**1D Scanline matching**

**研究方法**

**各項方法的可行性分析**

能夠在『區域平坦紋理』取得較正確深度資訊的方法主要有三種，在參考數篇研究文獻後，針對各種方法的優缺點在下面進行評述：

1. Region based local matching
   * 做法：對左右眼影像使用image segmentation方法，依據顏色切割出許多segment，之後用類似pixel matching的方式(只是primitive變成region)將左右影像的region對應起來，由此得到深度資訊。
   * 優點：由於primitive是region而非pixel，matching方式又與pixel based的相似，因此matching速度會相當的快。
   * 缺點：其結果很大程度的依賴於segmentation的結果，然而，實際上的狀況雜訊干擾甚多，segmentation的結果很難預料，一旦segment切的不好，由於是不規則的二維區域，程式將非常難進行好的處理；match的結果一旦有錯，就是整個區域都錯（因此很多是採用over-segmented方法來克服此問題，但是我們的目標是解決『區域平坦紋理』，一旦進行over-segmentation，便沒有辦法在此問題上有好的表現，也就失去意義了）。
   * 需求評估：如果真的要做，應該考慮將影像縮小至一定的程度，減少segmentation的成本，做完matching後再還原到原本的大小，細節失真的方式利用Local的濾波方法來加強；由於目的只是針對平坦區域，因此縮小造成紋理複雜區域資訊遺失也沒有關係，最後主要只擷取平坦區域的深度資訊。
2. Global Matching（1D Scanline Search）
   * 做法：由於本研究是以產生能夠即時運算的演算法為目標，因此，Global方法基於時間因素大多不可行；然而有一種Global方法速度相當的快，是目前見過數種Global方法中最快的一種，是利用微分擷取影像中的edge，之後將左右影像中每個Scanline被edge切割出的區段拿出來做matching，並使用dynamic programming加快速度，而這類做法中的翹楚便是Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming。
   * 優點：其實與Region based的Global方法類似，但由於用於比對的是一維的像素區段與edge，因此容易處理，速度相當的快。
   * 缺點：由於只考慮到1維的資訊（其實還有一些二維），因此有些地方會產生著名的streaking effect，且準確度相較於其它Global方法較低；尤其特別在紋理複雜的區域，結果會較差，而且會消耗更多的時間在上面（edge很多很複雜）。
   * 需求評估：考慮到其速度，以及能夠解決我們的目標『區域平坦紋理』問題，因此列為優先考慮方法；其缺點以本演算法評估是可以接受的，因為紋理複雜區域可以使用Pixel based matching取得很好的結果，兩者可說是互補。
3. SURF特徵點Matching，並對中間區域進行interpolation
   * 做法：使用SURF在左右影像中擷取特徵點，之後將兩影像中的特徵點Matching，並計算出深度資訊，之後將各個點連結，利用Interpolation估算出中間沒有特徵點的區域深度資訊（不用Corner detect之類的方法是因為很多紋理或是物體邊緣是平滑的）。
   * 優點：擷取特徵點的演算法適合利用平行處理加速，特徵點Matchimg也是，而且不需要像其它方法一樣左右眼分別計算深度資訊然後結合，特徵點對應完即取得特徵點的深度資訊，中間的值做Interpolation即可；由於SURF出眾的抗模糊、旋轉及不同光源的能力，可以在許多雜續干擾的狀況下仍能抓到Robust的特徵點。
   * 缺點：在平滑區域的表現較差；很難單靠擷取出的特徵值就還原出物體的輪廓，邊緣多會失真，產生出的深度圖細節部分較差。
   * 需求評估：偵測出『區域平坦紋理』後，可以限縮產生特徵點的範圍，來解決速度不夠的問題；由於本研究使用的是校正後影像，所以特徵點的matching將會便得相當簡單快速，在擷取特徵點時的一些不需要的步驟或許有辦法省略；因此也在考慮範圍之內；因為是well-known的方法在測試效果時，可使用opencv或matlab函式庫快速的實作出來。
4. 透過Adaptive Gradient Cost以及Conditional Match來改善平坦區域的matching狀況

* Adaptive Gradient Cost的做法：在計算平坦區域邊緣的gradient cost的時候，放大gradient的kernel，使平坦區域接近邊緣的部分被成功match出來的範圍大幅增加；由於邊緣的disparity被正確的取得，因此在cost aggregation的時候，將正確的disparity由邊緣區域傳遞到中心區域（說不定就是belief propagation，但是沒有看過相關的論文，可能要查一下）。

Conditional Match的做法：被標識為平坦區域的地方（基本上是平坦區域的中心點）不做Pixel Match，因為做了反而會增加受到noise影響的可能性。但是標識為平坦區域的演算法必須要足夠保守，否則反而可能造成錯誤的結果。

（抓平坦區域有可能要從１維掃描的方向著手）

* 優點：兩種做法都只需要做一個對影像紋理分析的前處理，而且不需要做到很細（因為目標只是在標示出平坦區域），因此應該不需要花太多時間。
* 缺點：仍然會有些微誤差；因為平坦區域邊緣的gradient cost應該也會ambiguous，只是因為會被kernel壟罩的範圍不大，所以disparity的誤差就會被限縮；因此掃平坦區域邊緣所使用的特殊kernel也不能太大。
* 需求評估：與方法２的目標殊途同歸，有別於方法２針對平坦區域使用額外的interval matching；此方法**期望透過在Local Cost的計算規則中加入一些Global或者說較大範圍的Local的考量**，使得原本只單純考慮像素相似度cost的pixel based matching，也被摻入了針對平坦區域所特殊設計的考量。

**方法的選擇**

經過評估後基於方法原理以及優缺點上的互補性，將1D Scanline search的演算法做為優先考量，並將Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming做為主要的參考依據，並依據本研究的需求進行改量。之後在時間允許的情況下，在依序實作另外兩種方法，因此本研究對於可行性評估中的三種方法的執行實驗的優先順序如下：

**紋理分析**

為了節省Matching的時間，並有效解決紋理平坦區域的錯誤率問題，

Adaptive Cost Matching and Conditional Matching

Conditional Match實驗

* 使用固定的Gradient Threshold

效果差的原因：

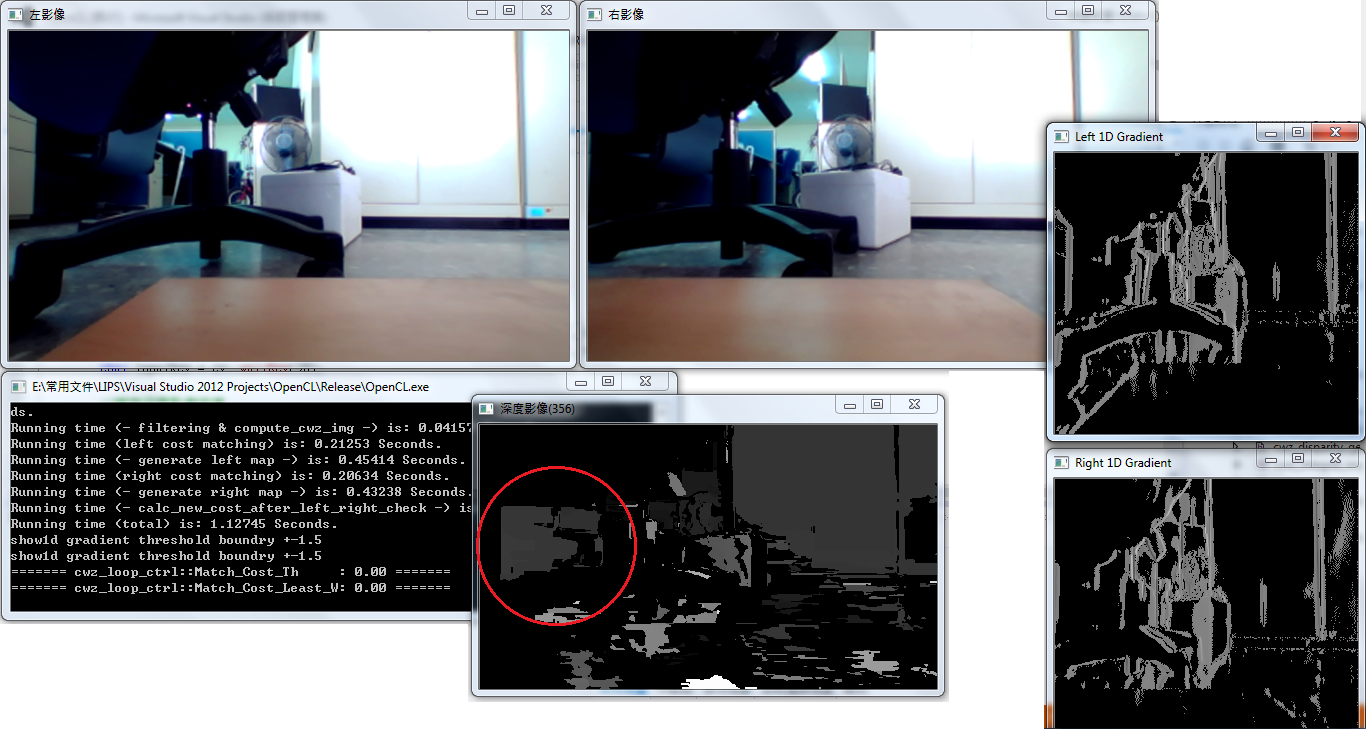
就算是Gradient值低的pixel仍很可能有所貢獻，舉幾個例子

* + - 旁邊都是Gradient高的，中間一小塊平坦區域，這平坦區域反而也變成一種容易Matching的特徵
    - Gradient低的完全不做，算出的Cost都是0，這樣就會把決定權完全交給周邊有Gradient的pixel決定，如果同區塊中有足夠Gradient的pixel很少，有極大的可能性會發生整個區塊的disparity在cost aggregate之後，全部得到完全錯誤的disparity值；在原本是不太會發生的，因為平坦區域match後的雜訊很強，反而對錯參雜，不會錯得這麼離譜。
    - 事實上就算Gradient低，也很可能是因為光源等等之類的問題，另一張影像的對應區塊說不定很清楚；因為諸如此類的可能性，Gradient低的地方做Matching算出來的cost，事實上，仍能夠產生一些正確的影響。

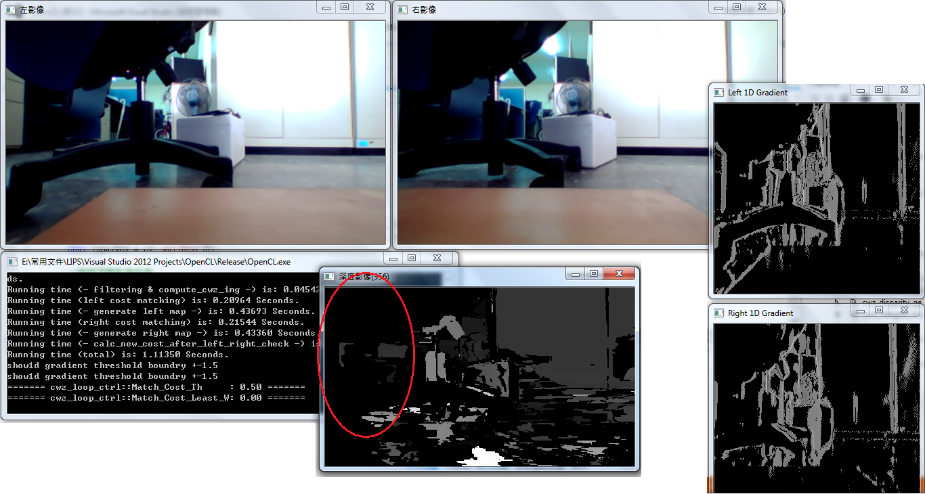
* 使用Weighted Gradient Mask

符合比例原則，也不會像只設threshold那樣武斷而造成部分區域反而錯得更嚴重，只是仍有一些觀測到的問題

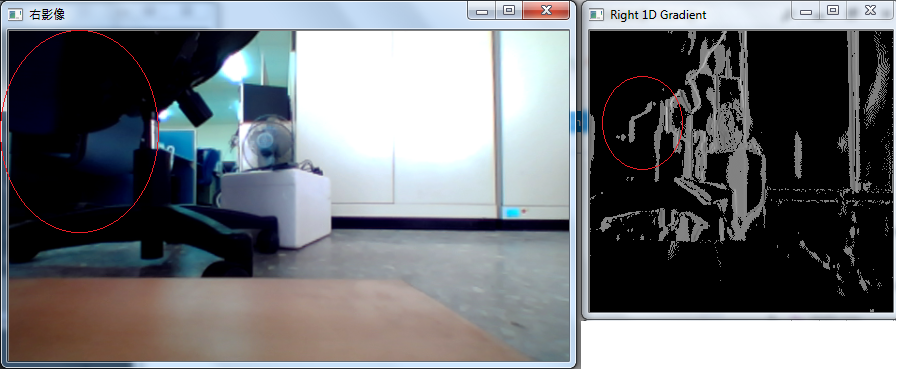
* + 背景部分由於光源不足，幾乎完全沒有gradient，在th設為0時(等於普通的cost matching狀態)，反正能夠計算出正確的disparity(因為物體很多地方仍然稜角分明，沒有gradient但是因為形狀有特徵性，所以仍然抓的到正確的disparity)。



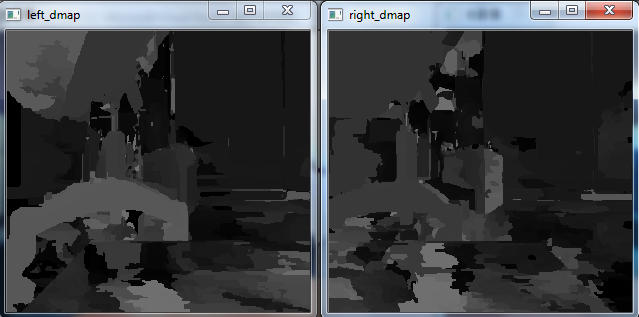
* + 然而同樣的情況，一旦使用gradient mask，那塊原本disparity正確的部分，因為整塊相關區域都缺乏Gradient，因而權重被設的非常小，整塊區域的disparity被臨近顏色相近的椅子拉走，整個disparity跑掉，這邊th才設到0.5而已。



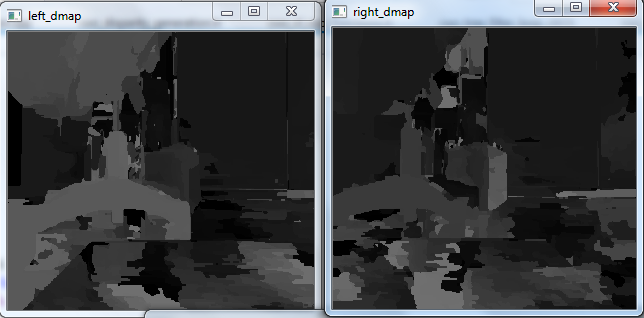
其實更詳細的原因是，那塊區域match完後，原本計算是正確的，真正的disparity是有算出較小的cost；然而權重很小，偏偏剛好右邊影像椅子很黑的地方，靠近了那塊黑色的區域，因為Tree Filter把他們連在了一起，偏偏椅子黑黑那邊擁有大的Gradient，因此導致了椅子的disparity擁有相當大的權重，被傳遞到了背景那塊區域，導致了右影像的disparity錯誤，最後left right check時被mark為unstable。



這是左右影像的disparity map(有gradient mask，背景disparity被拉走)



這是左右影像的disparity map(沒有gradient mask，椅子disparity被拉走)



這其實也間接秀出了Tree Filtering無範圍形狀限制進行濾波的缺點，不同的物體一旦顏色相近，disparity就會被互相影響。

**文獻方法萃取**

在文獻回顧之後，針對本研究的需求做了幾項調整：

1. **For Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming**

* Edge detection

在Kernel大小的部分，採用paper的方式，使用多個不同大小kernel，並將結果聯集來detect edge。

在Kernel方向的部分，我們並不只使用水平的gradient kernel，而是完整擷取了2維的edge。

目的在於區分影像的紋理複雜度，以及切分出需要1D scanline search處理的部分。

因此，預計採用3種kernel大小不同的Local Thresholding來處理這個問題。

* 只做Intra-Scanline Search（因此不做Edge Linking、Edge ordering、3D cost）

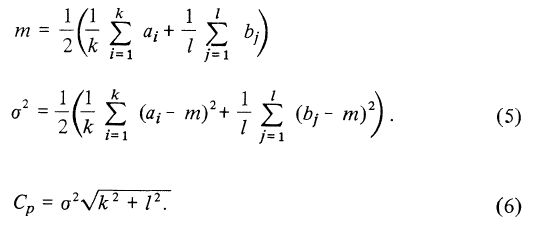
此方法是為了在做intra scanline的同時，加入考慮2維的資訊；從原理來看，此方法應該是為了增進在紋理較複雜區域進行matching時的抗雜訊能力，然而，本研究的需求目標是將紋理複雜區域交給其它演算法解決，因此決定不採用。

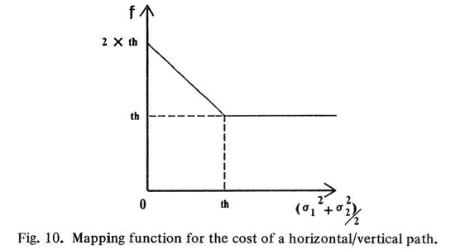
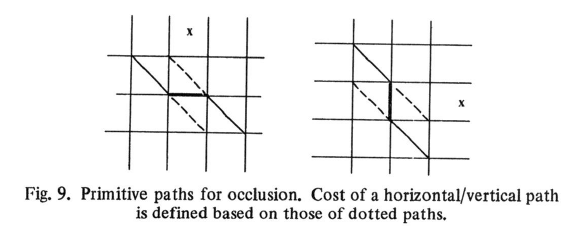
尤其是，Edge linking以及3D cost計算的演算法很大程度的增加了複雜度，可是解決的不是核心的問題。

核心問題{ 1 }：edge或segment切不好時造成整個區域都matching錯誤，大幅增加錯誤率的問題。

* 採用2D cost計算方法

包含計算Interval間相似度的規則



以及計算occlusion區域cost的規則

如果()/2的值很小，則f()會算出很大的cost；如果()/2算出的值很大，則f()會算出很小的cost；具體多大多小受th的值所影響。



1. **For Region-Tree Based Stereo Using Dynamic Programming Optimization**

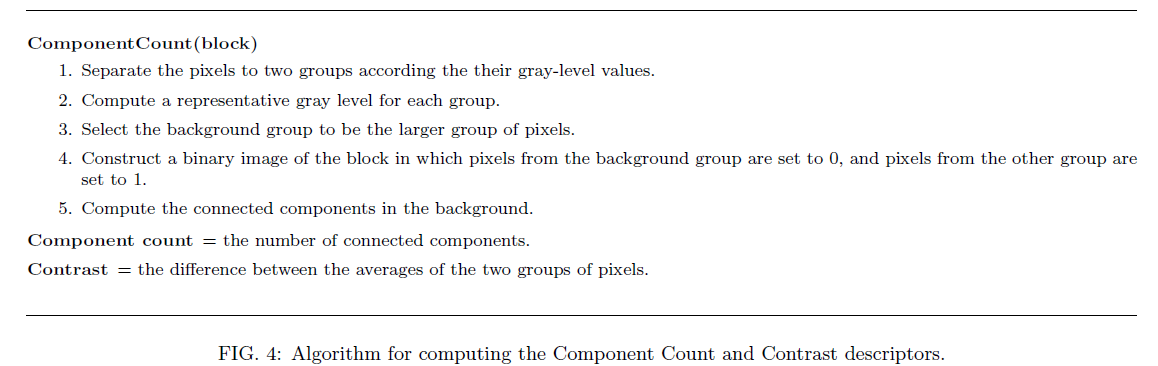
* (此方案最後棄置，因為在edge情況差的時候，無法避免streaking effect，使用over segmented tree應該比較有用，但要額外做segmentation太花時間)使用over segmented 1D “pixel intervals”做為Tree的primitive unit

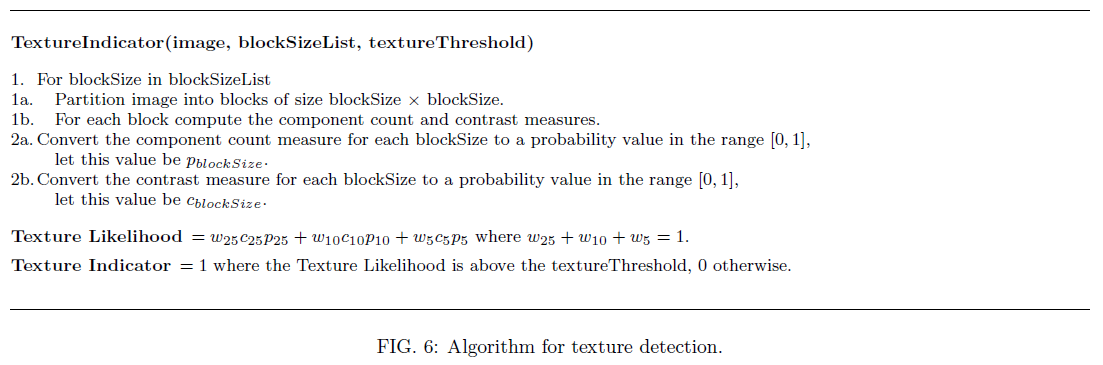
因為2維region在matching上較為複雜且困難，並且2維的Image segmentation也需要消耗較多的時間，因此，本研究才會採取1D Scanline Search的方式；由於1D Scanline Search的前置處理是使用1維的region（也就是1D pixel interval），因此region tree的建置將採用1D pixel intervals做為node。

然而，為了不遺漏任何edge資訊(見核心問題1)，在擷取edge時，會產生一個衍生的edge expand的問題(見核心問題3)，因此如果真的要建1D region tree，則需要特殊處理。

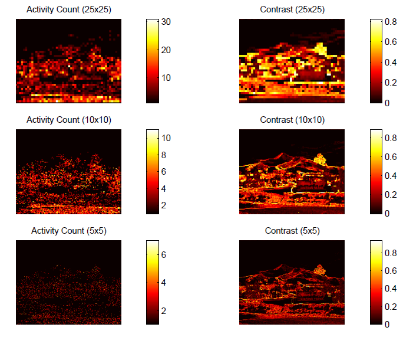
1. **Detection of Textured Areas in Images Using a Disorganization Indicator Based on Component Counts**

* 用block罩住pixel，將block內的pixel區分成兩個group，之後取得兩個Descriptor：”Component count”以及”Contrast”





Result



1. **A Robust Stereo Matching Method for Low Texture Stereo Images**

* 前面提到一種方法：

只採用gradient高或分析出來是有texture區域的pixel所計算出的disparity值；這個方法的主要問題在於面臨某些狀況時，可以用的pixel太少，不足以產生足夠的disparity資訊。

* [5]使用[8]中的”Quasi-Dense” Match Propagation方法，將Strong edge邊的pixel的disparity傳遞到low texture區域；這方法的問題點在於，物體邊緣要有足夠的edge讓演算法detect到，否則一樣沒辦法。

（這點我意見保留，如果連edge都找不到，應該怎麼樣都吃土吧？）

* 使用微修改的cosine function做為Cost Function；有提到SAD、STAD、NCC等方法再time complexity跟抗noise的能力上權衡，最後選出這個改過的cosine function（相似於NCC，但不用開平方根，故用之）。

(注：g是用sobel filter分別對rgb三個通道做完後的結果，實際上就是各自的gradient)



* 看最下面的Table1, 2來看，此演算法類似簡化版的semi-global；只是此paper使用edge map來讓cost只在edge內做aggregate，像是local版本的semi-global，與本研究所構想的方法似乎不一樣。
* 此paper主要是依據segmentation based cost aggregation來做的，因為segmentation太耗時間，因此，這個研究使用了edge map來做邊界內水平、垂直的cost aggregation；看前兩頁的敘述，此篇paper著墨於改善cost aggregation的方法，比較對象主要是moving average把邊界吃掉、Adaptive window太耗時間等等老問題。

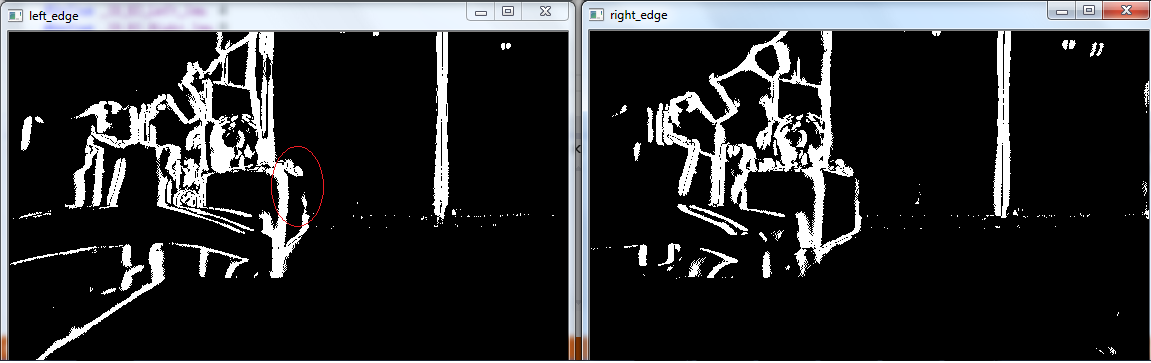
1. **OpenCL 2.0 Specification**

用device enqueue的功能似乎可以加速樹狀結構的平行效能，如果真的可以，可以套用在Edge Match時的Update DP Table和Tree Filtering的Cost Aggregation(說不定樹的建構也可以)。

**核心問題**

1. Segment取不好，很可能導致在做Intra-Scanline Search的時候，數個scanline的intervals都得到錯誤的結果

如下圖紅色圈起來的部分



兩個區域的漸層相接處很接近，但其實是兩個不同的Segment

假設像素值如下：

顏色10 10 10 10 10 10 10 9 | 7 6 6 5 5 5 6



左segment | 右segment

在上圖中（從箱子的邊緣放大擷取的像素），如果kernel太小，則看不出差異。

1. Edge expand的問題

問題敘述：edge範圍會擴大(下面mark部分0為平坦區域、**1**為edge區域)

Mark 0 0 0 0 0 **1 1 1** | **1 1 1** 0 0 0 0

顏色10 10 10 10 10 10 10 9 | 7 6 6 5 5 5 6



左segment | 右segment

此問題會產生兩種後續影響

* 1. 複雜區域平坦區域幾乎都會被吃掉，大部分都會被mark成edge
  2. 在建構1D region tree的時候，沒法直接使用edge map的資訊；在平坦區域還可以針對連續被mark為edge的部分處理，複雜區域幾乎沒有辦法。

**核心問題的解決方法**

1. 使用大小不同的kernel來偵測edge

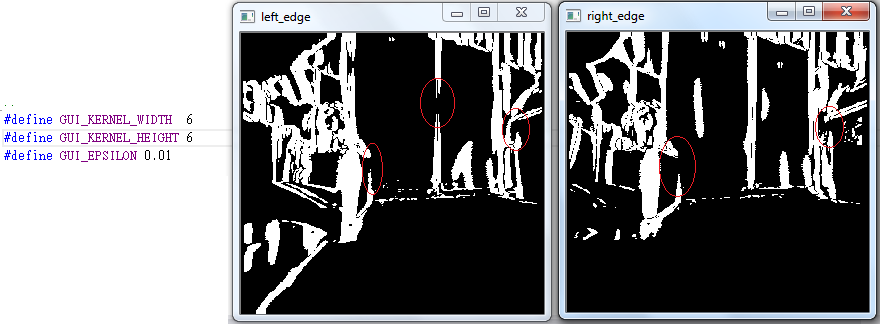
此問題是因為兩segment的相接處是漸層接近，因此如果小的kernel很可能會偵測不出來有edge，要換成大的kernel才能放大出兩個segment的差別。

但是會帶來衍生的edge expand問題。這個問題主要會對紋理複雜區域的segment切割造成影像，對本研究來說是可以接受的。然而，如果要建構1D region tree，則必須要針對此問題處理，因此將此問題歸類為核心問題{ 3 }。

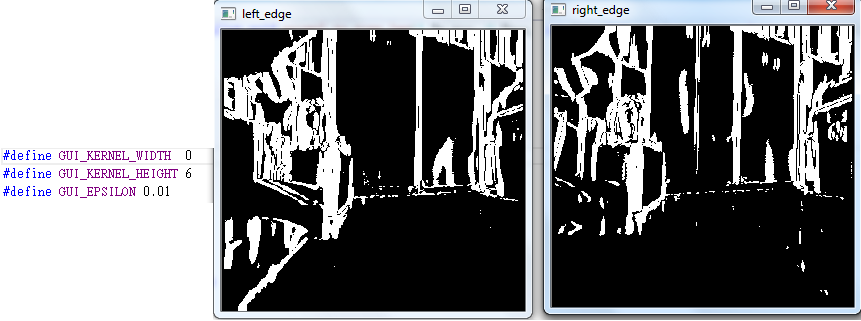
1. 改變濾波的kernel形狀

如果邊界顏色相近，在濾波之後便很可能造成Edge被吃掉的問題（因為Edge Preserving的濾波是靠顏色來分辨的），因此為了避免垂直的邊界被吃掉，改使用水平的Kernel做濾波。

下面影像是Guided Image Filtering (13\*13)的結果



下面影像是Guided Image Filtering (1\*13)的結果



1. 解決edge expand問題
2. 平坦區域特殊處理
3. 紋理複雜區域限制kernel的size

**文獻回顧**

參考自Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming

* Intra Scanline Search:

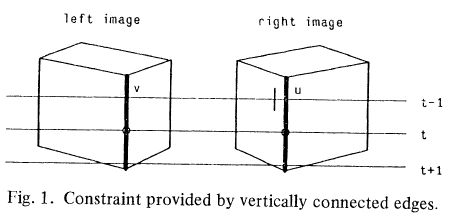
左右scanline中的edge做matching，以及被edge切開的平坦區域的matching (**edge delimited intervals**)。

理論上intra scanline就足夠了，在現實上會有很多ambiguity會影響，因此需要Inter-scanline constraint

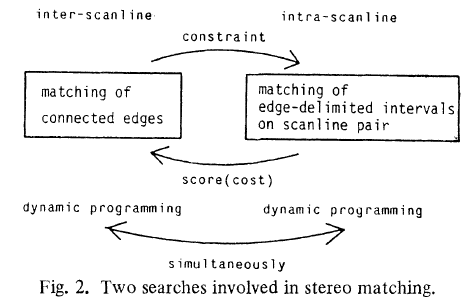
* Inter Scanline Consistency Constraint:

以下圖來解釋；如果left image的edge v跟right image的edge u在scanline t沒有被match在一起, 則在其它的scanline如t-1跟t+1，這兩個edge都不應該被match在一起

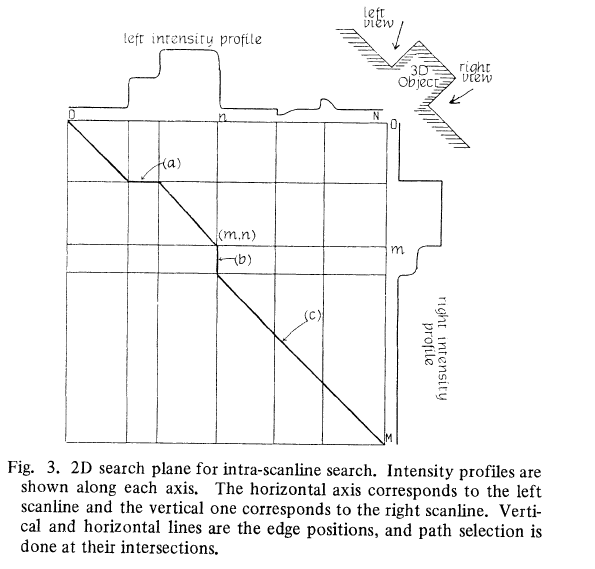
(we call this property inter-scanline consistency constraint.)。



文獻回顧的兩篇論文都沒有得到最佳解，因此此論文要探討如何利用DP在inter-scanline的限制下同時進行intra scanline search。



Intra-scanline Search的問題可以在一個2D space被解決

Cost定義：

* d(m, k): 從點k到點m的cost
* D(m, k): 從點k到點m的**最佳cost**
* D(m):當node k為原點(0,0)時, 以D(m)表示之
* Obviously d(m, k) >= D(m, k)當d(m, k)為最佳解時d(m, k) ≡ D(m, k)

由上面條件我們可以定義

D(m) =

從原點到m的最短距離 D(m)

|| ||

前一點到m的距離 d(m, m-i)

**+** +

原點到前一點的最短距離 D(m - i)

當D( O ) = 0時

m = (m, n) 、 i = (i, j)

0 <= I <= m 、 0 <= j <=n

i + j ≠ 0 、 O = (0, 0)

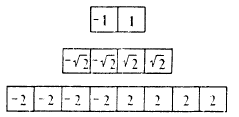
Vector i = (i, j)表示了到達點m時，必須經過的primitive path

* **圖中(a):** 當primitive path為水平線時，表示在left image的那個edge沒有出現在right image中**(occlusion in right image)**，因此被跳過。
* **圖中(b):** 當primitive path為垂直線時，表示在right image的那個edge沒有出現在left image中**(occlusion in left image)**，因此被跳過。
* **圖中(c): 當 i > 1 and/or j > 1 表示primitive path可能跳過了 i – 1 and/or j – 1條在right and/or left scanline上的edge**

以上這三種狀況，皆是intra-scanline search時，找不到對應edge時的狀況，可能是因為雜訊或其它原因造成的

（時間複雜度計算省略，需要時請見paper內文第三頁）

在matching時，不用把一個edge切成doublet來做，ㄧ個段interval其實就等於兩個對應的half edge；用interval來做如果有N個edge，就只有N-1個interval這也是最簡化的表示法，如果把edge切成doublet的話，反而會有N\*2個單位，無端增加複雜度。

* Edge detect的方式：針對影像，用三種不同的operator來detect edge，當小範圍的operator在某地方沒有擷取到edge時，才接著用第二個operator，再沒有才用第三個，避免同一個地方重複擷取到同個edge

注: 本paper對d(m,k)的真正定義在Section IV-B

Matching特徵

1. 整個區域的mean
2. 整個區域的variance(可能沒用, 因為應該是平坦區域)
3. 面積
4. 整個區域的數量(前兩項指標的權重要比較高, 前兩項分數相似才比第三項)
5. 左右edge強度的比較(edge有分正負)

擷取特徵流程

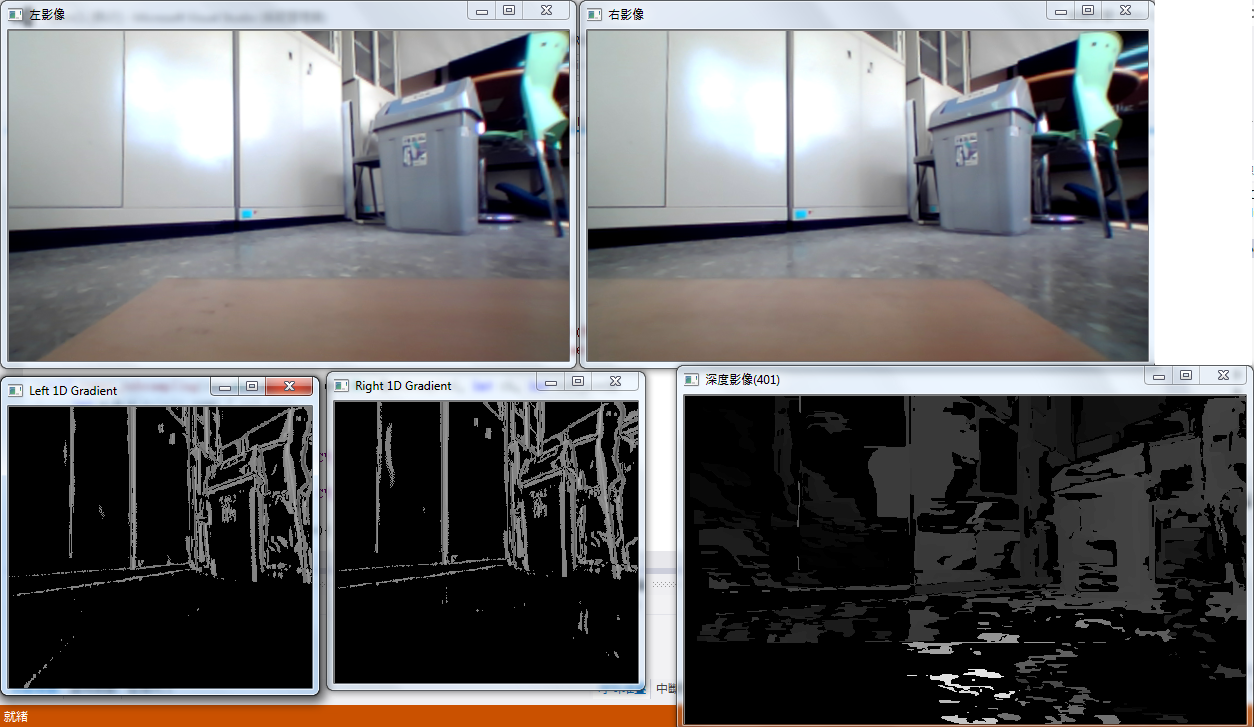
以下括弧部分表示方法有多種選擇, 撰寫程式時必須考慮到可變動性

1. 用(切edge的方法)取得一灰階影像, 影像中顏色全黑部分代表平坦區域, 有顏色部分{依據顏色強弱決定edge強度, 若, 只有單一白色代表edge}
2. (Linking edge)
3. Edge的global排序
4. 用計算cost的方法開始跑intra and inter scanline search

連續影像

* 這種dynamic programming方式，會先取得3D的edge
* 第一張取得整張影像的edge後，可以套用同樣的演算法，找尋下一張影像的相似edge
* 類似於optical flow的應用，由於各種edge還是有些不同
* 可否讓電腦利用上張影像的edge，比對下張影像的edge，來檢查某些比較weak的edge
* 如果該處gradient的pattern，跟之前影像gradient的pattern相似(也可加入分界線兩側的像素平均值為考率標準之ㄧ)
* 則鑒於前影像是有的edge，讓我們判斷該處也應該有edge，只是在這張影像較弱，但由於經驗，所以仍然將該點取edge，直到此edge在周圍再也找不到pattern類似的edge為止

理想針對目標(尤演算法先偵測出，哪邊適合用scanline search，哪邊適合用pixel based matching，哪邊應該混用)



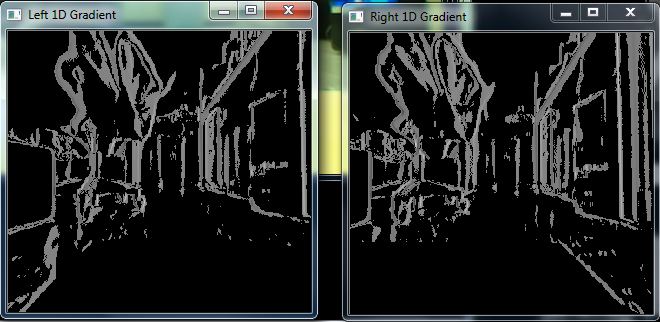
**研究動機及目標**

由於Pixel based的matching方式，不管使用何種方法濾波（Fast cost volume, Tree filtering, semi-global）在面對『紋理平坦區域』甚至是特殊的『紋理重複性區域』時，由於沒有考慮到全域的資訊，因此都很難計算出正確的深度資訊，並且，還浪費很多時間在這些『紋理平坦區域』的深度計算上（因為基本上做出來幾乎都是錯的，所以可以說是做白功）。

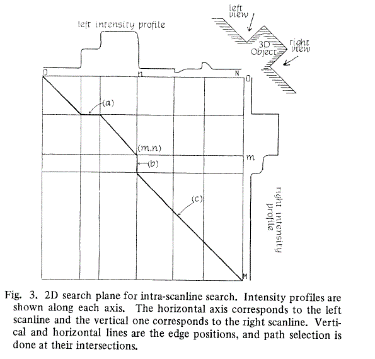
在閱讀過相關文獻後，發現可能可以解決『紋理平坦區域』的方式主要有『Region based local matching』、『Global Matching』以及『擷取特徵點做Matching之後，對中間的區域深度利用interpolation進行估算』等三種主要的方式。

因此本研究期望，參考上述這些方式，借由混和Pixel based的local matching的方式，產生出一個混合演算法；首先針對得到的影像做一次快速的分析，分析出影像紋理的分佈，將影像區分為紋理複雜、普通、平坦等三種不同的區域，之後針對演算法各自的優勢，進行工作分配，中間難以區分的部份，則依據效能需求，可以只做Local matching也可以一起做，並中和結果。

可以考慮做一個假2D的matching (符合inter scanline consistency), 以下圖的箱子為例, 如果上半部分只以純1維來match是有可能match錯誤的, 但如果換成2維正方形, 則就不會match錯誤



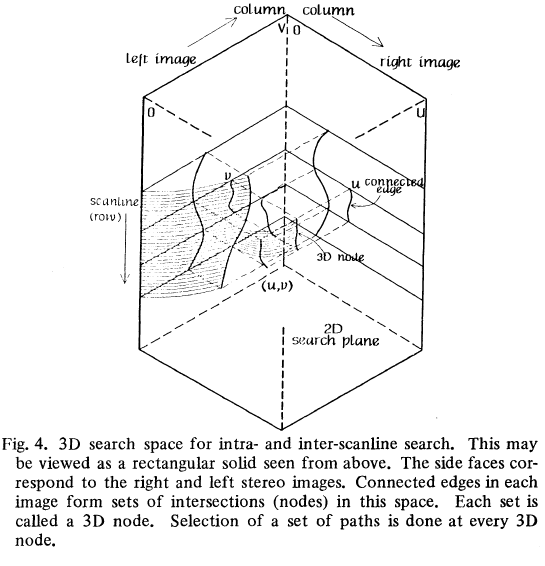
1D的問題可以在2D space被解決



假2D可以在3D space被解決, 上面那張2D space是一條scanline的intra scanline search

當考慮到整張影像時, 就可以被畫成像右邊的方塊一樣的3D space

在2D space每次要前進一個node時, 必須要計算optimal path

如果一張2D space在計算cost的時候, 疊在其上方與下方的其它2D space也能依據一個 (edge的相似度權重) 來影響其cost的計算的話, 則相信可以達到一個更好的效果。

edge的相似度權重計算方式：

問題說明

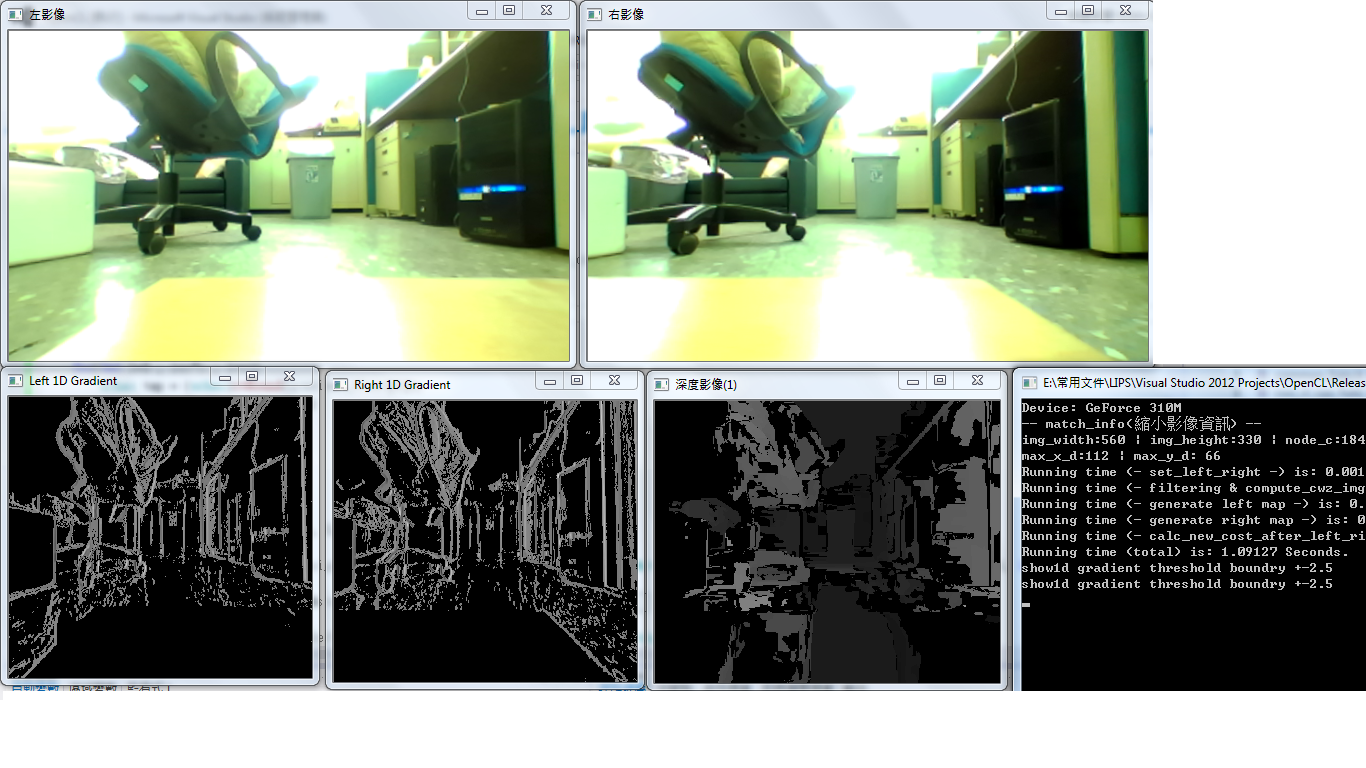
如果兩個被match的區域得到的disparity不大, 而區域長度卻差很多, 可能代表的意義: 一邊有某個edge沒切到所以有一個區域是另一邊兩個或多個區域的結合, 剛好在邊緣rectification沒有做很好導致沒對齊, 中間有障礙物擋到造成edge被切的很不一樣(如椅子擋住沙發的edge), 兩者本身就是不同區域被錯誤match在一起

當左影像有切出edge, 但右影像沒有的狀況,

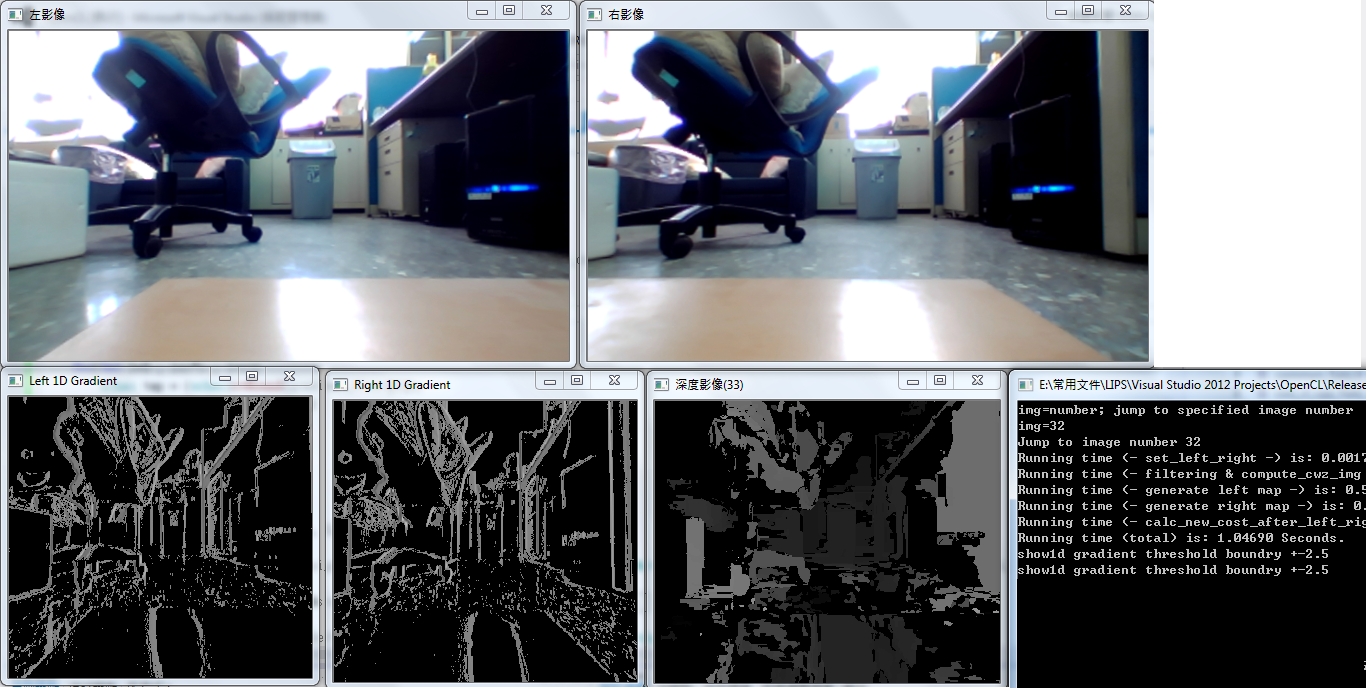
當左影像的沙發被椅子擋住的地方與右影像差距大的時候, 取特徵的機制, 會嚴重影響到matching結果是否正確

問題說明

(理想狀態)



(有問題的案例)



此處的數台電腦將會因segment沒有取好, 而matching到不正確的結果

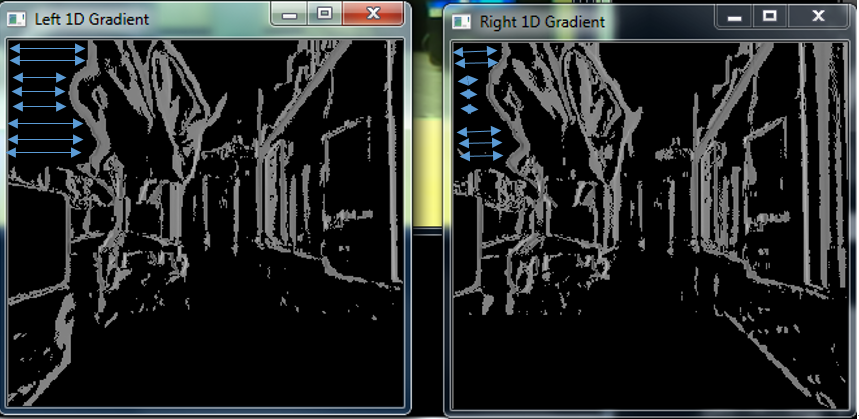
(兩台電腦會得到同一個disparity)

因此必須做光源處理，把pixel為255的部分忽略

問題說明

(正面) 下面的箱子可以由右邊界推算出正確disparity, 因為**右邊界跟左邊那塊平坦區域是屬於同一物體**

(反面) 但箱子上方的區塊的右邊界跟左邊平坦區域不屬於同個物體, **左邊平坦區域是被擋到的背景, 因為反光變白而形成一個平坦的區域**, 如果以inter, scanline search這樣去算, 左邊那塊背景的disparity會被前面的椅子混淆



問題描述

當edge很少時，有可能因為沒得選擇而被match在一起，如下方推車的板子

因此還是要有一個threshold，一旦兩個edge算出的cost在threshold之上，則兩者不能被match在一起

(threshold的值定法，可由程式自動收集正確match的cost，跟不正確match的cost，然後找出一個適合的臨界點。)

判斷正不正確的方式應該容易想，但由於程式需要偵錯，所以在學習過程時，需要有一個可以視覺化的偵錯程式，每當輸入一張影像，會循序將被match在一起的edge變色（紅是錯，綠是對），並可以由人為輸入來更正正確或錯誤的tag

