機器視覺函式庫開發與應用

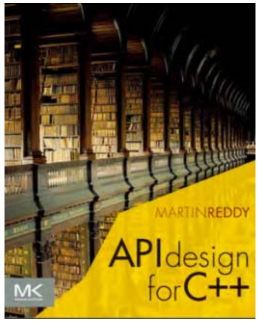
機器視覺函式庫

A novel Fourier descriptor based image alignment algorithm for automatic optical inspection. J. Visual Communication and Image Representation 20(3): 178-189 (2009)

A hybrid defect detection for in-tray semiconductor chip. International Journal of Advanced Manufacturing Technology 65: 43-56 (2013)

An accelerating CPU based correlation-based image alignment for real-time automatic optical inspection. Computers & Electrical Engineering 49: 207-220 (2016)

Optical Character Recognition Based on GA-based Optimal Neural Network. ICASI 2017

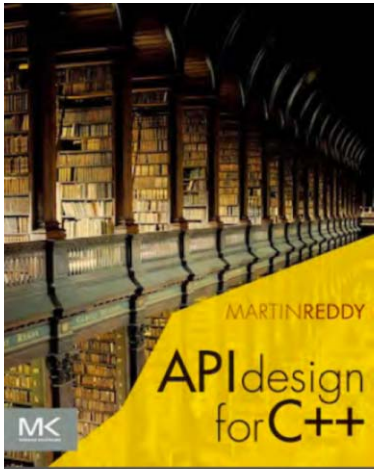


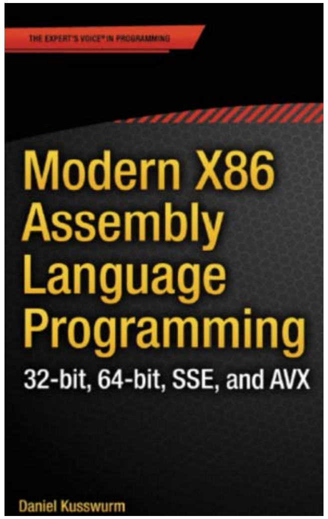
大綱

- 機器視覺函式庫開發之注意事項
- 多媒體指令集
- 函式庫開發與應用實務
  - 算術及邏輯運算
  - 遮罩運算
  - 影像二值化
  - 型態學
  - 影像量測
- 進階機器視覺演算法介紹 – 2D 視覺定位

機器視覺函式庫開發之注意事項

- 在工業應用中，重視:(1) 準確性,(2) 速度 及 (3) 穩定性。
- 除了持續開發影像演算法外，如何結合 CPU 所提供的 SIMD 指令集提升效能，則是商用函式庫之核心技術。
- 本課程將探討如何將常用之影像處理演算法，利用 SIMD 指令集加速，以符合實際應用上之需求。
- 除 CPU 之 SIMD 指令集外，利用 GPU 或 FPGA 進行影像的平行處理，也是函式庫開發之主流。



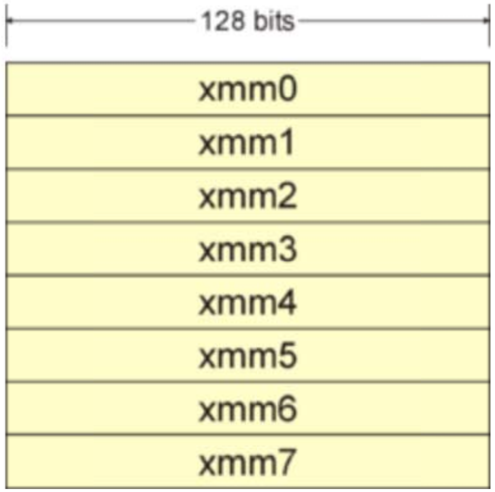


多媒體指令集

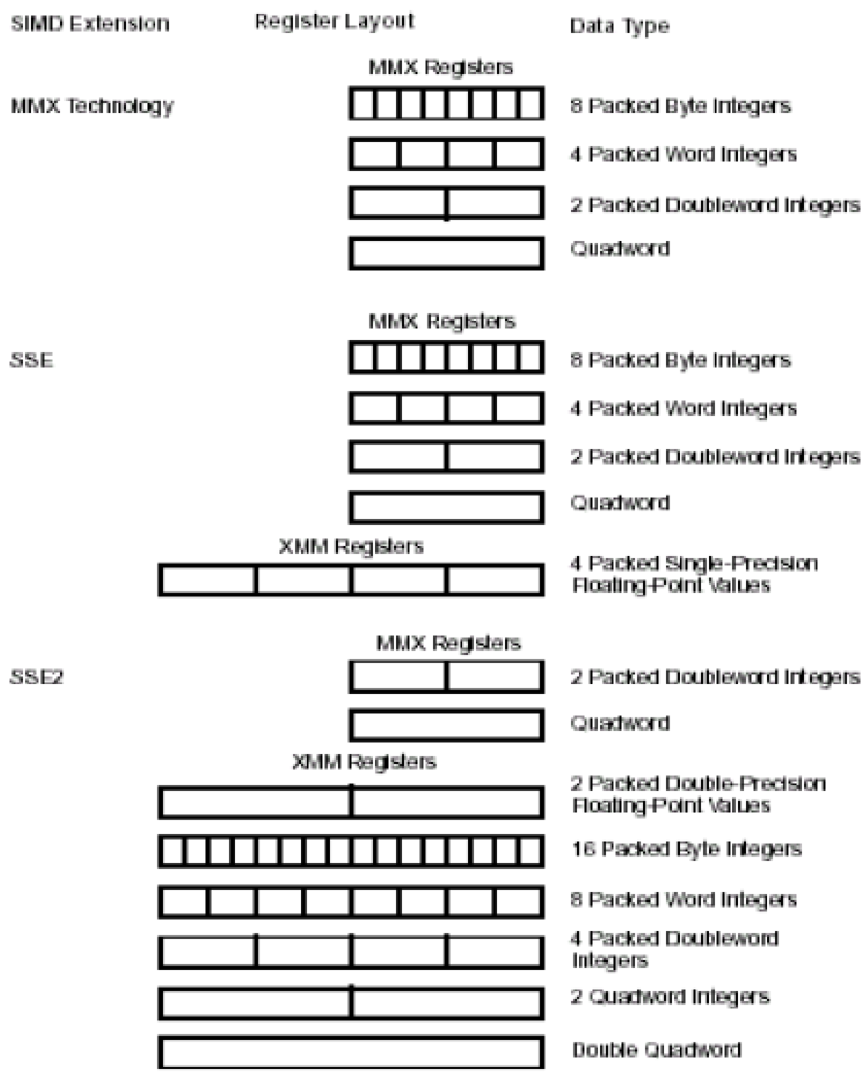
- 多媒體指令集 (Multimedia Extension/Matrix Math Extensions, MMX) 是 Intel 在 1997 年首先應用在 CPU-Pentium 中的 SIMD 技術。 !! ARM 的 SIMD 技術是 NEON
- 其核心架構是利用 CPU 中的浮點運算暫存器，置換為 MMX 用的暫存器，其名稱為 MM0-MM7 ，共 8 個暫存器。
- MMX 指令集可以切割其 64Bits 的暫存器為 8 個 Byte (8 Bits) / 4 個 Word (16 Bits) / 2 個 Double Word (32Bits) 。
- 在影像處理中，以灰階影像的 Pixel (8Bits) 為主要處理單位，因此可一次運算多筆資料，發揮 SIMD 的最大功效。

多媒體指令集

- Intel 後續提出 Streaming SIMD Extensions (SSE) 架構，在 CPU 中新增 8 個獨立的 128Bit 浮點暫存器，不需要佔用原本的浮點暫存器。
- 在 Intel SSE2 的指令手冊中， SSE2 讓整數指令全面支援 XMM 128Bit 暫存器，也就是在部份影像處理中，可以同步運算到 16 個像素 (8 Bits) 點。



多媒體指令集



SIMD Extension
Register Layout
Data Type
MMX Registers
MMX Technology
8 Packed Byte Integers
4 Packed Word Integers
2 Packed Doubleword Integers
Quadword
MMX Registers
SSE
8 Packed Byte Integers
4 Packed Word Integers
2 Packed Doubleword Integers
Quadword
XMM Registers
4 Packed Single-Precision Floating-Point Values
MMX Registers
SSE2
2 Packed Doubleword Integers
Quadword
XMM Registers
2 Packed Double-Precision Floating-Point Values
16 Packed Byte Integers
8 Packed Word Integers
4 Packed Doubleword Integers
2 Quadword Integers
Double Quadword

多媒體指令集

- Intel 最新提出 Advanced Vector Extensions (AVX) 架構，延伸 SSE 指令集，將暫存器 XMM 128Bit 暫存器提升至 YMM 256bit ，增加一倍的資料寬度。
- Fused Multiply Accumulate (FMA) 則是 Intel 的 AVX 擴充指令集。
- AVX2 指令集是 Haswell 處理器升級的一個重點，通過增加對 256 位元整數 SIMD 指令的支援、2 個新 FMA 單元和一些位移指令，讓處理器的整數和浮點運算速度翻倍。
- 由於最新的 Intel CPU 才有支援 AVX2/AVX512 及 FMA 指令集，考慮兼容性，目前商用機器視覺函式庫普遍還是使用 SSE 指令集架構進行加速。

多媒體指令集

- 在 Visual Studio 中開發基於 SSE2 指令集的程式，需要

  #include <emmintrin.h>

  #include <tmmintrin.h>

- 常用 SSE2 資料結構：

```
typedef union __declspec(intrin_type) _CRT_ALIGN(16) __m128i {
    __int8          m128i_i8[16];
    __int16         m128i_i16[8];
    __int32         m128i_i32[4];
    __int64         m128i_i64[2];
    unsigned __int8     m128i_u8[16];
    unsigned __int16    m128i_u16[8];
    unsigned __int32    m128i_u32[4];
    unsigned __int64    m128i_u64[2];
} __m128i;

typedef struct __declspec(intrin_type) _CRT_ALIGN(16) __m128d {
    double          m128d_f64[2];
} __m128d;
```


請參考：Intel C++ SSE2 Intrinsics Reference

多媒體指令集

New Data Type	MMX(TM) Technology	Streaming SIMD Extensions	Streaming SIMD Extensions 2	Streaming SIMD Extensions 3
__m64	Available	Available	Available	Available
__m128	Not available	Available	Available	Available
__m128d	Not available	Not available	Available	Available
__m128i	Not available	Not available	Available	Available

多媒體指令集

■ 常用 SSE2 指令集存取指令：

Intrinsic Name	Operation	Instruction
mm_load_si128	Load	MOVDQA
_mm_loadu_si128	Load	MOVDQU
_mm_loadl_epi64	Load and zero	MOVQ

```
__m128i _mm_load_si128(__m128i const*p)
```

Loads 128-bit value. Address p must be 16-byte aligned.

R
*p

```
__m128i _mm_loadu_si128(__m128i const*p)
```

Loads 128-bit value. Address p not need be 16-byte aligned.

R
*p

```
__m128i _mm_loadl_epi64(__m128i const*p)
```

Load the lower 64 bits of the value pointed to by p into the lower 64 bits of the result, zeroing the upper 64 bits of the result.

R0	R1
*p[63:0]	0x0

多媒體指令集

- 常用 SSE2 指令集存取指令：

Intrinsic Name	Operation	Corresponding SSE2 Instruction
_mm_stream_si128	Store	MOVNTDQ
_mm_stream_si32	Store	MOVNTI
_mm_store_si128	Store	MOVDQA
_mm_storeu_si128	Store	MOVDQU
_mm_maskmoveu_si128	Conditional store	MASKMOVDQU
_mm_storel_epi64	Store lowest	MOVQ

```
void _mm_storeu_si128(__m128i *p, __m128i b)
```

Stores 128-bit value. Address p need not be 16-byte aligned.

*p
a

□ 影像處理 - 算術及邏輯運算

函式名稱	功能說明
iImg_ImageAdd	影像相加
iImg_ImageSub	影像相減
iImg_ImageMul	影像相乘(指定變數)
iImg_ImageDiv	影像相除(指定變數)
iImg_ImageShiftLeft	像素位元左移
iImg_ImageShiftRight	像素位元右移
iImg_ImageBirwiseAND	像素位元 AND
iImg_ImageBirwiseOR	像素位元 OR
iImg_ImageBirwiseXOR	像素位元 XOR
iImg_ImageEqual	像素邏輯相等
iImg_ImageNOTEqual	像素邏輯相斥
iImg_ImageInvert	影像反向

影像相加

- 影像相加所需 SSE2 指令：

```
__m128i _mm_adds_epu8(__m128i a, __m128i b)
```

Adds the 16 unsigned 8-bit integers in a to the 16 unsigned 8-bit integers in b using saturating arithmetic.

R0	R1	...	R15
UnsignedSaturate (a0 + b0)	UnsignedSaturate (a1 + b1)	...	UnsignedSaturate (a15 + b15)

影像相減

- 影像相減所需 SSE2 指令：

```
__m128i _mm_subs_epu8 (__m128i a, __m128i b)
```

Subtracts the 16 unsigned 8-bit integers of b from the 16 unsigned 8-bit integers of a using saturating arithmetic.

R0	R1	...	R15
UnsignedSaturate (a0 - b0)	UnsignedSaturate (a1 - b1)	...	UnsignedSaturate (a15 - b15)

影像相乘

□ 影像相乘所需 SSE2 指令：

`__m128i _mm_set1_epi16(short w)`

Sets the 8 signed 16-bit integer values to w.

R0	R1	...	R7
w	w	w	w

`__m128i_mm_mullo_epi16(__m128i a, __m128i b)`

Multiplies the 8 signed or unsigned 16-bit integers from a by the 8 signed or unsigned 16-bit integers from b. Packs the lower 16-bits of the 8 signed or unsigned 32-bit results.

R0	R1	...	R7
(a0 * b0)[15:0]	(a1 * b1)[15:0]	...	(a7 * b7)[15:0]

`__m128i _mm_unpackhi_epi8(__m128i a, __m128i b)`

Interleaves the upper 8 signed or unsigned 8-bit integers in a with the upper 8 signed or unsigned 8-bit integers in b.

R0	R1	R2	R3	...	R14	R15
a8	b8	a9	b9	...	a15	b15

`__m128i _mm_setzero_si128()`

Sets the 128-bit value to zero.

R
0x0

`__m128i _mm_unpacklo_epi8(__m128i a, __m128i b)`

Interleaves the lower 8 signed or unsigned 8-bit integers in a with the lower 8 signed or unsigned 8-bit integers in b.

R0	R1	R2	R3	...	R14	R15
a0	b0	a1	b1	...	a7	b7

`__m128i _mm_packus_epi16(__m128i a, __m128i b)`

Packs the 16 signed 16-bit integers from a and b into 8-bit unsigned integers and saturates.

R0	...	R7	R8	...	R15
Unsigned Saturate(a0)	...	Unsigned Saturate(a7)	Unsigned Saturate(b0)	...	Unsigned Saturate(b15)

影像相乘

- 影像相除新增 SSE2 指令 (相較影像相乘):

```
__m128i _mm_slli_epi16(__m128i a, int count)
```

Shifts the 8 signed or unsigned 16-bit integers in a left by count bits while shifting in zeros.

R0	R1	...	R7
a0 << count	a1 << count	...	a7 << count

像素位元右移

- 位元右移新增 SSE2 指令：

```
__m128i _mm_srli_epi16(__m128i a, int count)
```

Shifts the 8 signed or unsigned 16-bit integers in a right by count bits while shifting in zeros.

R0	R1	...	R7
srl(a0, count)	srl(a1, count)	...	srl(a7, count)

像素位元 AND / OR / XOR

位元 AND 新增 SSE2 指令：

```
__m128i _mm_and_si128(__m128i a, __m128i b)
```

Computes the bitwise AND of the 128-bit value in a and the 128-bit value in b.

R0
a & b

Intrinsic Name	Operation	Corresponding SSE2 Instruction
_mm_and_si128	Computes AND	PAND
_mm_andnot_si128	Computes AND and NOT	PANDN
_mm_or_si128	Computes OR	POR
_mm_xor_si128	Computes XOR	PXOR

位元 OR 新增 SSE2 指令：

```
__m128i _mm_or_si128(__m128i a, __m128i b)
```

Computes the bitwise OR of the 128-bit value in a and the 128-bit value in b.

R0
a | b

位元 XOR 新增 SSE2 指令：

```
__m128i _mm_xor_si128(__m128i a, __m128i b)
```

Computes the bitwise XOR of the 128-bit value in a and the 128-bit value in b.

R0
a ^ b

像素邏輯相等/互斥

■ 像素邏輯相等 新增 SSE2 指令：

```
__m128i _mm_cmpeq_epi8(__m128i a, __m128i b)
```

Compares the 16 signed or unsigned 8-bit integers in a and the 16 signed or unsigned 8-bit integers in b for equality.

R0	R1	...	R15
(a0 == b0) ? 0xff : 0x0	(a1 == b1) ? 0xff : 0x0	...	(a15 == b15) ? 0xff : 0x0

■ 像素邏輯互斥 新增 SSE2 指令：

```
__m128i _mm_set1_epi8(char b)
```

Sets the 16 signed 8-bit integer values to b.

R0	R1	...	R15
b	b	b	b

```
__m128i _mm_subs_epi8(__m128i a, __m128i b)
```

Subtracts the 16 signed 8-bit integers of b from the 16 signed 8-bit integers of a using saturating arithmetic.

R0	R1	...	R15
SignedSaturate (a0 - b0)	SignedSaturate (a1 - b1)	...	SignedSaturate (a15 - b15)

影像反向

- 反向 新增 SSE2 指令：

```
__m128i _mm_subs_epu8 (__m128i a, __m128i b)
```

Subtracts the 16 unsigned 8-bit integers of b from the 16 unsigned 8-bit integers of a using saturating arithmetic.

R0	R1	...	R15
UnsignedSaturate (a0 - b0)	UnsignedSaturate (a1 - b1)	...	UnsignedSaturate (a15 - b15)

函式庫開發與應用實務

- 影像處理 - 遮罩運算

函式名稱	功能說明
iImg_Sobel	邊緣偵測
iImg_Laplacian	邊緣偵測
iImg_Robert	邊緣偵測
iImg_Prewitt	邊緣偵測
iImg_LaplacianSharping	邊緣強化
iImg_GaussianSmoothing3x3	3x3 高斯低通濾波
iImg_MeanSmoothing3x3	3x3 均值濾波
iImg_MedianFilter3x3	3x3 中值濾波
iImg_ArbitraryConvolution3x3	任意係數 3x3 遮罩濾波

函式庫開發與應用實務

□ 影像處理 - 二值化及形態學

函式名稱	功能說明
iImg_Threshold	二值化
iImg_DoubleThreshold	雙閥值二值化
iImg_OtsuThreshold	Otsu 二值化
iImg_Dilation	形態學 – 膨脹
iImg_Erosion	形態學 – 侵蝕

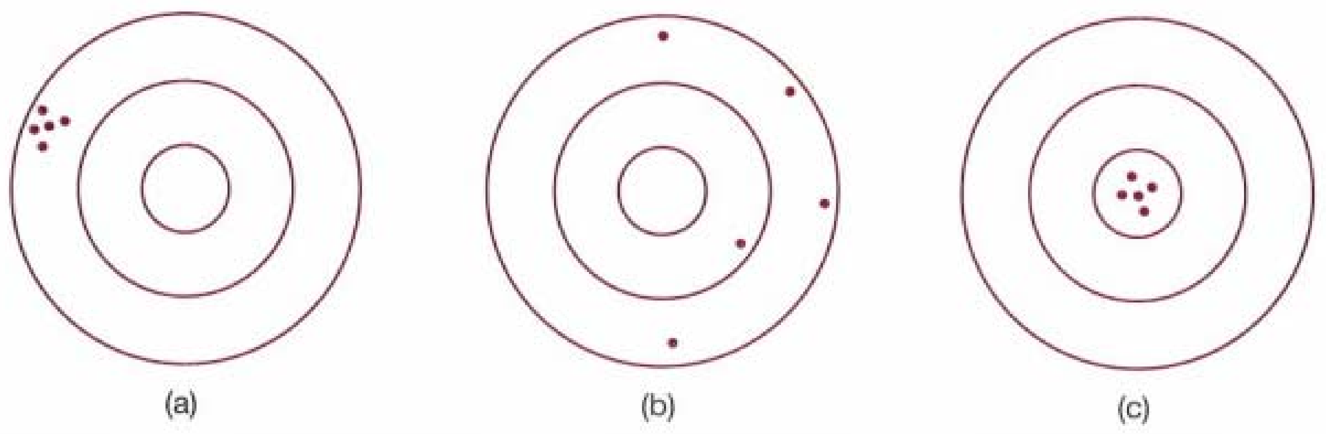
影像量測技術

準確度與精密度

- 準確度 (Accuracy) 與精密度 (Precision) 是量測學之基礎觀念，更普遍應用到與計量相關的多種領域。
- 準確度為觀測值或估值跟真值間差異的比較。
- 精密度為觀測值間相互之關係。
- 準確度之科學使用中，包含量測準確度 (Measurement Accuracy) 與分類準確度 (Classification Accuracy) 兩種，本課程主要探討量測準確度。

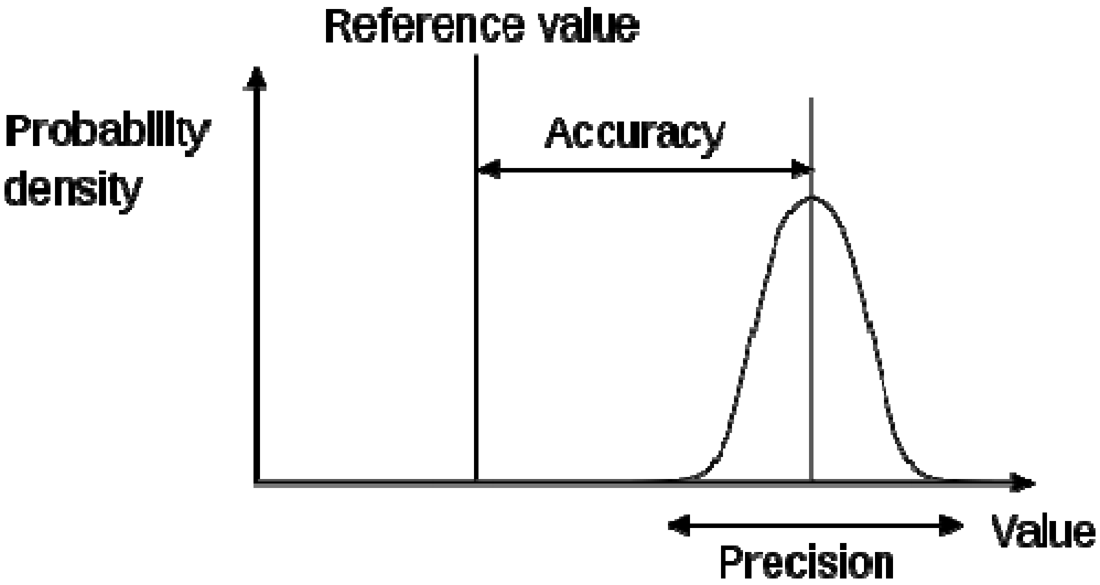
準確度與精密度

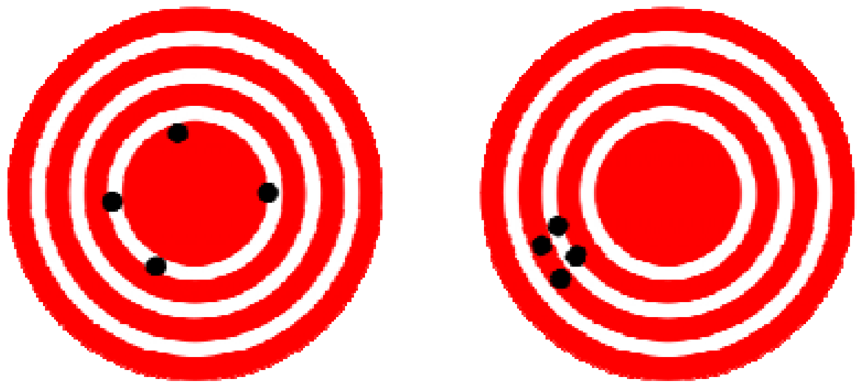
- 準確度表示觀測量與真實量間之絕對接近程度。
- 精密度為一群觀測量間之吻合程度，由其間之變異數或標準差大小評估。
- 量測系統之精密度也可稱為重現度，為在相同條件下重複性量測其值之重複程度。



Examples of precision and accuracy (a) Results are precise but not accurate. (b) Results are neither precise nor accurate. (c) Results are both precise and accurate.

準確度與精密度


Reference value
Probability density
Accuracy
Precision
Value

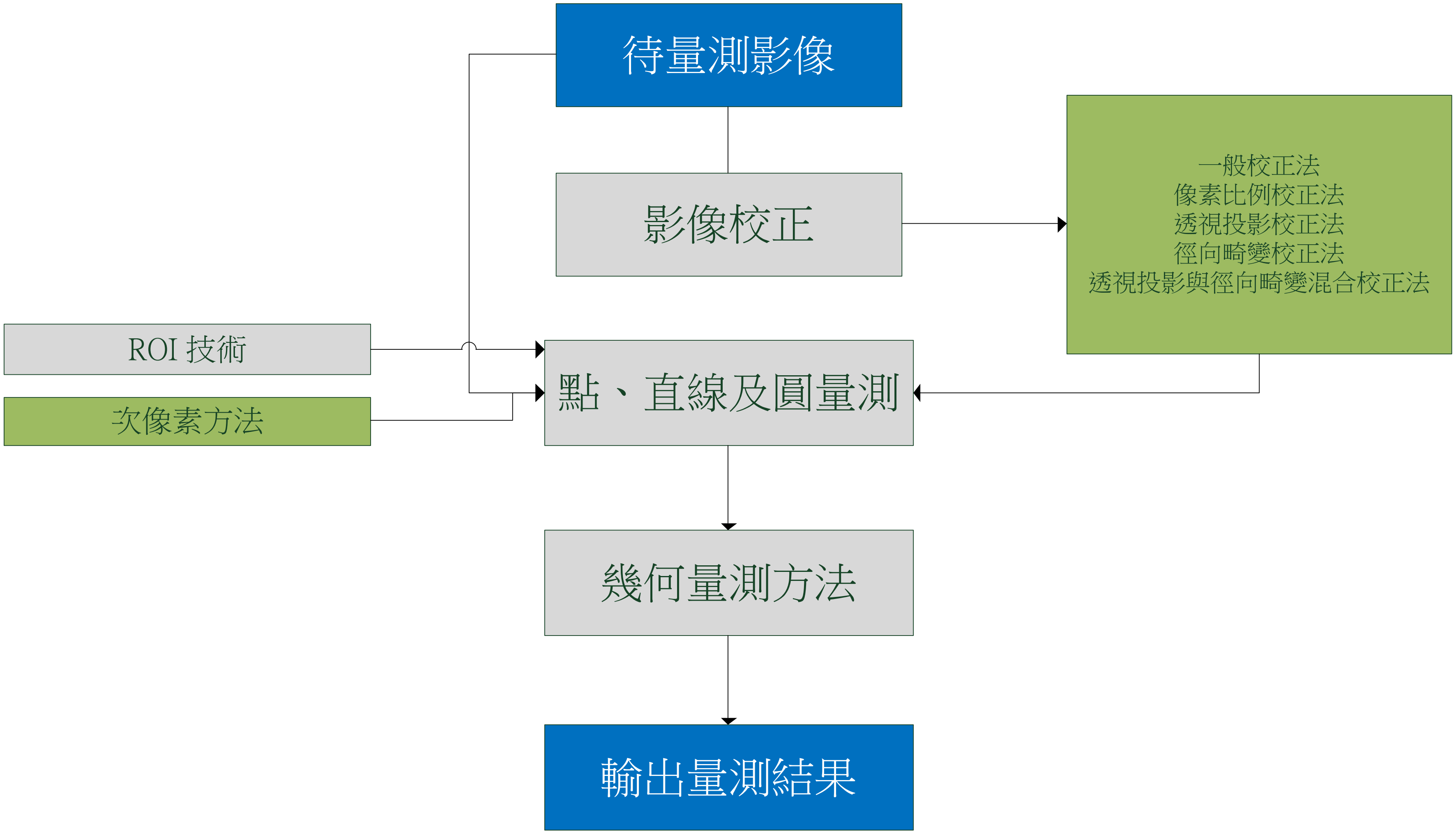


準確度與精密度

準確度與精密度

- 準確度 (Accuracy) 高是表示系統誤差小或不存在。
- 精密度等同隨機誤差的概念。
- 分析量測結果時，所謂的精度應包含系統誤差跟隨機誤差。
- 一個量測系統，必須要精密，才可以準確。

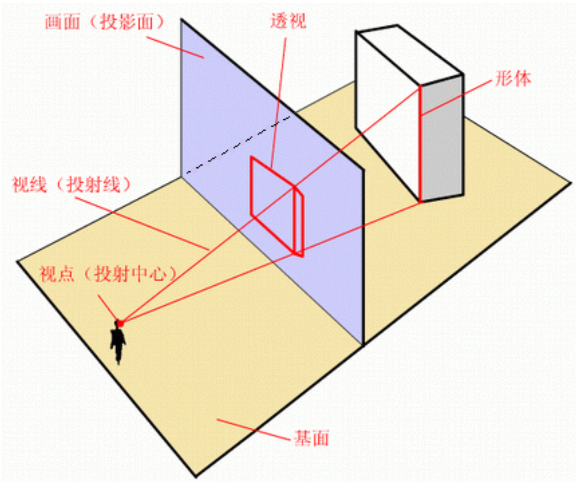
影像量測流程圖


待量測影像
影像校正
一般校正法
像素比例校正法
透視投影校正法
徑向畸變校正法
透視投影與徑向畸變混合校正法
ROI 技術
次像素方法
點、直線及圓量測
幾何量測方法
輸出量測結果

影像校正

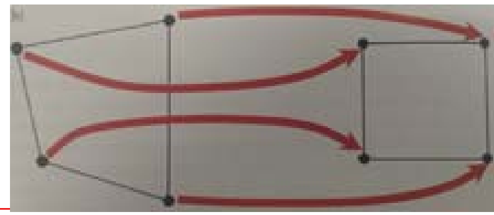
- 像素比例校正法：根據三組不共線點資料的影像座標與真實座標，校正出像素真實長度與寬度，輸出的單位即為校正時，點資料真實座標的單位。

- 透視投影校正法：校正因透視效應造成的畸變 (Perspective Distortion)。根據四組三三不共線的影像座標及真實座標資料，用以校正出真實座標與影像座標間的轉換關係。



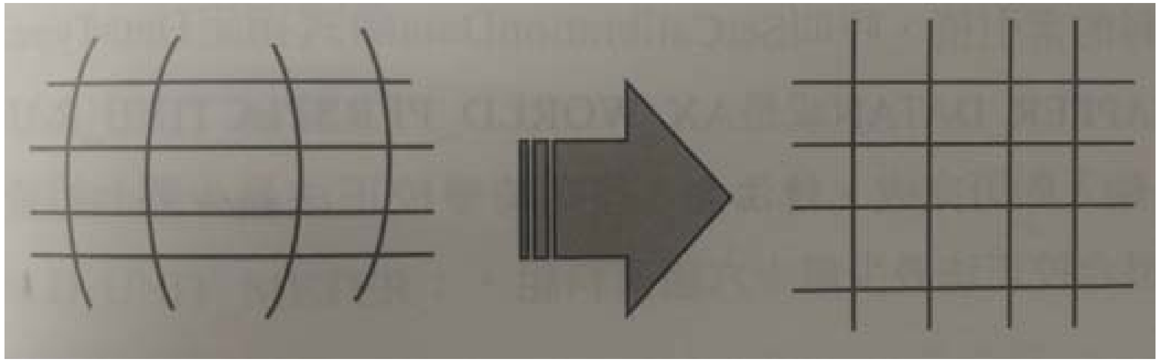
$$x' = k_1 x + k_2 y + k_3 xy + k_4$$

$$y' = k_5 x + k_6 y + k_7 xy + k_8$$

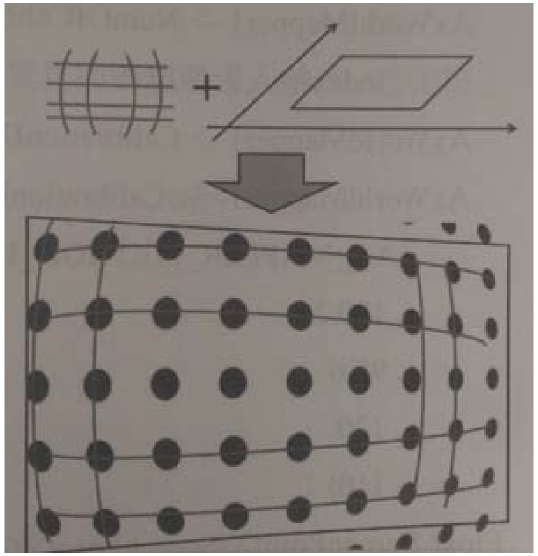


影像校正

- 徑向畸變：

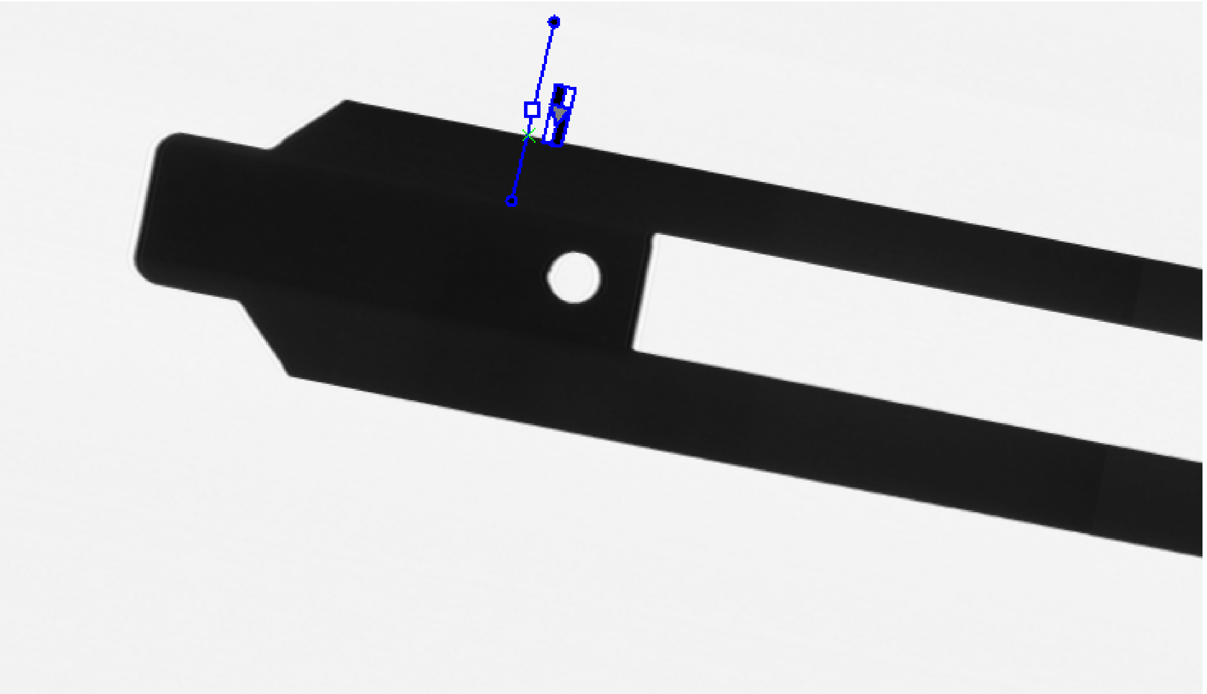


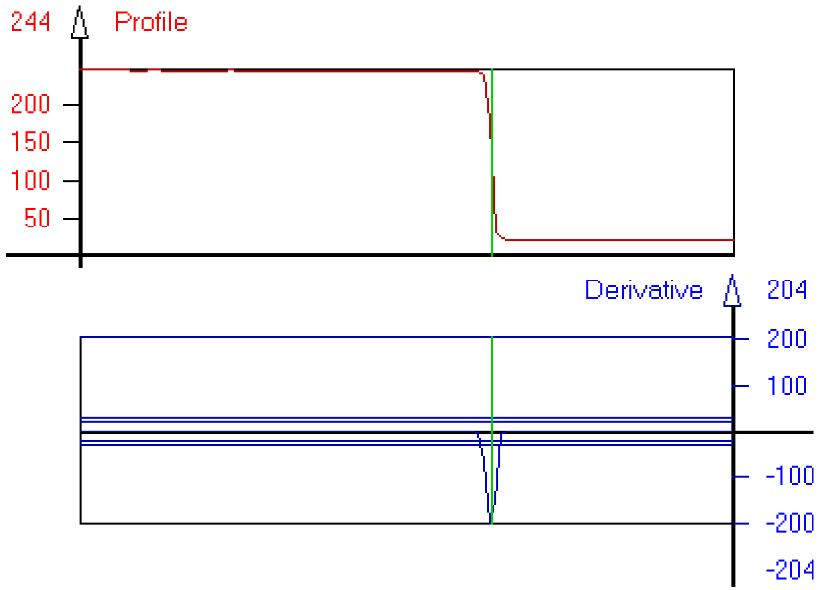
- 徑向與透視投影混合：



點量測方法 (Point Gauge)

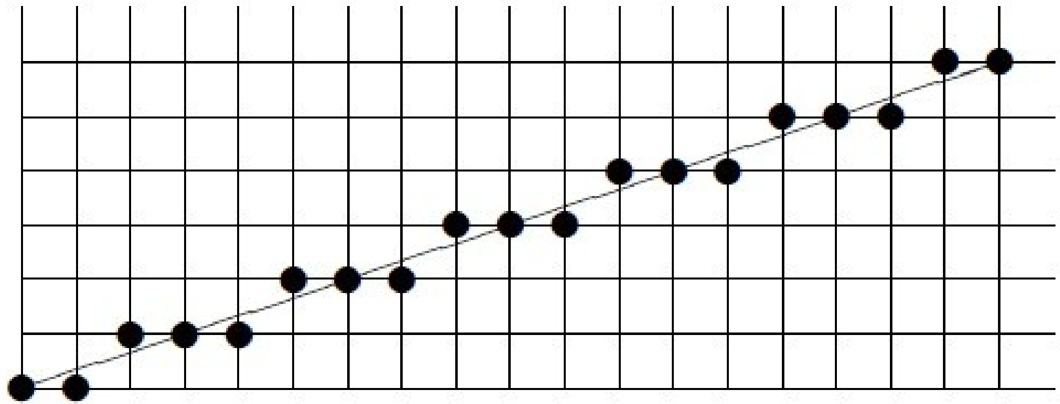
- 打背光後取得的影像如下：

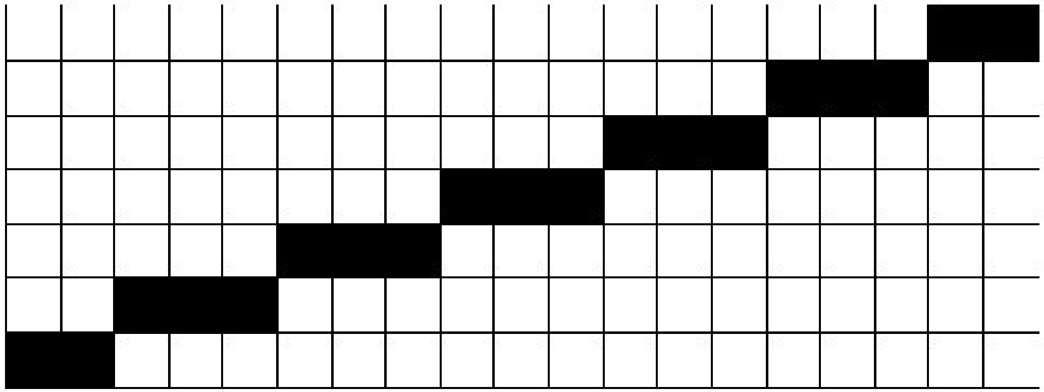




點量測方法 (Point Gauge)

- 數位直線生成演算法：
  - DDA 數值微分演算法
  - Bresenham 演算法





點量測方法 (Point Gauge)

◻ DDA 數值微分演算法

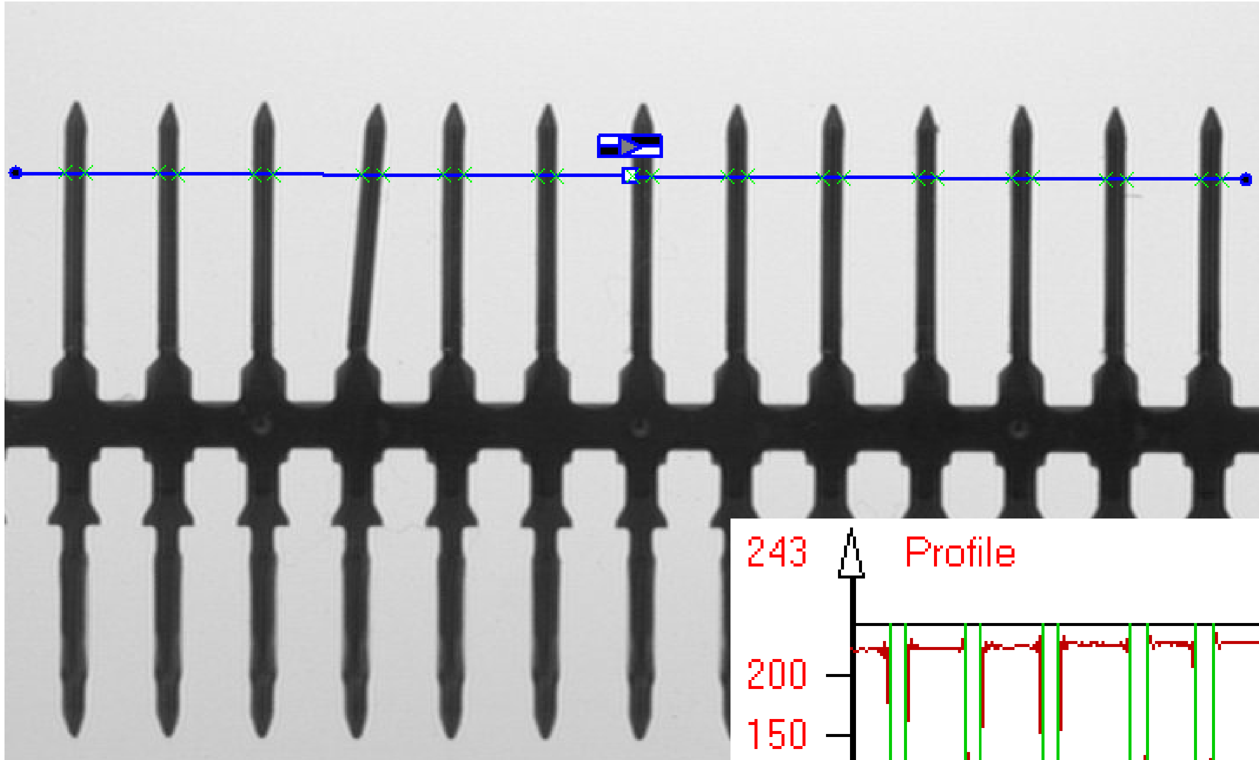
複雜度：加法 + 取整數

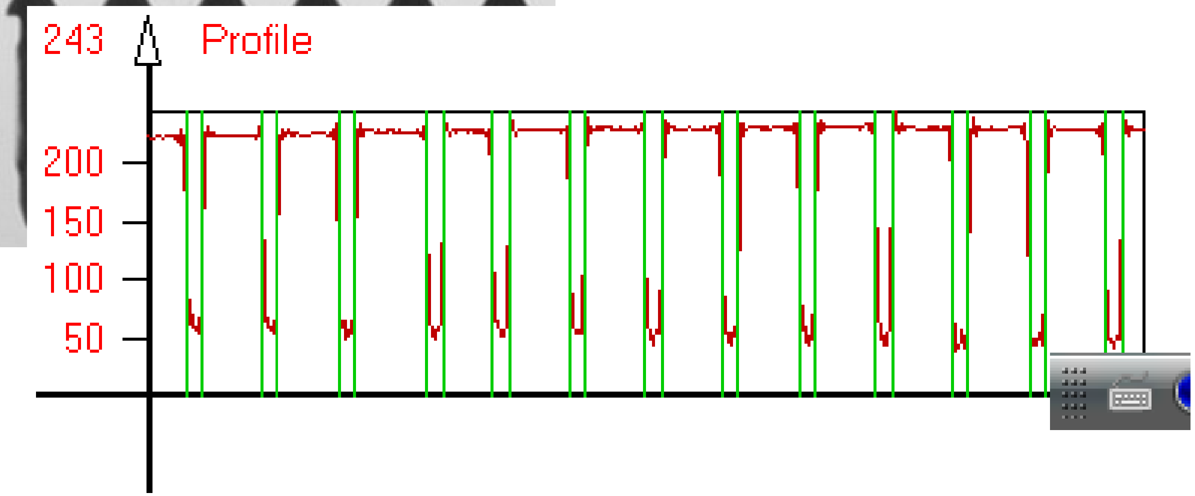
上述採用的增量计算方法稱为數值微分演算法 (Digital Differential Analyzer, DDA) 。數值微分演算法的本質，是用數值方法解微分方程，通過同时對 x 和 y 方向各增加一個小增量，來計算下一步的 x 、 y 值。

◻ Bresenham 演算法

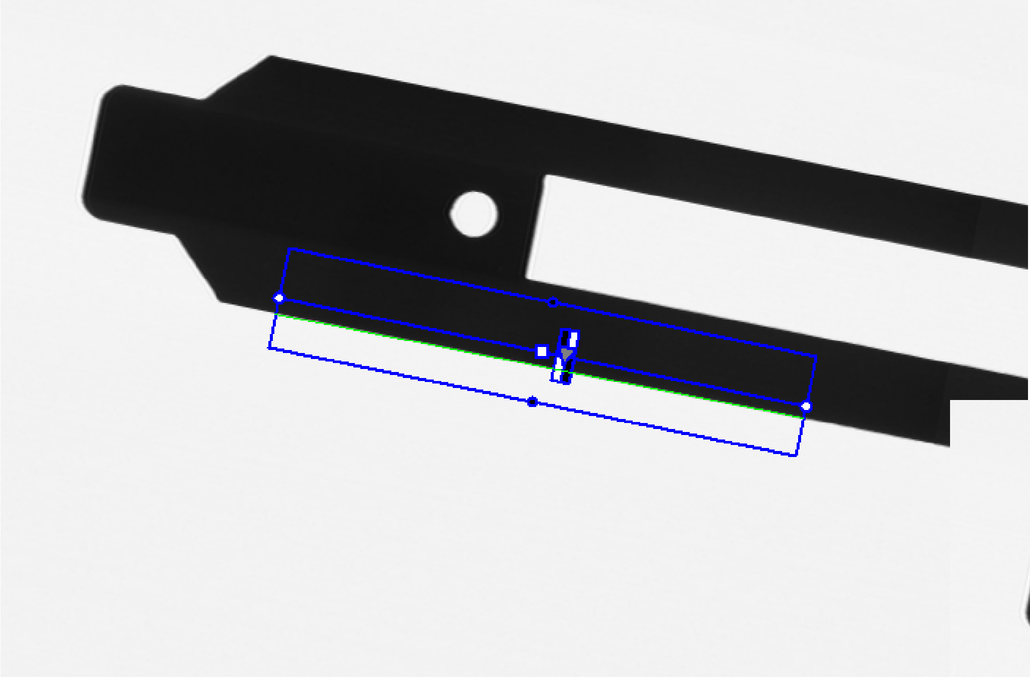
Bresenham 演算法是 1965 年提出，基本原理是：借助一個誤差量 (直線與當前實際繪製像素點的距離) ，來確定下一個像素點的位置。算法的巧妙之處在於採用增量計算，使得對於每一列，只要檢查誤差量的符号，就可以確定該下一列的像素位置。

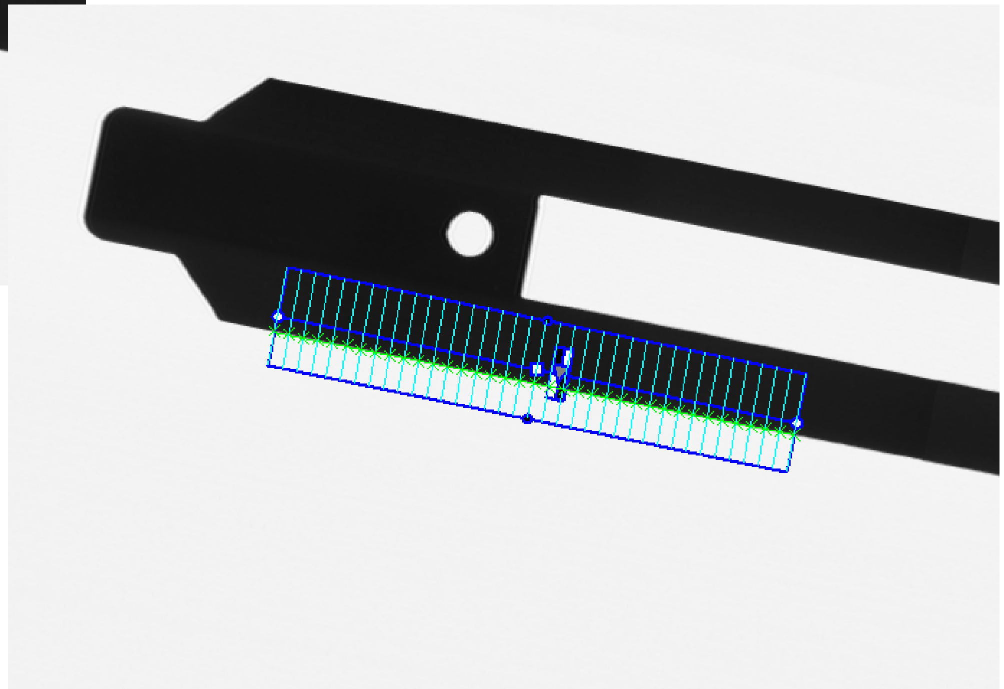
點量測方法 (Point Gauge)





直線量測方法 (Line Gauge)

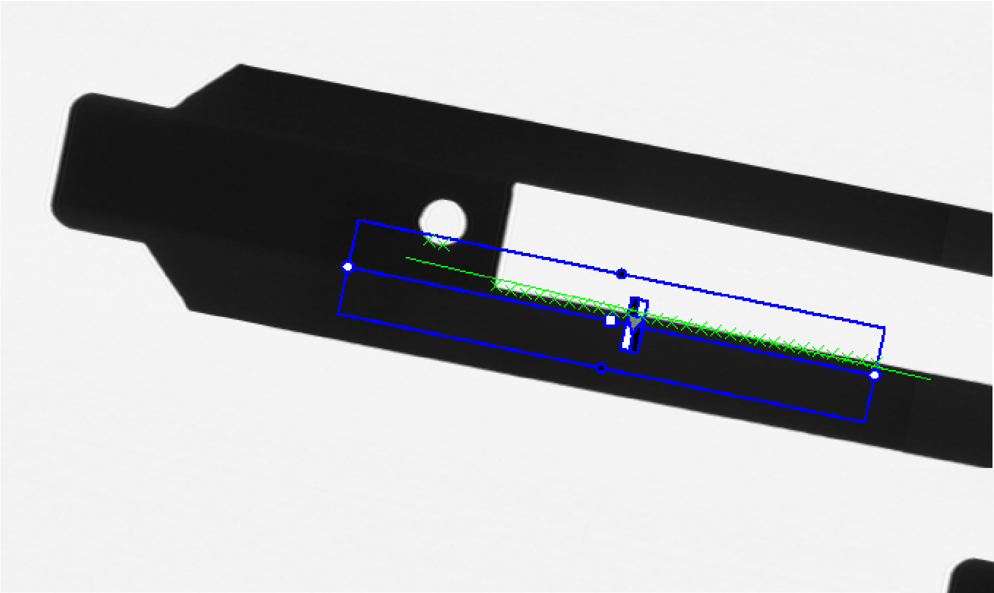


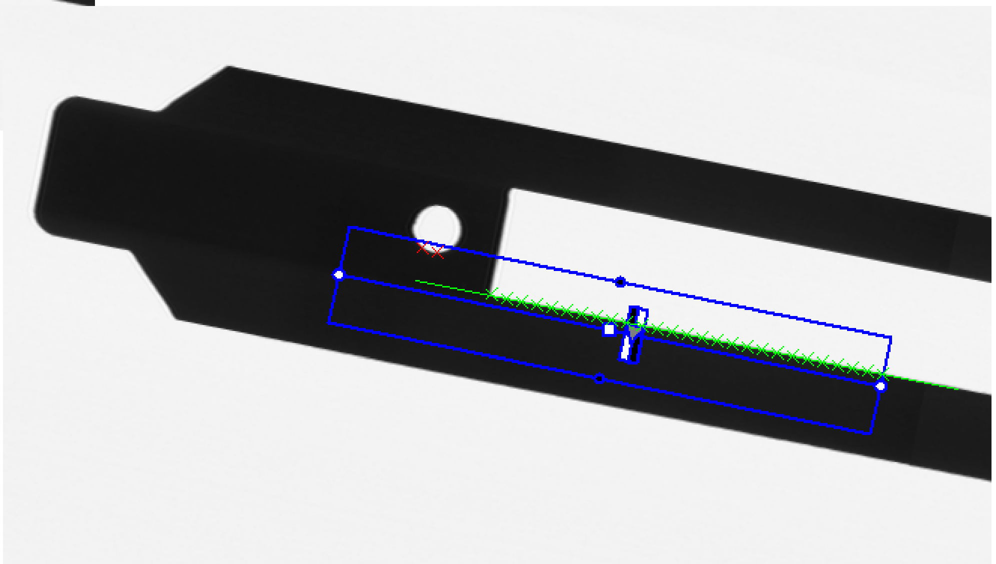


直線量測方法 (Line Gauge)

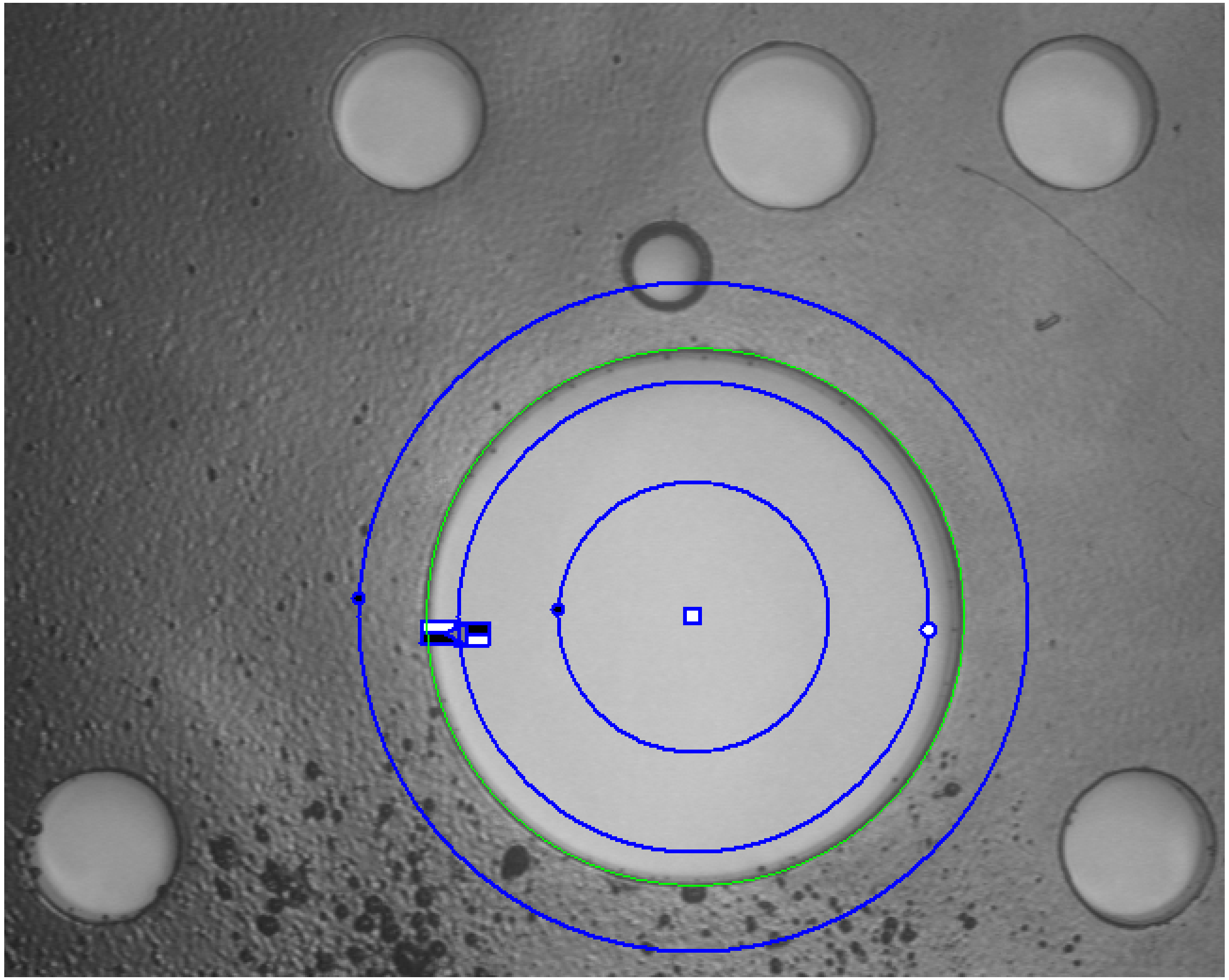
- Least-squares approximations (最小平方誤差)
  - 在科學或工程研究領域中，有限的實驗資料或實測資料，希望延伸實測資料之使用性，找出一代表性函數來近似，而此函數與資料之偏離誤差為最小，在此精神之下，所推導出來之結果，稱之為最小平方誤差法近似 (least-squares approximations) 。

直線量測方法 (Line Gauge)

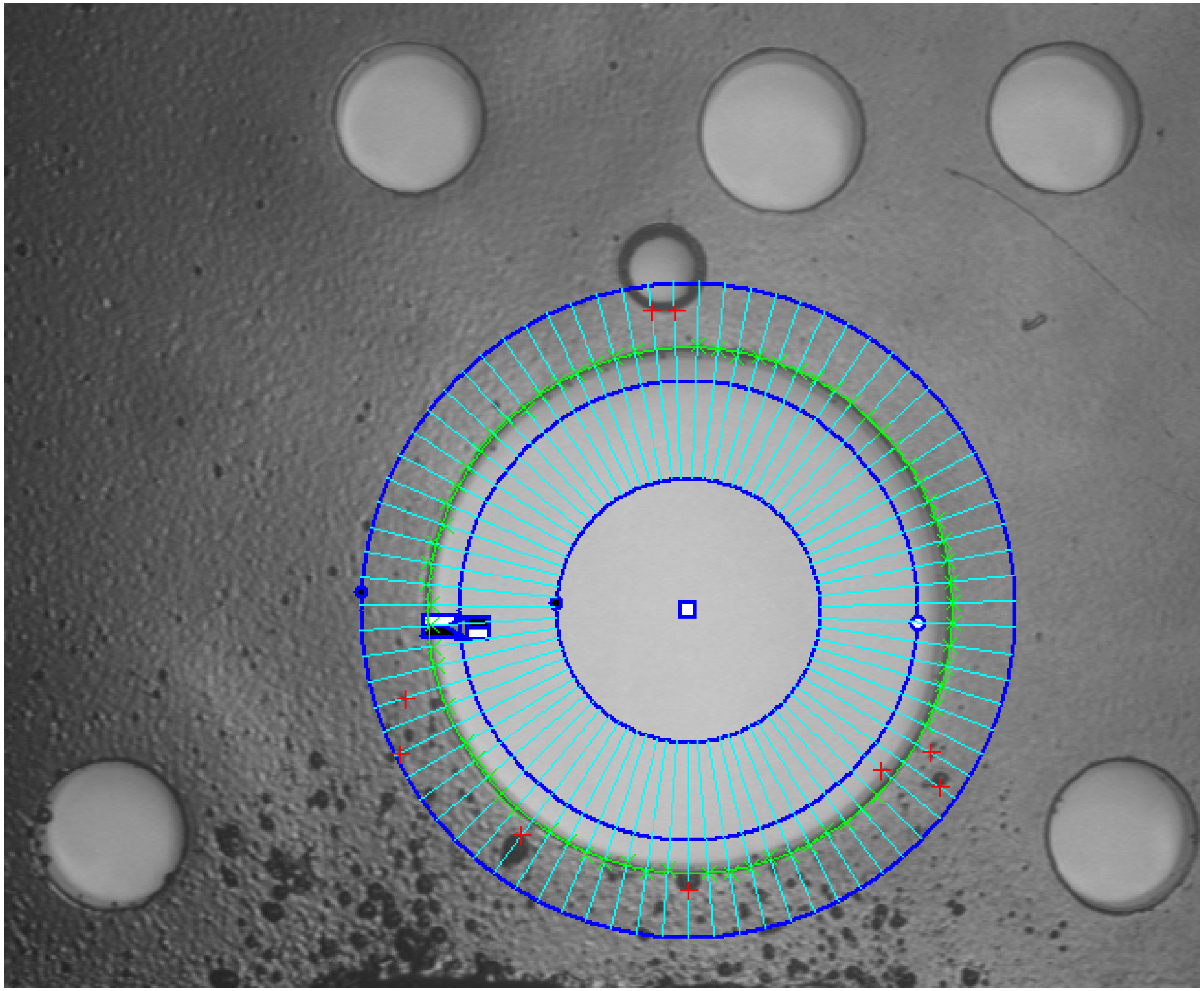




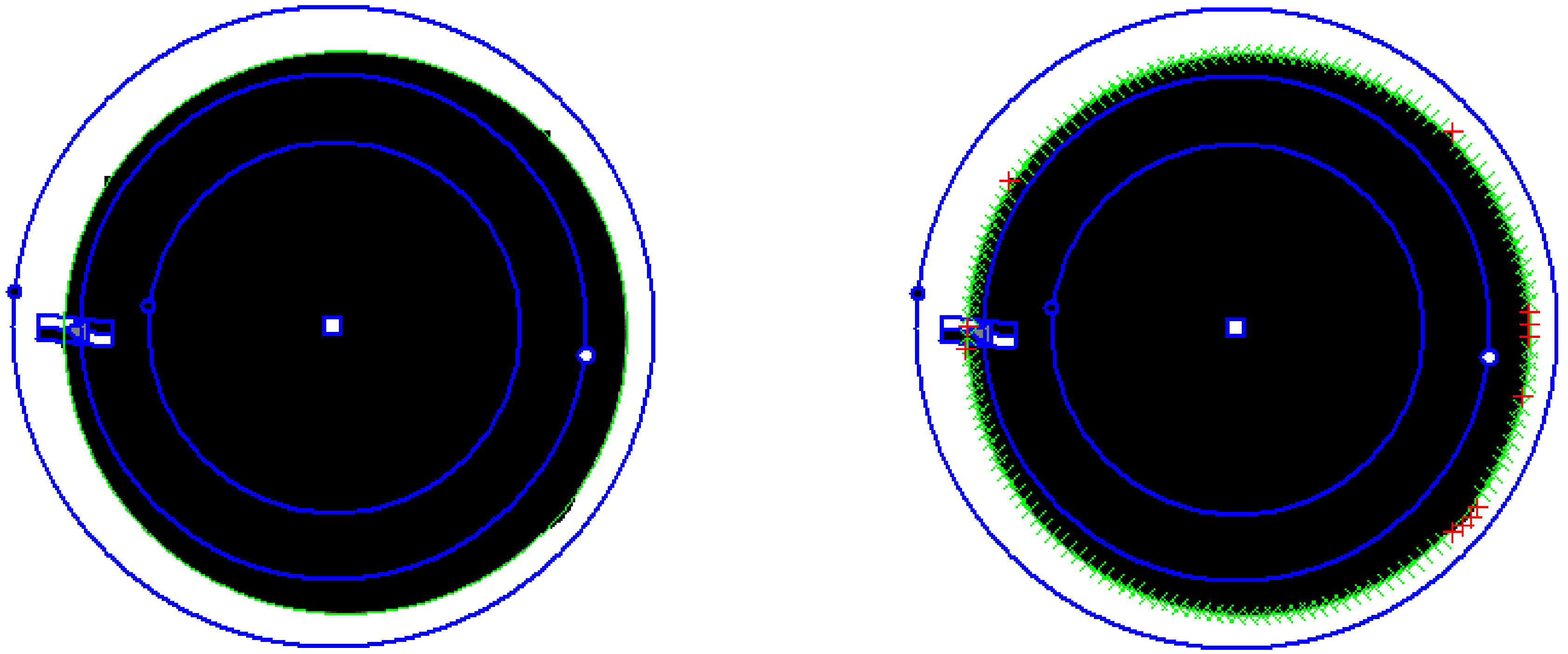
圓量測方法 (Circle Gauge)



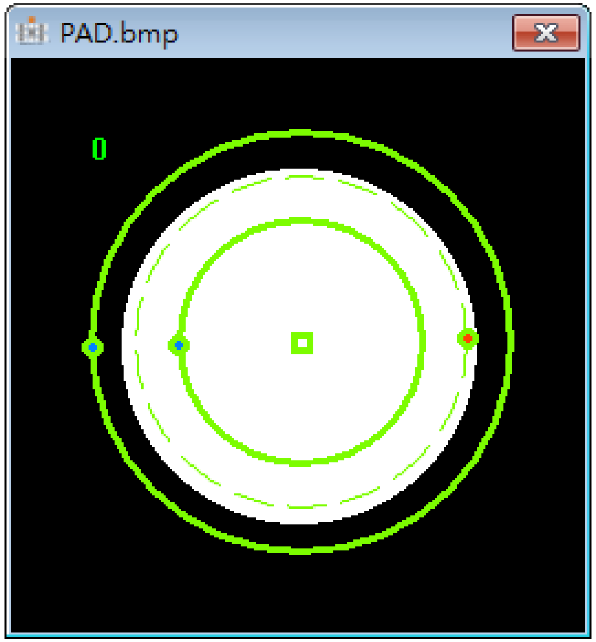
圓量測方法 (Circle Gauge)

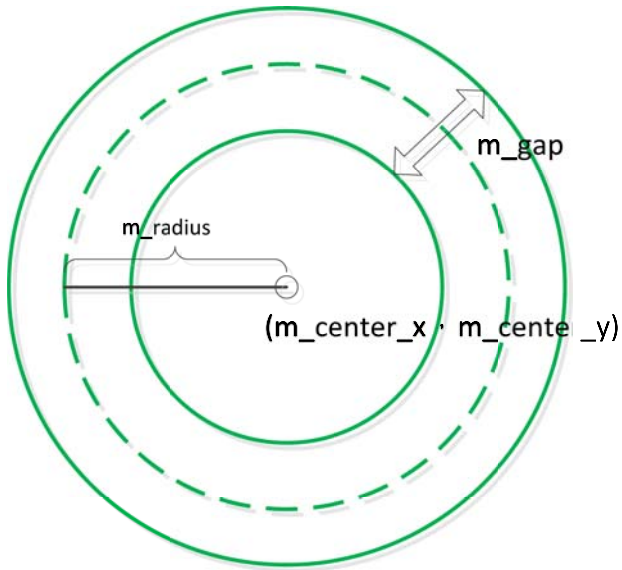


圓量測方法 (Circle Gauge)



圓量測方法 (Circle Gauge)

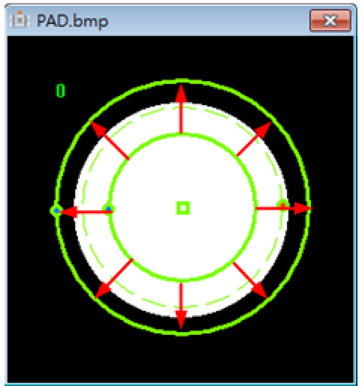


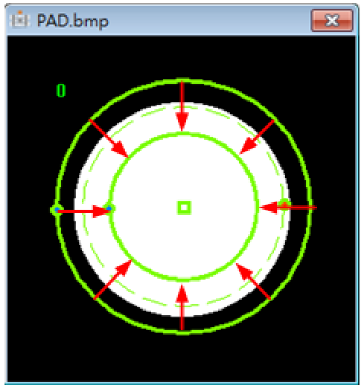


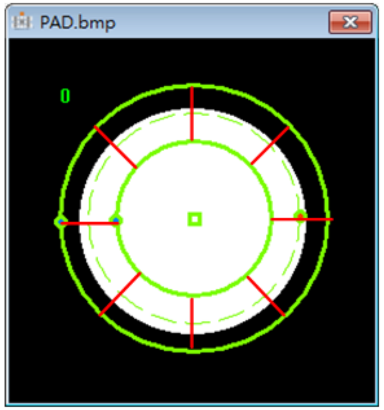
型態	名稱	說明
int	m_center_x	中心點 X 座標
int	m_center_y	中心點 Y 座標
int	m_ gap	環形間格
double	m_ radius	環形半徑

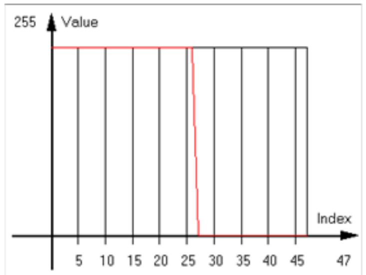
圓量測方法 (Circle Gauge)

- Edge mode
  - From begin
  - From end
  - Largest



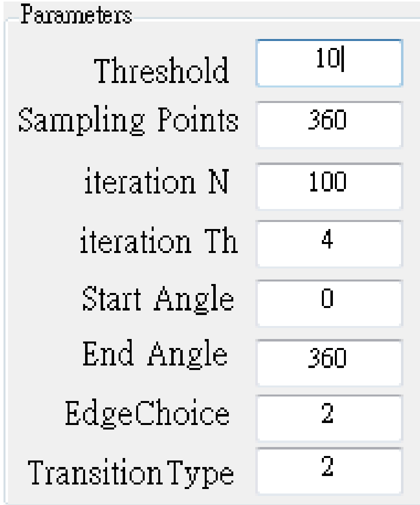




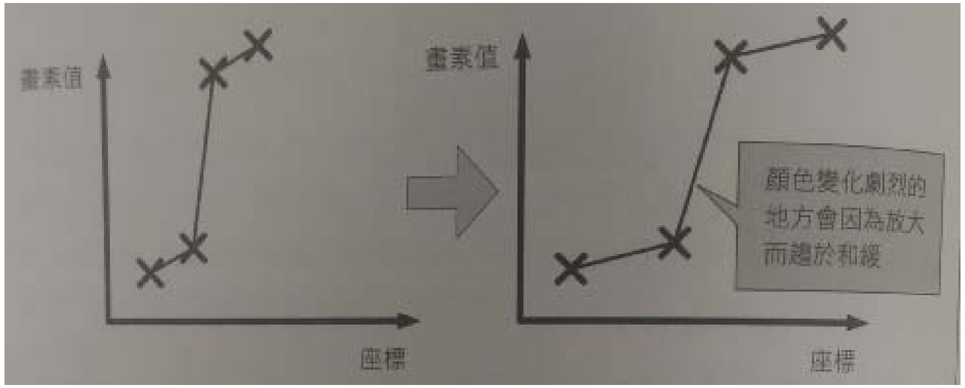


圓量測方法 (Circle Gauge)

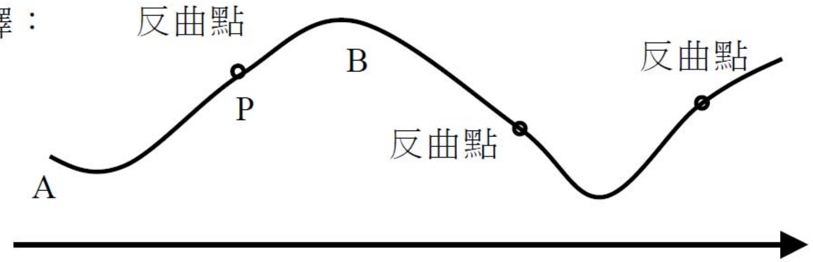
- Threshold ： 灰階變化量
- Sampling Points ： 取樣特徵點數量
- Start Angle ： 如果使用非全圓量測時， ROI 起始角度
- End Angle ： 如果使用非全圓量測時， ROI 終止角度
- Edge Choice ： 投射線上判斷特徵點的方向選擇
- Transition Type ： 邊緣偵測方向：黑白 :0 、白黑 :1 、 both:2
- Iteration N ： 圓估測中疊代次數
- Iteration Th ：疊代閥值

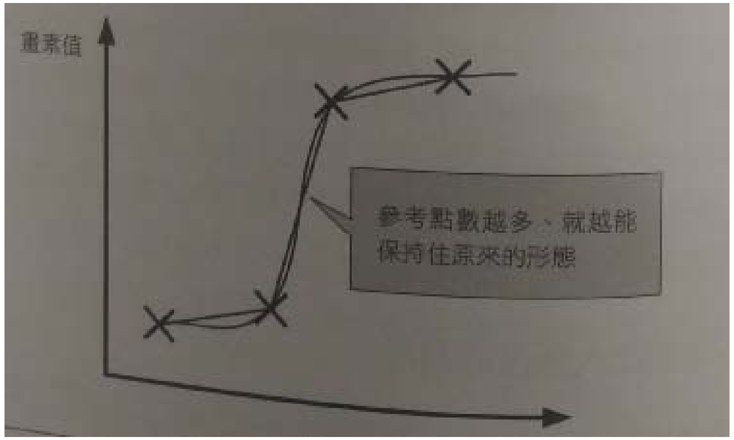


次像素方法



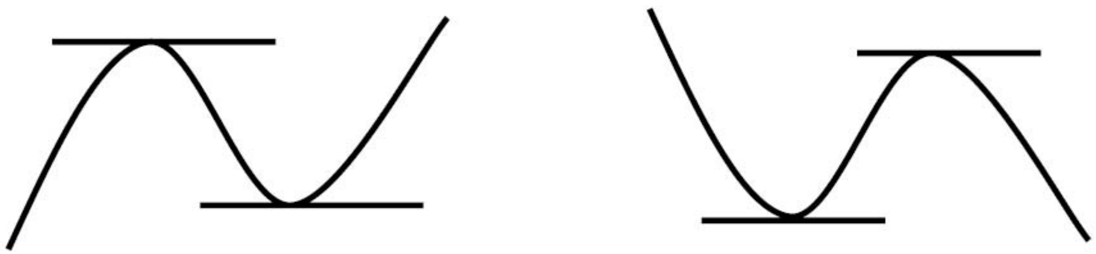
幾何解釋：





設三次函數$f(x)=ax^3+bx^2+cx+d$ ($a \neq 0$)

$a>0$ $a<0$



$f''(\frac{-b}{3a})=0$，當$x>\frac{-b}{3a}$與$x<\frac{-b}{3a}$，$f''(x)$異號，所以$(\frac{-b}{3a}$，$f(\frac{-b}{3a}))$爲反曲點。

有一個極大，一個極小，一個反曲點

次像素方法

- Lagrange polynomail (拉格朗日內插法)：通過 4 點的多項式可以用3次多項式表示：

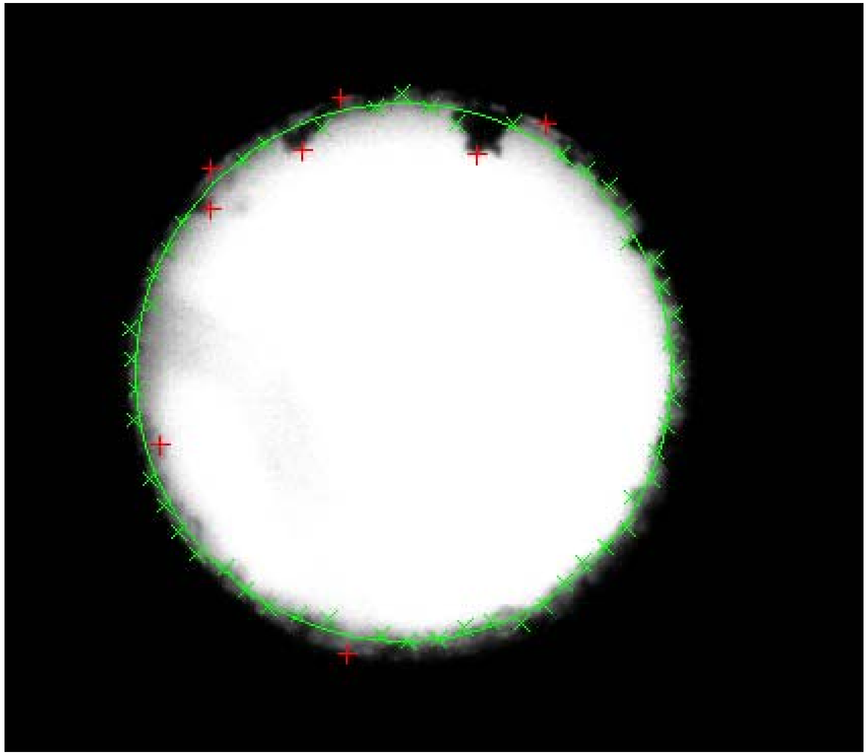
$$f(x) = ax^3 + bx^2 + cx + d$$

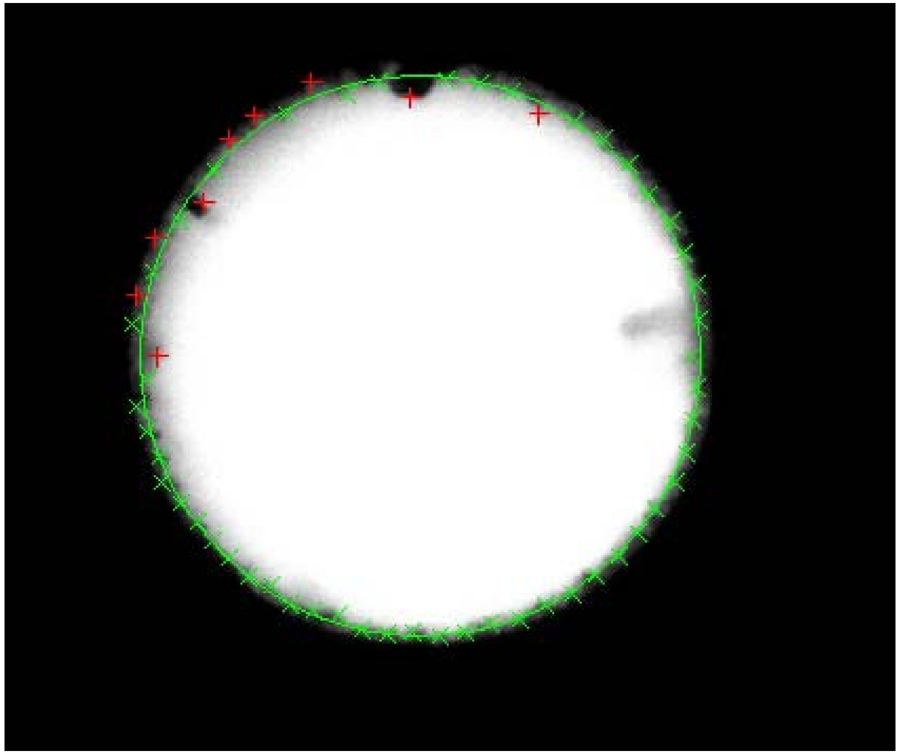
給應通過的 4 點就可以決定以上的多項式函數。 如果以 G(x) 表示在座標 x 上的像素值，則在參照 x-1, x, x+1及 x+2等4點時能建立以下方程式求解：

$$\begin{pmatrix} G(x-1) \\ G(x) \\ G(x+1) \\ G(x+2) \end{pmatrix} = \begin{pmatrix} (x-1)^3 & (x-1)^2 & (x-1) & 1 \\ x^3 & x^2 & x & 1 \\ (x+1)^3 & (x+1)^2 & (x+1) & 1 \\ (x+2)^3 & (x+2)^2 & (x+2) & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \end{pmatrix}$$

實際應用案例

- 噴料孔檢測與量測：





實際應用案例

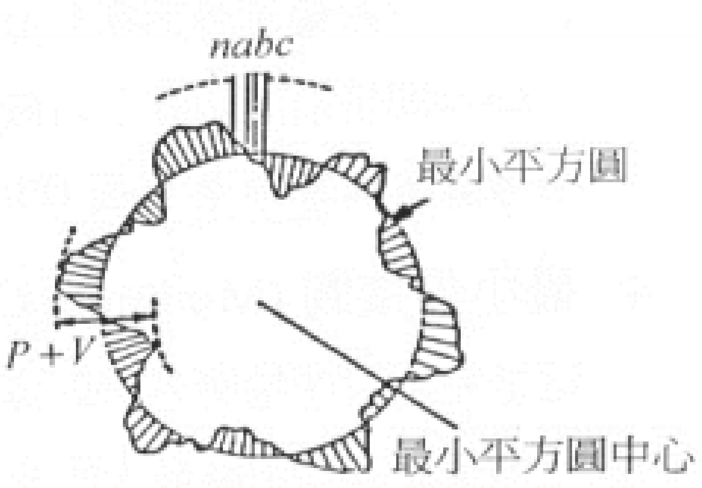
- 噴料孔檢測與量測：
  - 真圓度量測：
    - 以失圓尺寸大小表示
    - 圓形工件之輪廓形狀與理想形狀偏差量
    - 兩個能包絡圓形工件輪廓形狀的同心圓之最小半徑差異

實際應用案例

- 噴料孔檢測與量測：
  - 真圓度量測：
    - 最小平方圓
    - 最大內切圓
    - 最小外接圓
    - 最小環帶圓

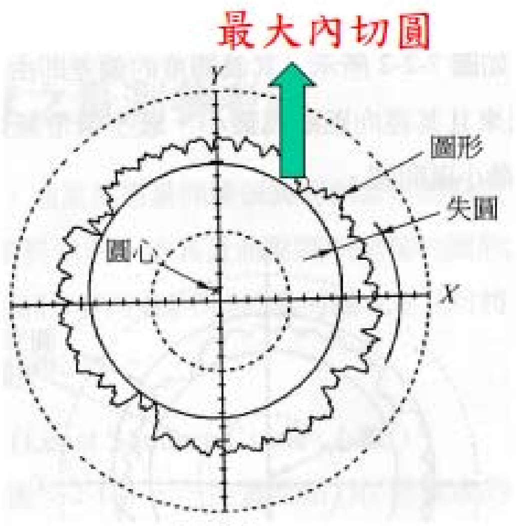
實際應用案例

- 最小平方圓：
  - 最小平方圓距離輪廓形狀之徑向距離之平方和為最小。
  - 待測工件圓斷面之失圓等於最小平方圓其偏差值的大小。
  - 失圓為其最大波峰到最大波谷的和。



實際應用案例

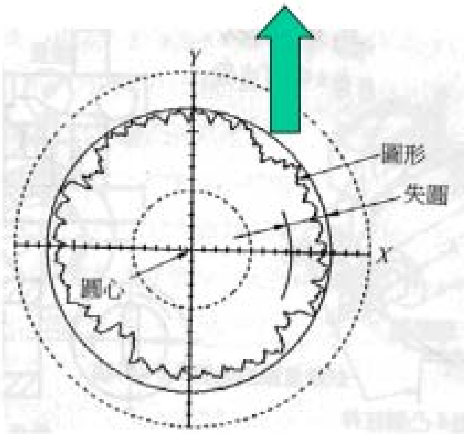
- 最大內切圓：
  - 內接於待測圓輪廓，且半徑為最大的內接圓。
  - 完全被輪廓外形所包圍，而無相交之最大圓。



實際應用案例

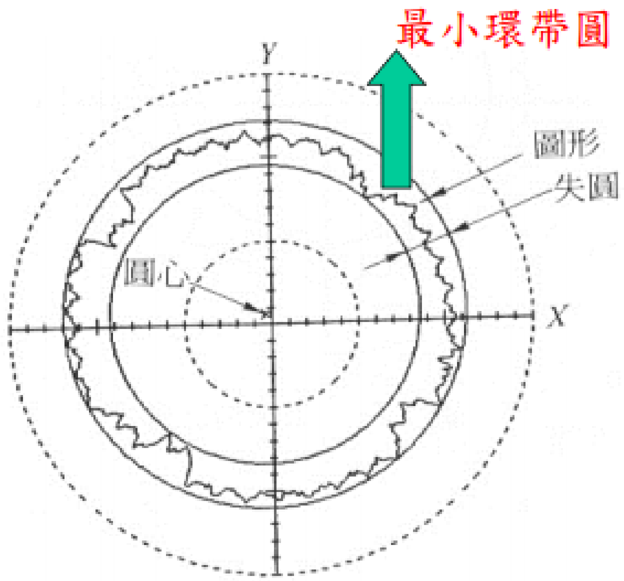
- **最小外接圓：**
  - 外接於待測圓輪廓，且半徑為最小的外接圓。
  - 完全封閉輪廓外形最小圓。

最小外接圓



實際應用案例

- 最小環帶圓：
  - 又稱最小區間圓。
  - 兩個同心圓將輪廓形狀包絡起來，且其徑向距離為最小。

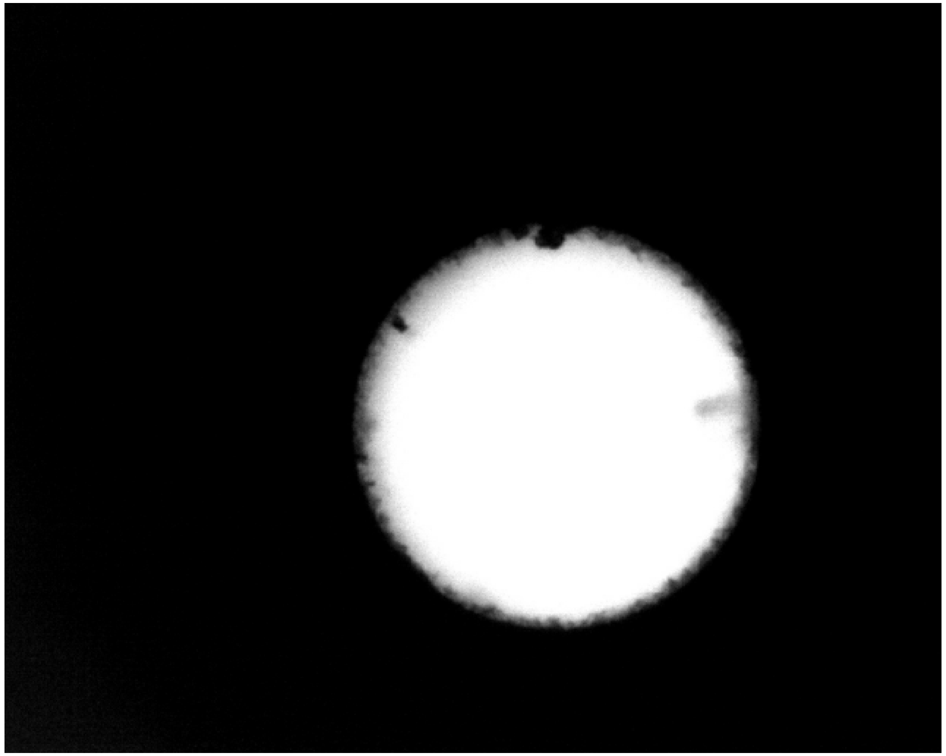


實際應用案例

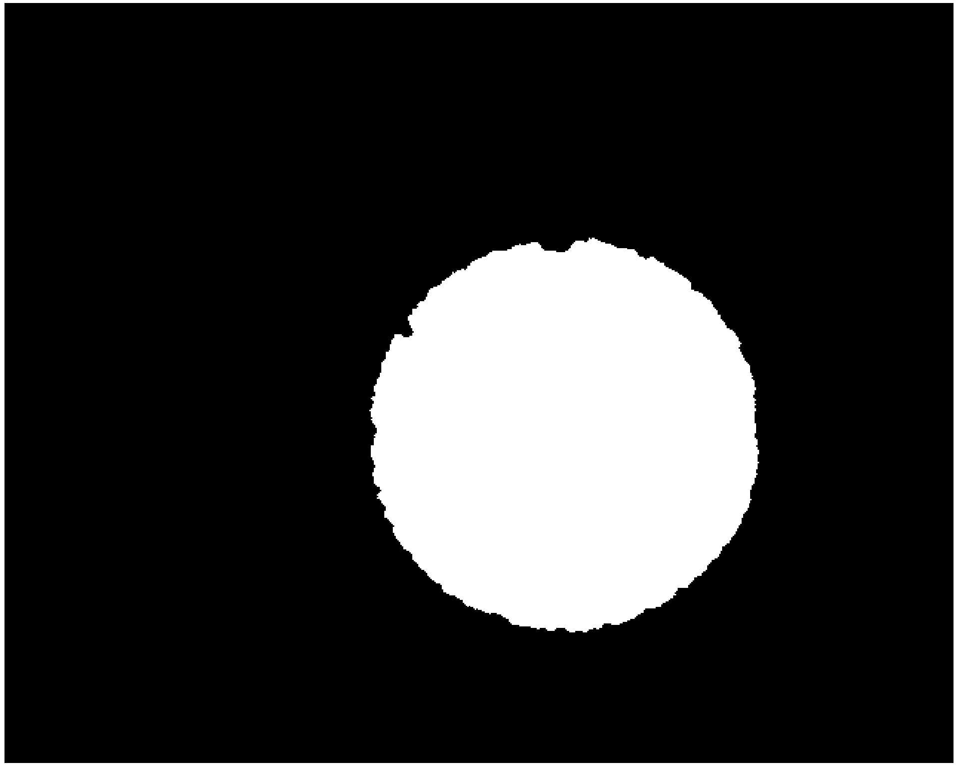
- 真圓度量測：
  - 最小外接圓 - 最大內切圓。
  - 實作方式：
    - 先計算最小平方圓
    - 利用最小平方圓所計算出的圓心位置，計算輪廓的最小外接圓及最大內切圓
    - 將最小外接圓半徑減去最大內切圓半徑，即可得真圓度(失圓)

實際應用案例

- 非固定式圓孔量測：



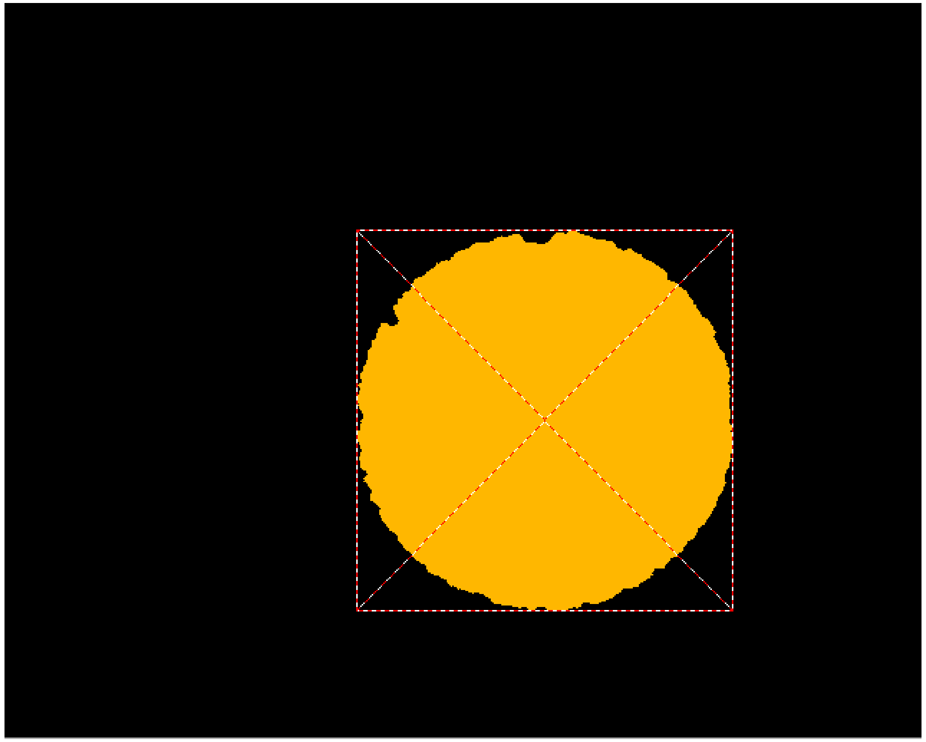
原始圖



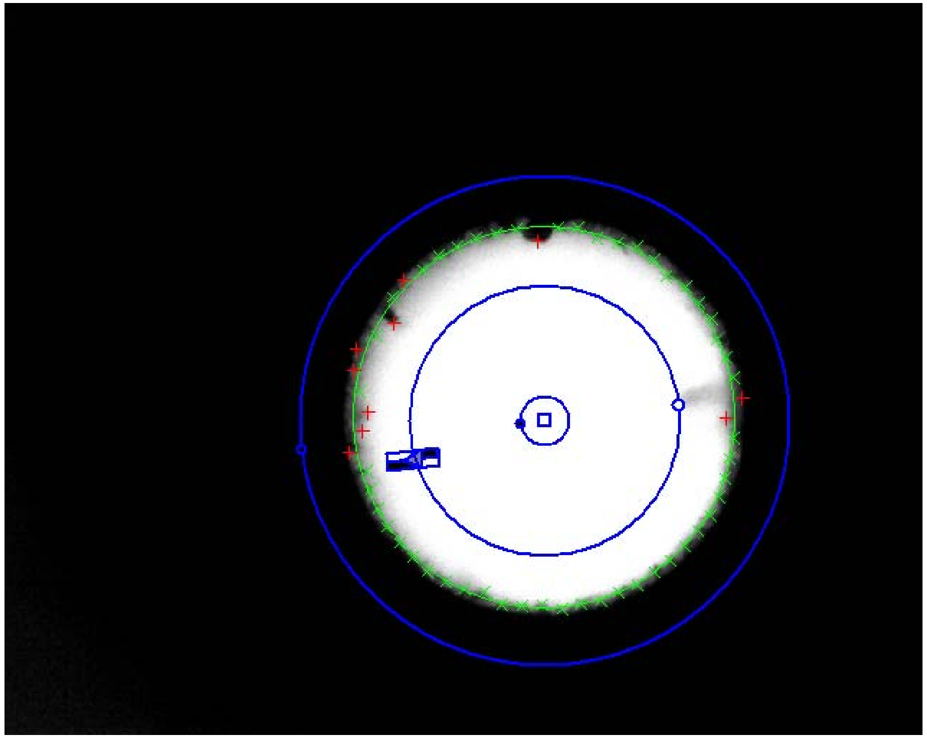
二值化

實際應用案例

□ 非固定式圓孔量測：



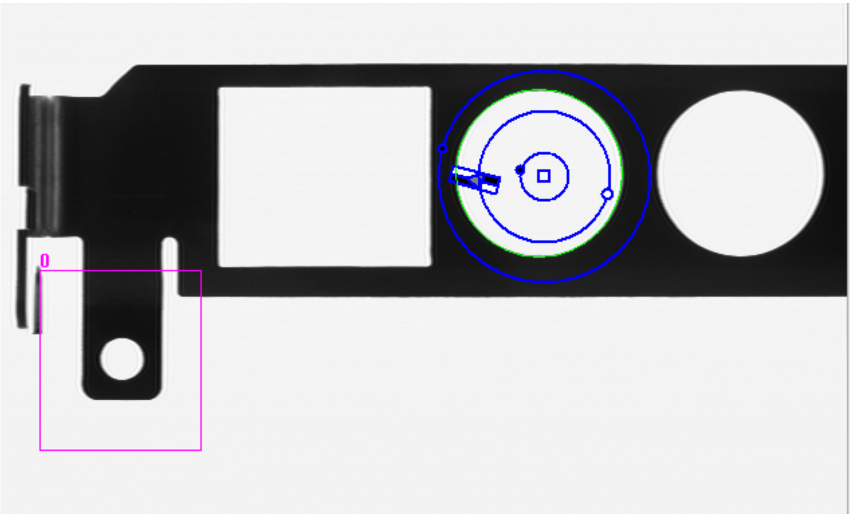
Connected Component Labeling

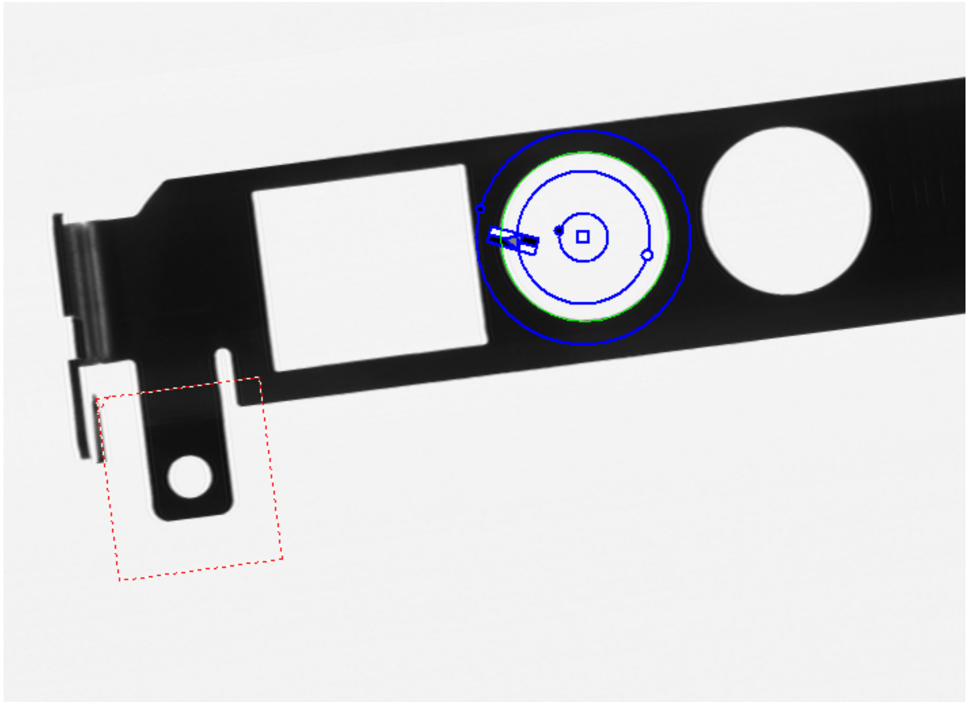


圓孔量測

實際應用案例

- 非固定式圓孔量測：

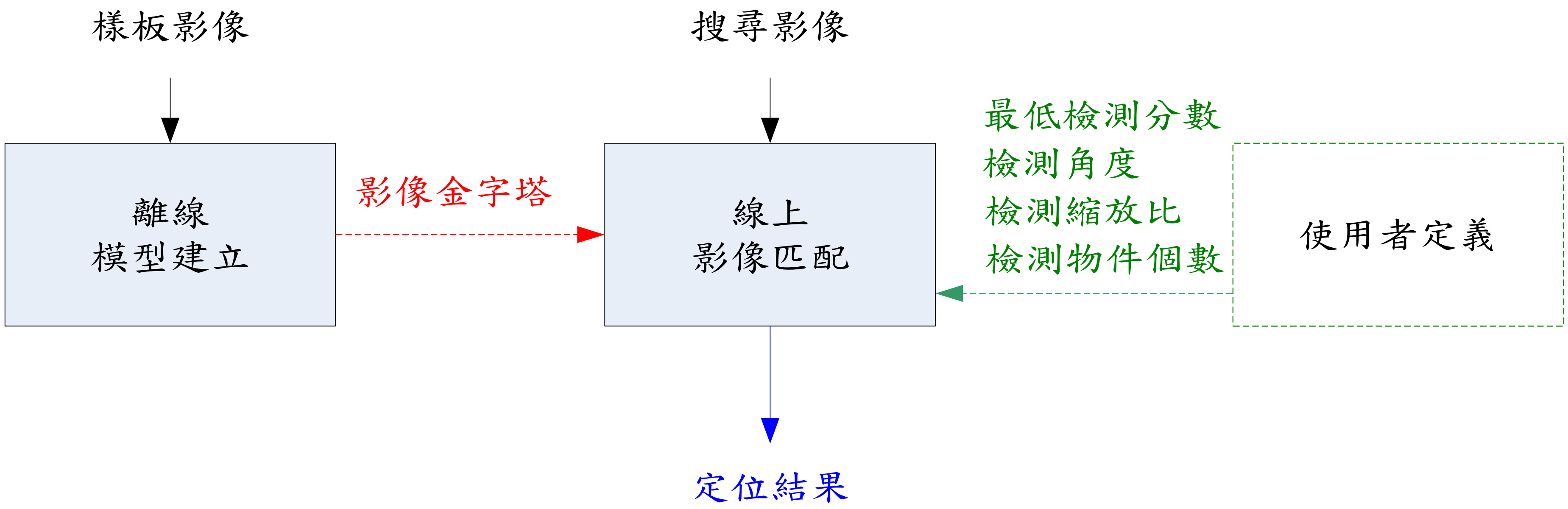




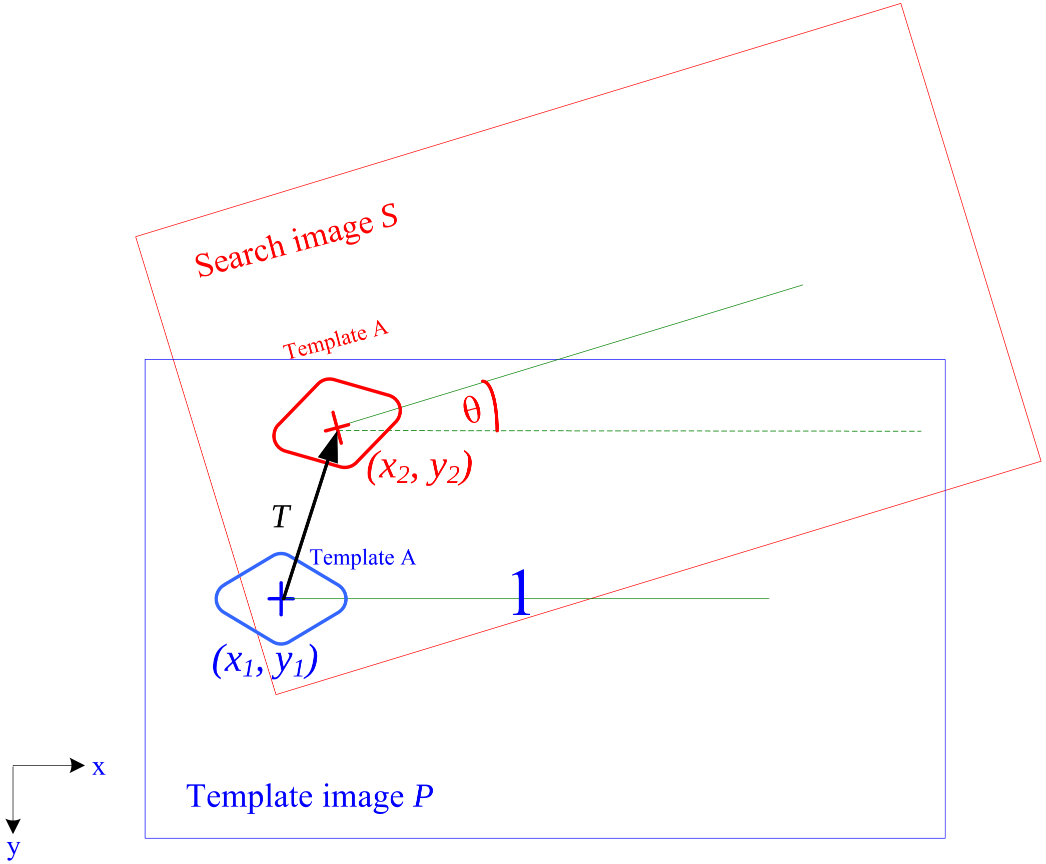
視覺定位技術

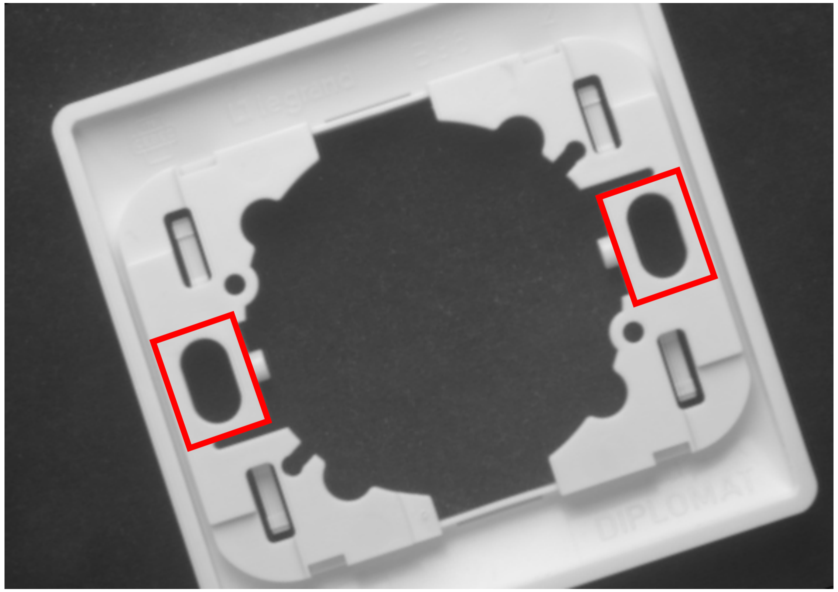
An accelerating CPU based correlation-based image alignment for real-time automatic optical inspection. Computers & Electrical Engineering 49: 207-220 (2016)

視覺定位流程圖

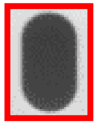

樣板影像
搜尋影像
離線
模型建立
影像金字塔
線上
影像匹配
最低檢測分數
檢測角度
檢測縮放比
檢測物件個數
使用者定義
定位結果

2D 視覺定位





Search image



Template

相似度

- Sum of Absolute Differences (SAD)
- Sum of Squared Differences (SSD)
- Normalized cross correlation (NCC)
- Shape-based matching (SBM)

$$-1 \leq \text{Simialrity coefficient} \leq 1$$

Perfect matching → Similarity coefficient =1
minimum difference

相似度(續)

- Sum of Absolute Differences (SAD)

$$SAD(x,y)=\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}\left|I(x+i,y+j)-T(i,j)\right|$$

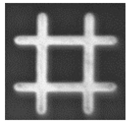


Image T

- Sum of Squared Differences (SSD)

$$SAD(x,y)=\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}\left|I(x+i,y+j)-T(i,j)\right|^2$$

- Matching result : Minimum difference

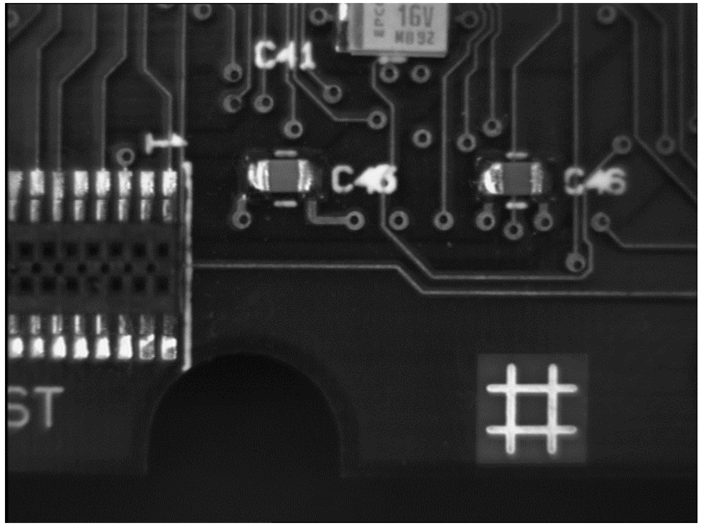


Image I

相似度(續)

- Normalized cross correlation (NCC)

$$\delta_g(x,y)=\frac{\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}[I(x+i,y+j)\cdot T(i,j)]-w\cdot h\cdot \mu_I\cdot \mu_T}{\sqrt{\left(\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2} I^2(x+i,y+j)-w\cdot h\cdot \mu_I^2\right)\cdot\left(\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2} T^2(i,j)-w\cdot h\cdot \mu_T^2\right)}},$$

The size of the template image $w\times h$

and μ_T are μ_I the gray-level averages of the template image and the windowed compared image.

$$\mu_T=\frac{1}{w\cdot h}\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2} T(i,j)$$

$$\mu_I=\frac{1}{w\cdot h}\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2} I(x+i,y+j)$$

相似度(續)

- Normalized cross correlation (NCC)

$$\delta_c(x,y)=\frac{\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}[I(x+i,y+j)\cdot T(i,j)]-3\cdot w\cdot h\cdot \mu_I\cdot\mu_T}{\sqrt{\left(\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}\left\|I(x+i,y+j)\right\|^2-3\cdot w\cdot h\cdot\mu_I^2\right)\cdot\left(\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}\left\|T(i,j)\right\|^2-3\cdot w\cdot h\cdot\mu_T^2\right)}},$$

The size of the template image $w\times h$

u_T and u_I are the color-level averages of the template image and the windowed compared image.

$$T(i,j)=\left(T_{\mathrm{R}}(i,j),T_{\mathrm{G}}(i,j),T_{\mathrm{B}}(i,j)\right)\qquad I(x+i,y+j)=\left(I_{\mathrm{R}}(x+i,y+j),I_{\mathrm{G}}(x+i,y+j),I_{\mathrm{B}}(x+i,y+j)\right)$$

$$\mu_T=\frac{1}{3\cdot w\cdot h}\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}[T_{\mathrm{R}}(i,j)+T_{\mathrm{G}}(i,j)+T_{\mathrm{B}}(i,j)]$$

$$\mu_I=\frac{1}{3\cdot w\cdot h}\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}[I_{\mathrm{R}}(x+i,x+j)+I_{\mathrm{G}}(x+i,x+j)+I_{\mathrm{B}}(x+i,x+j)]$$

相似度(續)

- Normalized cross correlation (NCC)

$$\delta_c(x,y)=\frac{\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}[I(x+i,y+j)\cdot T(i,j)]-3\cdot w\cdot h\cdot \mu_I\cdot \mu_T}{\sqrt{\left(\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}\left\|I(x+i,y+j)\right\|^2-3\cdot w\cdot h\cdot \mu_I^2\right)\cdot\left(\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}\left\|T(i,j)\right\|^2-3\cdot w\cdot h\cdot \mu_T^2\right)}},$$

The size of the template image $w\times h$

u_I and u_T are the color-level averages of the template image and the windowed compared image.

$$T(i,j)=\left(T_{\mathrm{R}}(i,j),T_{\mathrm{G}}(i,j),T_{\mathrm{B}}(i,j)\right) \qquad I(x+i,y+j)=\left(I_{\mathrm{R}}(x+i,y+j),I_{\mathrm{G}}(x+i,y+j),I_{\mathrm{B}}(x+i,y+j)\right)$$

$$\mu_T=\frac{1}{3\cdot w\cdot h}\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}[T_{\mathrm{R}}(i,j)+T_{\mathrm{G}}(i,j)+T_{\mathrm{B}}(i,j)]$$

$$\mu_I=\frac{1}{3\cdot w\cdot h}\sum_{i=-w/2}^{w/2}\sum_{j=-h/2}^{h/2}[I_{\mathrm{R}}(x+i,x+j)+I_{\mathrm{G}}(x+i,x+j)+I_{\mathrm{B}}(x+i,x+j)]$$

相似度(續)

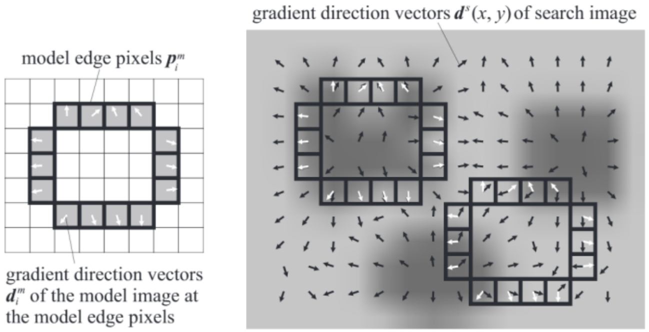
- Shape-Based Matching

$$\delta_{eg}(x,y)=\frac{1}{n}\sum_{i=1}^{n}\frac{\mathbf{d}_i\mathbf{e}_i}{\|\mathbf{d}_i\|\|\mathbf{e}_i\|}=\frac{1}{n}\sum_{i=1}^{n}\frac{t_i v_{x+r_i,y+c_i}+u_i w_{x+r_i,y+c_i}}{\sqrt{t_i^2+u_i^2}\sqrt{v_{x+r_i,y+c_i}^2+w_{x+r_i,y+c_i}^2}},$$

$\mathbf{d}_i=(t_i,u_i)^T$ gradients direction vector of template image

$\mathbf{e}_i=(v_{r_i,c_i},w_{r_i,c_i})^T$ gradients direction vector of scene image

$\mathbf{q}_i=(r_i,c_i)^T$ edge points, these points are relative to the center of gravity and the edge points of the object.



(b) Model edges (c) Search image

Picture Ref: Halcon

2D 視覺定位

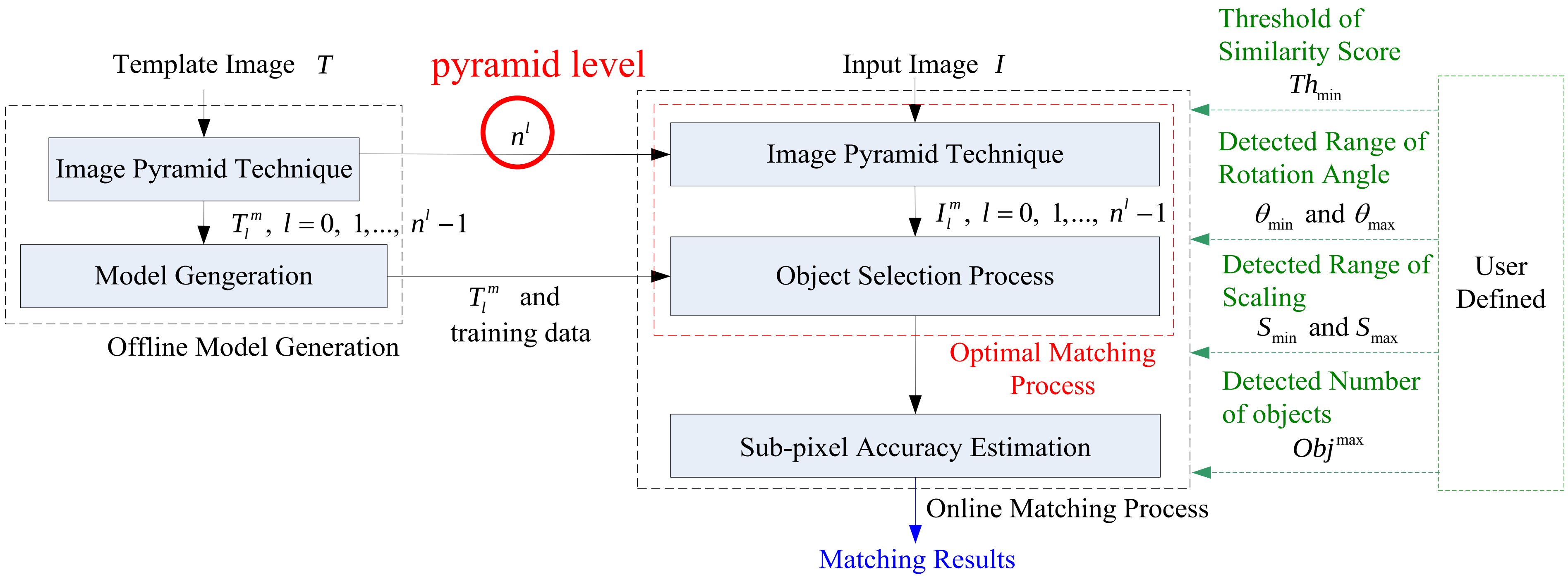
Table. Summary of image alignment methods.

Category	Features	Approaches	Disadvantages
Feature-based	Edge maps Interest point Invariant descriptors Orientation code	Hausdorff distance Feature correspondence Zernike moment Dissimilarity measurement Ring-projection	Inaccurate feature extraction
Area-based	Intensity	Cross correlation	Excessive computation time

2D 視覺定位

Category	Strategy	Improving methods
Area-based	Computation enhancement	Integral image GPU SIMD
	Skipping unnecessary computation	Coarse-to-fine Bounded conditions Elimination strategy Winner update Walsh-Hadamard kernels
Feature-based	Computation enhancement	SIMD Dominant gradients orientation
	Skipping unnecessary computation	Branch and bound Invariant descriptor Bounded conditions

2D 視覺定位

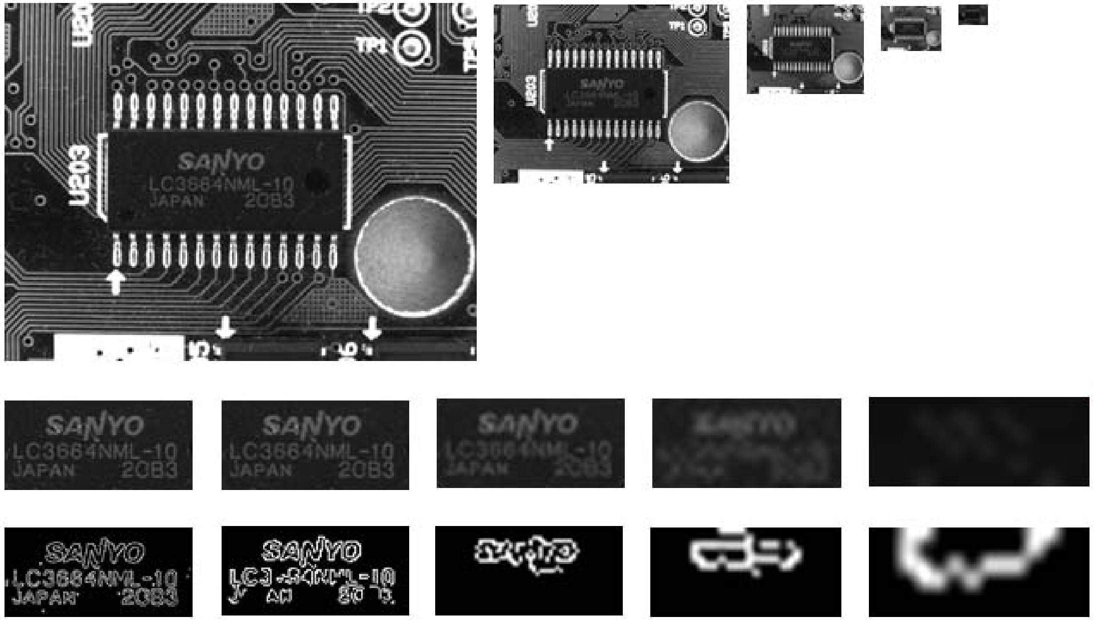


Optimal (Coarse-to-fine) :

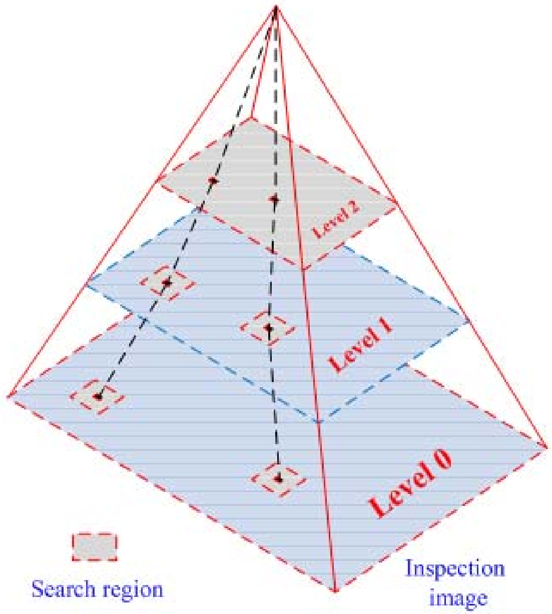
- Image pyramid technique
- SIMD
- Search strategy

2D 視覺定位

- Image pyramid technique

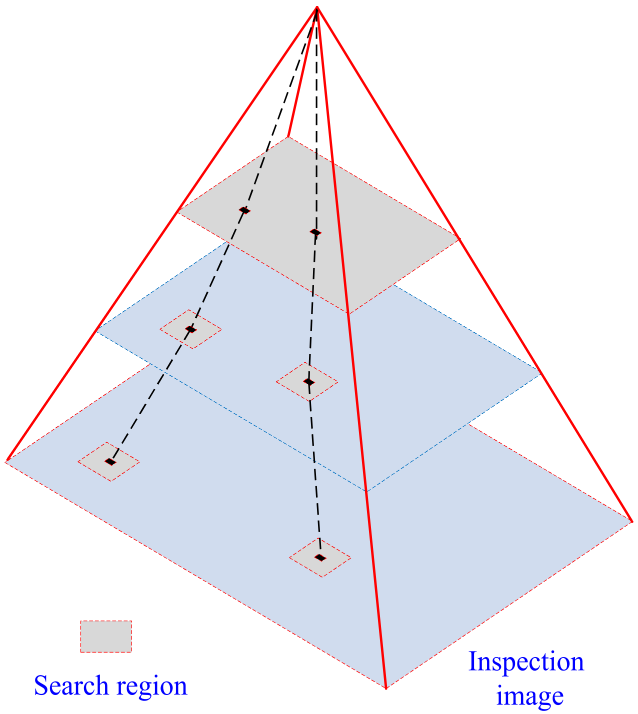


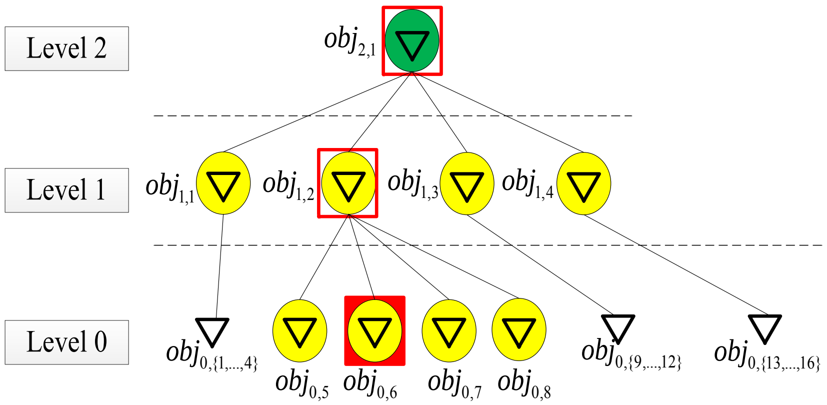
Level 0 1 2 3 4

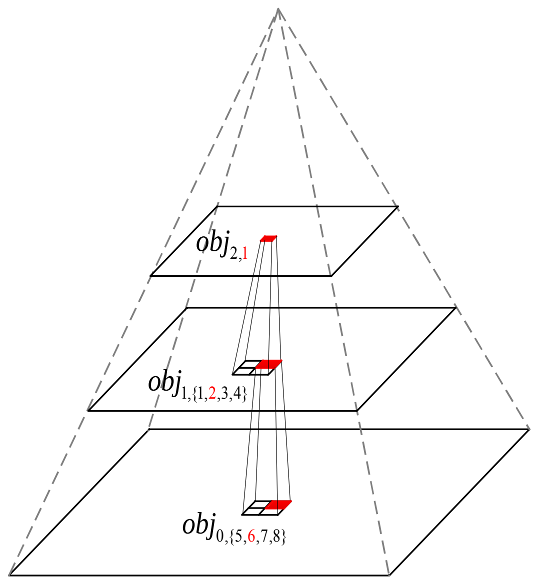


- Pyramid levels: as high as possible.
- Top level: recognizable.

搜尋策略



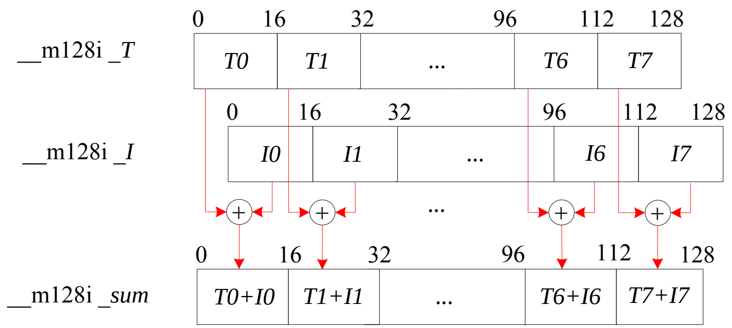




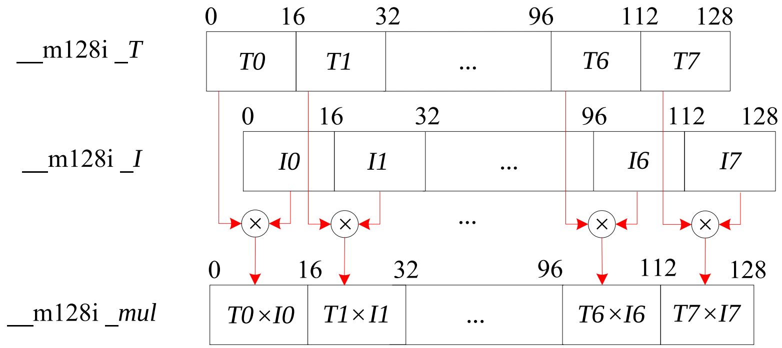
Yellow: search candidates

Red: with maximum similarity score and it exceeds the threshold of similarity

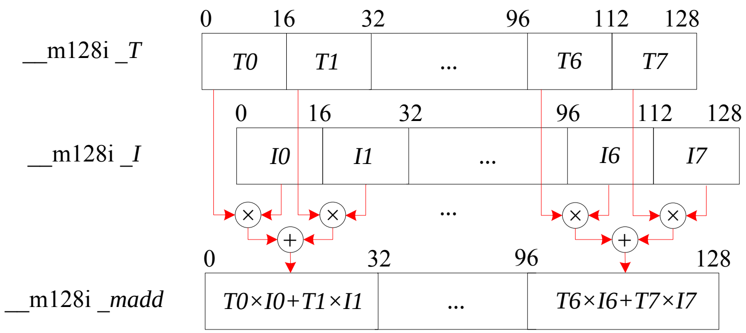
SSE2 加速機制



$$\_sum = \_mm\_add\_epi16(\_T,\ \_I)$$



$$\_mul = \_mm\_mullo\_epi16(\_T,\ \_I)$$



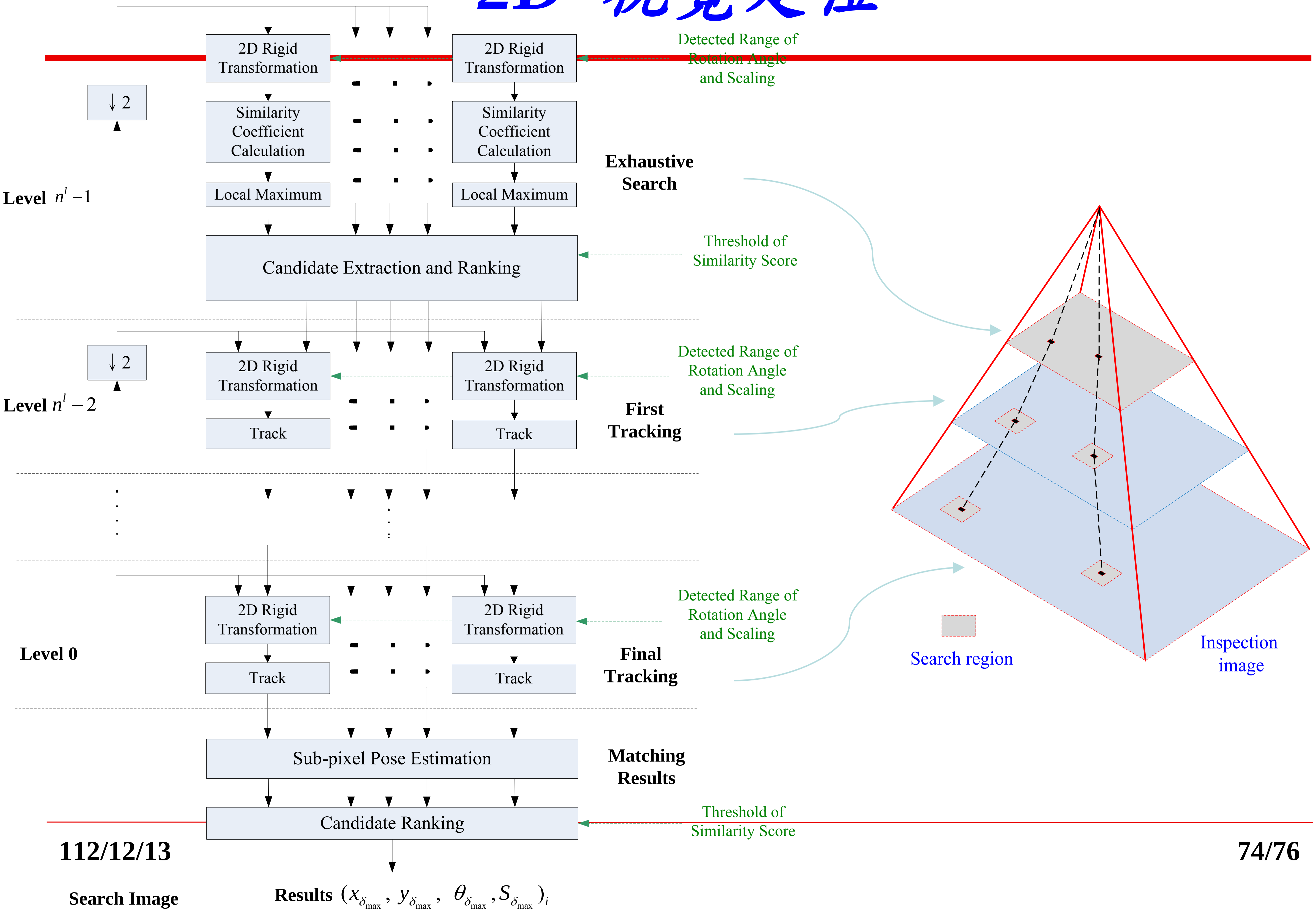
$$\_madd = \_mm\_madd\_epi16(\_T,\ \_I)$$

For instance, the size of the template and compared windows images are 8×8 pixel

	Instruction cycles
SSE2	469
Without SSE2	3736

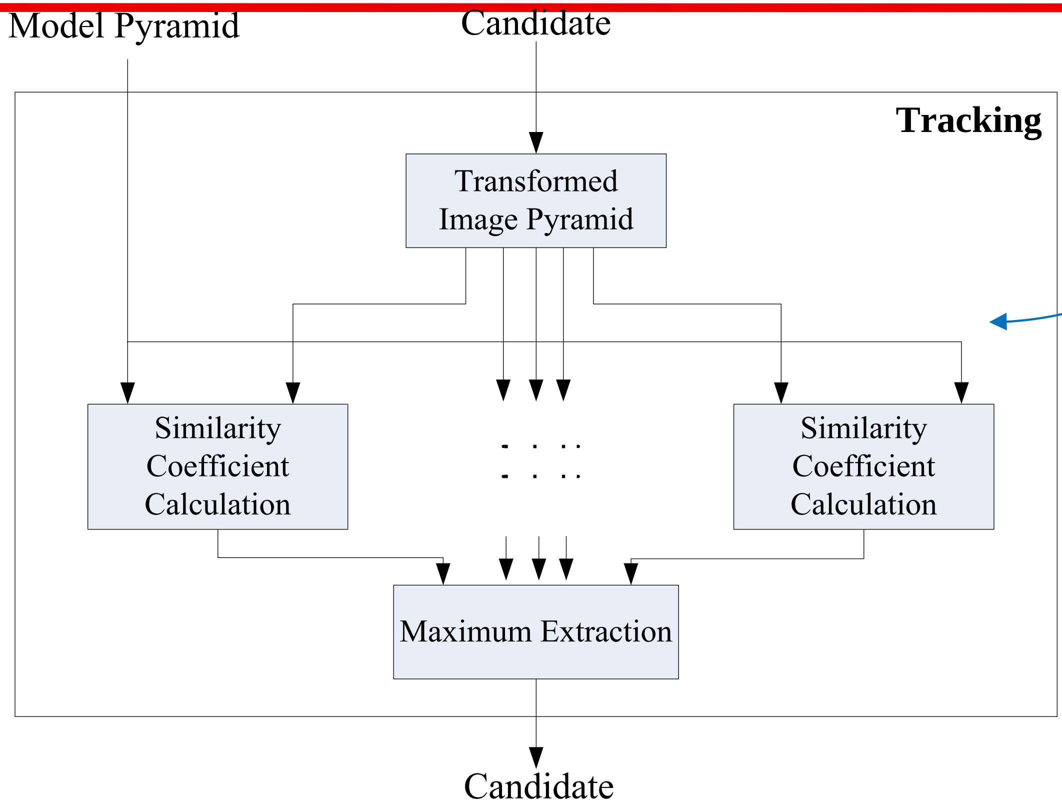
8 times

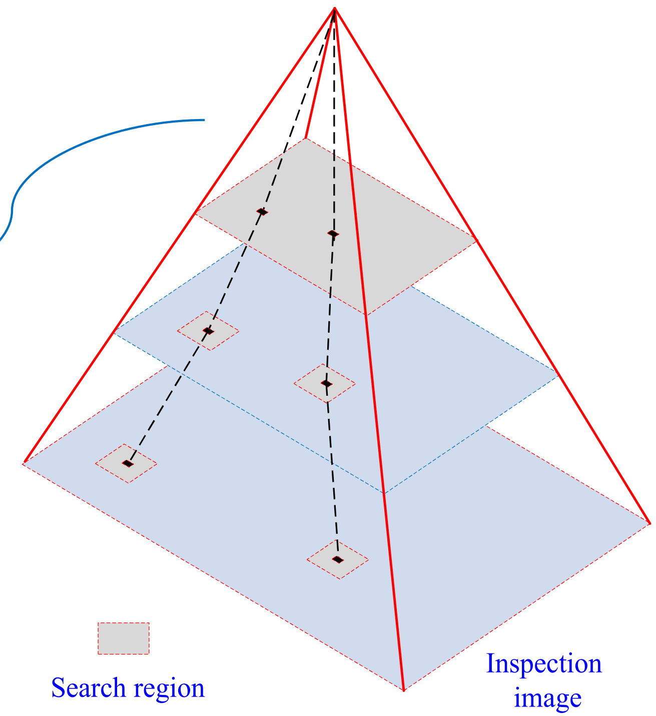
2D 視覺定位



2D Rigid Transformation
Similarity Coefficient Calculation
Local Maximum
Candidate Extraction and Ranking
Detected Range of Rotation Angle and Scaling
Threshold of Similarity Score
Exhaustive Search
Level $n^l - 1$
↓ 2
Level $n^l - 2$
Track
First Tracking
Level 0
Final Tracking
Sub-pixel Pose Estimation
Matching Results
Candidate Ranking
Search Image
Results $(x_{\delta_{max}}, y_{\delta_{max}}, \theta_{\delta_{max}}, S_{\delta_{max}})_i$
Search region
Inspection image

2D 視覺定位





$$(x_{\delta_{\max}}, y_{\delta_{\max}}, \theta_{\delta_{\max}}, s_{\delta_{\max}}) = \arg \max_{x,y,\theta,s} \delta(x, y, \theta, s),$$

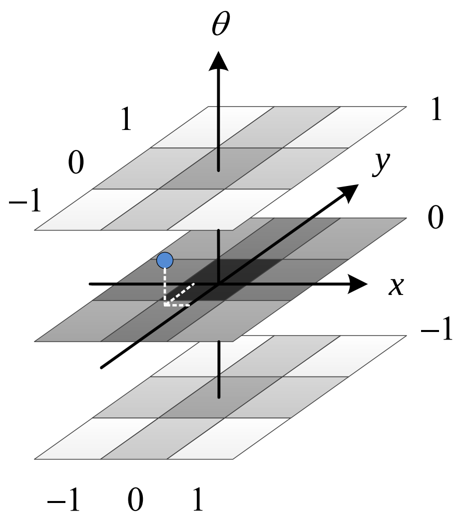
$$x \in [x' - t_x,\ x' + t_x],\ y \in [y' - t_y,\ y' + t_y],$$

$$\theta \in [\theta' - 2\Delta\theta,\ \theta' + 2\Delta\theta]$$

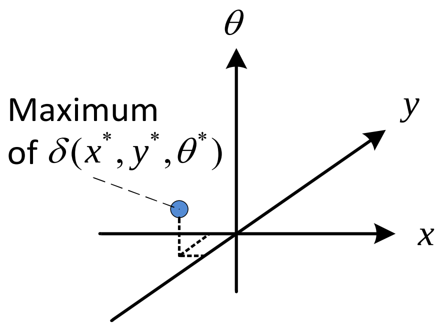
$$s \in [s' - \Delta s,\ s' + \Delta s]$$

2D 視覺定位

$$\delta(x,y,\theta)=k_0x^2+k_1y^2+k_2\theta^2+k_3xy+k_4x\theta+k_5y\theta+k_6x+k_7y+k_8\theta+k_9$$



$3\times3\times3$ neighborhood space



$$x\in[x_0'-1,\ \ x_0'+1],$$
$$y\in[y_0'-1,\ \ y_0'+1],$$
$$\theta\in[\theta'_0-\Delta\theta_0,\ \ \theta'_0+\Delta\theta_0]$$

$$\begin{bmatrix}x^*\\y^*\\\theta^*\end{bmatrix}=\begin{bmatrix}2k_0 & k_3 & k_4\\k_3 & 2k_1 & k_5\\k_4 & k_5 & 2k_2\end{bmatrix}^{-1}\begin{bmatrix}-k_6\\-k_7\\-k_8\end{bmatrix}$$