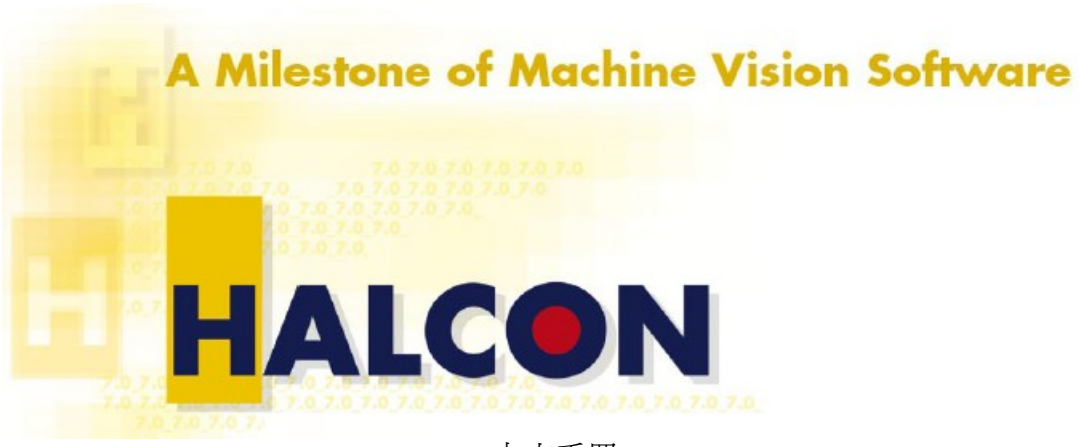


中文手冊

快速指引

目錄

第一章

Introducing HALCON

HALCON 是當今machine vision技術的代表，它總是以最新科技爲基礎，提供了現今市場中最強大的vision library。不論您的工作爲何，HALCON都能快速而精確的解決問題。

Vision Development Environment
一個專業的影像處理工具不能只具有影像處理功能。影像處理只是整個工作的其中一環，還要有其他軟體功能，像是程序控制，資料處理，硬體方面還有照明和取像設備，以及其他硬體機構等等。一個影像處理系統除了要易於使用，還必須能夠以富有彈性的方式將上述功能加入開發的流程之中。
爲此，HALOCN考量到各種重要的層面：
透過一個互動式的工具HDevelop快速達成軟體開發的工作，藉由程式碼的輸出，可以輕易的和標準的軟體開發工具，例如Micrisoft Visual C++ 整合。
問題導向式文件涵蓋了所有層次，包括取得重點資訊到進階的細項討論。
文件內容和上千個範例程式連接，讓使用者以最直覺的方式了解解決之道，各種範例還可以作爲開發的樣本以節省時間。
此外，HALCON也提供了開放的介面以便進行有效率的資料交換，整合自訂的運算子，以及周邊的硬體設備。

Vision Library
HALCON滿足了專業vision library的各種要求：
它包含了各種標準到高階的功能，從基本的影像處理，取像，到高階的功能shape- based matching等等。除了針對影像，HALCON還提供了機器視覺應用中常用的功能，例如socket通訊以及rs232的溝通，檔案存取，資料分析，算數運算，或是分類等等。富有彈性的平行計算方式可在多處理器的硬體上提昇速度。一般的使用者看不到系統是用HALCON開發的，安裝時需要的資源也很少，非常適合OEM廠商。

1.1 Key Features

最先進的科技
除了提供了完整的標準 machine vision功能，還有一系列優異的功能，例如，3D相機校正，形狀以及原件導向的匹配，次像元精度的物體擷取，計算，利用雙像立體量測，任意形狀的ROI，以及更多的功能。此外，某些library具有的功能，像是morphology，整合在HALCON之下，其計算效能比起其他產品提高近百倍之譜，也提供了更多的使用彈性。

能符合所有應用的單一軟體
HALCON包含了１１００多種各類功能，可用於任何和影像相關的研究以及產品開發，全世界已有許多使用者利用HALOCN解決了machine vision方面的問題。

保障您的投資
選擇HALCON，您選擇了獨立性。需要轉換一個作業平台？HALCON支援了許多作業平台，從微軟的Windows NT/2000/XP，Linux，到UNIX。要將程式由C++改爲C#來開發?HALCON可

用於多種程式語言與開發環境。您的計算需求日增，需要更有力的計算工具？換到一台多處理器的電腦上，HALCON可以**自動進行平行處理**。還有，您可以自行選擇想要使用的取像設備，HALCON已經提供了多種即用的取像設備連結介面，例如
analog，digital，IEEE1394，CameraLink等等不同的取像設備。

快速建立雛形

在許多狀況下，您必須在最短的時間內決定問題的解決方案。HALCON提供的HDevelop是一個互動式的快速發展工具，具有成熟的程式編譯以及除錯功能，同時還提供了可能使用的運算建議，並且自動顯示計算結果。藉由整合的工具，您可以檢視影像以及任意階段的計算結果，得以快速的決定各種參數。

開放的架構

HALCON提供了市場上最強大的 vision library，但它卻不是一個封閉的套件。它有個開放性的架構，也就是說您可以新增自己的計算功能。此外，如果您想要使用的frame grabber目前HALCON尚未支援，您還是可以直接讀取記憶體或是自行另外開發一個介面來整合。

1.2 誰應該用 HALCON？

簡單的說，就是所有用到機器視覺軟體的人。
HALCON可說是爲了下列的人們設計的：
OEMs設備商，例如晶片或印刷檢驗機；軟體開發者，例如車牌辨認和細胞分析；客戶指定功能之機器視覺設備的系統整合者。對於研究機構，大學，完整的計算功能使其或益良多，尤其是HDevelop這個互動式的工具用於雛形開發上有極佳的成效。

1.3 您需要的知識

影像處理

當然，您對影像處理的專業術語和標準方法越熟悉，越容易用HALCON來解決問題。在第 5 章簡要的介紹了這些方法，也說明了在HALCON中要如何使用。您也可以由第 3 章或是第 4 章著手，其中說明了不同工業以及應用的例子以及使用的計算方式。

程式設計

如果您要在某個程式語言中使用HALCON，您就必須熟悉該種語言以及相關工具。 HALCON Programmer’s Guide 說明了HALCON的語言介面，資料型態和類別，呼叫運算子的方法等等。

作業系統

您需要對目前使用的作業系統有基本的了解，才能安裝HALCON以及其license等等。

1.4 開始使用 HALCON

想對HALCON有所認識，我們建議您安裝demo版的HDevelop，不必使用license。它一樣提供了所有的影像計算功能，唯一的限制是不能連接取像介面，撰寫的程式也不能儲存。有了demo版的HDevelop，您就可以踏入HALCON的世界，透過範例程式，了解它的能力。2.2節有簡要的介紹。您可以由MVTec 的網站或是CD來安裝，要注意的是，CD之中並不包含所有的版本。如果要評估完整的功能和測試，例如包含取像設備的運作，或是在程式之中加入HALCON的功能，您可以向當地的代理商(新亞洲儀器公司)索取當月份的試用license。License安裝與使用的詳細說明請參閱 HALCON Installation Guide.

1.4.1如何從CD安裝demo版的 HALCON

在HALCON Installation Guide.中有詳細說明

1.4.1.1 Windows NT/2000/XP Platforms

要進行下列步驟，您必須有 **administrator 權限**
將 CD 置入 CD-ROM 中。如果安裝沒有自動開始，請由CD目錄的 nt-x86中，執行Setup.exe。安裝程式會自動說明相關事宜，並且引導您完成安裝程序。Setup之中會讓您選擇安裝型式，在此請選擇安裝Demo版，安裝完成後，你可以直接由Windows的程式集中執行HDevelop，不必再經過其他額外設定。

1.4.1.2 UNIX 平台

將CD置入CD-ROM中，並將其連接到系統中。您可能需要root 權限才能完成硬體連接設定。由CD根目錄執行 shell script **install-unix** 。這個 script 會問您想要安裝的路徑，您也可以選擇想要安裝的原件，不過您只要依照程式的建議，選用預設值即可。安裝後要設定環境變數，安裝程式會告知您要做的動作。要啓動展示版的HDevelop，由shell中執行之。

1.4.2如何由網際網路上安裝HALCON的展示版

MVTec的下載區位於 http://www.mvtec.com/download ，您必須先註冊才能下載。 選擇demo version以及您的作業系統，再依照指引操作。在 Windows NT/2000/XP 下，由"開始""程式集"下可以找到展示版的HDevelop來執行。 UNIX ，由 shell中執行。

1.5 如何取得更多資訊

藉由本手冊

第二章中說明了如何在HDevelop中發展程式，或是在支援的程式語言中使用HALCON。
第三以及第四章分別以machine vision industry 以及 application area 兩種分類方式來介紹HALCON的範例程式
第五章敘述了machine vision中主要的方法，以及在HALCON之中要怎樣使用這些功能，以及範例程式。

HALCON Installation Guide
這本手冊說明了不同的安裝授權方式，以及安裝，升級，解除安裝的詳細方法。

HDevelop User’ s Manual
這本書詳細的介紹了HDevelop這個程式設計環境，書中解釋了它的圖形介面，撰寫程式使用的語法，輸出成其他語言，等等。

HALCON Programmer’ s Guide
如果您要在C++或是VB等等程式語言中使用HALOCN，請參考這本手冊，其中說明了相關的介面，資料型態，類別等等。

Extension Package Programmer’ s Manual
這本手冊說明了如何將自訂或自行設計的計算功能加入HALCON之中。

Frame Grabber Integration Programmer’ s Manual
如果您使用的取像器HALCON尚未支援，這本手冊說明了開發取像系統連接介面的方法。

Application Guide
這本手冊目前包含了三個部分

1.Application Note on Shape-Based Matching: 說明了如何使用HALCON的shape-based matching 來找尋物體。內容以影像中的單一 model 爲例，說明以其進行次像元精度定位的方法。
2.Application Note on 3D Machine Vision: 這段落說明了 3D machine vision 的原理與方法。

3.Application Note on Image Acquisition: 這段落說明了取像介面的基本以及進階設定方式，以及各種計時模式等等。

Reference Manuals
包含了HALCON所有運算子的說明，針對 HDevelop, C++, COM, and C 等等四種語言各有其獨立的文件。

Release Notes
如果您有使用舊版的HALCON，請抽空看一看。這份文件位於您的HALCON安裝目錄下，名爲 notes.html

Example Programs
HALCON提供了爲數可觀的範例程式，除了HDevelop的程式，也提供了其他程式語言例如VC或VB等等的程式範例。這些程式位於安裝目錄下的 examples 目錄中。其中有許多的例子在手冊之中都會提到。

Support
對HALCON有任何問題，請聯絡當地代理商(新亞洲儀器公司)
http://www.idsvision.com.tw/

Training Seminars
某些代理商會舉辦 training seminars ，相關消息可以由各代理商或是 HALCON 網站
新亞洲儀器公司 http://www.idsvision.com.tw/
MVTec公司 http://www.mvtec.com/halcon/support/training.html 查詢

第二章

如何用 HALCON 來開發程式

您可以用不同的方式以HALCON的功能來建構一個應用程式。但是最有效率的方式是採用下圖所建議的流程：

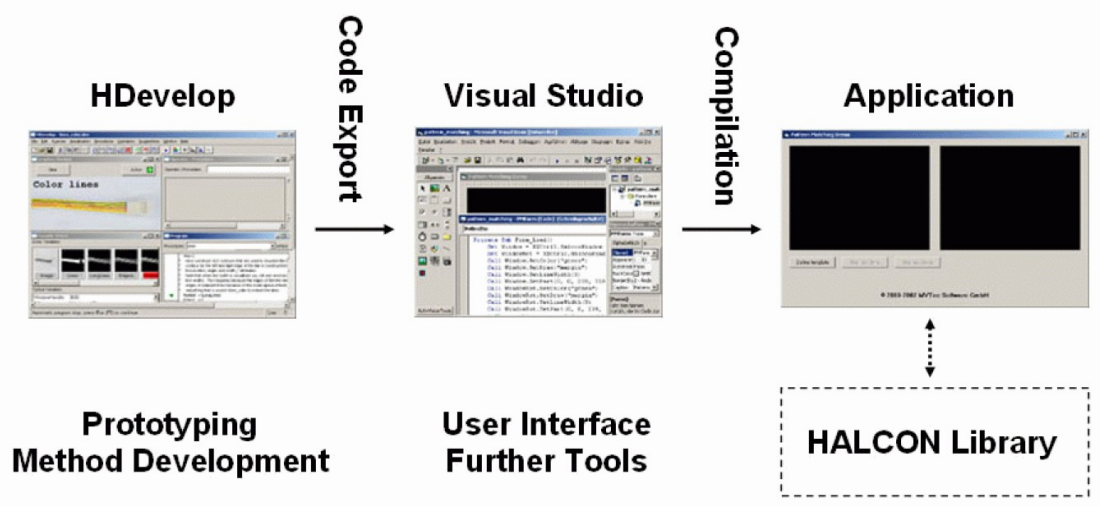


Figure 2.1: Three-step approach for the application development.

圖2.1發展軟體的三個步驟

上圖說明了這個流程：利用HDevelop檢視分析影像，建立計算雛形，最後完成視覺計算方法的發展。程式可以分成不同的子程序， 每個procedures可以只做一件事，像初始化，計算，或是清除。主程序用於呼叫其他子程序，傳遞影像或是接收顯示結果。最後，程式輸出成我們要用的程式碼，接續下一步工作。
完整的程式發展是在程式設計環境中進行，像是 Microsoft Visual Studio 。由HDevelop輸出的程式碼，透過指令加入程式中(例如include)。至於程式的介面等等則是利用程式語言的功能來建構，接下來，compiled and linked，產生應用程式。自行撰寫的程式和 HALCON library一起裝入機器中出貨，或是將程式賣到客戶處。
圖2.1說明了這個流程的概念，這個三步驟的發展方式有幾個好處：

1.由於HDevelop提供了極佳的影像檢視以及除錯機制，系統中的vision部分很容易新增功能或是修改，不需要在一般的程式語言環境下進行除錯。

2.HDevelop輸出的程式碼可以輕易的和程式語言結合，縮短撰寫時間。

3.由於vision計算的程式碼和程式其他部分無關，因此可以獨立執行開發測試，不必跟隨整個專案進行發展。

4.從技術支援的層面來說，只要將HDevelop程式碼交給廠商即可，可大幅提高服務效率。

5.由於HDevelop可在不同的作業系統下執行，像是Linux，因此開發好的程式可以輕易的達到跨平台的使用便利性，這和一般的微軟系列的程式開發工具大不相同。

2.2章說明了如何在HDevelop中建立雛形的基本資訊，2.3章說明如何將HDevelop輸出的程式碼整合到程式之中。至於程式語言的開發環境諸如 Microsoft Visual C++等等，請參考相關軟體廠商的文件。

2.1 HALCON 的核心: 組織架構以及資料結構

HALCON的架構，資料結構以及內部管理機制是依據以下的哲理來設計的

1.效率
2.開放
3.標準化
4.自我敘述

效率是指HALCON盡可能地縮短每個運算子的執行時間。進一步的，即使進行一連串的複雜計算，依舊能維持良好效率。開放的架構是基於兩種重要的考量，首先，HALCON要能用在不同的程式語言中，因此必須能對外進行資料交換以及讀取內部資料。開放而透明的架構才能整合使用者自訂的計算功能甚至是一些非標準的取像設備。此外，這種方式也使得週邊設備有更動或更新時，不必再重新安裝HALCON一次。標準化是指運算子的命名，功能以及使用方式，資料結構等等都遵循一個嚴格的限制，這使得使用者有最快的學習效率，也降低了犯錯機會。最後，HALCON對每個運算子都有詳細的說明，除了詳盡的文件，特定的運算子也可以讀取這些說明資料。

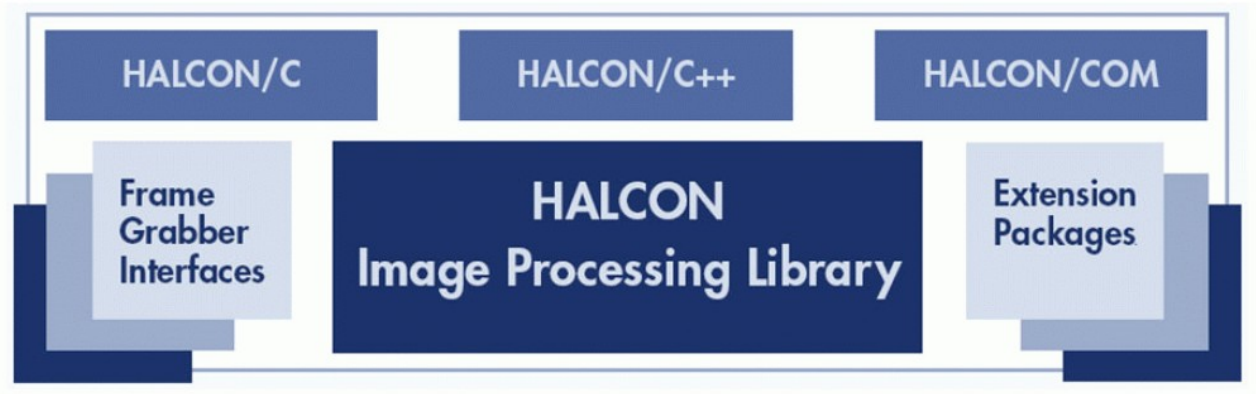


圖 2.2: HALCON 的基本架構

圖2.2中可見其基本架構。主要的部分是 image processing library，包含了大量的運算子，提供了所有的功能。運算子透過所謂的語言介面來操作，可以用不同的程式語言來呼叫其中的功能，或是用動態連結的方式載入。圖片的左側是所謂的取像介面，右側是使用者自訂的延伸功能。更深入來說，有兩大要點：即運算子和資料結構。

2.1.1 HALCON 的運算子

HALCON中各種運算功能皆是透過運算子來完成。目前的版本約有 1100 多個運算子。其中大部分都有多重的計算功能，你可以透過參數來選擇計算方法。在參考手冊或是HDevelop主選單中的Operators中都有完整的列表。這些運算子之間其實是沒有任何繼承關係，但是可以依據功能來分類。您可以在C++或是COM手冊中看到這些類別，相同類別裡的運算子成員處理類似的資料。所有運算子的輸入輸出參數排列方式都有相同的規則，請看 2.1.2。運算子的設計依循開放式的架構，您可以設計自己需要的運算子，加入HALCON，感覺上就像暨有的功能一般 (請看 2.4.1)。許多運算子可以藉由自動平行計算機制來加快速度，尤其是以多處理器的電腦來處理大尺寸的影像時效果尤著。 (請看 2.1.3).

2.1.2 參數和資料結構

HALCON 有兩種基本的資料型態: **圖像資料**(iconic，例如影像)以及**控制資料**(control，例如變數，整數，字串，handle等等)。所有運算子的參數都是以相同的方式排列：輸入圖像，輸出圖像，輸入控制，輸出控制。當然，並非所有的運算子都具有上列四類參數，不過參數排列的次序依舊相同。每個運算子都有一個自我敘述介面，除了標準文件，還有參數類型，或是可用的數值，都可在線上作業時獲得。計算過程中運算子的輸入參數內容不會改變，僅有的三個例外是 **set_grayval, overpaint_gray** 以及 **overpaint_region**。
開放的架構讓您可以取得內部資料，以便和外部資料整合。在處理2D影像時需要用到的各種資料結構像是影像(例如多頻道影像)， region， contours，tuples (一種陣列)等等，都提供了一種快速有效的存取功能。關於這些低階資料結構的說明，請參考 **Extension Package Programmer's Manual** 之第四章。 至於程式語言中可用的資料型態或類別，請參閱**HALCON Programmer's Guide**.

2.1.2.1 影像

影像屬於圖像資料
影像中的主要部分是個別的頻道，他們是由代表不同像元型態的灰階值陣列構成。每張影像都有其定義域(domain)，代表影像中要處理的資料範圍。這就成爲所謂的 *region of interest* (ROI)，定義域本身就是一個HALCON的region，具有無比的彈性，例如從一個簡單的矩形到任意形狀。

像元資料
像元資料幾乎可爲任何型態，從 8-bit 灰階到浮點數皆可。整數，從1到4bite。浮點數，或是 complex images 也可處理。此外，還包含了表現邊緣方向或是 hue 的特別資料型態。

影像頻道
頻道即是影像中某一個資料陣列。影像可含有任意數目的頻道。同一影像中的所有頻道都有相同的尺寸。最常見的影像是單頻道的灰階影像，或是三頻道的彩色影像(例如RGB)，多光譜影像，或是多種紋理分類使用的影像。

座標系統
影像是以左上角爲座標原點 (0,0)。每個像元是以row 和 column 表示其座標。座標值的範圍從 (0,0) 到 (height-1, width-1)。每個像元的尺寸爲 1 ，第一個像元的中心座標爲 (0,0)，因此第一

個像元的範圍是從(-0.5, -0,5) 到 (0.5,0.5)。

2.1.2.2 Regions

Region屬於圖像資料。
所謂region 即是一堆像元的集合。但是他們的座標範圍不受影像大小的限制。region中的像元不一定要相連，也就是說任意形狀的像元集合都可成爲一個region，如果要讓相連接的像元成爲一個region，只要呼叫運算子 connection 即可。由於region內的像元座標範圍並非受限於某張影像，region可大於影像範圍，例如進行region擴張運算時就有可能會超過。至於要不要將尺寸限制在影像大小範圍內，可透過運算子 set_system ，配合參數 'clip region'來設定。
region是由一種runlength encoding資料組成的，記憶體使用量極低，卻有很好的速度和效率。由於region不是以一般軟體的 label images 來控制，因此可以互相重疊，例如擴張運算的結果，一般的軟體無法做到這一點。至於程式中允許的region數目，並無特別限制。

2.1.2.3 XLDs

XLDs (屬於iconic data).
XLD 是 eXtended Line Description 的縮寫，包含了所有等值線以及多邊型的資料。像 edges_sub_ pix 之類的次像元精度運算子產生的資料即屬於 XLD 。
一條所謂的２Ｄ等值線是一連串座標點的串列，相鄰兩點間以直線相連。一般來說，資料點之間的距離大約是１Pixel。XLD物件中除了點座標資料，還包含了全域或區域屬性，例如edge方向，或是分割時的regression參數等等。除了取出XLD資料，還能做進一步的應用，例如選擇具有特定特徵的曲線，或是把曲線分割成直線段，圓弧，多邊型，或是平行線等等。

2.1.2.4 Control Tuples

Tuples就像一個陣列，其中的資料型態可爲整數，浮點數或是字串。一個tuple型態的變數可爲上述三種資料型態之一，甚至於混合。當我們計算一個region的某些特徵時，會傳回一個結果，如果計算的是一群region，會傳回一個tuple，其中含有每個region的特徵計算結果。
Control tuple的指標由０開始。**請注意，iconic tuple的指標由１開始**。

2.1.2.5 Handles

Handles是用於管理一組複合的資料，例如 shape-based matching 中的models。爲了程式設計的方便性以及資料安全與效率，這類資料只透過一個handle 讓使用者操控。每個Handle都有一個唯一的整數數值，由系統底層自行產生。例如圖形視窗，檔案，sockets，取像設備，OCR， OCV，measuring， matching等等，都會以handle來代表要操作的對象。

2.1.3 Parallel HALCON

簡單的說，標準版的HALCON 是爲了在單處理器的電腦上進行循序式計算而設計的。在 Windows NT/2000/XP， Linux，以及 Solaris等等系統下，HALCON 是 *thread-safe*，也就是能用於 multi-threaded 程式。然而程式執行時，每個運算子都是獨一的，也就是執行緒間要互相等待。相對的，Parallel HALCON 支援所謂的 *parallel programming* (e.g., multi-threaded programs)，他同時是thread-safe 以及 reentrant。這意味著多個執行緒可同時呼叫同一個運算子。平行運算版本的 HALCON 支援 Windows NT/2000/XP， Linux， 以及 Solaris。除了支援平行運算，平行運算版本的HALCON在多處理器的電腦上能自動進行資料平行化，分配到不同的處理器去作業，例如影像。舉個例子，在一個４處理器的電腦上進行影像的濾波計算，影像會被分成四

份，由４個執行緒分別在４個處理器上進行相同的運算。以HALCON處理影像以及region的設計哲理來說，由於影像不必再經複製，所以這種平行化是一個最有效率的方法。至於平行化的程度還能由線上控制，避免平行化進行過度。例如，太小的影像就不應該進行平行化，否則往往只會降低速度。此外，並非所有的 運算子都支援平行處理。這部分的詳細說明請參閱 **Programmer's Guide**,第一章。注意， Parallel HALCON 是針對 *shared-memory systems* 設計的，例如多個處理器使用共用記憶體的架構，主要理由就是因爲這種架構下各個執行緒需要的資料不必再複製一次。這也意味著HALCON不能用於cluster式記憶體或是處理器使用個別記憶體的工作站。

2.1.4 取像

目前 HALCON 提供了 40 餘種取像設備連接介面，並以動態連結的方式載入。(Windows: DLLs; UNIX: shared libraries).這些 libraries 隨著 HALCON一起安裝到電腦中。他們的名稱一律以 HFG開頭，至於以 parHFG開頭者是用於Parallel HALCON。關於取像作業的詳細說明請參考**Application Note on Image Acquisition** 。這些取像介面將HALCON與硬體製造商提供的驅動軟體結合，成爲一個標準化的共同介面，不論您使用的的硬體爲何，只要透過少數幾個運算子就能操作。當您安裝了影像卡以後，要透過 HALCON 使用它，只需要呼叫 **open_framegrabber**，並設定相關的參數。接下來，只要用**grab_image** 或 **grab_image_async** 即可取得影像。HALCON 的frame grabber 介面常常在更新，比HALCON library 本身還頻繁。理由之一是因爲 MVTec 持續在發展新的介面，另外就是爲了配合取像設備廠商的軟體更新。最新的介面以及相關文件可由http://www.mvtec.com/halcon/framegrabber下載。

2.2 HDevelop 快速入門

HDevelop 是一個強大的建構雛形以及發展方法的強大環境。要使用HDevelop ，您只需知道一點點東西。建議您從MVTec提供的 680 多個立即可以執行的程式著手瀏覽。這部分細節請您參考**HDevelop User's Manual**。在Windows NT/2000/XP下，由*開始* > *程式集* > MVTec HALCON 7.0 > HDevelop 來執行HDevelop。 UNIX系統下，在 shell 中執行 hdevelop。要載入範例程式，只要從程式主選單的**File > Open** 開啓一個檔案選擇對話框，此時的目錄是位於主要的範例所在目錄，建議各位初學者由 Applications這個目錄下的範例程式開始著手。此外，由程式主選單 File > Open Example Program... 則可開啓一個對話框，由此可以根據不同的分類方式來選擇範例。程式載入後，程式視窗中即顯示出程式碼，變數視窗中擇顯示出所有用到的變數，由於尚未執行，各個變數是以？表示其尚無內容。簡易使用方法如下：
1.要執行程式，按RUN(F5)即可，若程式停在stop指令上，再按一次RUN(F5)即可繼續。

2.除了RUN，HDevelop還提供了STEP(F6)，可讓您一行行的執行程式並且檢視成果。如果程式中含有其他的副程序，利用STEP則可 Step Into 以及 Step Out。

3.要重新執行程式，按Reset(F2)即可讓程式回到起點。

4.想要執行一部份的程式，只要在程式視窗中，用滑鼠左鍵在想要的程式行左側點一下，將綠色的 program counter 放在該處，程式就可以由該行開始執行。

HDevelop的提示：
1. 程式視窗下方有一個狀態列，這裡會顯示一些很有用的參考資訊。例如各個運算子的執行

時間，或是當程式因爲stop指令停止時，或是等候使用者輸入(例如畫region)等等。
2. 許多程式會在圖形視窗中顯示成果，您也可以在變數視窗中在想要顯示的物件上點兩下，便可以手動的方式將物體顯示在圖形視窗中。
3. 依據您選擇的安裝方式，並非所有的影像都會複製到電腦中。如果有短少程式需要的影像，建議您將 HALCON CD 置入光碟中，或是安裝需要的影像。
4. 有些程式會開啓 frame grabbers 進行取像，若系統中沒有對應的取像設備，則會出現錯誤訊息。遇到這種情況則建議您選擇另一個程式來執行，或是修改程式中的參數以符合現行使用的硬體。另外，如果您執行的是Demo版的HDevelop，將無法進行任何取像動作。

2.3 在程式語言中使用 HALCON

HALCON提供了三種語言介面， 讓您在所使用程式語言中呼叫使用 HALCON library 強大的功能卻不失使用上的便利。其中 C 和 C++ 介面是特定語言使用的，但是 COM 介面並不限定使用語言，你可以在 Visual Basic， C#，或是 Delphi中使用。依據您使用的語言，對應的介面(halconc.*, halconcpp.*,halconx.*) 要和HALCON library (halcon.*) Link到程式中。在 C 以及 C++ 語言中，要加入對應的 include files 。在著手進行開發前，建議您先試著執行HALCON中提供的各種語言範例程式，由此您可以了解專案中必要的設定以及相關的資料型態宣告方式。各種語言介面中用的語法以及資料型態，類別等等稍有不同，詳細的說明詳見 HDevelop，C++， COM (Visual Basic) 以及 C 的參考文件。

2.3.1 C

這個介面是最簡單的一種。每個運算子有一到二種全域函數，其名稱和參數順序和在HDevelop中一致。所有的參數都能接受tuple型態的輸入資料。這些運算子名稱都是以T開頭，如果不會用到tuple作爲輸入，則可用另一種只使用基本資料型態(long, double, and char*)的運算子。 Hobject 以及 Htuple 分別用來宣告圖像以及控制資料。
由於　C 不含解構功能，因此你必須利用 **clear_obj** 來釋放宣告的圖像變數。複製，產生，清除或是處理tuple資料時，會用到 macros 功能。詳細資料請看HALCON **Programmer's Guide**， 第四部分。下列的程式碼說明了如何由檔案讀取一張影像，並且顯示在圖形視窗中。

```
read_image(&Monkey, "monkey");
get_image_pointer1(Monkey, &Pointer, Type, &Width, &Height);
open_window(0, 0, Width, Height, 0, "visible", "", &WindowHandle);
disp_obj(Monkey, WindowHandle);
```

2.3.2 C++

C++ 介面要比C精巧得多。C++語言的特長以及物件導向的語法讓使用上更爲便利，例如自動型態轉換，建構與解構，或是將處理同類型資料的功能集合成一個類別。和在C介面中一樣，每個運算子都有一個全域函數讓您可以使用程序式的程式設計語法，此時可用Hobject 以及 HTuple 兩個類別，以物件導向式的來說，可用到的包含HDataCode2d，HMeasure或是HshapeModel....等等，至於其他的類別請參考 HALCON **Programmer's Guide, part II**.
下列程式碼說明了讀取影像，顯示在圖形視窗中，並且進行一些基本的 blob 分析。

```
HImage Mandrill("monkey");
HWindow w(0, 0, 512, 512);
Mandrill.Display(w);
HRegion Bright = (Mandrill >= 128);
HRegionArray Conn = Bright.Connection();
HRegionArray Large = Conn.SelectShape("area", "and", 500, 90000);
```

2.3.3 Visual Basic

和C++類似，您可以用物件式或是程序式的語法。以程序式的語法，您使用的是 HOperatorSetX 這個類別，其中提供了所有運算子的呼叫，HUntypedObjectX 代表所有圖像資料，內建的 Variant 則是用於控制資料。物件導向式的寫法，用到了各種類別，像是 HDataCode2dX， HMeasureX，或是 HShapeModelX ....等等。至於其他的類別請參考 HALCON **Programmer's Guide, part III**
下列程式碼說明了讀取影像，並且進行一些基本的 blob 分析。

```
Dim image As New HImageX
Dim region As HRegionX
Call image.ReadImage("monkey")
Set region = image.Threshold(128, 255)
```

2.3.4 C#

如同 Visual Basic， C# 使用 HALCON 的 COM 介面。也就是說和VB中的用法相同。唯一的差異是在.NET中 Variant 資料型態稱之爲 Object， C# 中也是如此。
下列是C# 程式範例：

```
HImageX image = new HImageX();
HRegionX region;
image.ReadImage("monkey");
region = image.Threshold(128, 255);
```

2.4 延伸 HALCON 功能

2.4.1Extension Packages (User-Defined Operators)

HALCON 可透過新增運算子以延伸功能。 HALCON 雖然已提供了1100 多個運算子，或許您還是想新增一些功能，例如整合某些硬體或是新的運算方法。爲此， HALCON 提供了所謂的 Extension Package 介面，讓您可以整合加入新的運算子(以C撰寫)。他提供了一些預先設定的程序和巨集，讓您可以掌控影像物件以及記憶體的使用。一旦新的運算子完成整合，使用上就和HALCON暨有的功能一般。在 **Extension Package Programmer's Manual** 包含了詳細的說明。

2.4.2Frame Grabber 介面

frame grabber 介面的整合是透過動態結資料庫。以這種方式新增介面就不必動到系統中的其他部分。介面產生以及整合方法在**Frame Grabber Integration Programmer's Manual.** 有敘述。此外，HALCON 包含了一套樣板程式碼作爲自行開發時的基礎。

2.5 HALCON 的使用極限

影像尺寸　32768*32768
記憶體中影像陣列數目　100000
每個參數包含的物件數目　100000
每張影像的頻道數　1000
tuple中的數值資料數目　1000000
一條等值線上的取樣點數目　30000
一個多邊型上的控制點數目　10000
影像座標　-32768 到　+32768
字串長度　1024個字元

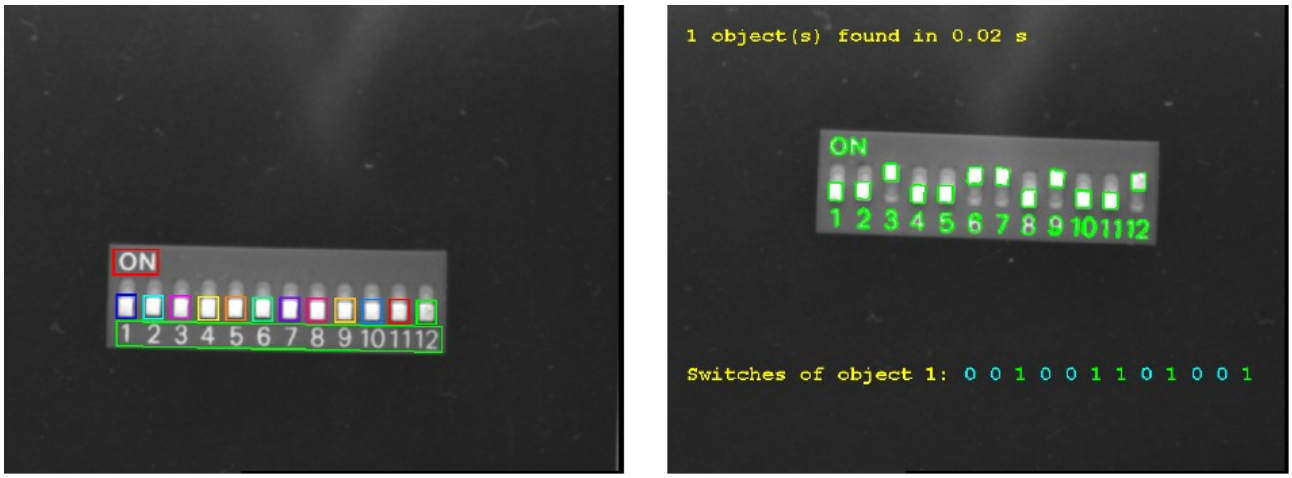
第三章

各種行業的應用

3.1 電子零件和設備

3.1.1 檢查 Dip Switch 的狀態

範例：examples\hdevelop\Applications\FA\cbm_dip_switch.dev
這個範例的任務是要檢查 Dip Switch ，即計算外殼與各個開關間的相關位置(見圖 3.1)。



(a) (b)

圖 3.1(a)dip switch 的影像中各部份的 ROI。(b)位於 dip switch 的開關狀態。

這個任務是使用以元件爲基礎的比對方法來解決。在圖 3.1a 中，Dip switch 一共有 14 個部份，其中包含 12 個開關和 2 個在外殼上的印刷文字。在影像顯示出所有可能的開關位置後，即完成對影像的教導，此外系統也會學習外殼與開關間相對位置的變動。

```
train_model_components (ModelImage, InitialComponents, TrainingImages,
                        ModelComponents, 45, 45, 30, 0.95, -1, -1, rad(20), 'speed',
                        'rigidity', 0.2, 0.5, ComponentTrainingID)
```

在教導中加入少量的誤差容許值變動，使影像的識別更加可靠。

```
modify_component_relations (ComponentTrainingID, 'all', 'all', 0, rad(4))
```

現在可以產生這個模型了

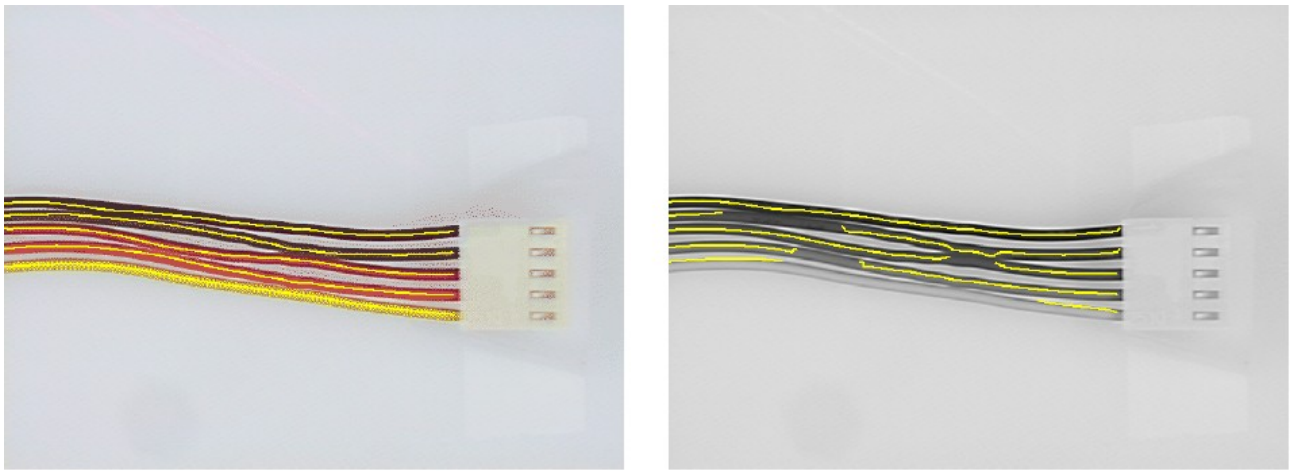
```
create_trained_component_model (ComponentTrainingID, 0, rad(360), 10,
                                MinScoreComp, NumLevelsComp, 0, 'none', 'use_polarity', 'false',
                                ComponentModelID, RootRanking)
```

在搜尋 dip switch 時，不止是搜尋外殼與 dip 的相對位置關係，還包含每個個別 dip 間的相對位置。經由這些資訊，可以很容易的得知開關所有的狀態。

```
find_component_model (SearchImage, ComponentModelID, RootRanking, 0,
                      rad(360), 0, 0, 0.5, 'stop_search', 'prune_branch', 'none',
                      MinScoreComp, 'interpolation', 0, 0.9, ModelStart, ModelEnd, Score,
                      RowComp, ColumnComp, AngleComp, ScoreComp, ModelComp)
```

3.1.2 檢視電源供應線

範例：examples\hdevelop\Filter\Lines\lines color.dev
這個範例的任務是電源線的定位和檢查，詳細見圖 3.2



(a) (b)

圖 3.2：(a)利用有顏色的線條，在原始的顏色影像中，取出電源線的中心(b)相同的影像結果使用灰階值的影像來作。

底下程式的輸入項，是一個有顏色的電源線影像樣本。它的任務是爲了取出每條電源線的中心和寬。完成此行程式後，使用次像元精度的顏色線來解析，並將太短的輪廓線刪掉，去除不相關的部份。

```
lines_color (Image, Lines, 3.5, 0, 12, 'true', 'false')
select_contours_xld (Lines, LongLines, 'contour_length', 450, 100000, 0,
                     0)
```

經由擷取線寬的特性，判定電源線的寬度。爲了顯示這個目的，要產生每一個邊的輪廓線。

```
Number := |LongLines|
EdgesL := []
EdgesR := []
for K := 1 to Number by 1
    Line := LongLines[K]
```

```
        get_contour_xld (Line, Row, Col)
        get_contour_attrib_xld (Line, 'angle', Angle)
        get_contour_attrib_xld (Line, 'width_right', WidthR)
        get_contour_attrib_xld (Line, 'width_left', WidthL)
        EdgeRR := Row+cos(Angle)*WidthR
        EdgeRC := Col+sin(Angle)*WidthR
        EdgeLR := Row-cos(Angle)*WidthL
        EdgeLC := Col-sin(Angle)*WidthL
        gen_contour_polygon_xld (EdgeR, EdgeRR, EdgeRC)
        gen_contour_polygon_xld (EdgeL, EdgeLR, EdgeLC)
        EdgesL := [EdgesL,EdgeL]
        EdgesR := [EdgesR,EdgeR]
endfor
```

把解析的結果和傳統的方法作比較，讓線形解析也應用於灰階值影像。這個結果被顯示在圖 3.2(b)中。這裡，由於影像僅僅由光線的照明而變的明顯，因此可以清楚的看到其成果是較不如用有顏色的線條來作分析。

```
rgb1_to_gray (Image, GrayImage)
lines_gauss (GrayImage, LinesGray, 3.5, 0, 0.7, 'dark', 'true', 'true',
        'false')
```

3.1.3 其它的範例

HDevelop

- examples\hdevelop\Applications\FA\cbm bin switch.dev
  使用以元件爲基礎的比對，找出一個開關並且測試它的狀態。

- examples\hdevelop\Applications\FA\holes.dev
  取出洞的位置和半徑

- examples\quick guide\hdevelop\fuse.dev
  測量保險絲的厚度
  --->在 Quick Guide 中有這裡的描述

3.2 食物

3.2.1 "食用期限"的日期

範例：examples\hdevelop\Applications\OCR\bottle.dev
這個範例的任務是檢視在瓶蓋上"食用期限"的日期

(a) (b)

圖 3.3：(a)原始影像(b)讀取的日期

這個必需使用多個步驟才能解決。首先，要擷取黑暗的區域，之後處理太薄的部份。

```
threshold (Bottle, RawSegmentation, 0, 95)
fill_up_shape (RawSegmentation, RemovedNoise, 'area', 1, 5)
opening_circle (RemovedNoise, ThickStructures, 2.5)
fill_up (ThickStructures, Solid)
```

下一步，將 region 分割成單獨的字元;即使是太過於靠近而互相接觸的字元也都可以被分離出來。

```
opening_rectangle1 (Solid, Cut, 1, 7)
connection (Cut, ConnectedPatterns)
intersection (ConnectedPatterns, ThickStructures, NumberCandidates)
select_shape (NumberCandidates, Numbers, 'area', 'and', 300, 9999)
sort_region (Numbers, FinalNumbers, 'first_point', 'true', 'column')
```

最後，進行 OCR 的讀取

```
read_ocr_class_mlp (Name+'.fnt', OCRHandle)
do_ocr_multi_class_mlp (FinalNumbers, Bottle, OCRHandle, RecNum, Confidence)
```

3.2.2 其它的範例

HDevelop

- examples\hdevelop\Applications\OCR\bottlet.dev
  分離和教導啤酒蓋上的數字

C++

- examples\cpp\source\bottle.cpp
  讀取啤酒蓋上的數字

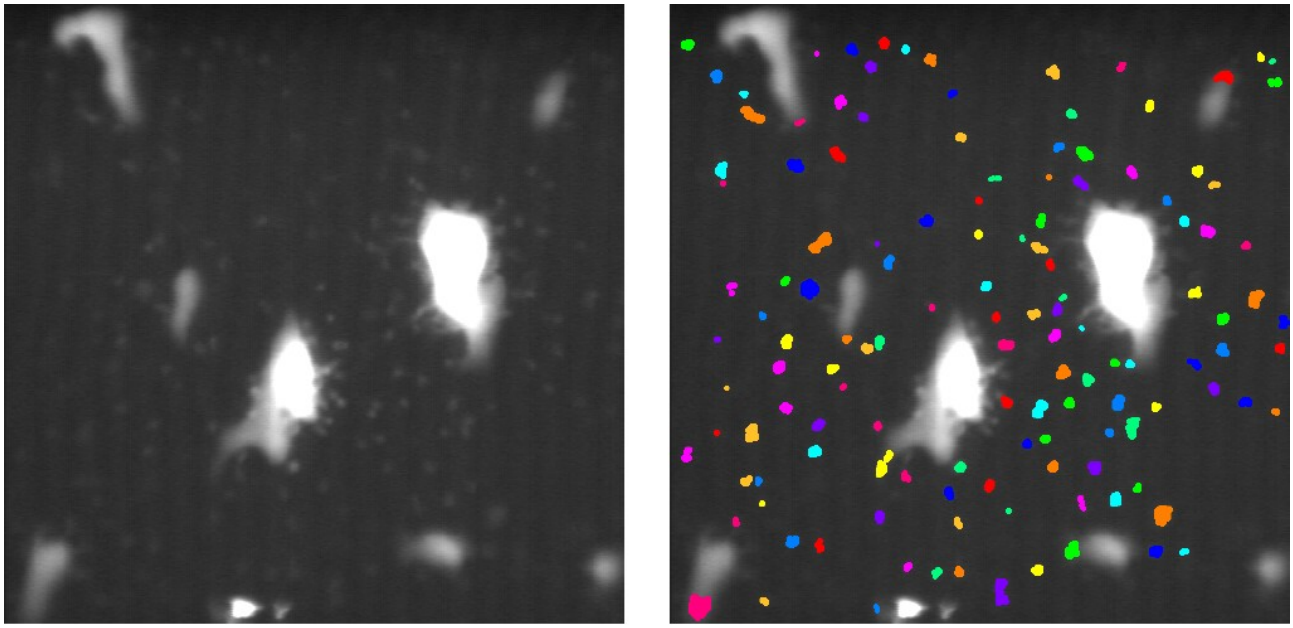
3.3 醫療和生命科學

3.3.1 分析顆粒

範例：examples\hdevelop\Applications\Medicine\particle.dev

這個範例的任務是分析液體中的顆粒。在這項應用中的主要困難是，存在二個不同類型的物體，一個是較大又明顯的物體，另一個是較小且對比不明顯的物體。此外雜訊的存在更使這個分離更複雜。

程式片斷中的二個件物類別，分別使用二個不用的方法：全域和區域的臨界值。增加一個後處理，讓較小的顆粒以可靠的方法被擷取出來。



(a) (b)

圖 3.4:取出較小的顆粒(a)原始影像(b)結果

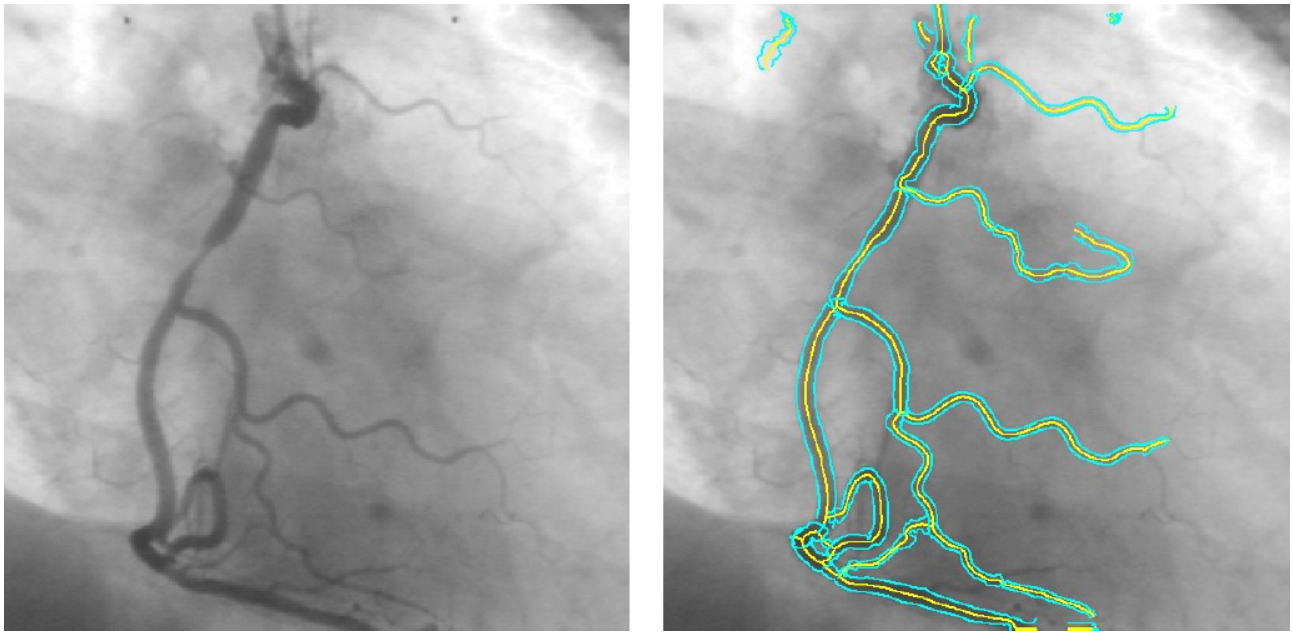
```
threshold (Image, Large, 110, 255)
dilation_circle (Large, LargeDilation, 7.5)
complement (LargeDilation, NotLarge)
reduce_domain (Image, NotLarge, ParticlesRed)

mean_image (ParticlesRed, Mean, 31, 31)
dyn_threshold (ParticlesRed, Mean, SmallRaw, 3, 'light')
opening_circle (SmallRaw, Small, 2.5)
connection (Small, SmallConnection)
```

3.3.2 血管造影法

範例：examples\hdevelop\Filter\Lines\lines gauss.dev

這個範例的任務是去擷取，在圖 3.5 中心臟的 x 光照片影像中的血管。在經由適當的對比之後，血管可被突顯出來。然後擷取血管的寬度，來判斷區域中狹小的部份(狹窄症)，這對診斷而言是很重要的。



(a) (b)

圖 3.5:(a)心臟的 x 光照片影像(b)擷取血管

使用 line_gauss 來擷取血管影像。這項作業的結果是從 XLD 輪廓線中找出血管的中心。其中一項的屬性是區域線的寬度，這個寬度如同輪廓線一般是必要的，並且要被顯示出來。

```
lines_gauss (Angio, Lines, 2.2, 0, 0.8, 'dark', 'true', 'true', 'true')
Number := |Lines|
for I := 1 to Number by 1
        Line := Lines[I]
        get_contour_xld (Line, Row, Col)
        get_contour_attrib_xld (Line, 'angle', Angle)
        get_contour_attrib_xld (Line, 'width_left', WidthL)
        get_contour_attrib_xld (Line, 'width_right', WidthR)
        RowR := Row+cos(Angle)*WidthR
        ColR := Col+sin(Angle)*WidthR
        RowL := Row-cos(Angle)*WidthL
        ColL := Col-sin(Angle)*WidthL
        disp_polygon (WindowID, RowL, ColL)
        disp_polygon (WindowID, RowR, ColR)
endfor
```

3.3.3 其它範例

HDevelop

- examples\hdevelop\Applications\Medicine\angio.dev
  從血管造影中擷取血管和它的直徑

- examples\hdevelop\Applications\Medicine\vessel.dev
  血管的分離和測量

- examples\hdevelop\Manuals\HDevelop\particle.dev
  測量較小的顆粒
  --->描述在 HDevelop User's Manual

- examples\hdevelop\Manuals\HDevelop\vessel.dev
  微血管的擷取
  --->描述在 HDevelop User's Manual

C++

- examples\cpp\source\example5.cpp
  分析細胞大小的分佈

3.4 鐵、鋼和金屬

3.4.1 檢視鑄造零件

範例：examples\hdevelop\Applications\Measure\measure arc.dev

這個範例的任務是要檢視零件在挖槽之後，其瘦長凹洞間的距離(見圖 3.6)。注意，要達成一個最正確的測量，這裡我們建議使用背光板配合 telecentric 鏡頭

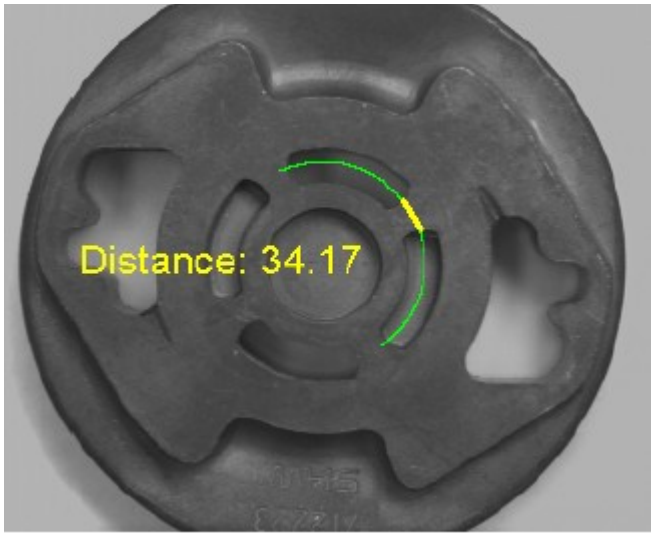


圖 3.6：測量洞與洞之間的距離

使用環形量測的 ROI 來作爲測量的工具，可以容易的解決這個任務。我們將 ROI 的中心放置於 cast part 的中心處；並將它的半徑設成從中心到瘦長凹洞的距離。

```
Row := 275
Column := 335
Radius := 107
AngleStart := -rad(55)
AngleExtent := rad(170)
gen_measure_arc (Row, Column, Radius, AngleStart, AngleExtent, 10, Width,
Height, 'nearest_neighbor', MeasureHandle)
```

現在，洞與洞之間的距離，只要用底下一行運算子呼叫即可測量出：

```
measure_pos (Zeiss1, MeasureHandle, 1, 10, 'all', 'all', RowEdge,
ColumnEdge, Amplitude, Distance)
```

3.4.2 其它的範例

Hdevelop

- examples\application guide\shape matching\hdevelop\align measurements.dev

使用以形狀爲導向的匹配之直線 ROI 測量工具，來檢視個別的刮鬍刀片
--->在 Application Note on Shape-Based Matching 有詳細的介紹

- examples\application guide\shape matching\hdevelop\multiple models.dev

同時搜尋二種形狀的物體
--->在 Application Note on Shape-Based Matching 有詳細的介紹

- examples\application guide\shape matching\hdevelop\multiple objects.dev

搜尋安全環的例子
--->在 Application Note on Shape-Based Matching 有詳細的介紹

- examples\application guide\shape matching\hdevelop\multiple scales.dev

Searches for nuts of different sizes
--->description in the Application Note on Shape-Based Matching
搜尋不同尺寸的螺帽
--->在 Application Note on Shape-Based Matching 有詳細的介紹

- examples\application guide\shape matching\hdevelop\reuse model.dev

儲存模型並重複使用
--->在 Application Note on Shape-Based Matching 有詳細的介紹

- examples\hdevelop\Applications\FA\cbm pipe wrench.dev

搜尋套筒扳手的二個組成要件
--->在 Quick Guide 有詳細的介紹

- examples\hdevelop\Applications\FA\pm multiple models.dev

Finds multiple different models in a single pass using shape-based matching
使用以形狀爲導向的匹配，來找尋各種不同的模形

- examples\hdevelop\Applications\FA\pm world plane.dev

Recognizes planar objects using shape-based matching in perspectively distorted images
在透視變形的影像中使用以形狀爲導向的匹配來識別平面物體

- examples\hdevelop\Applications\OCR\engraved.dev

Segmenting and reading characters on a metal surface
--->description here in the Quick Guide
在金屬表面上分離和閱讀字元
--->在 Quick Guide 有詳細的介紹

- examples\hdevelop\Applications\OCR\engravedt.dev

Segmenting and training characters on a metal surface
在金屬表面上分離和教導字元

- examples\hdevelop\Matching\Gray-Value-Based\best match mg.dev

Finding the best match of a gray value template in a pyramid
在影像金字塔中找出最匹配的灰階值樣板位置

- examples\hdevelop\Tools\Geometry\distance pc.dev

Calculating the distance between a point and a contour
計算點到一條曲線間的距離

- examples\quick guide\hdevelop\measure metal part.dev

由最適線和圓來檢查金屬形狀

- examples\quick guide\hdevelop\surface scratch.dev

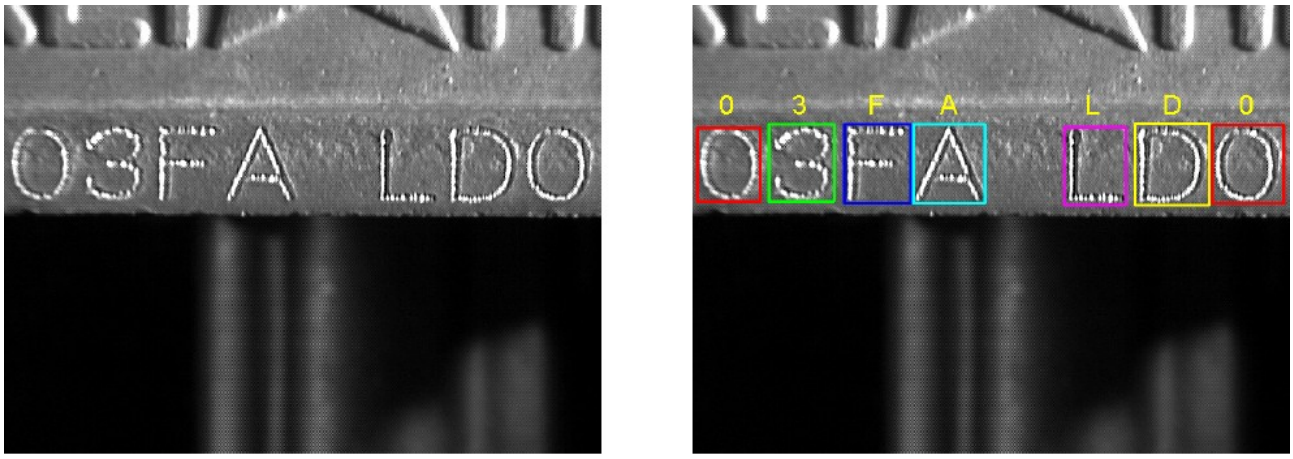
由區域的二值化和形態學來檢查平面上的刮痕
--->在 Quick Guide 有詳細的介紹

3.5 機械

3.5.1 讀取雕刻文字

範例：examples\hdevelop\Applications\OCR\engraved.dev

這個範例的任務是在圖 3.7 中，去讀取在金屬表面所雕刻的文字



(a)

(b)

圖 3.7：(a)原始影像(b)讀取的文字

這個任務要使用進階的 blob 分析來解決：
若僅由選擇較黑或較亮的像元作解析，並不能擷取出其字元，一個簡單的分割反而產生伴隨雜訊物體的片斷字元。所以在分割動作之前，先使用灰階值型態學處理，更可分離出正確的字元。

```
gray_range_rect (Image, ImageResult, 7, 7)
threshold (ImageResult, Region, 128, 255)
connection (Region, ConnectedRegions)
select_shape (ConnectedRegions, SelectedRegions, 'area', 'and', 1000, 99999)
sort_region (SelectedRegions, SortedRegions, 'first_point', 'true',
'column')
```

最後，執行底下的函數即完成讀取的動作

```
read_ocr_class_mlp (Name+'.fnt', OCRHandle)
for i := 1 to Number by 1
ObjectSelected := SortedRegions[i]
do_ocr_single_class_mlp (ObjectSelected, ImageResult, OCRHandle, 1,
Class, Confidence)
endfor
clear_ocr_class_mlp (OCRHandle)
```

3.5.2 檢視工具的輪廓

範例：examples\hdevelop\Applications\FA\circles.dev

這個範例的任務是在圖 3.8 中檢視工具的輪廓

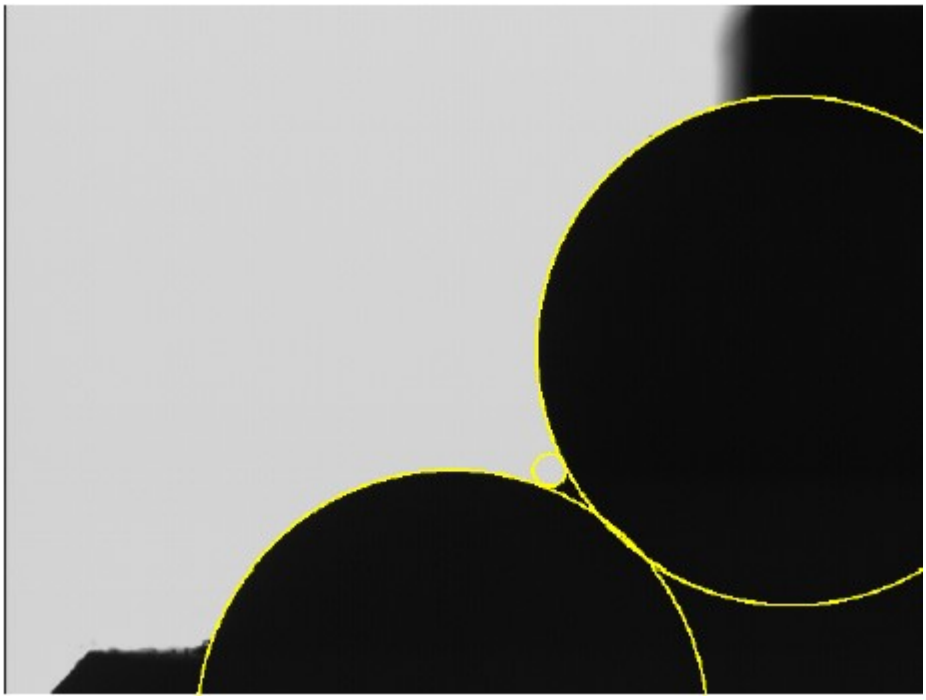


圖 3.8：工具輪廓的最適圓

因爲擷取次像元精度的輪廓是很費時的，所以我們會在第一個步驟中先粗略的找尋出邊緣：
1.首先我們先使用二值化的操作來擷取測量的物體。
2.之後將物體的 region 轉變成到它的邊界。
3.最後再刪去在影像的邊線像元。

```
fast_threshold (Image, Region, 0, 120, 7)
boundary (Region, RegionBorder, 'inner')
clip_region_rel (RegionBorder, RegionClipped, 5, 5, 5, 5)
```

接下來使用一個較小的 region 去 close 物體的邊緣，再使這個 region 的邊界(即邊緣)擴張變成另一個 ROI。下一步我們將影像 domain 縮小，只擷取其邊緣部分。現在，呼叫次像元精度的邊緣擷取函數，再將輪廓線切割成爲平直的線和圓弧的線。

```
dilation_circle (RegionClipped, RegionDilation, 2.5)
reduce_domain (Image, RegionDilation, ImageReduced)
edges_sub_pix (ImageReduced, Edges, 'lanser2', 0.5, 40, 60)
segment_contours_xld (Edges, ContoursSplit, 'lines_circles', 5, 4, 3)
```

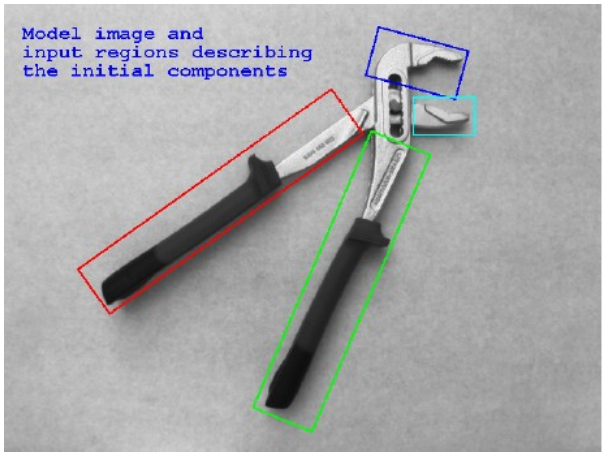
爲了求出最適切的圓弧，要設定相關參數。爲了肉眼檢視的目的，最後將結果和影像顯示在一起。

```
get_contour_global_attrib_xld (ObjectSelected, 'cont_approx', Attrib)
if (Attrib > 0)
fit_circle_contour_xld (ObjectSelected, 'ahuber', -1, 2, 0, 3, 2, Row,Column, Radius, StartPhi,
EndPhi, PointOrder)
gen_ellipse_contour_xld (ContEllipse, Row, Column, 0, Radius, Radius, 0,rad(360), 'positive', 1.0)
dev_display (ContEllipse)
endif
```

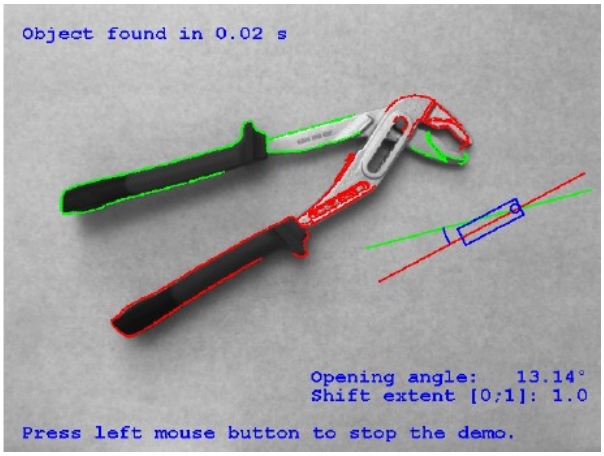
3.5.3 在不同的狀態下找出套筒板手

範例：examples\hdevelop\Applications\FA\cbm pipe wrench.dev

這個範例的任務是以四個預先定義好的 ROI 爲基礎下，去找出套筒板手(見圖 3.9)。四個 ROI 中有兩組相對位置固定的 ROI，兩 ROI 組間的相對位置是不固定的。



(a)



(b)

圖 3.9：(a)在模型影像中依構造指定其 ROI；(b)在其它狀態中找尋套筒板手

展示同一系統中多個影像來教導相關部份的移動。

```
read_image (ModelImage, 'pipe_wrench/pipe_wrench_model')
gen_rectangle2 (InitialComponentRegions, 212, 233, 0.62, 167, 29)
gen_rectangle2 (Rectangle2, 298, 363, 1.17, 162, 34)
gen_rectangle2 (Rectangle3, 63, 444, -0.26, 50, 27)
gen_rectangle2 (Rectangle4, 120, 473, 0, 33, 20)
InitialComponentRegions := [InitialComponentRegions,Rectangle2]
InitialComponentRegions := [InitialComponentRegions,Rectangle3]
InitialComponentRegions := [InitialComponentRegions,Rectangle4]
for i := 1 to 4 by 1
read_image (TrainingImage, 'pipe_wrench/pipe_wrench_training_'+i)
TrainingImages := [TrainingImages,TrainingImage]
endfor
train_model_components(ModelImage,InitialComponentRegions,TrainingImages,
ModelComponents, 22, 60, 30, 0.65, -1, -1, rad(60),'speed', 'rigidity', 0.2,0.4,
ComponentTrainingID)
```

它們的構造和關係會由底下的函數顯示：

```
get_training_components (ModelComponents, ComponentTrainingID,
'model_components', 'model_image', 'false', RowRef, ColumnRef,
AngleRef, ScoreRef)
dev_display (ModelComponents)
get_component_relations (Relations, ComponentTrainingID, i, 'model_image',
```

```
Row, Column, Phi, Length1, Length2, AngleStart, AngleExtent)
dev_display (Relations)
```

根據教導，實際的構造模型會被產生出。其構造的關係會被表現在樹狀結構中。

```
create_trained_component_model (ComponentTrainingID, rad(-90), rad(180),
10, 0.6, 4, 0, 'none', 'use_polarity', 'false', ComponentModelID,
RootRanking)
get_component_model_tree (Tree, Relations, ComponentModelID, RootRanking,
'model_image', StartNode, EndNode, Row, Column, Phi, Length1,
Length2, AngleStart, AngleExtent)
dev_display (ModelImage)
dev_display (Tree)
dev_display (Relations)
```

最後，我們用測試的影像來找尋套筒板手；在每個影像中，我們可以看到模型的輪廓被覆蓋著、轉動著並且套筒板手打開的角度也有所不同。

```
find_component_model (SearchImage, ComponentModelID, RootRanking,
rad(-90), rad(180), 0, 0, 1, 'stop_search', 'prune_branch', 'none',
0.6, 'interpolation', 0, 0.7, ModelStart, ModelEnd, Score, RowComp,
ColumnComp, AngleComp, ScoreComp, ModelComp)
dev_display (SearchImage)
NumFound := |ModelStart|
if (NumFound)
get_found_component_model (FoundComponents, ComponentModelID,
ModelStart, ModelEnd, RowComp, ColumnComp, AngleComp, ScoreComp,
ModelComp, 0, 'false', RowCompInst, ColumnCompInst,
AngleCompInst, ScoreCompInst)
dev_display (FoundComponents)
endif
```

3.5.4 其它的範例

HDevelop

- examples\application guide\shape matching\hdevelop\multiple models.dev

同時尋找二個不同形狀的物體
--->在 Application Note on Shape-Based Matching 有詳細的介紹

- examples\application guide\shape matching\hdevelop\multiple objects.dev

搜尋安全環的例子
--->在 Application Note on Shape-Based Matching 有詳細的介紹

- examples\application guide\shape matching\hdevelop\multiple scales.dev

搜尋不同尺寸的螺帽
--->在 Application Note on Shape-Based Matching 有詳細的介紹

- examples\application guide\shape matching\hdevelop\reuse model.dev

儲存並重複使用一個模形
--->在 Application Note on Shape-Based Matching 有詳細的介紹

- examples\hdevelop\Applications\Measure\measure arc.dev

沿著圓形弧度測量物體的寬
--->在 Quick Guide 有詳細的介紹

- examples\hdevelop\Applications\OCR\engravedt.dev

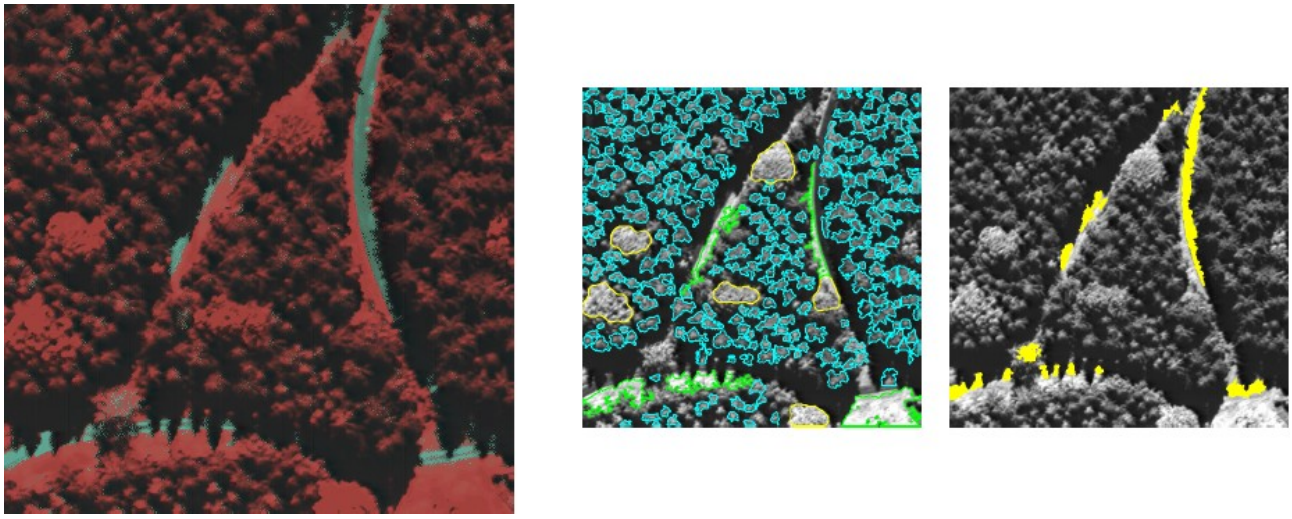
Segmenting and training characters on a metal surface
在金屬表面上分離和教導字元

3.6 航空攝影測量和遙感

3.6.1 從紅外線影像擷取森林的特徵

範例：examples\hdevelop\Applications\Aerial\forest.dev

這個範例的任務是在圖 3.10 中，去檢查在紅外線影像中不同的物體分類：樹、草地、道路。



(a) (b) (c)

圖 310：(a)原始影像；(b)擷取樹和草地(c)擷取道路

這張影像資料是屬於紅外線假色影像，由於它特別的顏色，因此，可容易的擷取出道路的部份。

```
read_image (Forest, 'forest_air1')
decompose3 (Forest, Red, Green, Blue)
threshold (Blue, BlueBright, 80, 255)
connection (BlueBright, BlueBrightConnection)
select_shape (BlueBrightConnection, Path, 'area', 'and', 100, 100000000)
```

在紅色的 channel 中，根據其光的強度和最小的尺寸之下，使山毛櫸樹被分離出來。

```
threshold (Red, RedBright, 120, 255)
connection (RedBright, RedBrightConnection)
select_shape (RedBrightConnection, RedBrightBig, 'area', 'and', 1500,
10000000)
closing_circle (RedBrightBig, RedBrightClosing, 7.5)
opening_circle (RedBrightClosing, RedBrightOpening, 9.5)
connection (RedBrightOpening, RedBrightOpeningConnection)
select_shape (RedBrightOpeningConnection, BeechBig, 'area', 'and', 1000,
100000000)
select_gray (BeechBig, Blue, Beech, 'mean', 'and', 0, 59)
```

草地有著相似的光譜屬性，但稍稍較爲明亮。

```
union1 (Beech, BeechUnion)
complement (BeechUnion, NotBeech)
difference (NotBeech, Path, NotBeechNotPath)
reduce_domain (Red, NotBeechNotPath, NotBeechNotPathRed)
threshold (NotBeechNotPathRed, BrightRest, 150, 255)
connection (BrightRest, BrightRestConnection)
select_shape (BrightRestConnection, Meadow, 'area', 'and', 500, 1000000)
```

先使用 watershed 方法，再由二值化中清除陰暗區域，來擷取結毬果的樹，

```
union2 (Path, RedBrightClosing, BeechPath)
smooth_image (Red, RedGauss, 'gauss', 4.0)
invert_image (RedGauss, Invert)
watersheds (Invert, SpruceRed, Watersheds)
select_shape (SpruceRed, SpruceRedLarge, 'area', 'and', 100, 5000)
select_gray (SpruceRedLarge, Red, SpruceRedInitial, 'max', 'and', 100, 200)
LocalThresh := []
NumSpruce := |SpruceRedInitial|
for i := 1 to NumSpruce by 1
SingleSpruce := SpruceRedInitial[i]
min_max_gray (SingleSpruce, Red, 50, Min, Max, Range)
reduce_domain (Red, SingleSpruce, SingleSpruceRed)
threshold (SingleSpruceRed, SingleSpruceBright, Min, 255)
connection (SingleSpruceBright, SingleSpruceBrightCon)
select_shape_std (SingleSpruceBrightCon, MaxAreaSpruce, 'max_area', 70)
LocalThresh := [MaxAreaSpruce,LocalThresh]
endfor
opening_circle (LocalThresh, FinalSpruce, 1.5)
```

3.6.2 在彩色影像中分離物體

範例：examples\hdevelop\Filter\Edges\edges color.dev

這個範例的任務是在圖 3.11 中分割彩色影像。

這個例子是顯示在 multi-channel 中過濾邊緣的可能性。
首先，從彩色影像中取得灰階值影像，從這我們可以看出某些物體的邊緣不能被看到了。
例如，足球場(綠色的)和周圍的跑道(紅色的)不能被識別出。

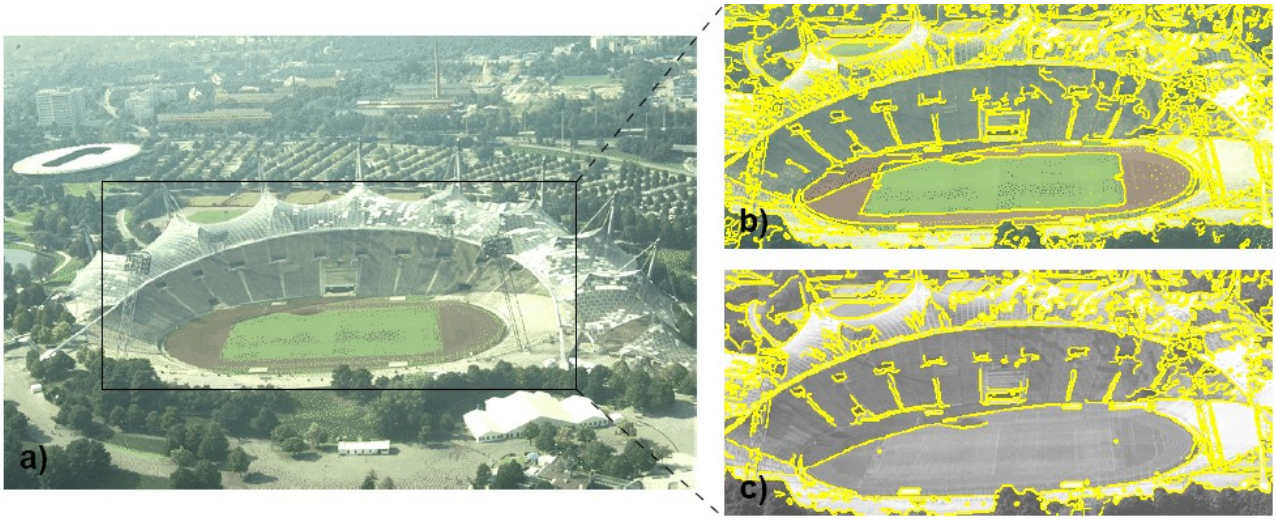


圖 3.11：(a)原始影像(b)擷取彩色邊緣覆蓋在彩色影像上(c) 擷取灰階值邊緣覆蓋在灰階值影像上

```
read_image (Image, 'olympic_stadium')
rgb1_to_gray (Image, GrayImage)
```

應用彩色邊緣的過濾並且顯示其邊緣線。假使你比較這二種邊緣的不同就可以很容易的了解其差異。

```
edges_color (Image, ImaAmp, ImaDir, 'canny', 1, 'none', -1, -1)
edges_image (GrayImage, ImaAmpGray, ImaDirGray, 'canny', 1, 'none', -1, -1)
```

最後，從彩色和灰階影像擷取出邊緣線，並且覆蓋在原始的影像中。

```
edges_color (Image, ImaAmpHyst, ImaDirHyst, 'canny', 1, 'nms', 20, 40)
threshold (ImaAmpHyst, RegionColor, 1, 255)
skeleton (RegionColor, EdgesColor)
dev_display (Image)
dev_display (EdgesColor)
stop ()
edges_image (GrayImage, ImaAmpGrayHyst, ImaDirGrayHyst, 'canny', 1, 'nms',20, 40)
threshold (ImaAmpGrayHyst, RegionGray, 1, 255)
```

```
skeleton (RegionGray, EdgesGray)
dev_display (GrayImage)
dev_display (EdgesGray)
```

3.6.3 擷取道路

範例：examples\hdevelop\Applications\Aerial\roads.dev

這個範例的任務是在圖 3.12 的空照影像中去擷取道路。程式開始首先在放大尺寸的影像中擷取 lines。而這道路本身就是一個很好的線形。

圖 3.12：(a)原始影像；(b)縮放影像的部份；(c)擷取道路

```
threshold (Mreut43, Bright, 160, 255)
reduce_domain (Mreut43, Bright, Mreut43Bright)
lines_gauss (Mreut43Bright, RoadCenters, 1.2, 5, 14, 'light', 'true',
'true', 'true')
```

爲了消除一些可能的錯誤，在較大的比例尺影像中擷取邊緣進行計算。在道路擷取上，我們假定它是由二個平行的邊緣所組成，而在這二平行線之間即爲道路部份。使用 contour processing 可以一步一步的找出較爲精確的結果。

```
edges_image (Part, PartAmp, PartDir, 'mderiche2', 0.3, 'nms', 20, 40)
threshold (PartAmp, EdgeRegion, 1, 255)
clip_region (EdgeRegion, ClippedEdges, 2, 2, PartWidth - 3, PartHeight - 3)
skeleton (ClippedEdges, EdgeSkeleton)
gen_contours_skeleton_xld (EdgeSkeleton, RoadEdges, 1, 'filter')
gen_polygons_xld (RoadEdges, RoadEdgePolygons, 'ramer', 2)
gen_parallels_xld (RoadEdgePolygons, ParallelRoadEdges, 10, 30, 0.15,
'true')
mod_parallels_xld (ParallelRoadEdges, Part, ModParallelRoadEdges,
ExtParallelRoadEdges, 0.3, 160, 220, 10)
combine_roads_xld (RoadEdgePolygons, ModParallelRoadEdges,ExtParallelRoadEdges,
RoadCenterPolygons, RoadSides, rad(40),rad(20), 40, 40)
```

3.6.4 其它的範例

HDevelop

- examples\hdevelop\Applications\Aerial\dem.dev

從一個數位高度模型來擷取出物體的高

- examples\hdevelop\Applications\Aerial\dem trees.dev

Extraction of trees using texture and a digital elevation model
使用紋理和數位高度模型來擷取樹

- examples\hdevelop\Applications\Aerial\high.dev

用二種不同的方法來擷取物體的高

- examples\hdevelop\Applications\Aerial\texture.dev

找出紋理區域(樹和灌木)

- examples\hdevelop\Manuals\HDevelop\dtm.dev

從一個數位高度模型來擷取出物體的高

3.7 印刷

3.7.1 讀取條碼

範例: examples\hdevelop\Applications\Barcode\EAN13AddOn5.dev

這個範例的任務是去讀取條碼，見圖 3.13。

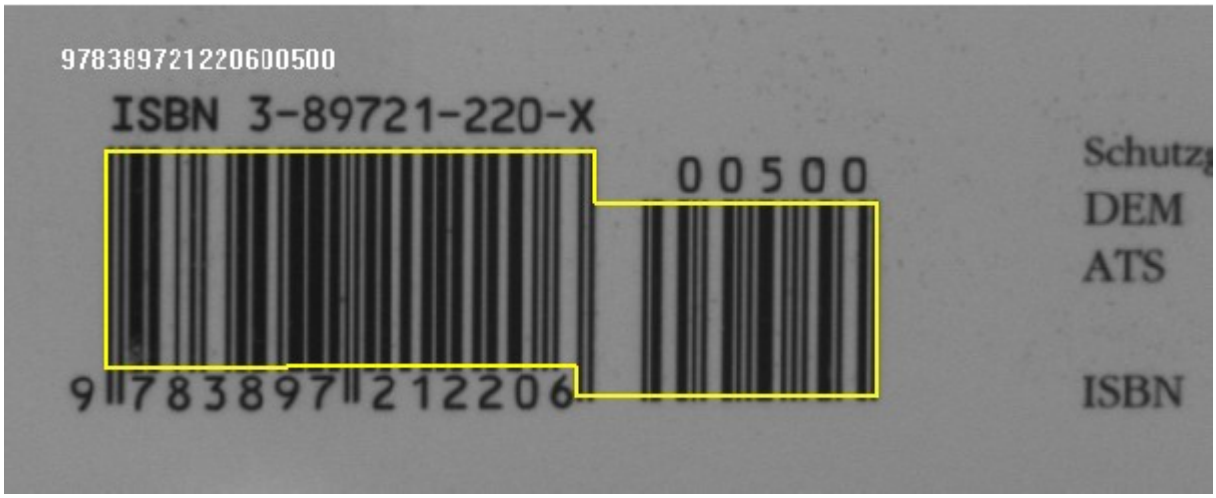


圖 3.13:解讀條碼

這個例子顯示出條碼(甚至是複合的條碼)，能夠使用 HALCON 容易的去解讀。
首先，必需區分條碼的類型

```
gen_1d_bar_code_descr ('EAN 13 Add-On 5', 13, 13, BarCodeDescr)
```

從檔案讀取影像之後，條碼的擷取器會被呼叫。這項作業的結果，即每個元素的寬度，之後再將結果傳給解讀的作業。

```
read_image (image, 'barcode/ean13addon5/ean13addon5'+(i$'.2'))
find_1d_bar_code (image, CodeRegion, BarCodeDescr, 'dilation_factor', 2,
BarcodeFound, BarCode, Orientation)
decode_1d_bar_code (BarCode, BarCodeDescr, Characters, Reference, IsCorrect)
```

3.7.2 其它的範例

HDevelop

- examples\hdevelop\Applications\OCR\stamp catalogue.dev

在雜亂的頁面上分離和群集字元

- examples\hdevelop\Applications\OCV\adaption ocv.dev

從報告的字元特性分析改變的影響

- examples\hdevelop\Applications\OCV\print quality.dev

在不同的影像中顯示字母 A 的特性

examples\hdevelop\Tools\OCV\write ocv.dev
寫入 OCV 資料到檔案中(並再讀取出來)

3.8 橡膠、合成纖維材料、金屬薄片

3.8.1 Checking a Boundary for Fins

範例: examples\hdevelop\Applications\FA\fin.dev

這個範例的任務是要檢查塑膠零件的外部邊界。在這個案中，某些物體在一個應爲無缺點的部份卻有突起物，這是不被允許的(見圖 3.14)。

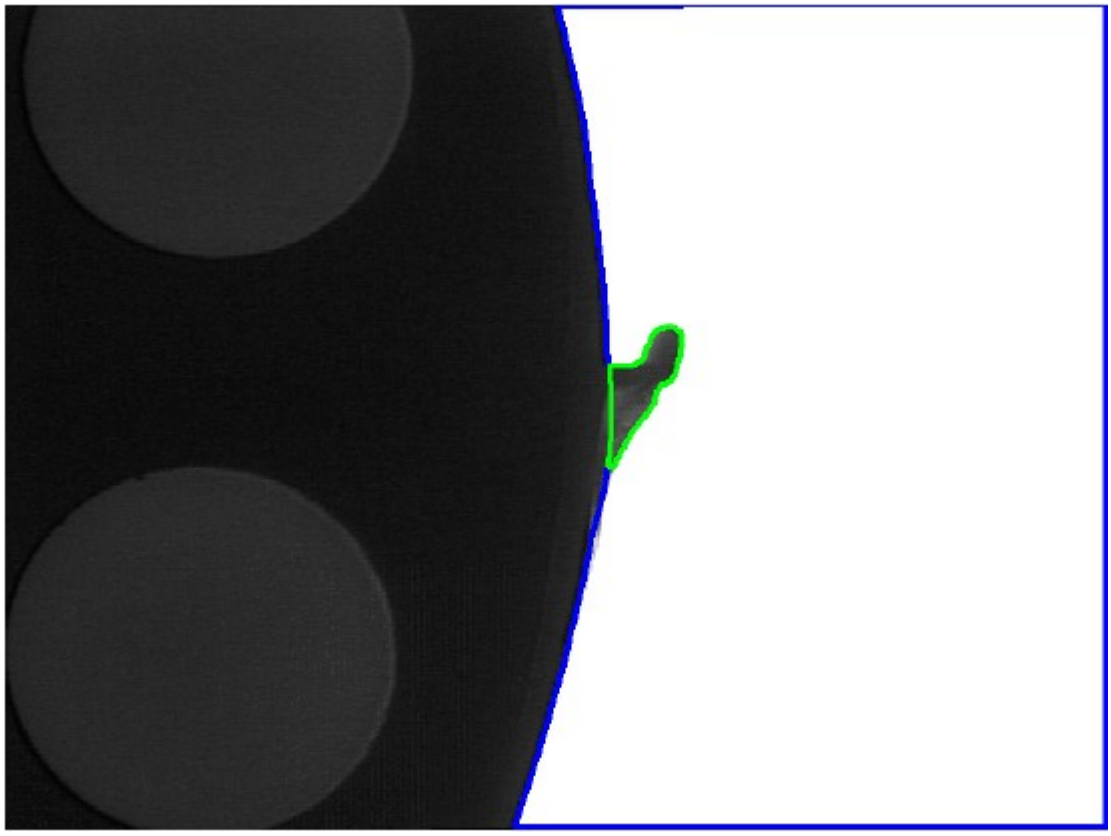


圖 3.14:擷取邊界的突起物

首先程式會先擷取塑膠零件，然後利用 complement 方法去擷取背景的 region(邊界的突起物如同背景 region 的凹陷)

```
bin_threshold (Fin, Dark)
difference (Fin, Dark, Background)
```

在背景 region 的這個凹陷，使用型態學的方法 close 將它填滿

```
closing_circle (Background, ClosedBackground, 250)
```

在 close 之後的 region 和原始的 region 之間，有一個重要的不同就是缺陷的產生。

```
difference (ClosedBackground, Background, RegionDifference)
opening_rectangle1 (RegionDifference, FinRegion, 5, 5)
```

3.8.2 其它的範例

HDevelop

- examples\hdevelop\Applications\FA\hull.dev

檢查注射模型的噴嘴

3.9 半導體

3.9.1 Bonding Balls

範例: examples\hdevelop\Applications\FA\ball.dev

這個範例的任務是要檢查圖 3.15 中 ball bonds 的直徑。

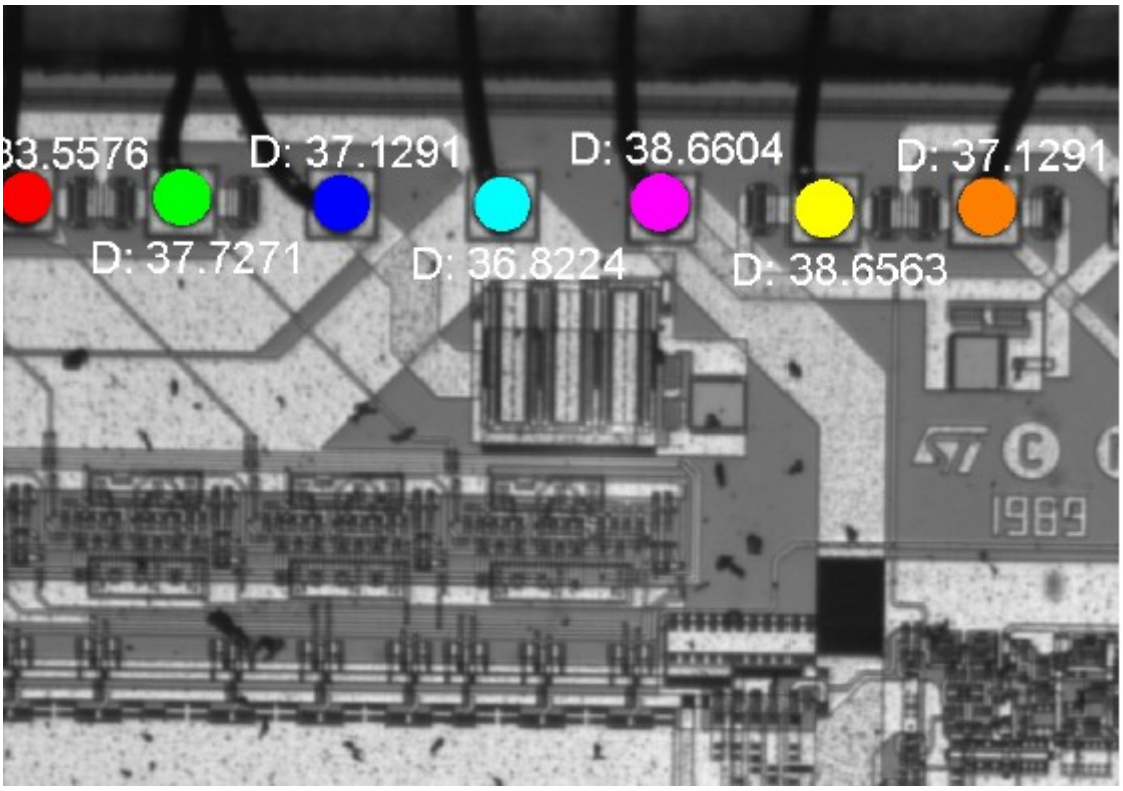


圖 3.15:測量 ball bonds 的直徑

爲了找出 ball bonds 需要經過二個步驟：首先，分離比較明亮的區域以找出 die，然後將找出來的結果，當成一個輸入項，轉換成最小的平行矩形。

```
threshold (Bond, Bright, 100, 255)
shape_trans (Bright, Die, 'rectangle2')
```

現在，處理的重點是在 die 裡面的 region，這我們使用 reduce_domain 將原始影像和經過 shape_trans 的影像相減，產生一個只含 ROI 的原始影像。之後在這個 ROI，程式會檢查較暗的區域，這個較黑的區域是金屬接線的地方

```
reduce_domain (Bond, Die, DieGrey)
threshold (DieGrey, Wires, 0, 50)
fill_up_shape (Wires, WiresFilled, 'area', 1, 100)
```

由 opening_circle 指令來移除不符的結構，然後把物體抽出變成一個獨立的物的之後再用 select_shape 找出圓形物體，並且排列物體，就能擷取想要的特徵了。

```
opening_circle (WiresFilled, Balls, 15.5)
connection (Balls, SingleBalls)
select_shape (SingleBalls, IntermediateBalls, 'circularity', 'and', 0.85,
1.0)
sort_region (IntermediateBalls, FinalBalls, 'first_point', 'true', 'column')
smallest_circle (FinalBalls, Row, Column, Radius)
```

3.9.2 使用 Fuzzy 測量來檢查 IC

範例: examples\hdevelop\Applications\Measure\fuzzy measure pin.dev

這個範例的任務是檢查在圖 3.16 中 IC 的接腳的寬度和距離。

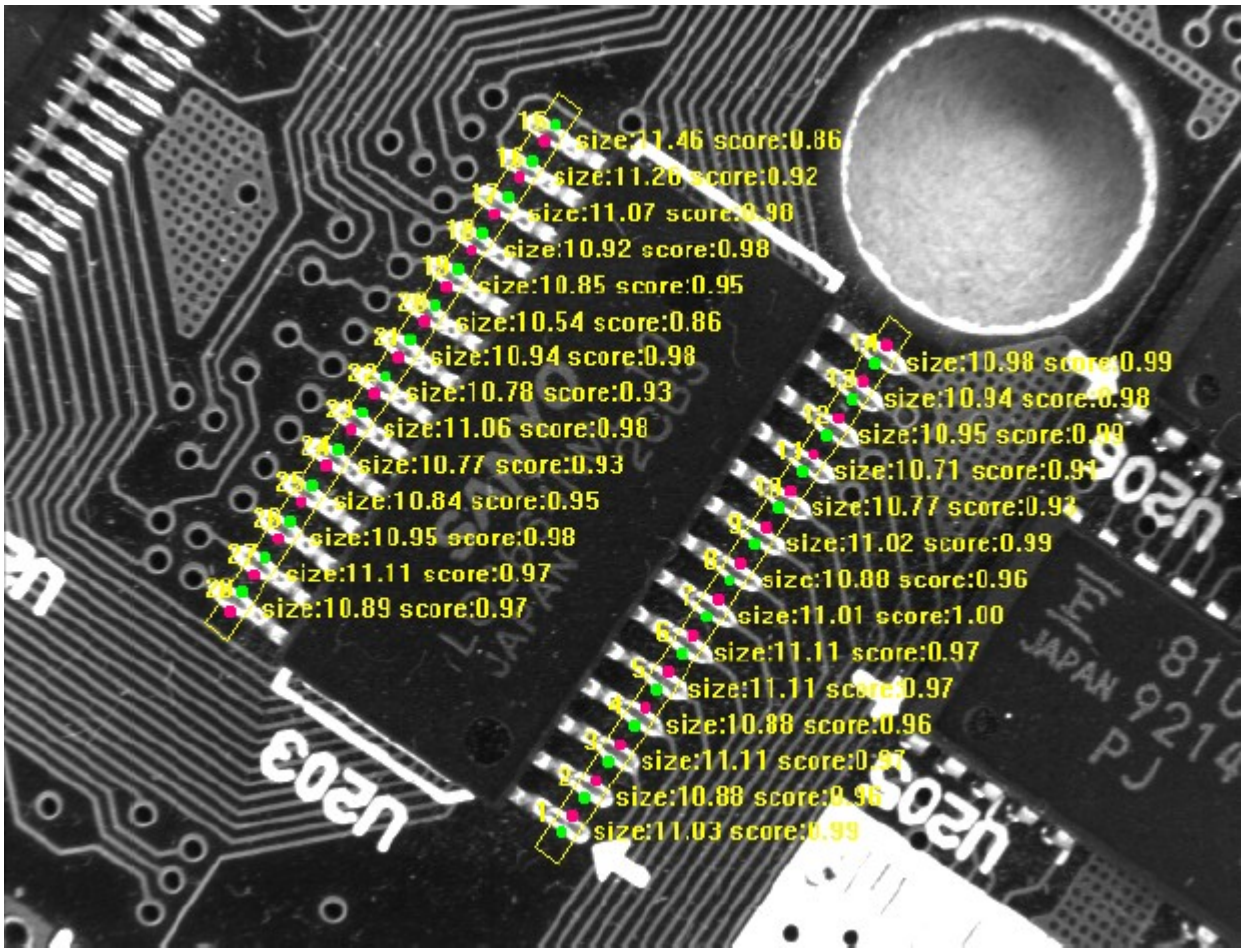


圖 3.16：測量接腳的寬度和距離

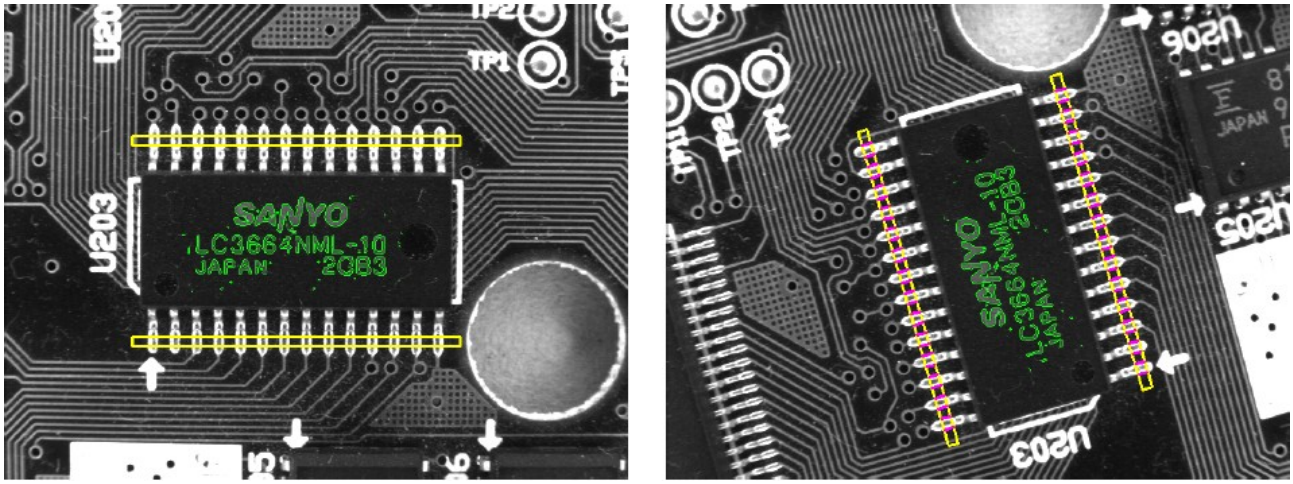
在這個範例中照明的條件是相當的困難，因爲照明會很明顯的影響到每個接腳的四個邊。因此可使用 Fuzzy 的方式來限制正確的接腳數目。

```
gen_measure_rectangle2 (305.5, 375.5, 0.982, 167, 7.5, Width, Height,
'nearest_neighbor', MeasureHandle1)
create_funct_1d_pairs ([0.0, 0.3], [1.0,0.0], FuzzyAbsSizeDiffFunction)
set_fuzzy_measure_norm_pair (MeasureHandle1, 11.0, 'size_abs_diff',
FuzzyAbsSizeDiffFunction)
fuzzy_measure_pairs (Image, MeasureHandle1, 1, 30, 0.5, 'positive',
RowEdgeFirst1, ColumnEdgeFirst1, AmplitudeFirst1, RowEdgeSecond1,
ColumnEdgeSecond1, AmplitudeSecond1, RowEdgeMiddle1,
ColumnEdgeMiddle1, FuzzyScore1, IntraDistance1, InterDistance1)
```

3.9.3 測量移動中 IC 的接腳

範例: examples\hdevelop\Applications\FA\pm measure board.dev

這個範例的任務是測量晶片接腳的位置(見圖 3.17)，因爲晶片可以用不同的位置和角度下出現，所以在測量 ROI 時必需先使用校準。



(a) (b)

圖 3.17:(a)測量 ROI 的模型影像(b)對已校準的 ROI 上測量其接腳

在這個案中，經由以型狀導向的匹配，搜尋晶片上的印刷字完成定位。

```
gen_rectangle1 (Rectangle, Row1, Column1, Row2, Column2)
reduce_domain (Image, Rectangle, ImageReduced)
create_shape_model (ImageReduced, 4, 0, rad(360), rad(1), 'none',
'use_polarity', 30, 10, ModelID)
```

找到印刷字之後，就可以知道測量的 ROI 位置與印刷字的位置間的相對關係。

```
find_shape_model (ImageCheck, ModelID, 0, rad(360), 0.8, 1, 0.5,
'interpolation', 4, 0.9, RowCheck, ColumnCheck, AngleCheck, Score)
hom_mat2d_identity (HomMat2DIdentity)
hom_mat2d_translate (HomMat2DIdentity, RowCheck, ColumnCheck,
HomMat2DTranslate)
hom_mat2d_rotate (HomMat2DTranslate, AngleCheck, RowCheck, ColumnCheck,
```

```
HomMat2DRotate)
affine_trans_pixel (HomMat2DRotate, Rect1Row, Rect1Col, Rect1RowCheck,
Rect1ColCheck)
```

然後，產生一個測量的工具並進行量測

```
gen_measure_rectangle2 (Rect1RowCheck, Rect1ColCheck, AngleCheck,
RectLength1, RectLength2, Width, Height, 'bilinear', MeasureHandle1)
measure_pairs (ImageCheck, MeasureHandle1, 2, 90, 'positive', 'all',
RowEdgeFirst1, ColumnEdgeFirst1, AmplitudeFirst1, RowEdgeSecond1,
ColumnEdgeSecond1, AmplitudeSecond1, IntraDistance1, InterDistance1)
```

3.9.4 產生一個 mosaic 影像

範例: examples\hdevelop\Tools\2D-Transformations\gen projective mosaic.dev

這個範例的任務是在圖 3.18 產生一個長條型的電路板影像。若使用標準的影像大小，則部份的影像會被清掉，而解決的方法就是從各種不同的位置來擷取影像，以產生 mosaic 影像。

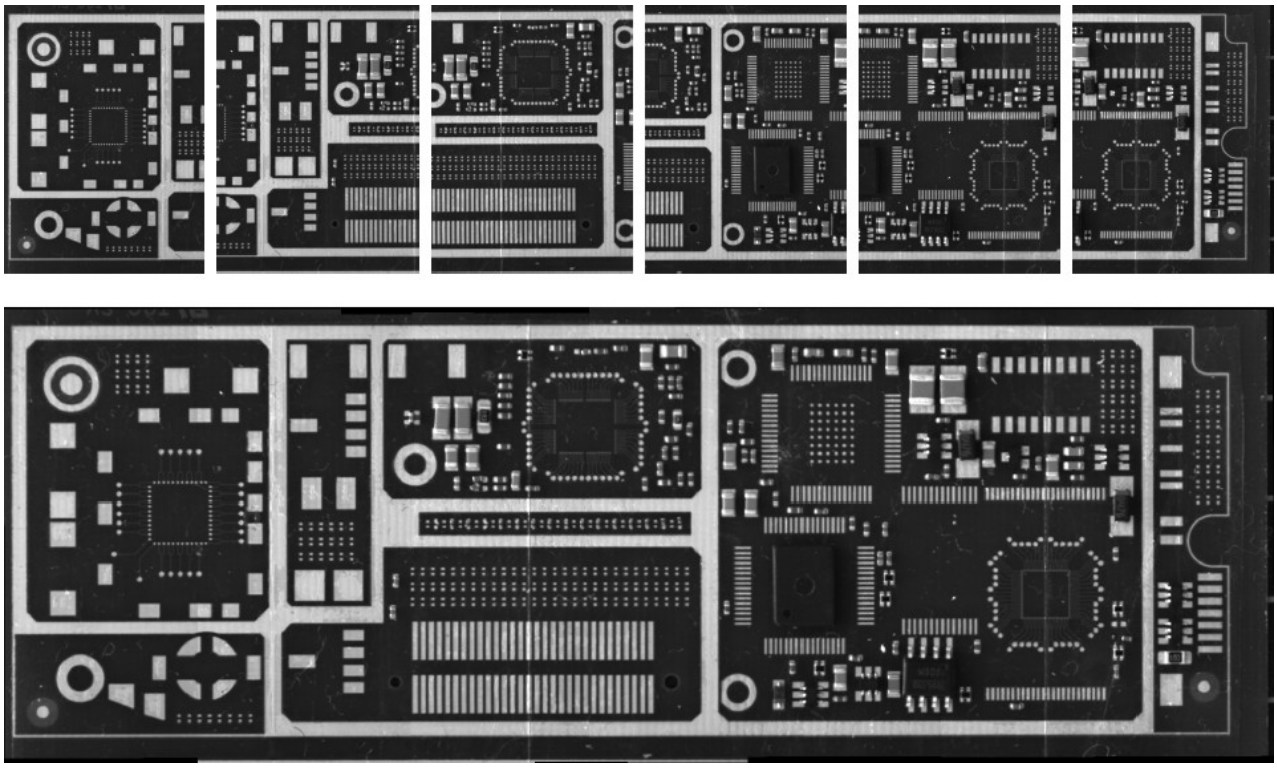


圖 3.18:從若干重疊的影像產生一個 mosaic 影像

將電路板各部份重疊的影像輸入到程式中；第一個步驟，在每張影像擷取其特徵部份。這些部份會被輸入到以特徵點爲匹配的輸入中。之後的每個步驟，相繼的兩張影像可以配對起來。將這張影像配對到下張影像直到結束，就是這項處理的結果。

```
points_foerstner (ImageF, 1, 2, 3, 200, 0.3, 'gauss', 'false',
RowJunctionsF, ColJunctionsF, CoRRJunctionsF, CoRCJunctionsF,
CoCCJunctionsF, RowAreaF, ColAreaF, CoRRAreaF, CoRCAreaF, CoCCAreaF)
points_foerstner (ImageT, 1, 2, 3, 200, 0.3, 'gauss', 'false',
RowJunctionsT, ColJunctionsT, CoRRJunctionsT, CoRCJunctionsT,
CoCCJunctionsT, RowAreaT, ColAreaT, CoRRAreaT, CoRCAreaT, CoCCAreaT)
proj_match_points_ransac (ImageF, ImageT, RowJunctionsF, ColJunctionsF,
RowJunctionsT, ColJunctionsT, 'ncc', 21, 0, 0, 480, 640, 0, 0.5,
'gold_standard', 1, 4364537, ProjMatrix, Points1, Points2)
ProjMatrices := [ProjMatrices,ProjMatrix]
```

將這些影像收集起來，最後建構一個單一高解析影像的完整 PCB

```
gen_projective_mosaic (Images, MosaicImage, 2, From, To, ProjMatrices,
'default', 'false', MosaicMatrices2D)
```

3.9.5.利用顏色找出電路板上的元件

範例:examples\hdevelop\Applications\FA\ic.dev

這個範例的任務是在圖 3.19 中，去找出電路板上所有的元件

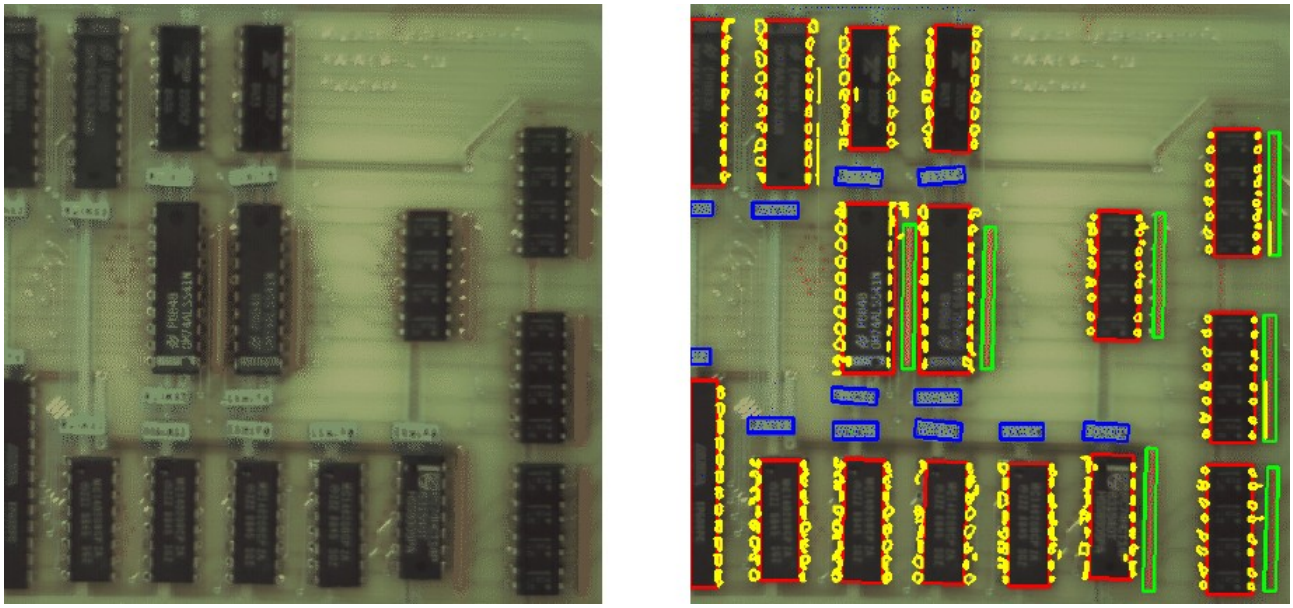


圖 3.19(a)原始影像(b)擷取 IC、電阻與電容

輸入的資料是一張有顏色的影像，它不同的顏色找出元件，像是電容或電阻：使用 color space 的轉換，利用不同顏色的值找出相對應的元件。接下來的程式會去擷取電阻、電容，其擷取方式，用相同的方法進行。

```
decompose3 (Image, Red, Green, Blue)
trans_from_rgb (Red, Green, Blue, Hue, Saturation, Intensity, 'hsv')
threshold (Saturation, Colored, 100, 255)
reduce_domain (Hue, Colored, HueColored)
threshold (HueColored, Red, 10, 19)
connection (Red, RedConnect)
select_shape (RedConnect, RedLarge, 'area', 'and', 150.000000, 99999.000000)
shape_trans (RedLarge, Resistors, 'rectangle2')
```

IC 的擷取是較困難的，因爲在這個步驟中明亮的部份不允許一個簡單的二值化處理，所以另一個替代的方法是，先選擇較暗的區域，然後使用 dilation 結合。

```
threshold (Intensity, Dark, 0, 50)
dilation_rectangle1 (Dark, DarkDilation, 14, 14)
connection (DarkDilation, ICLarge)
```

之後，重覆的進行分離的動作，直到相連接的元件分離爲止。

```
add_channels (ICLarge, Intensity, ICLargeGray)
threshold (ICLargeGray, ICDark, 0, 50)
shape_trans (ICDark, IC, 'rectangle2')
```

爲了找出接腳，在每一個 IC 的左邊和右邊產生了許多個較小的 ROI

```
dilation_rectangle1 (IC, ICDilation, 5, 1)
difference (ICDilation, IC, SearchSpace)
dilation_rectangle1 (SearchSpace, SearchSpaceDilation, 14, 1)
union1 (SearchSpaceDilation, SearchSpaceUnion)
```

在這些區域中附近的亮點會被發現

```
reduce_domain (Intensity, SearchSpaceUnion, SearchGray)
mean_image (SearchGray, SearchMean, 15, 15)
dyn_threshold (SearchGray, SearchMean, PinsRaw, 5.000000, 'light')
connection (PinsRaw, PinsConnect)
fill_up (PinsConnect, PinsFilled)
select_shape (PinsFilled, Pins, 'area', 'and', 10, 100)
```

3.9.6 其它的範例

HDevelop
- examples\application guide\3d machine vision\hdevelop\
height above reference plane from stereo.dev
從雙像立體視覺中，使用高度資訊擷取晶片
--->在 Application Note on 3D Machine Vision 中有詳細的描述

- examples\application guide\3d machine vision\hdevelop\mosaicking.dev
使用 mosaick 來合併 BGA 的局部影像成爲一個較大的影像
--->在 Application Note on 3D Machine Vision 有詳細的描述

- examples\application guide\shape matching\hdevelop\
first example shape matching.dev
介紹 HALCON 型狀導向的匹配
--->在 Application Note on Shape-Based Matching 有詳細的描述

- examples\application guide\shape matching\hdevelop\process shape model.dev
經由更改顯示型狀模型的結果產生另一個 ROI 的模型
--->在 Application Note on Shape-Based Matching 有詳細的描述

- examples\application guide\shape matching\hdevelop\synthetic circle.dev
使用合成的模型(圓)來搜尋電路板上的電容
--->在 Application Note on Shape-Based Matching 有詳細的描述

- examples\hdevelop\Applications\FA\ball seq.dev
檢視 ball bonding(複合影像)

- examples\hdevelop\Applications\FA\board.dev
發現遺失的桿接點

- examples\hdevelop\Applications\FA\cbm dip switch.dev
找尋 dip switches 並使用以元件導向的匹配來測試它的狀態
--->在 Quick Guide 有詳細的描述

- examples\hdevelop\Applications\Measure\measure pin.dev
測量 IC 的接腳
--->在 Quick Guide 有詳細的描述

- examples\hdevelop\Applications\Stereo\board components.dev
Segments board components by height using binocular stereo
--->在 Quick Guide 有詳細的描述

- examples\hdevelop\Manuals\HDevelop\ball.dev
檢查 bonding of balls
--->在 HDevelop User's Manual 有詳細的描述

- examples\hdevelop\Manuals\HDevelop\ic.dev
結合不同的分割方法

--->在 HDevelop User's Manual 有詳細的描述

- examples\hdevelop\Matching\Component-Based\cbm modules simple.dev
在電路板中使用元件導向的匹配來找尋一個模型
--->在 Quick Guide 有詳細的描述

- examples\hdevelop\Matching\Component-Based\cbm sbm.dev
比較元件導向的匹配和型狀導向的匹配

- examples\hdevelop\Morphology\Gray-Values\pcb inspection.dev
使用灰階值型態學在 PCB 上找尋缺陷

- examples\hdevelop\Regions\Transformations\distance transform.dev
檢查相似的電路板之缺陷

- examples\hdevelop\Segmentation\Classification\class 2dim sup.dev
使用 2D 像元來分割影像

- examples\hdevelop\Segmentation\Classification\class ndim box.dev
使用 hyper-cuboids 分類像元

- examples\hdevelop\Segmentation\Classification\class ndim norm.dev
使用 hyper-spheres 分割像元

- examples\hdevelop\XLD\Transformation\union contours xld.dev
連接原本相連的線之片斷

C++

- examples\cpp\source\fuzzy measure pin.cpp
使用 fuzzy 量測來量 IC 的接腳

- examples\mfc\Matching\Matching.cpp
使用 HALCON/C++和 MFC 找尋 IC，並產生 HALCON 視窗

- examples\mfc\MatchingCOM\Matching.cpp
使用 HALCON/COM 和 MFC 找尋 IC

- examples\mfc\MatchingExtWin\Matching.cpp
使用 HALCON/C++和 MFC 找尋 IC，並顯示在現存的視窗中

- examples\motif\Matching\matching.cpp
使用 HALCON/C++和 Motif 來找出 IC

Visual Basic

- examples\vb\Tools\Matching\matching.vbp
找出電路板上的 IC 並測量接腳的距離

- examples\vb\Tools\Measure\measure.vbp

用參數的 interactive control 測量接腳

C#

- examples\c#\Matching\Matching.csproj

找出電路板上的 IC 並測量接腳的距離

C

- examples\c\example multithreaded1.c

在 Parallel HALCON 下使用多執行緒

Delphi

- examples\delphi\Matching\matching.dpr

找出電路板上的IC並測量接腳的距離