafruit DC & Stepper Motor HAT for Raspberry Pi - Mini Kit*. [online] Adafruit.com. Available at: https://www.adafruit.com/product/2348 [Accessed 6 Dec. 2018].

Industries, A. (2018). *Mini Robot Rover Chassis Kit - 2WD with DC Motors*. [online] Adafruit.com. Available at: https://www.adafruit.com/product/2939 [Accessed 6 Dec. 2018].

Computer Vision Laboratory, Linkoping University (2018). *Two-Frame Motion Estimation Based on Polynomial Expansion*. Linkoping, Sweden.

Souhila, K. and Karim, A. (2007). Optical Flow Based Robot Obstacle Avoidance. *International Journal of Advanced Robotic Systems*, 4(1), p.2.

School of Information Technology and Electrical Engineering, University of Queensland (2018). *Obstacle Detection using Optical Flow*. [online] St Lucia, Australia. Available at: http://www.araa.asn.au/acra/acra2005/papers/low.pdf [Accessed 7 Dec. 2018].

Bonin-Font, F., Ortiz, A. and Oliver, G. (2008). Visual Navigation for Mobile Robots: A Survey. *Journal of Intelligent and Robotic Systems*, 53(3), pp.263-296.

Iida, F. (2003). Biologically inspired visual odometer for navigation of a flying robot. *Robotics and Autonomous Systems*, 44(3-4), pp.201-208.

Menze, M. and Geiger, A. (2018). *Object Scene Flow for Autonomous Vehicles*.

Docs.opencv.org. (2018). *OpenCV: Optical Flow*. [online] Available at: https://docs.opencv.org/3.4/d7/d8b/tutorial_py_lucas_kanade.html [Accessed 8 Dec. 2018].