Auto obstacle-avoidance of automatic mobile base on optical flow

自走車之光流自動避障模組

專業領域:系統組　組別:A94　指導教授:鄭桂忠
組員姓名:張智凱、蘇揚智、趙駿騰

Abstract

近年來自走車的研究越來越盛行，包含方向調整、速度操控與避開障礙等領域。其中，預測障礙物的方法多採用紅外線感測或超音波探測。然而，紅外線易受是外光線影響，而超聲波則易受溫度風向影響，因此，我們希望加入第三種測障方法，也就是光流，來提升自走車的測障精確度。

光流是透過連續兩張照片的差異而計算出的資訊，透過這些差異可以得知物體的遠近關係。我們採用單鏡頭相機拍攝連續照片，並藉此算出每個時刻的光流。再透過座標轉換、相似三角形等數學原理應用，藉由相機的運動資訊計算出物體的旋轉與平移光流，進一步推算物體深度資訊。由於地板並非障礙物而是可行走的路徑，用坐標軸轉換的方法將地板去除後，我們可以得到地板外的物體深度資訊。藉由這些深度資訊，可以得到離相機最近的物體在何處，並提供給自走車系統進行避障判斷。

我們的系統分為軟體與硬體兩部分。軟體的部分藉由 Opencv 的函式庫，使用 DIS 光流演算法搭配數學理論實現深度資訊推算。過程中，採用 Minecraft 提供的理想環境進行測試，並成功得到符合預期深度資訊。硬體的部分選用 RaspberrtPi 板子作為計算器，並搭載馬達與相機。相機拍攝連續照片，而馬達驅動輪子進行我們想要的操作。

系統整合時，我們使用兩種方法測式，第一種方法是用相機錄影再經由個人電腦分析光流與深度資訊，雖因缺乏精確的相機運動資訊而有不少誤差，但仍達到預期的結果。第二種方法為即時拍照計算，然而受限於硬體計算速度過慢而無法進行。

我們成功完成了測障與避障方案的設計，雖然未能將二者成功合併為完整的即時避障系統，若將做出的光流測障模組進行優化與整合，有望加入現有的自走車避障系統中，達成最初希望提升避障精確度的目標。

Introduction

本專題旨在透過單鏡頭相機拍攝連續照片所算得的光流資訊，推算深度並提供系統進行避障判斷。我們使用的硬體設備為 RaspberryPi 板子、單鏡頭相

機、馬達與車體。外型如下圖 1.。

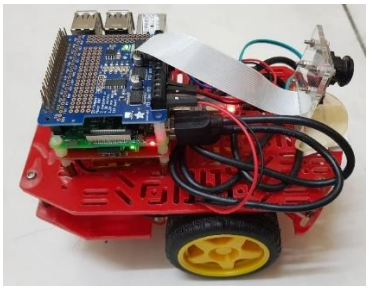

圖 1.

1. 理論

光流是一種透過連續拍攝相片，並將前後兩幀進行比較，估算每一個像素的強度變化所得出的資料。他可以用於物件追蹤或碰撞時間估算等。而我們所需要的是運用光流資訊推算障礙物方向與距離，以利車子提前進行避障。

光流法的基本假設有三：1. 兩幀照片間隔時間很短 2. 兩幀照片間亮度恆定 3. 場景中相同表面的相鄰點具有相似的運動

在認知這三個條件並了解光流基本算法後，我們在各種光流演算法中做選擇，最後採用 2016 年由 Till Kroeger 等人開發 DIS 演算法作為我們的光流法。

得到光流後，接下來說明如何由光流推算深度。我們假設一面虛擬的牆始終立在相機正前方二十公尺的位置。由於這面牆會隨相機鏡頭轉動，以保持在相機正前方，因此我們可以透過相機的移動資訊，估算出這面牆的理想光流。再將理想光流和實際算得的光流比較，即可算得深度。

深度的算法是使用相似三角形的概念。以下分為相機僅平行位移和僅旋轉討論產生的平移光流與旋轉光流和深度關係。

當相機平行位移時，物體和其在虛擬牆上對應的點在相機的焦距處產生的偏移，而這個偏移即為光流。由相似三角形關係分別推出以下式子

$$\frac{Real\ \ optical\ \ flow\ \ of\ \ object}{focal\ \ length} = \frac{camera\ \ shift}{Depth\ \ of\ \ object}$$

$$\frac{ideal\ \ optical\ \ flow\ \ of\ \ virtual\ \ wall}{focal\ \ length} = \frac{camera\ \ shift}{20}$$

將兩式相除可得

$$Depth\ \ of\ \ object = 20 \times \frac{ideal\ \ optical\ \ flow\ \ of\ \ virtual\ \ wall}{Real\ \ optical\ \ flow\ \ of\ \ object} \quad (式\ 1)$$

此即平移光流和深度的關係。

接下來考慮僅旋轉的部分，當相機旋轉θ角度時，物體和其對應的虛擬牆上的點會在焦距處產生一樣大小的光流。因此僅旋轉時，由於沒有光流差，所以沒辦法推算深度。也就是說僅旋轉造成的光流和深度沒有關聯。

將平移和旋轉兩個狀況組合起來，便可描述相機的所有運動情形。而總光流可由下式表達。

$$\text{Total optical flow} = \text{optical flow}_{translation} + \text{optical flow}_{rotation}$$

物體和其對應點的光流可表為

$$Total\ opt_{object} = translation\ opt_{object} + \text{optical flow}_{rotation}\text{(式 2)}$$

$$Total\ opt_{virtual\ wall} = translation\ opt_{virtual\ wall} + \text{optical flow}_{rotation}\text{(式 3)}$$

由於計算深度需要物體和對應點的平移光流比值，加上旋轉光流與深度無關，所以需要將式 2、3 先減去旋轉光流，才能得到平移光流的比值，並用式 1 推算深度。

有了上述基礎後，我們尚缺乏的資訊便是旋轉光流與虛擬牆的平移光流。這裡我們定義所使用到的兩個坐標系，圖片坐標系和相機坐標系。

1. 圖片的坐標系(u, v)：為圖片上的二維坐標系，光流是直接在這個坐標系上計算。
2. 相機坐標系(x, y, z)：為空間中的三維坐標系，代表相機看出去的現實世界的座標，可用來推算現實世界障礙物坐標與光流的關係。

這兩個坐標系間的關聯為

$$(\mathrm{u}, \mathrm{v}, \mathrm{w}) = K(x, y, z)$$

$$\mathrm{K} = \begin{bmatrix} focalLength & 0 & width/2 \\ 0 & focalLength & height/2 \\ 0 & 0 & 1 \end{bmatrix}$$

運用這兩個坐標系，我們可以推算需要的光流。假設在下一幀照片中，圖片坐標系新的點標示為(ur, vr, wr)。我們使用以下公式去推算旋轉後的牆在圖片上的位置。

$$\begin{bmatrix} ur \\ vr \\ wr \end{bmatrix} = \mathrm{K} \cdot \left[R_x \cdot R_y \cdot R_z \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \right.$$

由於是圖片坐標系，因此 wr 需為 1。將算得的(ur, vr, wr)等比例放大至 wr=1。

$$ur = \frac{ur}{wr}\ ,\ vr = \frac{vr}{wr}\ ,\ wr = 1$$

得到新的圖片座標後，將此座標減去原本的座標(u, v)，即可得旋轉光流。可表為：

$$Optical\ flow_{rotation} = (ur - u, vr - v)$$

得到旋轉光流後，具前面所述的推算深度方法，將總光流減去旋轉光流，即可得平移光流，進一步推算深度。

虛擬牆的理想平移光流算法和旋轉光流雷同，我們藉由下式將相機座標系的變動轉換到圖片坐標系上。

$$\begin{bmatrix} ut \\ vt \\ wt \end{bmatrix} = \mathrm{K} \cdot \left[\begin{array}{ccc|c} 1 & 0 & 0 & xd \\ 0 & 1 & 0 & yd \\ 0 & 0 & 1 & zd \end{array}\right] \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

可得理想平移光流。表為：

$$Optical\ \ flow_{translation} = (ut - u, vt - v)$$

如此，便得到計算深度的所有資訊。然而，接續的問題是地板會被視為距離很近的障礙物，消除方法同樣透過坐標系轉換算出理想中地板的深度。

$$depth\ \ of\ \ floor = \ \ ratio \times \overrightarrow{cfloor_z}$$

再和前面得到的深度圖做比較，若深度差不多的點，則歸類為地板，反之，則為障礙物。

2. 系統設計與實驗結果

首先我們透過 Minecraft 測試我們的軟體，其結果如下圖 3.

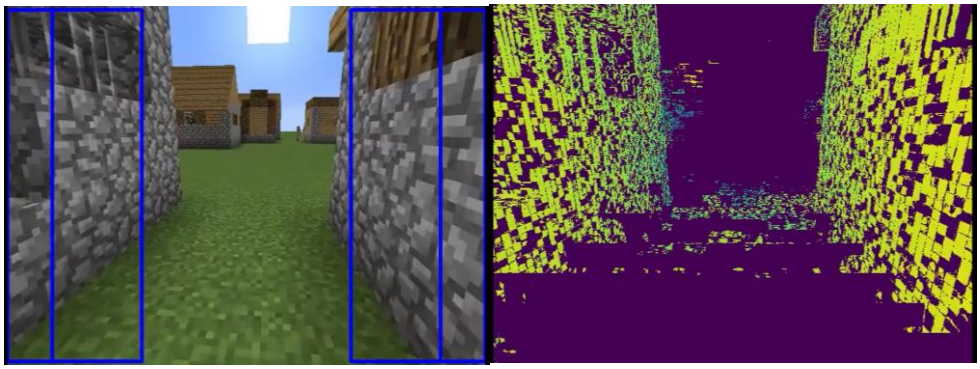

圖 3. a. 框選障礙物 b. 深度圖

我們將螢幕分為五個方向，左、左前、正前、右前、右，若該方向有在深度三公尺內的障礙物，則在畫面上將該方向框選出來。由圖 3. a. 可見，我們成功框出牆壁，且對照圖 3. b. ，越近的物體顏色越亮，所以兩側的牆壁特別明顯，而後面的其他建築則很黯淡。值得注意的是，我們成功將地板消除，讓地板被判定為深度很遠的物體，可由圖 3. b. 看出。

確認完軟體功能與預期相符後，我們將軟體與硬體整合。共分兩種方法實驗：錄製影片、即時拍照。

下圖為我們在錄製影片測試時，其中幾張車子在相對平穩前進時所測得的障礙物資訊。

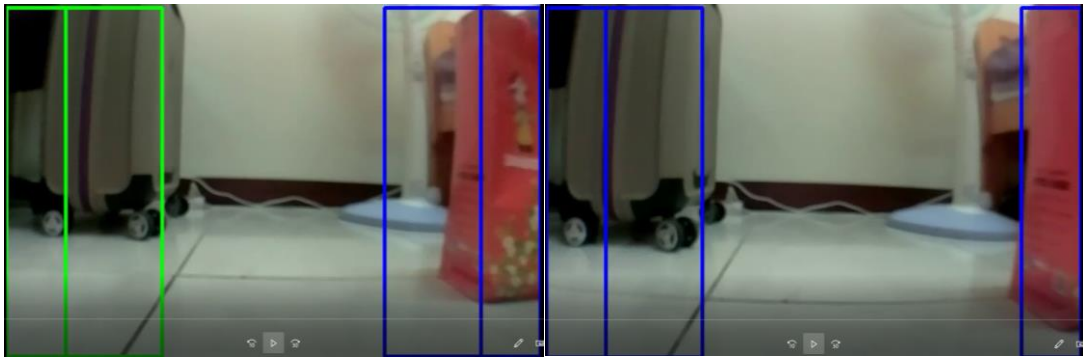

圖 4.a.b.

為了更好的顯示深度關係，我們設定三種距離區間，最近框藍色、中等框綠色、最遠框紅色。更遠則視為沒有障礙物。圖 4.a.b.可看出隨車子前進，左邊的行李箱由藍色變為綠色，代表距離變近了。

以上呈現的是成果中較佳的部分。由於車子的晃動與偏移，整部影片中有許多時間點，整個畫面都被判定到有障礙物。其原因我們認為是因為車子仍有些微旋轉與部小的晃動。些微的旋轉會造成旋轉光流，而晃動會導致理想平移光流和實際的有落差。由於我們給相機運動資訊時，假設不旋轉且除 z 方向外位移為零，這會導致算出來的深度因實際光流較我們計算的大而變的比實際上近。

接下來是即時拍照測試，就結果而言，我們認為我們的設備不適合進行即時算深度。要做到即時的最關鍵問題在於運算速度。前面光流演算法部分提過，光流的基本假設中有一項，兩幀照片間隔時間很短。因此若每幀照片的計算時間太長，算完再拍下一幀照片會造成兩幀照片間隔時間太長以致失去準度。

3. 結論

在這個專題中，我們實現了當車子前行時能透過光流與計算出的旋轉與平移光流計算出相機看出去的深度資訊，並運用深度資訊判斷五個方向是否有障礙物，以及得到五個方向的判斷結果後，下一步車子該如何避障。我們使用到的理論有 DIS 光流演算法、由座標軸旋轉得出的平移與旋轉光流計算方法和以相似三角形概念為基底推導出的光流與深度關係。硬體設備的部分，我們採用 RaspberryPi 板子，控制相機與馬達以得到我們的輸入資料。有了以上設備與理論，我們得以使用 python 及 opencv 等函示庫實現測障的功能。

整體來說，我們完成了測障與避障方法，但未能將二者成功合併為完整的即時避障系統。未來的研究方向我們認為可以大致往三個方向進行。首先，若搭載 IMU，則可得到相機的位移、旋轉資訊，不須被限制於僅能直走時測障。第二個方向是可以將車子的輪子改為同軸，以此減少因左右輪摩擦力不同造成的車子控制困難，也能做到更精確的方向控制。最後，便是我們遇到的運算速度不足的問題。若能有更高的計算速度，我們就可以測試即時測障與避障是否能符合我們的預期，並根據結果調整避障方案與演算法。

總結而言，我們的系統提供了以光流為基底的測障模組。在應用方面，可以將此模組導入其他避障系統中，提供額外的障礙物資訊以提升避障判斷的準確率。

心得

從最初對光流一無所知到現在實作出用光流測障，我們在過程中遇到許多

的問題，並在不斷的討論以及在學長與教授的各種建議下，一步一步地克服。

去年暑假，我們第一次接觸到和光流相關的論文，發現光流和想像中的不太一樣，並不需要複雜的儀器去做測量，僅透過相機拍攝連續照片再用數學方法推算照片間的變化，進而算得光流。大致了解了光流概念後，我們遇到了一個大問題，由單鏡頭相機取得的光流推算深度的論文非常少，大部分都是裡用雙鏡頭之間的光流差來推算深度。由於雙鏡頭推算深度時，光流不佔最重要的角色，因此我們還是希望使用單鏡頭實現光流推算深度。雖然在網路上還是有尋找到相關的論文，但是實用後發現精準度很低，所以我們認為可能是光流演算法的問題。

第一個選用的是 Lucas-Kanade 光流演算法，因為它是計算稀疏光流，所以計算速度很快。然而，因為現實環境不一定能滿足他的假設，因此會有很多不準的點。接下來我們選擇精準度較高的 Farneback 光流演算法，由於它是計算稠密光流，因此精準度較高。然而計算時間過長，若再搭配深度推算，會花太多時間，違背光流的基本假設：兩幀照片間的間隔時間要很短。

在準確度和速度之間難以取捨時，我們參加了 IEEE 的 student competition，並在比賽中從學長那邊得知了 DIS 演算法，以及計算深度時，應該要考慮平移和旋轉光流。DIS 是 2016 年才被提出的演算法，比起 Farneback 和 Lucas-Kanade，擁有更高的準確度與速度。而且我們也發現，前面提到的深度不準問題並非我們數學式弄錯，而是需要扣除旋轉光流。

完成軟體後，又遇到沒有辦法提供相機移動資訊，造成無法計算旋轉光流的問題。所以我們只好設定成車子只在前行時計算深度。接著新的困難是 RaspberryPi 板子計算深度時間過長，基本違背光流假設。所以我們又改採錄影的方式再透過個人電腦分析。

簡而言之，整個專題的過程遇到了各種瓶頸，並需要在問題之間取捨。在這個專題，我們學到如何在茫茫資料中尋找可能解決方法，也透過學長認識到 DIS 光流法，以及旋轉光流、平移光流的計算方法。