for (int index4 = max\_index \* 8 - 4; index4 <= max\_index \* 8 + 4; index4 += 4)

{

//旋转模板

cv2DRotationMatrix(cvPoint2D32f(pCMatTemplate4->width / 2, pCMatTemplate4->height / 2), index4, 1, matrix);

cvWarpAffine(pCMatTemplate4, pMatTemplate4, matrix, CV\_INTER\_LINEAR + CV\_WARP\_FILL\_OUTLIERS, cvScalarAll(0));

GetTemplatePara(pMatTemplate4);//计算当前所用的模板的参数（平方和等），该函数自行实现

nWidth = pMatTemplate4->width;

nHeight = pMatTemplate4->height;

pIData = pMatImage4->data.ptr;

nIstep = pMatImage4->step;

pTData = pMatTemplate4->data.ptr;

nTstep = pMatTemplate4->step;

TempPointAngle[(index4 + 4 - max\_index \* 8) / 4].score = -1;

NumCounter = 0;

for (k = 0; k < LEVEL3POINT; k++)//将1/8图像的坐标反映(映射)到1/4(高一级分辨率)图像下。

{

h = 2 \* SelectNcc[k].height;

w = 2 \* SelectNcc[k].width;

for (i = 0; i < 2; i++)

{

for (j = 0; j < 2; j++)

{

if (((w + j + nWidth) <= pMatImage4->width) && ((h + i + nHeight) <= pMatImage4->height))

{

ReSelectNcc[NumCounter].score = 0.0;

ReSelectNcc[NumCounter].width = w + j;

ReSelectNcc[NumCounter].height = h + i;

NumCounter++;

}

}

}

}

//对每个点计算NCC,求出NCC最大的点

for (k = 0; k < NumCounter; k++)

{

h = ReSelectNcc[k].height;

w = ReSelectNcc[k].width;

nSumIM = 0;

nSquareSumImage = 0;

nSumGrayImage = 0;

for (n = 0; n < nHeight; n++)

{

nTemp1 = (h + n) \* nIstep + w;

nTemp2 = n \* nTstep;

for (m = 0; m < nWidth; m++)

{

nGray = pIData[nTemp1 + m];

nSumGrayImage += nGray;

nSquareSumImage += nGray \* nGray;

nSumIM += nGray \* (pTData[m + nTemp2]);

}

}

fNsumIM = (double)m\_nTemplateSizeNow \* nSumIM;

fNSquareSumImage = (double)m\_nTemplateSizeNow \* nSquareSumImage;

fSumSquareImage = (double)nSumGrayImage \* nSumGrayImage;

temp = (double)(fNSquareSumImage - fSumSquareImage) \* (m\_NSquareSumTemplateNow - m\_SumSquareTemplateNow);//这里利用方差公式DX=E(X^2)-(EX)^2

if (temp < 0) temp = -temp;

if (temp == 0)

NCC = 0.0;

else

NCC = SquareFunc(fNsumIM - (double)nSumGrayImage \* m\_nSumGrayTemplateNow) / temp;

//选出NCC分数最大的点

if (NCC > TempPointAngle[(index4 + 4 - max\_index \* 8) / 4].score)

{

TempPointAngle[(index4 + 4 - max\_index \* 8) / 4].score = NCC;

TempPointAngle[(index4 + 4 - max\_index \* 8) / 4].width = w;

TempPointAngle[(index4 + 4 - max\_index \* 8) / 4].height = h;

}

}

if (TempPointAngle[(index4 + 4 - max\_index \* 8) / 4].score > score\_4max)

{

score\_4max = TempPointAngle[(index4 + 4 - max\_index \* 8) / 4].score;

TempPoint = TempPointAngle[(index4 + 4 - max\_index \* 8) / 4];

rotation4 = index4;

}

}