義守大學電機所「電腦視覺」報告

單元四

影像切割 II — Region Growing

參考解答

MIAT(機器智慧與自動化技術)實驗室

中　華　民　國　93　年　10　月　18　日

Region-based segmentation:

如果 R 表示一張影像的全部範圍，影像切割的目的是將 R 切割成 n 個區塊：R1,R2,…,Rn 並滿足下列條件：

a) $\bigcup_{i=1}^{n} R_i = R$

b) $\mathrm{R_i}$ is a connected region.

c) $\mathrm{R_i} \cap R_j = \phi$

d) $\mathrm{P(R_i)} = \mathit{True}$

e) $\mathrm{P(R_i} \cup R_j) = \mathit{False}$

Seeded Region growing：

選取一批種子(seed)pixels，以 seed 為核心進行成長(grow)，判斷 seed 周圍 pixels 是否與 seed 具有相似的特性(灰階值、節理、色彩) ，如果是，則接受該 pixel 為同一 region，再以此新的 pixel 為核心，繼續偵查周圍尚未被歸類到任一 region 的 pixel，直到影像所有 pixels 都分類完成。

Following Adams and Bischof, we say that the seeds are grouped into n sets, A1; A2; . . .; An. Each step of the algorithm adds a single pixel to one of these sets. To achieve this and maintain a homogeneity criterion, the set T of all as-yet unallocated pixels bordering at least one region is employed

$$T = \left\{ x \notin \bigcup_{i=1}^{n} A_i \,\middle|\, N(x) \cap \bigcup_{i=1}^{n} A_i \neq 0 \right\}$$

where N(x) is the nearest eight neighbors of the pixel x. If for x∈T we have that N(x) meets just one of the Ai, then the index i(x) ∈{1; 2; . . .; n} is defined such that $N(x) \cap A_{i(x)} \neq 0$.We define δ(x) to be a measure of how different x is from the region it adjoins. The simplest definition for δ(x) is

$$\delta(x) = \left| g(x) - \underset{y \in A_{i(x)}}{\text{mean}} [g(y)] \right|$$

where g(x) is the gray-scale intensity value of x. If N(x) meets two or more of the Ai, the value i(x) is taken to be the value of i such that N(x) meets Ai and δ(x) is minimized. A z∈T is then taken such that

$$\delta(z) = \min_{x \in T} \{\delta(x)\}$$

and append z to Ai(z). This completes a single step of the algorithm and the same process is iterated until all pixels have been allocated to a set.

The implementation of SRG employs a linked list storing the data of T , which is ordered according to δ(x) . Adams and Bischof refer to this as a sequentially sorted list (SSL). The SSL remains ordered throughout the progression of the algorithm, so that by simply processing the first entry at each time step, δ(x) is satisfied. Thus, a search cost must be incurred to locate appropriate positions when adding new members to the SSL.

演算法：

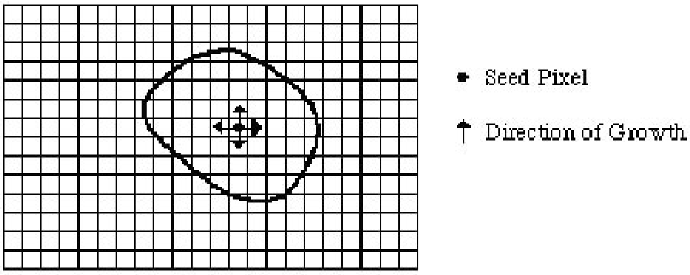
Initialization:

根據起始分群標示(labeling)每一個seed所屬region，把每一個seed鄰近點放入SSL(sequentially sorted list)。

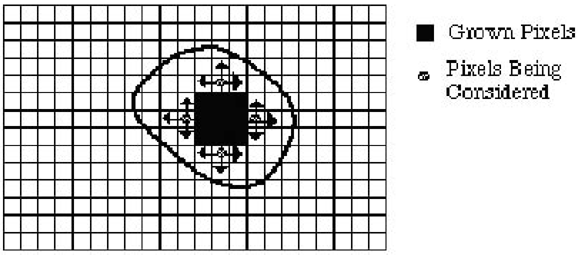
Region Growing:

```
While SSL不是空的 do
  從SSL移除第一個pixel y.
  測試y的鄰近點:
     if 所有y的鄰近點都已標示爲相同region的label A then
        標示y爲region A；
        更新region A的平均值mean；
        擇出y的鄰近點中不在SSL的pixel，計算該點灰階值與region A的平均灰階值(mean)的差距

          $\delta(x) = g(x) - mean_{y \in A_i(x)}[g(y)]$

    else
       標示y爲邊界點label。
```

Region Growing 示意圖



(a) Start of Growing a Region



(b) Growing Process After a Few Iterations

主程式：

```
//---------------------------------------------------------------------------
#pragma hdrstop
#include <fstream.h>
```

```
#include <iostream.h>
#include "array.h"
#pragma argsused
void RegionGrow(uc2D &ima1, uc2D &ima2,int x,int y,int Count,int Total,int Threshold);
int main(int argc, char* argv[])
{
   uc2D sima,dima;
   float Threshold;
   char c;
   int Xseed,Yseed,x,y;
   int HalfSize;
   int Count,Total;

   HalfSize=5;
   Xseed=45;
   Yseed=50;
   Threshold=40;

   ifstream in("finger300x300.raw",ios::binary);
   sima.Initialize(300,300);
   dima.Initialize(300,300);
   for(int i=0;i<sima.nr;i++)
   for(int j=0;j<sima.nc;j++)
   {
      in.get(c);
      sima.m[i][j]=c;
   }
   in.close();

   for(int i=0;i<sima.nr;i++)
   for(int j=0;j<sima.nc;j++)
      dima.m[i][j]=0;

   /* Initialize region statistics */
   Total = Count = 0;
   for (y = Yseed - HalfSize; y <= Yseed + HalfSize; y++)
   for (x = Xseed - HalfSize; x <= Xseed + HalfSize; x++)
      if ((x >= 0) && (y >= 0) && (x < sima.nc-1) && (y < sima.nr-1))
      {
          Count++;
           Total += sima.m[y][x];
        }
   /* Perform recursive seeded region growing */
   RegionGrow(sima,dima,Xseed,Yseed,Count,Total,Threshold);
   ofstream out("result.raw",ios::binary);
   for(int i=0;i<dima.nr;i++)
   for(int j=0;j<dima.nc;j++)
   {
      out<<dima.m[i][j];
   }
   out.close();

   return 0;
}
//-----------------------------------------------------------------------------
void RegionGrow(uc2D &ima1, uc2D &ima2,int x,int y,int Count,int Total,int Threshold)
{
  float Diff, Mean;

  /* Check to see if point already part of region */
  if (ima2.m[y][x] == 0)
  {
     /* See if point is close enough to add */
     Mean = Total / Count;
     Diff = ima1.m[y][x] - Mean;
     if (Diff < 0) Diff = -Diff;
     if (Diff < Threshold)
     {
        /* Add point to region and consider neighbors */
        Total += ima1.m[y][x];
        Count++;
        ima2.m[y][x] = 255;
        if (x > 0) RegionGrow(ima1, ima2,x - 1, y,Count,Total,Threshold);
        if (y > 0) RegionGrow(ima1, ima2,x, y - 1,Count,Total,Threshold);
        if (x < ima1.nc - 1) RegionGrow(ima1, ima2,x + 1, y,Count,Total,Threshold);
        if (y < ima1.nr - 1) RegionGrow(ima1, ima2,x, y + 1,Count,Total,Threshold);
     }
     else
```

```
        {
        ima2.m[y][x] = 127;
        }
    }
}
```

結果一(Threshold=40)

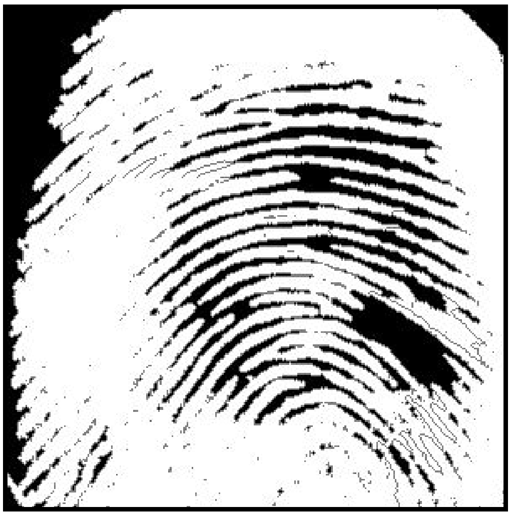


圖1.1 Seed x=126,y=104。

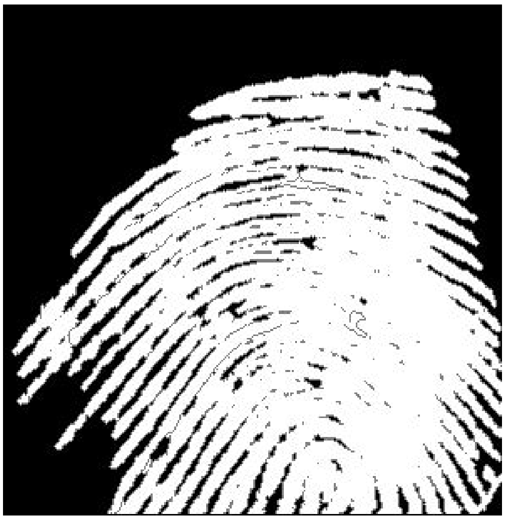


圖1.2 Seed x=210,y=187。

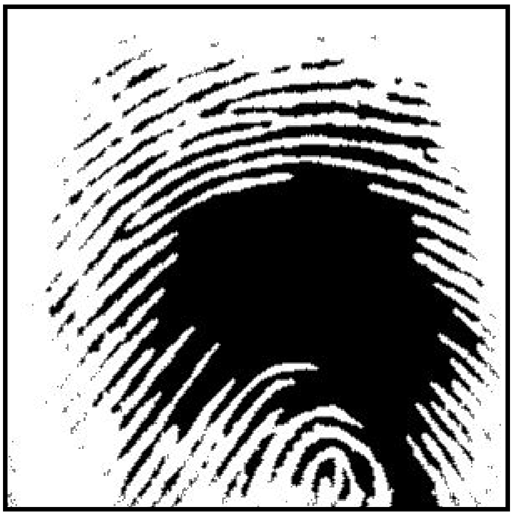


圖1.3 Seed x=45,y=50。

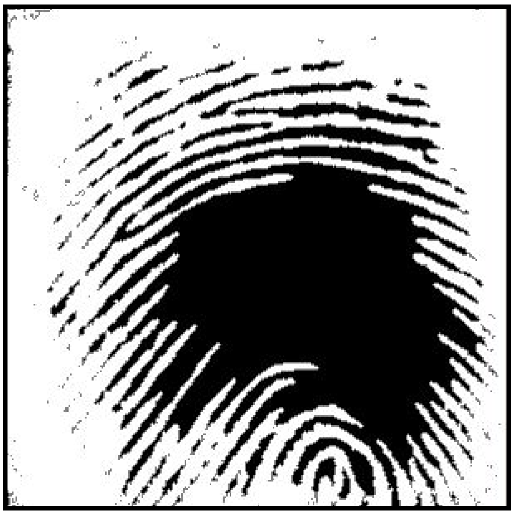
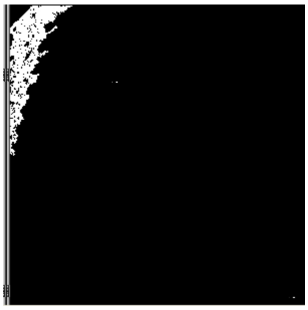
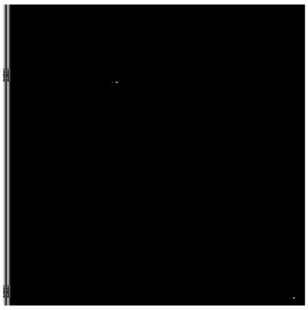
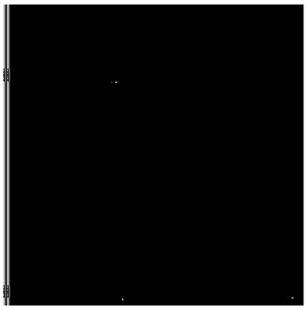
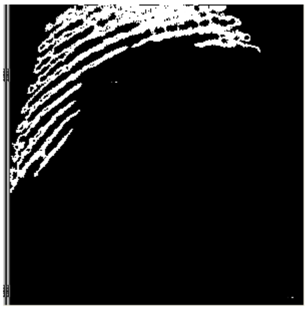
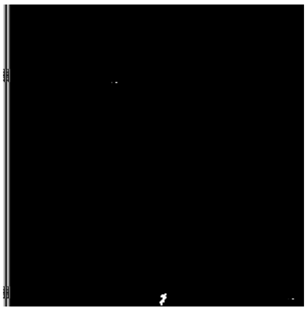
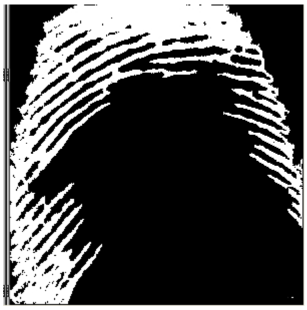
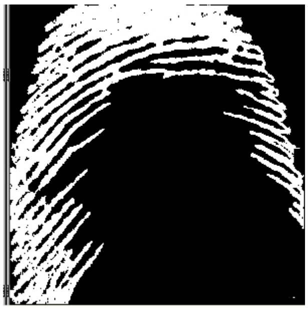
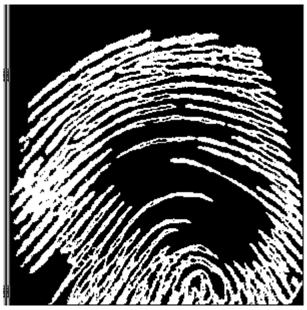
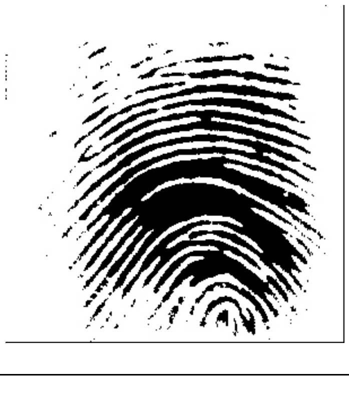
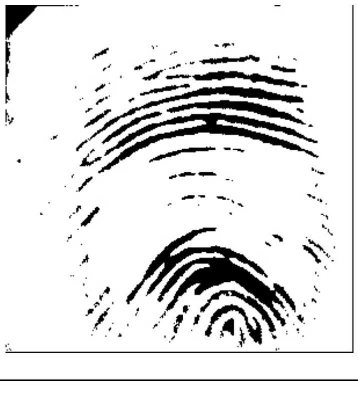


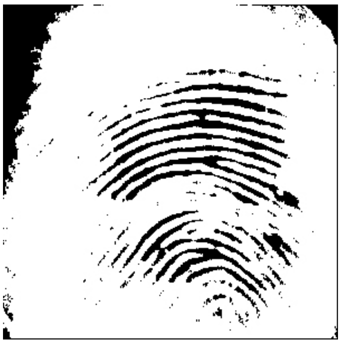
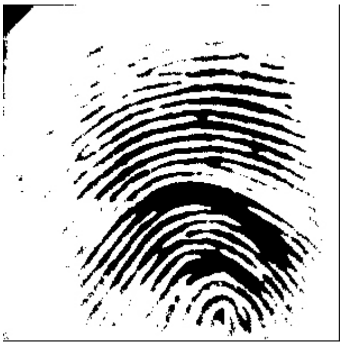
圖1.4 Seed x=45,y=114。

結果二：

 x=5,y=48,t=5	 x=243,y=5,t=5	 x=294,y=119,t=5
 x=7,y=186,t=10	 x=78,y=10,t=10	 x=294,y=160,t=10
 x=25,y=169,t=20	 x=143,y=294,t=20	 x=294,y=286,t=20

結果三：

Xseed=77; Yseed=88; **region contain 121 points with mean value = 177.521** 	**Xseed=113; Yseed=169;** **region contain 121 points with mean value = 130.388** 

Xseed=267; Yseed=192; **region contain 121 points with mean value = 160.058** 	**Xseed=97; Yseed=178;** **region contain 121 points with mean value = 140.421** 

解答二、

Segmentation by Region Growing

Algorithm：

```
Let f be an image for which regions are to be grown.
 Define a set of regions,R1,R2,R3.....Rn,each consisting
 of a single seed pixel.
  repeat
    for i=1 to n do
      for each pixel,p,at the border of Ri do
        for all neighbours of p do
          Let x,y be the neighbour's coordinates
          Let  ui be the mean grey level of pixels in Ri
          if the neighbours is unassigned and |f(x,y)-ui|<= ◎ then
             Add neighbour to Ri
             Update ui
          end if
        end for
      end for
    end for
  until no more pixels are being assigned to regions
```

Source Code：

主程式

```
#include<fstream.h>
#include"array.h"
#include"regiongrowing.h"

void main(void)
{
   ifstream in("test15x17.raw",ios::binary);
   ofstream out("test15x17.txt");

   uc2D ima;
   ima.Initialize(17,15);

   regiongrowing f;

   char c;

   for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
   {
      in.get(c);ima.m[i][j]=c;
   }

   f.growing(ima,20);

   for(int i=0;i<ima.nr;i++)
   {
      for(int j=0;j<ima.nc;j++)
      {
         out<<f.buffered.m[i][j]<<"\t";
      }
      out<<endl;
   }
}
```

物件區域轉圖形程式

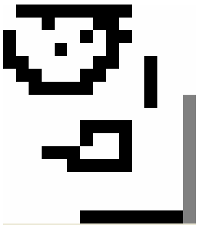
```
#include<fstream.h>
#include"array.h"

void main(void)
{
   ifstream in("test15x17.txt ");
   ofstream outim("out.raw",ios::binary);

   i2D ima;
   ima.Initialize(17,15);
   int c;

   for(int i=0;i<ima.nr;i++)for(int j=0;j<ima.nc;j++)
   {
      in>>ima.m[i][j];
   }

   for(int i=0;i<ima.nr;i++)
   {
      for(int j=0;j<ima.nc;j++)
      {
         if(ima.m[i][j]==1) outim<<(unsigned char) 0;
         else               outim<<(unsigned char) 255;
      }
   }
}
```

Regiongrowing 類別程式

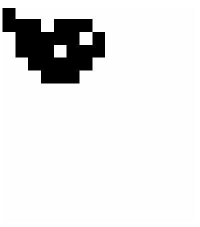
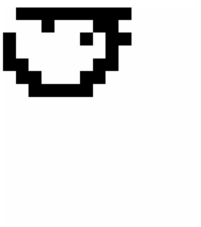
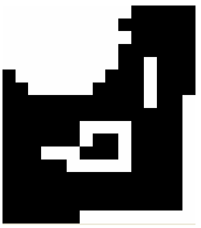
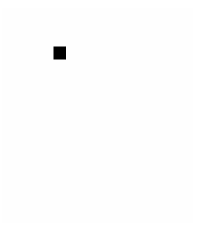
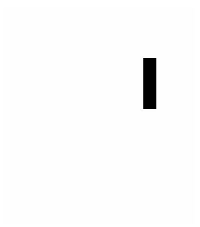
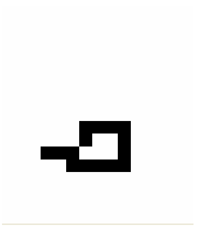
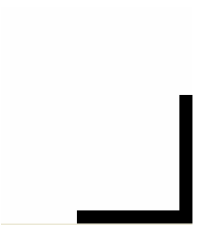
```
#if !defined(REGIONGROWING_H)
#define REGIONGROWING_H

#include <math.h>
#include "array.h"

class regiongrowing
{
 private:
        int u;                                              //region R之平均灰階值
        int index;                                          //索引值
        int T;                                              //分割閥值
        bool again;
```

```
       uc2D grayimg;                               //灰階影像
       void replaceR(int x,int y,int threshold);  //求出邊界點(x,y)的鄰近點是否數屬於 R
       void searchP(int index);                   //尋找 region  R 的邊界點
       void initbuffered();                       //對照圖初值化
       bool testsegmentation();                   //測試是否分割完畢
       int averageR(int index);                   //求出 R 的平均值
       void seedP();                              //尋找種子點
 public:
      i2D buffered;                                //對照索引圖
      void growing(uc2D &gimg,int threshold);     //區域分割
};
////////////////////////////////////////////////////////////////////////////////
void regiongrowing::growing(uc2D &gimg,int threshold) //區域分割
{
   grayimg=gimg;
   T=threshold;
   index=1;
   P=0;
   initbuffered();
   while(testsegmentation())
   {
      seedP();
      do
      {
         again=false;
         u=averageR(index);
         searchP(index);
      }while(again);
      index++;
   }
}
////////////////////////////////////////////////////////////////////////////////
void regiongrowing::replaceR(int x,int y,int threshold)   //求出點的四周是否同 R
{
   for(int i=-1;i<2;i++)for(int j=-1;j<2;j++)
   {

if((x+i>=0)&&(y+j>=0)&&(x+i<grayimg.nr)&&(y+j<grayimg.nc)&&((abs(grayimg.m[x+i][y+j]-
u))<=threshold)&&((buffered.m[i+x][j+y]==0)||(buffered.m[i+x][j+y]==-1)))
      {
         buffered.m[x+i][y+j]=index;
         again=true;
      }
   }
}
////////////////////////////////////////////////////////////////////////////////
void regiongrowing::searchP(int index)                      //尋找 R 的邊點
{
   for(int i=0;i<grayimg.nr;i++)for(int j=0;j<grayimg.nc;j++)
   {
      if(buffered.m[i][j]==index)
      {
         for(int x=-1;x<2;x++)for(int y=-1;y<2;y++)
         {

if((x+i>=0)&&(y+j>=0)&&(x+i<grayimg.nr)&&(y+j<grayimg.nc)&&((buffered.m[i+x][j+y]==0)
||(buffered.m[i+x][j+y]==-1)))
            {
               replaceR(i,j,T);
            }
         }
      }
   }
}
////////////////////////////////////////////////////////////////////////////////
void regiongrowing::initbuffered()                           //對照圖初值化
{
   buffered.Initialize(grayimg.nr,grayimg.nc);
   for(int i=0;i<grayimg.nr;i++)for(int j=0;j<grayimg.nc;j++)
   {
      buffered.m[i][j]=0;
      if((i==0)||(i==grayimg.nr-1)||(j==0)||(j==grayimg.nc-1)) buffered.m[i][j]=-1;
   }
}
////////////////////////////////////////////////////////////////////////////////
bool regiongrowing::testsegmentation()                    //測試是否分割完畢
{
   for(int i=0;i<grayimg.nr;i++)for(int j=0;j<grayimg.nc;j++)
   {
      if(buffered.m[i][j]==0) return true;
   }
   return false;
}
////////////////////////////////////////////////////////////////////////////////
int regiongrowing::averageR(int index)                       //求出R的平均值
{
   long sum=0;
   int num=0;
   for(int i=0;i<grayimg.nr;i++)for(int j=0;j<grayimg.nc;j++)
   {
      if(buffered.m[i][j]==index)
      {
         num++;
         sum=sum+grayimg.m[i][j];
      }
   }

   if(num!=0) sum=sum/num;
   return sum;
}
////////////////////////////////////////////////////////////////////////////////
void regiongrowing::seedP()                                  //尋找種子點
{
   for(int i=0;i<grayimg.nr;i++)for(int j=0;j<grayimg.nc;j++)
   {
      if(buffered.m[i][j]==0)
      {
         buffered.m[i][j]=index;
         return;
      }
   }
}
////////////////////////////////////////////////////////////////////////////////
#endif
```

我們自製一簡單影像 test15x17.raw，來測試程式的正確性，在 threshold 小於 20 所分割出來的情形，總共分出七個區域。

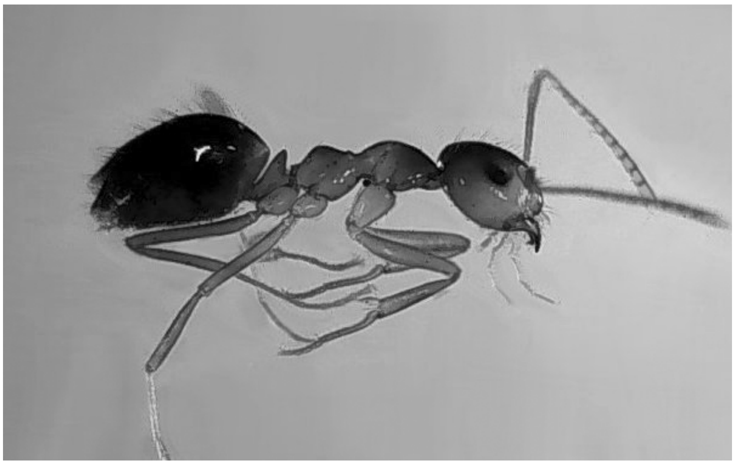


原始影像 test15x17.raw

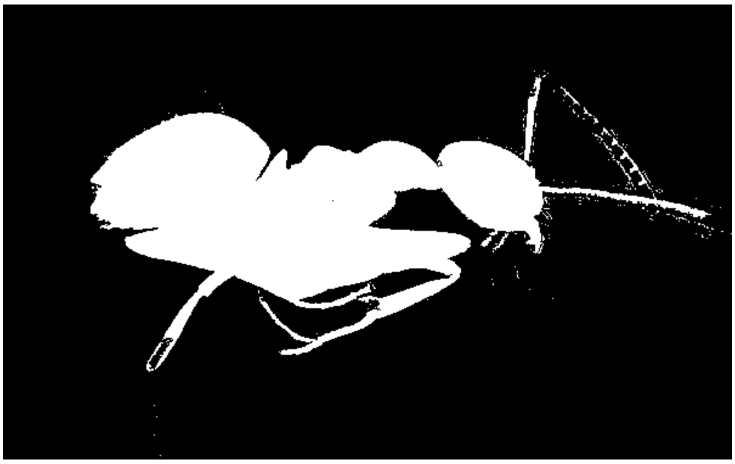
 物件 1 範圍	 物件 2 範圍	 物件 3 範圍	 物件 4 範圍	 物件 5 範圍	 物件 6 範圍	 物件 7 範圍

1	2	2	2	2	2	2	2	2	2	3	3	3	3	3
1	1	1	2	1	1	1	2	2	3	3	3	3	3	3
2	1	1	1	1	1	2	1	2	2	3	3	3	3	3
2	1	1	1	4	1	1	1	2	3	3	3	3	3	3
2	2	1	1	1	1	1	2	2	3	3	5	3	3	3
3	2	2	1	1	1	2	2	3	3	3	5	3	3	3
3	3	2	2	2	2	2	3	3	3	3	5	3	3	3
3	3	3	3	3	3	3	3	3	3	3	5	3	3	-1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	-1
3	3	3	3	3	3	6	6	6	6	3	3	3	3	-1
3	3	3	3	3	3	6	3	3	6	3	3	3	3	-1
3	3	3	6	6	6	3	3	3	6	3	3	3	3	-1
3	3	3	3	3	6	6	6	6	6	3	3	3	3	-1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	-1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	-1
3	3	3	3	3	3	3	3	3	3	3	3	3	3	-1
3	3	3	3	3	3	-1	-1	-1	-1	-1	-1	-1	-1	-1

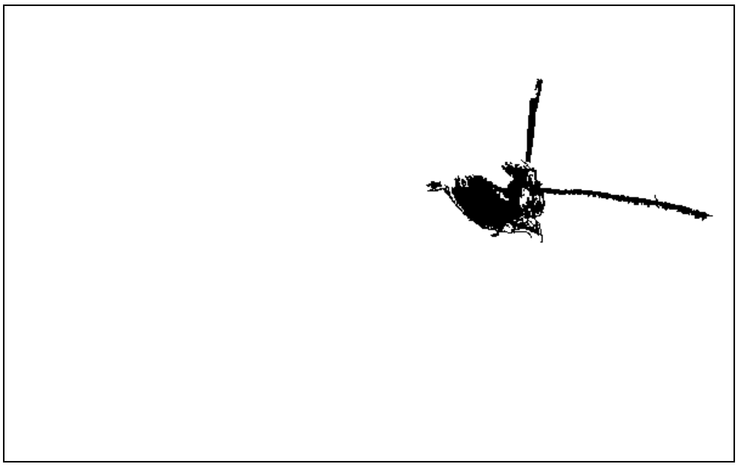
物件範圍分布情形 test15x17.txt



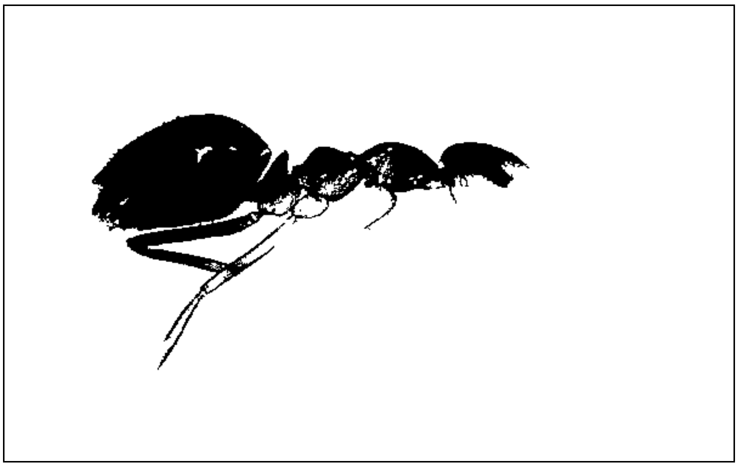
原始影像 ant(gray)600x400.raw



物件 1 分布範圍 out_1.raw



物件 7 分布範圍 out_7.raw



物件 22 分布範圍 out_22.raw