MACHINE VISION LEAD FRAME INSPECTION

YU-JUI, CHEN

B053021023

ABSTRACT

- Open Picture
- Binary
- Sobel Edge Detection
- Find Circle
- Find Angle
- Find Width

OPEN PICTURE

```
private: System::Void button1_Click(System::Object^  sender, System::EventArgs^  e) {
    //宣告點陣圖的格式
    //Bitmap^ image1;
    //new一個新的openFileDialog1開啟檔案
    OpenFileDialog^ openFileDialog1 = gcnew OpenFileDialog;
    //設定Filter，用以限定使用者開啟的檔案
    openFileDialog1->Filter = "點陣圖 (*.bmp)| *.bmp| All files (*.*)| *.*";
    //預設檔案名稱為空值
    openFileDialog1->FileName = "";
    //設定跳出選擇視窗的標題名稱
    openFileDialog1->Title = "載入影像";
    //設定Filter選擇模式(依照Filter數，如本例選擇1表示起始出現的為點陣圖，選擇2表示All filts)
    openFileDialog1->FilterIndex = 1;


    //跳出檔案選擇視窗(ShowDialog)，並且如果使用者點選檔案，並且檔案名稱超過0個字元，則判斷是為True，進入處理程序
    if (openFileDialog1->ShowDialog() == System::Windows::Forms::DialogResult::OK && openFileDialog1->FileName->Length > 0)
    {
        //將選取的檔案讀取並且存至Image1
        image1 = safe_cast<Bitmap^>(Image::FromFile(openFileDialog1->FileName));
        //設定rect範圍大小
        rect = Rectangle(0, 0, image1->Width, image1->Height);
        //將影像顯示在pictureBox1
        pictureBox1->Image = image1;

    }
}
```

BINARY

```
private: System::Void button2_Click(System::Object^  sender, System::EventArgs^  e) {
    //影像轉二值化
    //宣告binaryImage為BitMap^型態
    binarybmp = gcnew Bitmap(image1->Width, image1->Height);
    //宣告grayImageData為BitmapData^型態，表示像素資料位置
    Imaging::BitmapData^ binarybmpData;
    //鎖定欲處理像素位置
    binarybmpData = binarybmp->LockBits(rect, System::Drawing::Imaging::ImageLockMode::ReadWrite, image1->PixelFormat);
    //將int指標指向Image像素資料的最前面位置
    ResultPtr = binarybmpData->Scan0;
    //設定指標
    R = (Byte*)((Void*)ResultPtr);

    //鎖定原圖欲處理像素位置
    ImageData1 = image1->LockBits(rect, System::Drawing::Imaging::ImageLockMode::ReadWrite, image1->PixelFormat);
    //將int指標指向原圖像素資料最前面位置
    ptr = ImageData1->Scan0;
    //設定指標
    p = (Byte*)((Void*)ptr);

    int threshold = 50;
```

```
    int threshold = 50;

    //瀏覽所有像素點，取用像素，處理像素區域
    for (int y = 0; y < image1->Height; y++)
    {
        for (int x = 0; x < image1->Width; x++)
        {

            if (p[0] > 50) {
                p[0] = 255;
            }
            else {
                p[0] = 0;
            }
            R[0] = p[0];

            R++;
            p++;

        }
    }

    //解鎖像素位置
    image1->UnlockBits(ImageData1);
    binarybmp->UnlockBits(binarybmpData);
    //顯示在PictureBox上
    pictureBox2->Image = binarybmp;

}
```

SOBEL EDGE DETECTION

```
private: System::Void button4_Click(System::Object^  sender, System::EventArgs^  e)
{
    int kernely[9] = { -1,0,1,-2,0,2,-1,0,1 };
    int kernelx[9] = { -1,-2,-1,0,0,0,1,2,1 };
    int32_t* data = new int32_t[image1->Width * image1->Height];
    int32_t* dataPointer = data;


    binarybmp = gcnew Bitmap(image1->Width, image1->Height);
    //宣告grayImageData為BitmapData^型態，表示像素資料位置
    Imaging::BitmapData^ binarybmpData;
    //鎖定欲處理像素位置
    binarybmpData = binarybmp->LockBits(rect, System::Drawing::Imaging::ImageLockMode::ReadWrite, image1->PixelFormat);
    //將int指標指向Image像素資料的最前面位置
    ResultPtr = binarybmpData->Scan0;
    //設定指標
    k = (Byte*)((Void*)ResultPtr);

    //鎖定原圖欲處理像素位置
    ImageData1 = image1->LockBits(rect, System::Drawing::Imaging::ImageLockMode::ReadWrite, image1->PixelFormat);
    //將int指標指向原圖像素資料最前面位置
    ptr = ImageData1->Scan0;
    //設定指標
    p = (Byte*)((Void*)ptr);

    Byte* r[9];

    for (int y = 0; y < image1->Height; y++) {
        for (int x = 0; x < image1->Width; x++) {
            int sumx = 0;
            int sumy = 0;
```

```
    for (int y = 0; y < image1->Height; y++) {
        for (int x = 0; x < image1->Width; x++) {
            int sumx = 0;
            int sumy = 0;

            //最外圖像素不處理
            if (y > 0 && x > 0 && y < image1->Height - 1 && x < image1->Width - 1)
            {

                int Masksize = 0;
                int kernelindex = 0;
                //尋訪3X3遮罩的九個像素 並將像素值存下來
                for (int i = -1; i <= 1; i++)
                {
                    for (int j = -1; j <= 1; j++)
                    {
                        //指標指向目標像素位址
                        r[Masksize] = (Byte*)(Void*)p + i * image1->Width + j;
                        //儲存像素
                        sumx += (int)r[Masksize][0] * kernelx[kernelindex];
                        sumy += (int)r[Masksize][0] * kernely[kernelindex];
                        Masksize++;
                        kernelindex++;
                    }
                }
            }
            dataPointer[0] = abs(sumx) + abs(sumy);
            if (dataPointer[0] > 255) {
                dataPointer[0] = 255;
            }
            k[0] = dataPointer[0];
            p++;
            k++;
            dataPointer++;
        }
    }
```

Vertical Sobel =

-1	0	1
-2	0	2
-1	0	1

Horizontal Sobel =

-1	2	-1
0	0	0
1	2	1

$$F(x) = \sqrt{Vertical\ Sobel^2 + Horizontal\ Sobel^2}$$

FIND CIRCLE

- A. Connected component labeling
- B. Find Diameter

A. CONNECTED COMPONENT LABELING

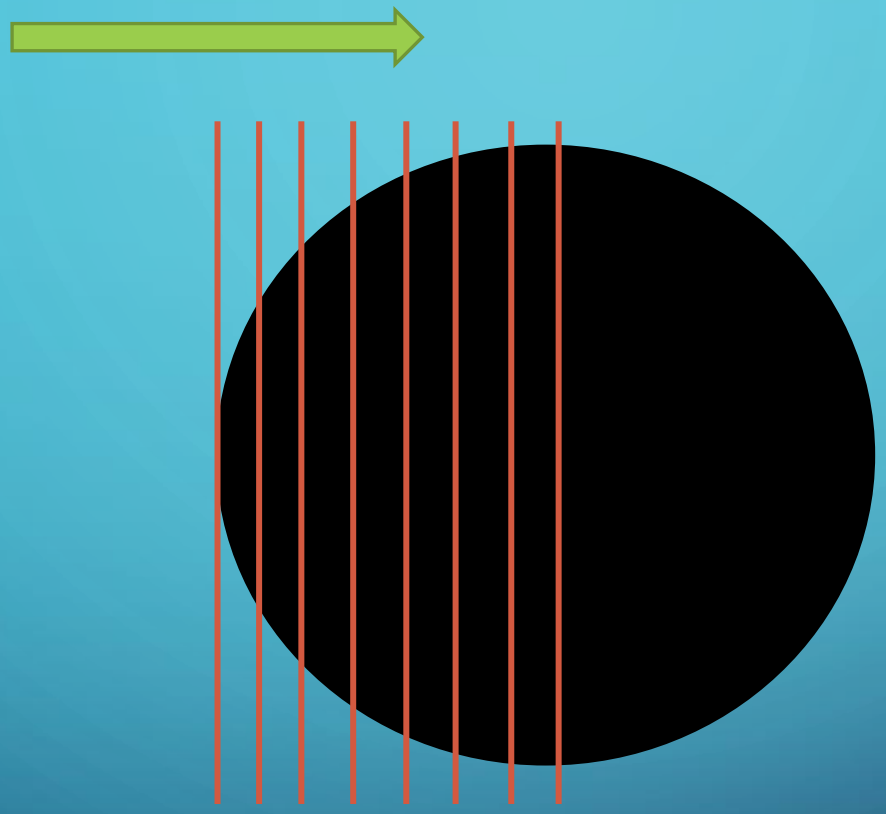
```
if (Label->GetPixel(i, j).R == 0) {


    if (((Label->GetPixel(i - 1, j).R == 0) || (Label->GetPixel(i - 1, j).R == 255)) && ((Label->GetPixel(i, j - 1).R == 0) || (Label->GetPixel(i, j - 1).R == 255))) {

        L[check] = check;
        Color set = Color::FromArgb(r, g, b);
        Label->SetPixel(i, j, set);
        color[check][0] = r;
        color[check][1] = g;
        color[check][2] = b;

        check++;
        r += 1;
        if (r == 250) {
            r = 1;
            g += 1;
            if (g == 250) {
                g = 1;
                b += 1;
            }
        }
    }
    else if (((Label->GetPixel(i, j - 1).R != 0) && (Label->GetPixel(i, j - 1).R != 255)) && ((Label->GetPixel(i - 1, j).R == 0) || (Label->GetPixel(i - 1, j).R == 255))) {

        Color set = Label->GetPixel(i, j - 1);
        Label->SetPixel(i, j, set);
    }

    else if (((Label->GetPixel(i - 1, j).R != 0) && (Label->GetPixel(i - 1, j).R != 255)) && ((Label->GetPixel(i,

        Color set = Label->GetPixel(i - 1, j);
        Label->SetPixel(i, j, set);
```

```
else if (((Label->GetPixel(i, j - 1)).R == Label->GetPixel(i - 1, j).R) && ((Label->GetPixel(i, j - 1)).G == Label->GetPixel(i - 1, j).G) && ((Label->GetPixel(i, j - 1)).B == Label->GetPixel(i - 1, j).B)) {

    Color set = Label->GetPixel(i, j - 1);
    Label->SetPixel(i, j, set);
}
else if (((Label->GetPixel(i, j - 1)).R != Label->GetPixel(i - 1, j).R) || ((Label->GetPixel(i, j - 1)).G != Label->GetPixel(i - 1, j).G) || ((Label->GetPixel(i, j - 1)).B != Label->GetPixel(i - 1, j).B)) {

    int checkup, checkleft;
    Color set = Label->GetPixel(i, j - 1);
    Label->SetPixel(i, j, set);
    for (int x = 1; x < 5000; x++) {
        if ((color[x][0] == Label->GetPixel(i, j - 1).R) && (color[x][1] == Label->GetPixel(i, j - 1).G) && (color[x][2] == Label->GetPixel(i, j - 1).B)) {

            checkup = x;
            break;
        }
    }
    for (int x = 1; x < 5000; x++) {
        if ((color[x][0] == Label->GetPixel(i - 1, j).R) && (color[x][1] == Label->GetPixel(i - 1, j).G) && (color[x][2] == Label->GetPixel(i - 1, j).B)) {

            checkleft = x;
            break;
        }

    }
    int pre = L[checkleft];
    L[checkleft] = L[checkup];
    for (int x = 1; x < 5000; x++) {
        if (L[x] == 0) {
            break;
        }
        else if (L[x] == pre) {
            L[x] = L[checkup];
```

B. FIND DIAMETER

```
    int d, dmin;
    int dmax = 0;
    int sum = 0;
    for (int x = 0; x < New->Width; x++)  //掃垂直  if holes are vertical change x,y
    {
        for (int y = 0; y < New->Height; y++)
        {
            if (New->GetPixel(x, y).R == 0) {
                sum++;
            }
            else if (New->GetPixel(x, y).R == 255) {
            }
        }

        if (sum >= 0) { dmin = sum; }
        if (dmax <= dmin) { dmax = sum; }
        else if (dmax > dmin) { dmax = dmax; }
        sum = 0;
    }


    this->label2->Visible = true;
    this->label4->Visible = true;
    this->label2->Text = "" + n + "";
    this->label4->Text = "" + dmax +"";

    diameter = dmax;
     Bitmap^result= New;
    pictureBox3->Image = result;
}
```

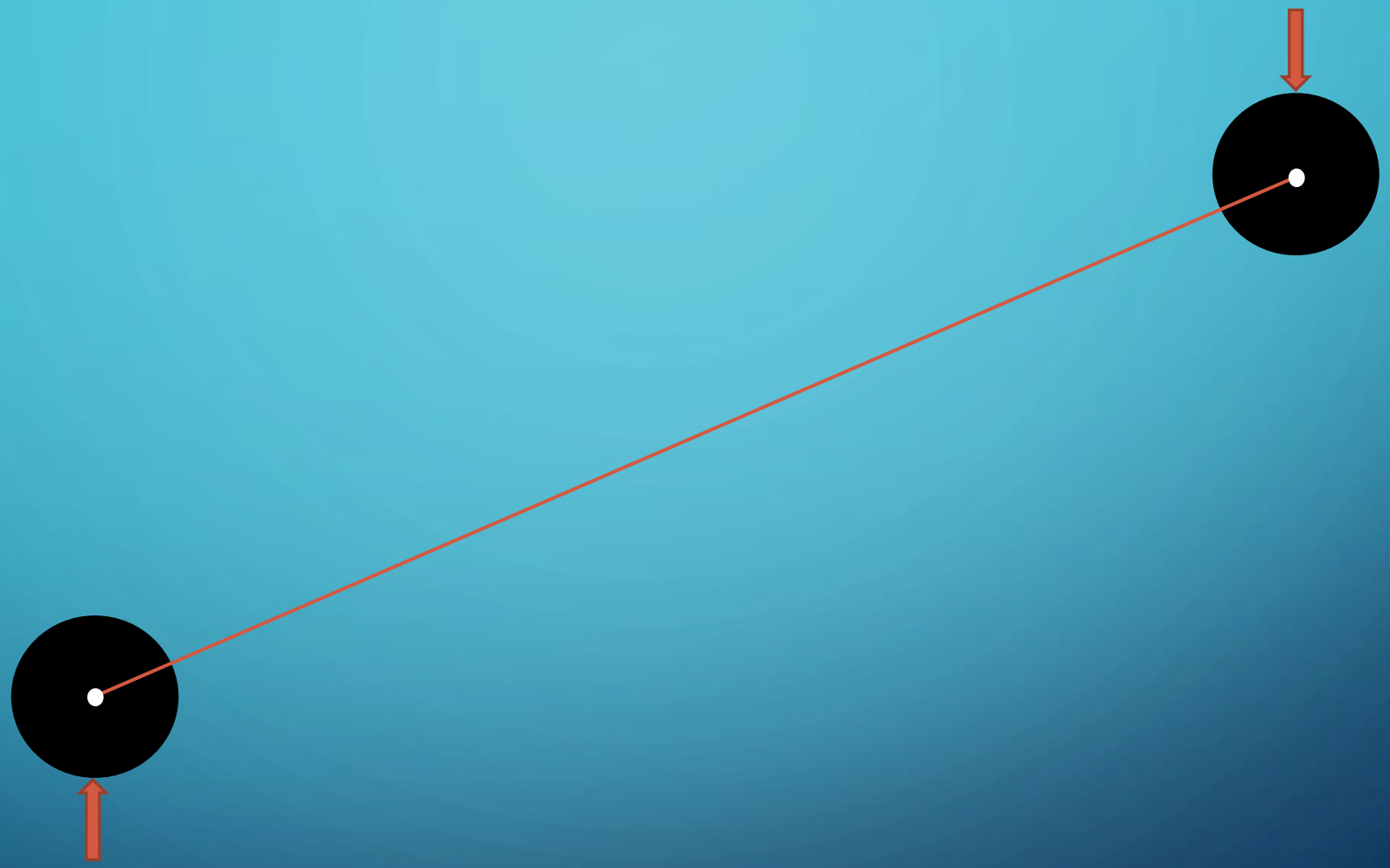
Scan Direction



FIND ANGLE

```
private: System::Void button6_Click(System::Object^  sender, System::EventArgs^  e) {
    int a; ////上方
    int b;
    int c;  ////下
    int d;

    for (int j = 0; j < binarybmp->Height; j++)
    {
        for (int i = 0; i < binarybmp->Width; i++)
        {
            int KK = binarybmp->GetPixel(i, j).R;
            if (KK == 0)
            {
                a = i;
                b = j + diameter / 2;
                j = binarybmp->Height;
                i = binarybmp->Width;
            }
        }
    }
    for (int j = binarybmp->Height - 1; j >0; j--)
    {
        for (int i = binarybmp->Width - 1; i >0; i--)
        {
            if (binarybmp->GetPixel(i, j).R == 0)
            {
                c = i;
                d = j - diameter / 2;
                j = 0;
                i = 0;
            }
        }
    }
    float w = abs(a - c);
    float h = abs(b - d);
    slide = (atan(h / w) * 180) / PI;
```

First Step

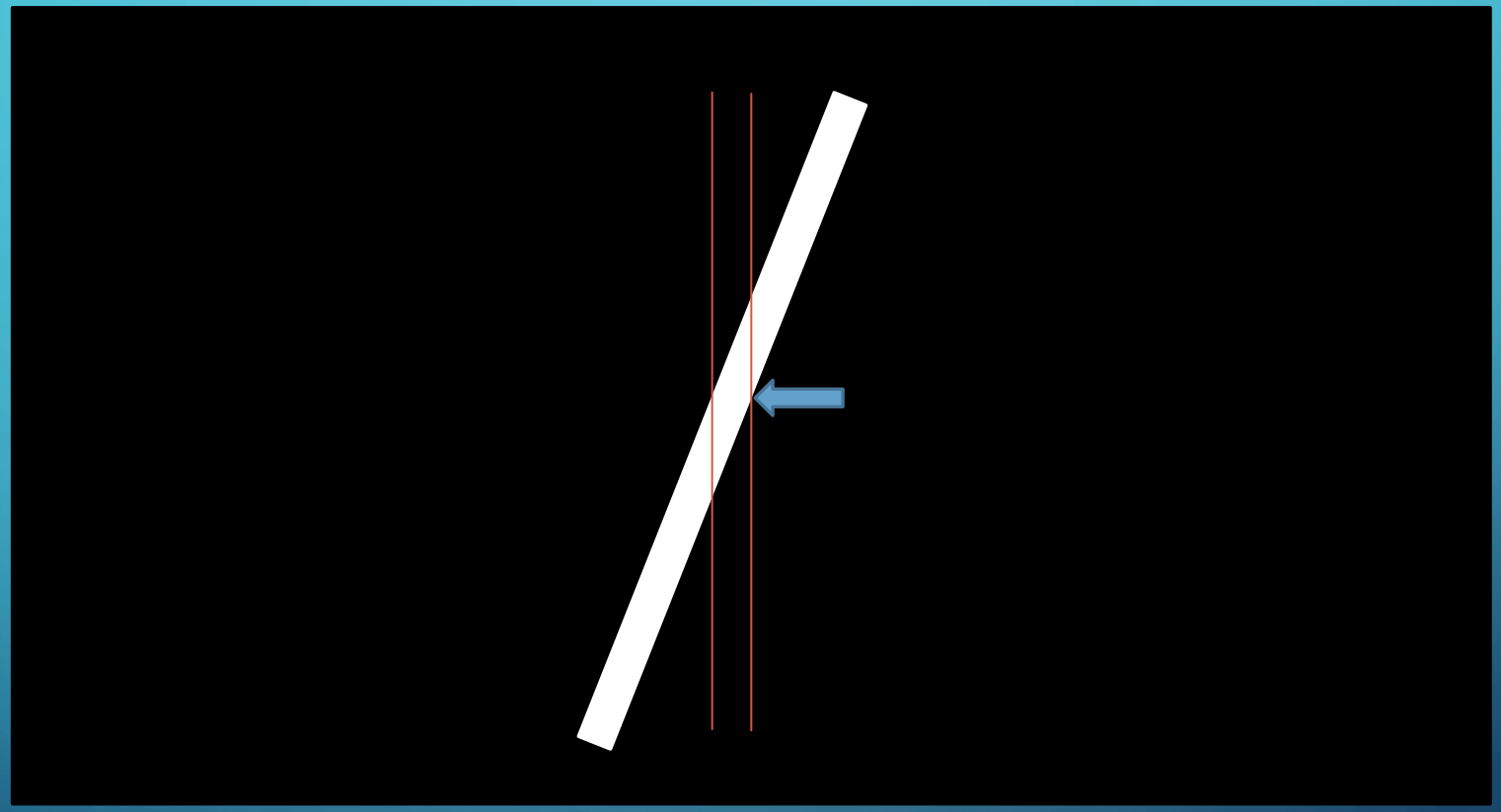


Second Step

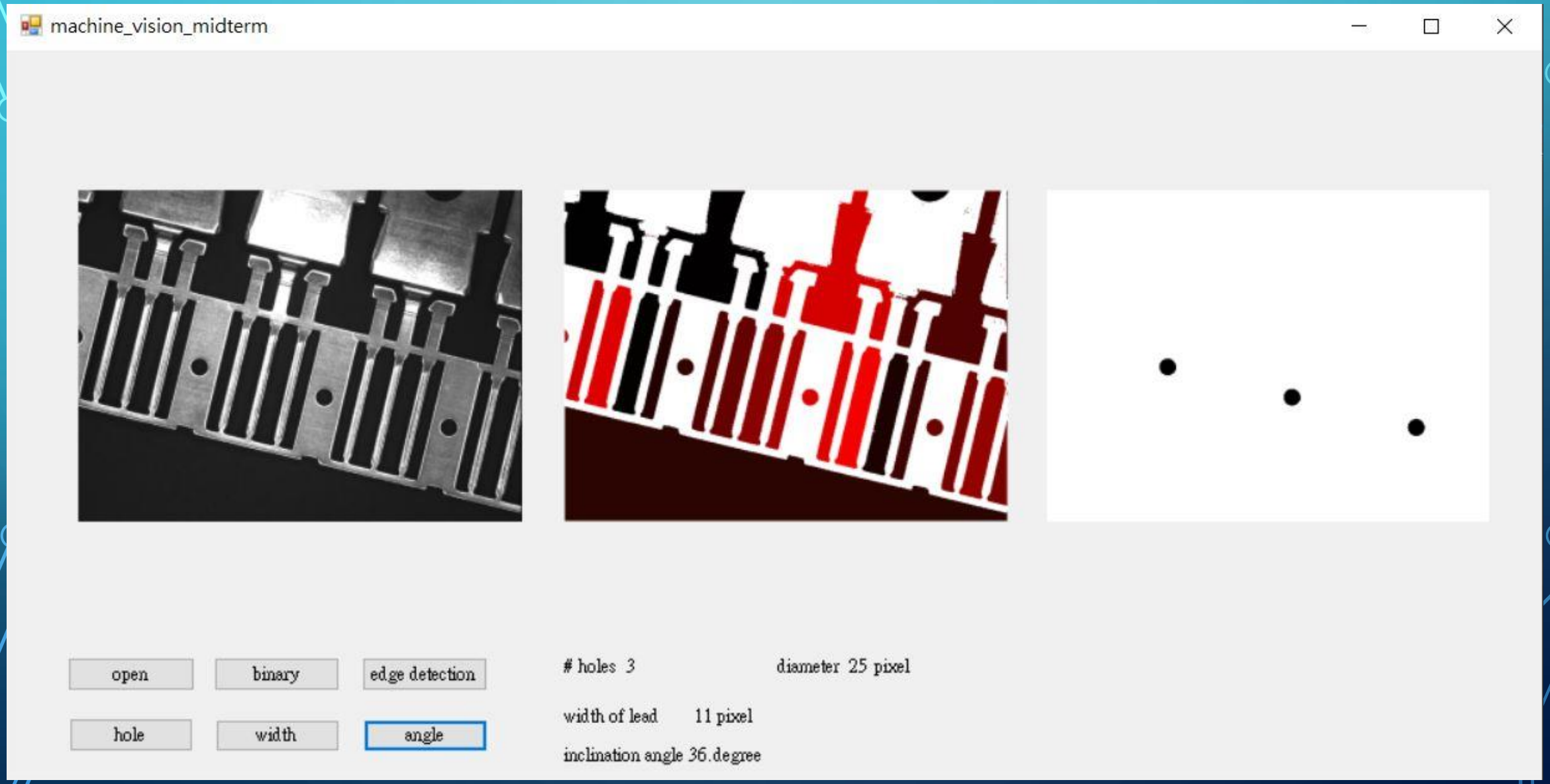
FIND WIDTH

```
private: System::Void button5_Click(System::Object^  sender, System::EventArgs^  e) {
    int x, y;
    int n = 0;
    int length;

    if (binarybmp->GetPixel(320, 240).R == 0)
    {
        x = 320;
        y = 240;
     }
    else if (binarybmp->GetPixel(320, 240).R == 255)
    {
        for (int left = 1; left < 100; left++)
        {
            length = 320 - left;

            if (binarybmp->GetPixel(length, 240).R == 0)
            {
                x = 320 - length;
                y = 240;
                left = 100;

            }
        }
    }
```

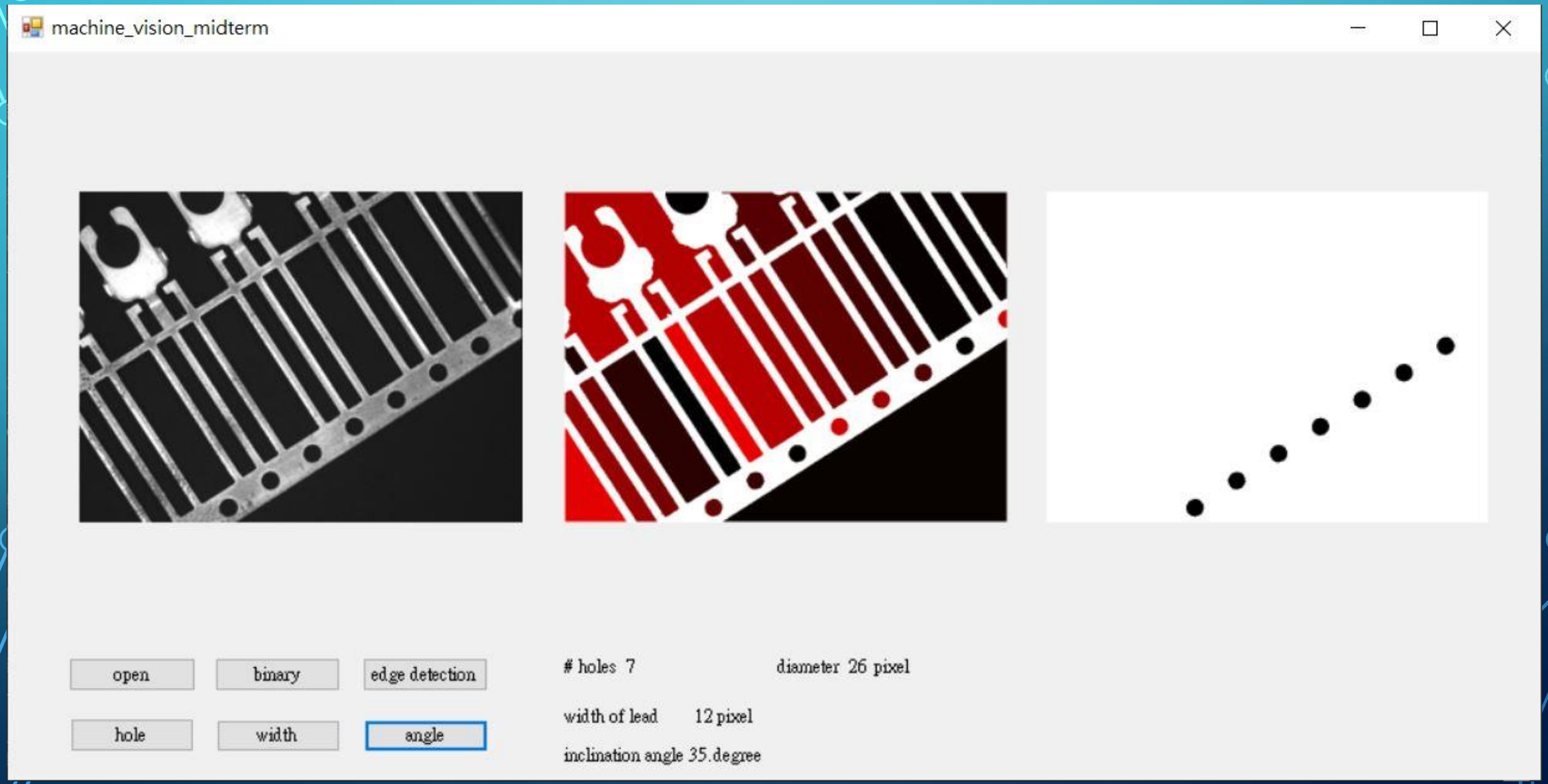
```
for (int i=x ; i>0; i--)
{
    if (binarybmp->GetPixel(i,y).R==255)
    {
        n++;
    }

    else if (binarybmp->GetPixel(i, y).R == 0)
    {
        if (n !=0)
        {
            break;
        }

        else if (n == 0)
        {

        }
    }
}

int width = n*cos(slide*(PI) / 180);
this->label6->Visible = true;
this->label6->Text = "" + width + "";
```



RESULT


machine_vision_midterm
open
binary
edge detection
hole
width
angle
holes 3
diameter 25 pixel
width of lead 11 pixel
inclination angle 36.degree

RESULT


machine_vision_midterm
open
binary
edge detection
hole
width
angle
holes 7
diameter 26 pixel
width of lead 12 pixel
inclination angle 35.degree

COMPUTER TEST