植物辨識

學生：蔡澔緯　　教授: 雷智偉 教授

摘要

這次作業主要是針對植物來做影像辨識，辨識的部分主要是包含:葉子的數量、植物的大小、莖的粗度、植物分支點(節)的數量等項目，我會先針對圖的照片進行處理，其中包括高斯濾波、二值化...等項目來進行處理，在主要內容中會更加詳細的說明，最後在做出辨識出該辨識的項目。

在現在的影像辨識中已有許多關於辨識植物的程式，也有關於辨識植物的產品的誕生，但現在大多數看到的產品都是針對植物中的葉進行辨識，針對葉來進行辨識來取得該葉子是甚麼植物，在這方面的技術已有許多程式開發者研究完畢。

目錄

第一章 前言

1.1 前言介紹

經過了這學期的課堂後，在這次作業六中，我們的作業內容是要辨識植物在圖片中的大小、植物葉片數量、植物有幾個分支點以及植物莖的粗度，這次作業對我來說像是考驗我這學期學習後的實作應用，我認為是相當有趣的。我所使用的工具是 Visual Studio 2019 下去執行的，影像辨識則是透過 opencv 去撰寫 Code。下圖為我本次作業辨識的圖片。

1.2 前言背景

在開始作業之前，查閱了許多關於市面上判斷植物的方法，但大多數判斷的方式是針對葉來進行判斷的，裡面內容敘述主要會先透過邊緣檢測去得出此葉子的形狀，後來去取得此葉子上的紋路特徵，透過這兩個特徵去比對開發者輸入到數據庫的資料進行比對，但在看完之後，我認為這對我來說收穫不大。後來我去觀看一些過去其他人所撰寫的論文中，有看到是針對莖的影像辨識的內容，在內容中看到許多之前在課堂中學習到的技術，透過論文上提供的方式，我去實測進而得到最後的結果。

第二章 本文

2.1 內文說明

以下我分別對判斷四個主要判斷的實際步驟來敘述，在實作部分會透過圖片搭配文字更佳了解我的實際的步驟得出來我的圖的部分。

葉片搜尋方法；最一開始將圖片匯入，因為圖片本身會有雜訊所以第一步將雜訊濾波，透過高斯濾波來進行。第二步我想將背景轉為黑色方便我來判斷，透過二維陣列來表示圖片每個 pixel，進而移除背景，將背景變為黑色。第三步進行二值化，第四步 Distance transform 演算法，第五步再取一次二值化，第六步 create a marker ，第七步填色，最後在進行判斷有幾個顏色，後面幾部是透過水壩法其中的方式來呈現的，再辨識莖跟節的方法我也是透過這樣的方式去呈現的。

植物大小搜尋方法:去背後透過二維列抓取圖像的 x、y 來擷取取得植物的大小。

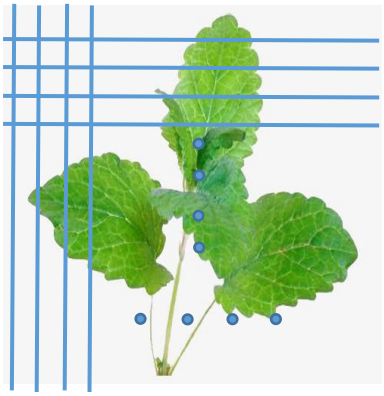
節的搜尋方式:在葉片搜尋中第三步二值化後，我將圖片進行較大(10*10)的侵蝕再膨脹，因為莖的粗度沒有像葉或者是節那樣的粗，在較大的侵蝕後，莖的部分會直接整個被侵蝕。將圖片的莖切割之後執行 create a maker 填色，最後得出來節的個數。

莖的粗度搜尋方式:與上述方式類似，主要是在圖片切割莖後讓原圖去撿去，就可以得出莖的位置，最後只要侵蝕再膨脹就可得出主要的莖幹，最後透過陣列算法去計算他的粗度。

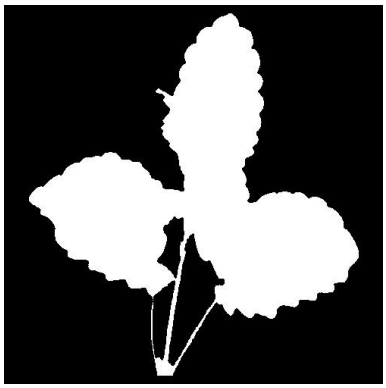
2.2 實作結果

第一個步驟將原圖匯入，在經過高斯 3*3 模板進行濾波。

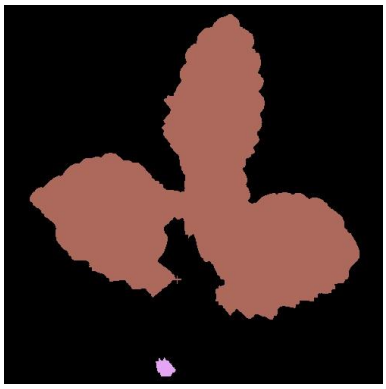
第二步將圖片背景去背，將圖片套入到二維陣列中，每個單位為一個 pixel，判斷該 pixel 是否為白色，如果是就讓他轉為黑色。

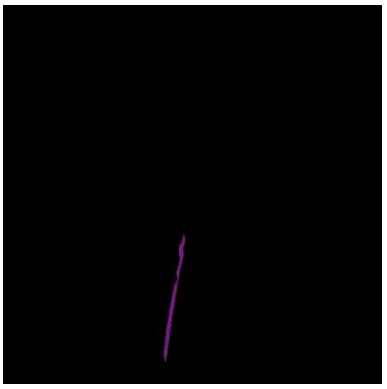


第三步我將圖片進行二值化，我使用的是一般的 Global threshold 的方式。



二值化後我利用(10*10)的模板進行侵蝕並填色，以及侵蝕後與原圖相減，為了取得主要莖幹我在做了一次侵蝕，這練利用的是(3*10)模板，將不必要的莖幹給侵蝕掉，最後擴散來達到原圖莖幹的比例。如以下兩圖所示。





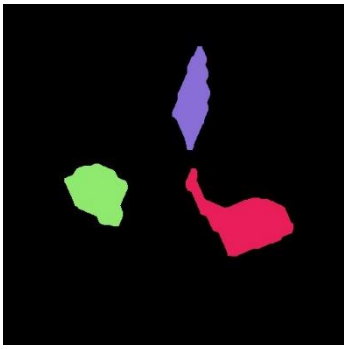
這邊我所用來計算節的方式是，我將圖片進行擴張讓葉子相連，這樣填色在葉子的部分就只會出現一個顏色，最後在記錄總共有幾個顏色，就可得知節的數量

莖的方式我是將圖片丟入二維列陣列中，每個陣列代表一個 pixel，去記錄有莖的哪幾列中，在針對這幾列來去記錄每列莖占多少欄位 pixel，而最終我是以佔最多欄位來當作莖的粗度。

第四部 回到二值化後，我利用之前在作水壩法時的其中的步驟，利用 Distance transform 演算法來去取得下圖，最終我想以水壩法來分辨每個葉片的部分。

第五步在取一次二值化後，得出下圖。並利用水壩法的製作過程中，給他填色便能分辨有幾張葉片了。





在判斷有幾張葉片時，我的方法是利用陣列來記錄葉片的顏色，重複獨到同一個顏色的值時，我則不記錄該值。

在最後判斷該植物大小時，我也是透過二維陣列的方式去記錄有出現不是黑色的顏色時的最小欄數、最大欄數、最小列數、最大列數，就可以取得該植物的大小，以及該植物為在圖片的哪一個點。

以下是我辨識出的項目的輸出。

```
這張值物照片有1個分節點
植物莖的粗度為4pixel
葉片有3片
高度度為477pixel
寬度為434pixel
```

第三章 結論

3.1 結論

最終可以看出時作的部分透過文字顯示來表示該植物的量測值，轉換的方式都是透過顏色上的辨識為主軸，來去辨認項目。在計算尺寸大小時，因為沒有相對的長度比 pixel，所以這邊用 pixel 來表示，

在一開始作的時候，找不到相關研究的程式碼，網路上找到的文章都是針對於葉的研究，後來我改變一個方向，透過課堂中學到的方式，我就想到之前有做過類似撲克牌辨識的題目，利用水壩法去分開每個撲克牌我就開始想到是否葉片的辨認是否也可以套用，並開始我實作的開端。

在這次實作，對於該圖片的辨識雖然辨識出來了，但對於考量到匯入圖片如果是呈現橫的話，再分辨植物莖時就會出錯，所以對於此專案還尚未有更完善的處理。

3.2 參考文獻

水壩法製作
https://docs.opencv.org/3.4/d2/dbd/tutorial_distance_transform.html
玫瑰切割
http://ir.lib.nchu.edu.tw/bitstream/11455/75647/1/147633-9.pdf
Distance transform
https://homepages.inf.ed.ac.uk/rbf/HIPR2/distance.htm
文心蘭切花
https://book.tndais.gov.tw/Magazine/mag60/fourth.pdf

3.3 程式碼

```
#include <opencv2/core.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/highgui.hpp>
#include <iostream>
# include <opencv2/opencv.hpp>
using namespace std;
using namespace cv;

Mat photo(Mat dist,int i)
{//maker image
	Mat dist_8u;
	dist.convertTo(dist_8u, CV_8U);
	vector<vector<Point> > contours;
	findContours(dist_8u, contours, RETR_EXTERNAL,
CHAIN_APPROX_SIMPLE);

	Mat markers = Mat::zeros(dist.size(), CV_32S);
	//
	for (size_t i = 0; i < contours.size(); i++)
	{
		drawContours(markers, contours, static_cast<int>(i),
Scalar(static_cast<int>(i) + 1), -1);
	}

	circle(markers, Point(5, 5), 3, Scalar(255), -1);

	Mat mark;
	markers.convertTo(mark, CV_8U);
	bitwise_not(mark, mark);

	vector<Vec3b> colors;
	for (size_t i = 0; i < contours.size(); i++)
	{
		int b = theRNG().uniform(0, 256);
```

```
            int g = theRNG().uniform(0, 256);
            int r = theRNG().uniform(0, 256);
            colors.push_back(Vec3b((uchar)b, (uchar)g, (uchar)r));
        }

        Mat dst = Mat::zeros(markers.size(), CV_8UC3);

        for (int i = 0; i < markers.rows; i++)
        {
            for (int j = 0; j < markers.cols; j++)
            {
                int index = markers.at<int>(i, j);
                if (index > 0 && index <= static_cast<int>(contours.size()))
                {
                    dst.at<Vec3b>(i, j) = colors[index - 1];
                }
            }
        }

        return dst;
}

int main(int argc, char* argv[])
{
        // 載入圖片

        Mat src = imread("C:\\Users\\ASUS\\Desktop\\108-1\\影像辨識\\
圖.jpg");
        if (src.empty())
        {
            cout << "Could not open or find the image!\n" << endl;
            cout << "Usage: " << argv[0] << " <Input image>" << endl;
            return -1;
        }    imshow("原圖", src);
        ///高斯綠波
        GaussianBlur(src, src, Size(3, 3), 0, 0);
        //去背 黑色
```

```
    for (int i = 0; i < src.rows; i++) {
        for (int j = 0; j < src.cols; j++) {
            if (src.at<Vec3b>(i, j)[0] >= 200)
            {
                src.at<Vec3b>(i, j)[0] = 0;
                src.at<Vec3b>(i, j)[1] = 0;
                src.at<Vec3b>(i, j)[2] = 0;
            }
        }
    }imshow("背景黑色", src);

    ///2 值化//////////////////////////
    Mat bw;
    Mat be_erode;
    cvtColor(imgResult, bw, COLOR_BGR2GRAY);
    threshold(bw, bw, 10, 255, THRESH_BINARY );
    Mat element = getStructuringElement(cv::MORPH_CROSS, cv::Size(3,
3));
    dilate(bw, be_erode, element);
    erode(be_erode, be_erode, element);

    imshow("OPEN_2 值化",be_erode);



    /////distanceTransform////////////////////////////

    Mat dist;
    distanceTransform(be_erode, dist, DIST_L2, 3);

    normalize(dist, dist, 0, 1.0, NORM_MINMAX);
    imshow("Distance Transform Image", dist);

    /////抓節//////////////////////////////////////
    Mat element1 = getStructuringElement(cv::MORPH_CROSS,
cv::Size(10, 10));
    Mat eroded, temp;
    Mat canny_2;
```

```
Canny(be_erode, canny_2, 50, 100, 3);
erode(be_erode, eroded, element1);
dilate(eroded, temp, element1);

Mat fin1 = photo(temp,1);
imshow("判斷幾個節", fin1);
int stem_count = 0;
int stem_rember[] = { 0 };
for (int i = 0; i < fin1.rows; i++) {
	for (int j = 0; j < fin1.cols; j++) {
		if (fin1.at<Vec3b>(i, j)[0] != 0)
		{
			int test = fin1.at<Vec3b>(i, j)[0];
			//初始
			if (stem_rember[0] == 0)
				stem_rember[0] = test;
			int repeat = 0;
			for (int x = 0; x < stem_count; x++)
			{
				if (stem_rember[x] == test)
				{
					repeat++;
				}
			}
			if (repeat == 0)
			{
				stem_rember[stem_count] = test;
				stem_count++;
			}
		}
	}
}
cout << "這張值物照片有" << stem_count-1 << "個分節點\n";
//////////////////抓莖//////////////////
Mat stem = be_erode - temp;
stem = stem - canny_2 ;
```

```
	Mat element3 = getStructuringElement(cv::MORPH_CROSS,
cv::Size(3, 10));
	erode(stem, stem, element3);
	dilate(stem, stem, element3);
	dilate(stem, stem, element3);
	erode(stem, stem, element3);




	Mat fin2 = photo(stem,2);
	imshow("判斷莖粗度", fin2);
	int stem_weight1 = 0,stem_weight2=0;
	int stem_weight[] = { 0 };
	int stem_weightx[] = { 0 };
	int stem_weight_count = 0;
	for (int i = 0; i < fin2.rows; i++) {
		stem_weight_count = 0;
		for (int j = 0; j < fin2.cols; j++) {
			if (fin2.at<Vec3b>(i, j)[0] != 0)
			{
				//cout << i <<"__有" << j<< "點\n";
				stem_weight[stem_weight_count] = j;
				stem_weight_count++;
				if (stem_weight_count > sizeof(stem_weightx))
				{
					for (int g = 0; g < sizeof(stem_weight); g++)
					{
						stem_weightx[g] = stem_weight[g];
					}
				}

			}
		}
	}
	for (int g = 0; g < sizeof(stem_weightx); g++)
	{
```

```
            //cout << "植物莖的粗度為" << stem_weightx[g] << "pixel\n";
            if (g == 0)
                stem_weight1 = stem_weightx[g];
            else
                stem_weight2 = stem_weightx[g];
        }
        cout << "植物莖的粗度為" << stem_weight2 - stem_weight1+1 <<
"pixel\n";




        ///2 值化膨脹//////////////////

        threshold(dist, dist, 0.55, 1.0, THRESH_BINARY);
        // Dilate a bit the dist image
        Mat kernel1 = Mat::ones(7,7, CV_8U);
        dilate(dist, dist, kernel1);
        imshow("Peaks", dist);

        //水壩法////////////////
        Mat dist_8u;
        dist.convertTo(dist_8u, CV_8U);

        vector<vector<Point> > contours;
        findContours(dist_8u, contours, RETR_EXTERNAL,
CHAIN_APPROX_SIMPLE);

        Mat markers = Mat::zeros(dist.size(), CV_32S);

        for (size_t i = 0; i < contours.size(); i++)
        {
            drawContours(markers, contours, static_cast<int>(i),
Scalar(static_cast<int>(i) + 1), -1);
        }
```

```
circle(markers, Point(5, 5), 3, Scalar(255), -1);
imshow("Markers", markers * 10000);

Mat mark;
markers.convertTo(mark, CV_8U);
bitwise_not(mark, mark);

vector<Vec3b> colors;
for (size_t i = 0; i < contours.size(); i++)
{
	int b = theRNG().uniform(0, 256);
	int g = theRNG().uniform(0, 256);
	int r = theRNG().uniform(0, 256);
	colors.push_back(Vec3b((uchar)b, (uchar)g, (uchar)r));
}

Mat dst = Mat::zeros(markers.size(), CV_8UC3);
for (int i = 0; i < markers.rows; i++)
{
	for (int j = 0; j < markers.cols; j++)
	{
		int index = markers.at<int>(i, j);
		if (index > 0 && index <= static_cast<int>(contours.size()))
		{
			dst.at<Vec3b>(i, j) = colors[index - 1];
		}
	}
}

imshow("Final Result", dst);
////尋找幾個葉片////////////////

int red_count = 1;
int red_rember[] = { 0 };
for (int i = 0; i < dst.rows; i++) {
	for (int j = 0; j < dst.cols; j++) {
		if (dst.at<Vec3b>(i, j)[0] != 0 & dst.at<Vec3b>(i, j)[1] != 0 &
```

```
dst.at<Vec3b>(i, j)[2] != 0)
                {
                    int test = dst.at<Vec3b>(i, j)[0];
                    //初始
                    if (red_rember[0] == 0)
                        red_rember[0] = test;
                    int repeat = 0;
                    for (int x = 0; x < red_count; x++)
                    {
                        if (red_rember[x] == test)
                        {
                            repeat++;
                        }
                    }
                    if (repeat == 0)
                    {
                        red_rember[red_count] = test;
                        red_count++;
                    }
                }
            }
        }
        cout << "葉片有" << red_count<<"片\n";
        //////////////////////找大小//////////////////////

        int height1 = 0, height2 = 0;
        int sch =0;
        ////////////////找高////////////////////
        for (int i = 0; i < src.rows; i++) {
            for (int j = 0; j < src.cols; j++) {
                if (src.at<Vec3b>(i, j)[0] != 0)
                {
                    if (sch == 0)
                    {
                        height1 = i;
                        sch++;
                    }
```

```
                else {
                    if (height2 < i)
                    {
                        height2 = i;
                    }
                }

            }
        }
    }cout << "高度度為" << height2 - height1 << "pixel\n";

    ////////////////////找寬//////////////////////////
    int weith1 = 0, weigh2 = 0;
    sch = 0;
    for (int j = 0; j < src.cols; j++) {
        for (int i = 0; i < src.rows; i++) {
            if (src.at<Vec3b>(i, j)[0] != 0)
            {
                if (sch == 0)
                {
                    weith1 = j;
                    sch++;
                }
                else {
                    if (weigh2 < j)
                    {
                        weigh2 = j;
                        //cout << weigh2;
                    }
                }

            }
        }
    }
    cout << "寬度為" << weigh2 - weith1 << "pixel\n";
    Mat image_src;
    Mat imageROI;
```

```
	Mat TempImg;
	int x_begin =weith1, y_begin = height1, width=weigh2-weith1,
height=height2-height1;
	image_src = src.clone();
	imageROI = image_src(Rect(x_begin, y_begin, width, height));
	imageROI.convertTo(TempImg, TempImg.type());
	imshow("剪過後", TempImg);
	waitKey();
	return 0;
}
```