FPGA Implementation of Visual Odometry by Using High-Level Synthesis

使用FPGA及高階合成技術應用於視覺里程計

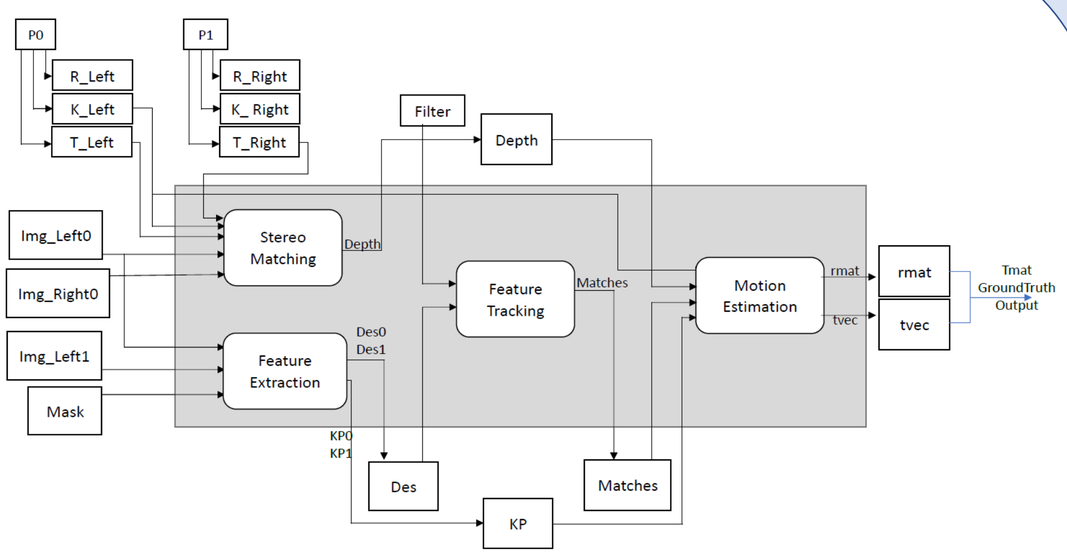
組別： A288　　指導教授： 賴瑾　　組員：林奕杰、李承澔、鄧文瑜、林妤諠、郭朝恩

摘要

高階合成(High Level Synthesis，HLS)為一種將高階語言(C/C++、SystemC) 轉換成RTL硬體電路的自動化設計技術。由於傳統FPGA特定應用加速器的開發普遍以Verilog、VHDL等硬體描述語言於RTL層級進行開發。但隨著近年來晶片的集成度提升、及開發周期的縮短，因此高階合成於晶片設計領域已愈趨重要。

本專題使用Xilinx Vitis HLS平台搭配Xilinx Alveo™ U50加速卡進行開發。我們使用電腦視覺、系統設計、硬體設計、HLS開發技術等相關知識，將視覺里程計（Visual Odometry，VO）演算法實現於FPGA上。

演算法架構

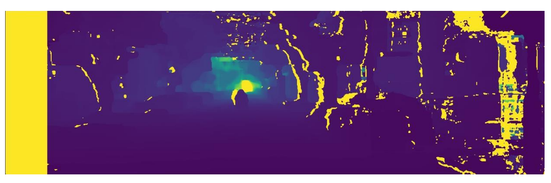


圖一 Visual Odometry 演算法架構圖

VO演算法由四個子算法建構而成，分別為深度估計(Stereo Matching)、特徵點擷取 (Feature Extraction)、特徵點配對 (Feature Tracking)、以及位移估測 (Motion Estimation)(圖一)。

圖二 雙目視覺圖(Img_Left0與Img_Right0)

Stereo Matching由相機參數(P0、P1)分析同一時間點的左右圖像 (Img_Left0、Img_Right0)，計算出視差 (Disparity)，進而得到視覺深度 (Depth)。

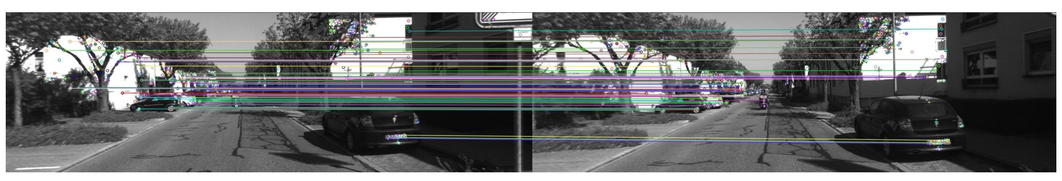


圖三 視覺深度圖(Depth)

圖四 提取之特徵點(Extracted Keypoints)

Feature Extraction則對左相機位移前後擷取的連續圖像(Img_Left0、Img_Left1)提取特徵點(KP0、KP1)以及其描述子(Des0、Des1)。

提取特徵點後，Feature Tracking以描述子比對特徵點從而形成配對 (Matches)，並以給定閾值(Filter)去除偏差配對。



圖五 特徵點形成之配對(Matches)

最後Motion Estimation利用深度、配對、相機參數，計算出旋轉矩陣(rmat)及平移向量(tvec)，並合成軌跡矩陣(Tmat)。

實作流程與硬體架構

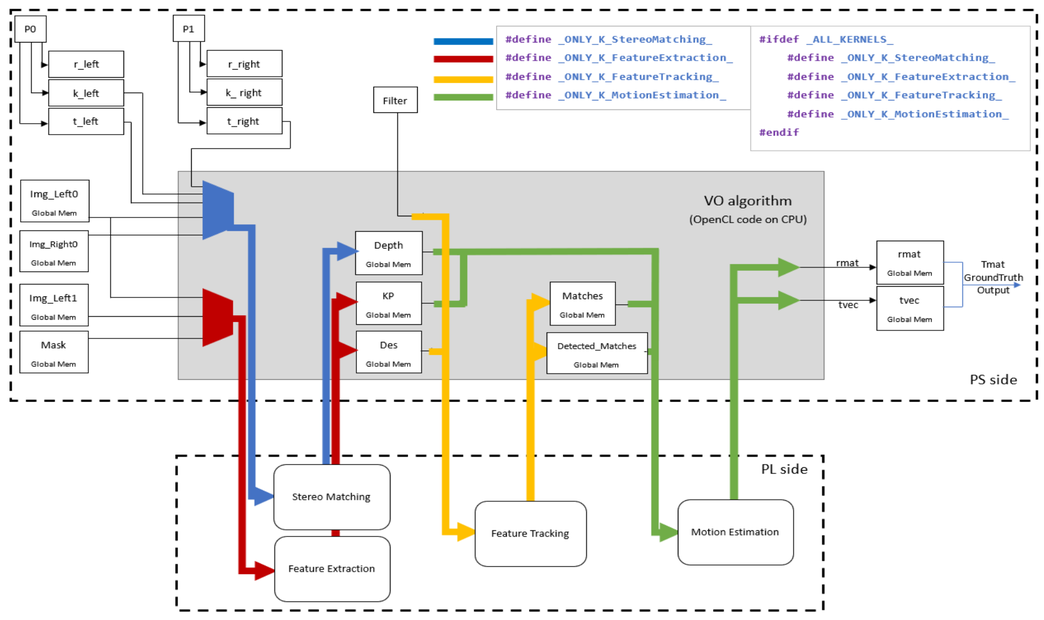
將VO的四個子算法實作成為四個Kernel Function，並依以下流程逐步完成Kernel開發及驗證。

1. 首先基於OpenCV C++ 4.4.0 Library，將四個子算法改寫為未呼叫外部函式庫的self-contained C code，並透過Vitis HLS以C-sim驗證此階段結果。
2. 因Vitis的HLS v++編譯器尚無法將前一階段的self-contained C code轉換為RTL硬體電路。

實作流程與硬體架構

此階段需改寫原先的資料存儲結構、去除雙重指標、使用靜態分配記憶體，及重新定義Kernel Function間的介面I/O，使其成為synthesizable HLS code。以Vitis HLS co-sim驗證此階段結果以及RTL 波形。

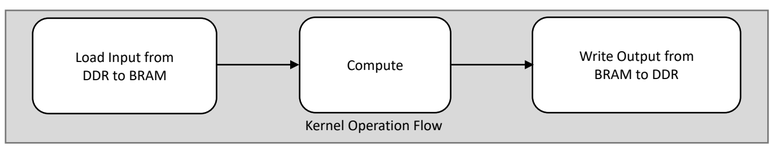
3. 以HLS code合成bitstream，寫入FPGA驗證實際運行結果，並與self-contained C code 結果比較。
4. 持續優化FPGA的資源使用率及運行速度。



圖六 硬體架構及macro

我們利用OpenCL API溝通Host PS side與FPGA Kernel PL side，輔助驗證結果，進而達成PS/PL side的協同運算。

在Host PS side中定義了四種Kernel編譯模式(圖六)。每種模式都對應一條運算路徑，將單獨Kernel置於PL side運行。



圖七 Kernel運作流程

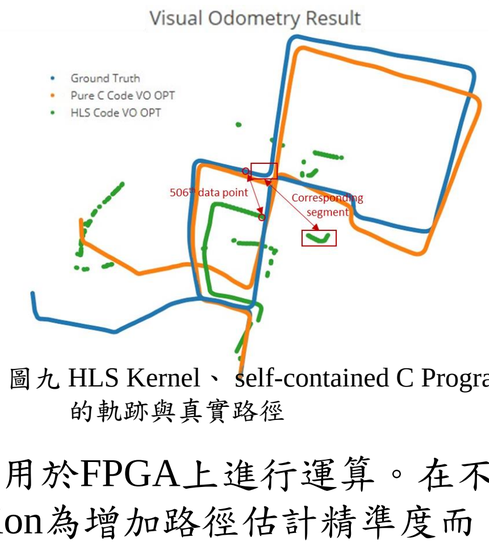
意即Host會先將此Kernel所需的Input置於DDR memory，此Kernel在被OpenCL啟動後，將Input由DDR讀入高速記憶體(BRAM)，並進行計算。運算完成後將Output結果存回DDR(圖七)。此模式用於驗證單獨Kernel運算的正確性。

最後透過定義_ALL_KERNELS_同時啟用四條運算路徑，即可達成四種Kernel同時運行在PL side的目標。

結果與結論

Kernel	FPGA time (Baseline Version)	FPGA time (Optimized Version)	CPU time
StereoMatching	8290.4ms(216.3Mhz)	7880ms(190.3Mhz)	439.3ms
FeatureExtraction	910.3ms(300Mhz)	321.6ms(104.7Mhz)	58.6 ms
FeatureTracking	11.6ms(300Mhz)	5.1ms(300 Mhz)	4 ms
MotionEstimation	60.6ms(300Mhz)	408.6ms(258.1 Mhz)	2.5 ms

圖八 各Kernel不同版本運行時間與CPU運行時間



圖九 HLS Kernel、 self-contained C Program 的軌跡與真實路徑

本次實作成功將VO演算法應用於FPGA上進行運算。在不斷迭代優化後，除Motion Estimation為增加路徑估計精準度而增加用時，已大幅縮短整體FPGA的運行時間。儘管其運行時間仍不及CPU，但與最初版的運行時間相比已有顯著進步。

由於self-contained C code計算出的軌跡與真實路徑(Ground Truth)間的偏差實屬VO演算法已知難以避免的問題。因此我們著重比較Kernel與self-contained C Program計算出的軌跡，可發現前506個iterations中，兩者具有相似的軌跡，說明我們的系統在較短的路徑內可以做出相對準確的軌跡估計。

在此次專題實作中，即便硬體資源與CPU平台相比較為貧乏許多，我們仍成功在FPGA上開發出整套系統。我們學習到如何應用HLS技術於系統的設計開發、驗證硬體行為，並找到可能導致結果不如預期的原因，未來將進一步持續平行優化。