int main()

{

SurfFeatureDetector surf( 3000 , 2 , 3 , true , true );

std::vector<KeyPoint> keypoints,keypoints2;

SurfDescriptorExtractor extractor;

Mat descriptors,descriptors2;

Ptr<DescriptorMatcher> matcher = DescriptorMatcher::create("BruteForce");

std::vector< DMatch > matches;

//for(int n=1;n<300;n++){

int n=200;

printf("第A%d次¡M\n",n);

char \*cstr=new char[20];

char \*cstr2=new char[20];

char \*cstr3=new char[20];

char \*cstr4=new char[20];

sprintf(cstr,"mobile%d%s",n,".jpg");

sprintf(cstr2,"mobile%d%s",n+40,".jpg");

sprintf(cstr3,"mobile\_compensation%d%s",n,".jpg");

sprintf(cstr4,"mobile\_difference%d%s",n,".jpg");

Mat img=imread(cstr);

Mat img2=imread(cstr2);

surf.detect(img,keypoints);

extractor.compute( img, keypoints, descriptors );

surf.detect(img2,keypoints2);

extractor.compute( img2, keypoints2, descriptors2 );

matcher->match(descriptors, descriptors2, matches);

double max\_dist = 0; double min\_dist = 100;

//-- Quick calculation of max and min distances between keypoints

for (int i = 0; i < descriptors.rows; i++)

{

double dist = matches[i].distance;

if (dist < min\_dist) min\_dist = dist;

if (dist > max\_dist) max\_dist = dist;

}

std::vector< DMatch > good\_matches;

for (int i = 0; i < descriptors.rows; i++)

{

if (matches[i].distance < 3 \* min\_dist)good\_matches.push\_back(matches[i]);

}

Mat img\_matches;

drawMatches(img, keypoints, img2, keypoints2, good\_matches, img\_matches, Scalar::all(-1), Scalar::all(-1), std::vector<char>(), DrawMatchesFlags::NOT\_DRAW\_SINGLE\_POINTS);

namedWindow("Lena",WINDOW\_AUTOSIZE);

float dx=0.0,dy=0.0;

/\*for( int i = 0; i < (int)keypoints.size(); i++ ){

Point2f point\_all=keypoints[i].pt;

Point2f point\_all2=keypoints2[i].pt;

circle(img, point\_all, 3, Scalar(0, 0, 255), 1);

circle(img2, point\_all2, 3, Scalar(0, 255, 255), 1);

}\*/

Point2f srcTriangle[3];

Point2f dstTriangle[3];

for( int i = 0; i < (int)good\_matches.size(); i++ )

{

Point2f point\_old = keypoints[good\_matches[i].queryIdx].pt;

Point2f point\_new = keypoints2[good\_matches[i].trainIdx].pt;

//circle(img, point\_old, 3, Scalar(0, 0, 255), 1);

//circle(img2, point\_new, 3, Scalar(0, 255, 255), 1);

dx=point\_old.x-point\_new.x+dx;

dy=point\_old.y-point\_new.y+dy;

if(i==0){srcTriangle[2]=point\_old;dstTriangle[2]=point\_new;}

else if(i==1){srcTriangle[1]=point\_old;dstTriangle[1]=point\_new;}

}

double distance=norm(srcTriangle[2]-srcTriangle[1])/norm(dstTriangle[2]-dstTriangle[1]);

printf("distance==%f\n",distance);

printf("%f\t%f\t%f\t%f\n",srcTriangle[2].x,srcTriangle[2].y,dstTriangle[2].x,dstTriangle[2].y);

printf("%f\t%f\t%f\t%f\n",srcTriangle[1].x,srcTriangle[1].y,dstTriangle[1].x,dstTriangle[1].y);

//imshow("img\_matches",img\_matches);

//imwrite(cstr3,img\_matches);

int ddx,ddy;

ddx=floor(dx/good\_matches.size());

ddy=floor(dy/good\_matches.size());

printf("dx=%d and dy=%d \n", ddx,ddy);

Mat dst;

CV\_Assert(img2.depth() == CV\_8U);

dst.create(img2.rows, img2.cols, img2.type());

Vec3b \*p;

//評gu儀oimg2得Óo到Lidst

for (int i = 0; i < img2.rows; i++){

p = dst.ptr<Vec3b>(i);

for (int j = 0; j < img2.cols; j++){

int x,y;

if(ddx<-5)x = j - ddx-4;

else if(ddx>1) x = j - ddx+2;

else x = j-ddx ;

if(ddy<0)y = i -ddy-9;

else if(ddy>0) y = i -ddy;

else y = i-ddy ;

if (x >= 0 && y >= 0 && x < img2.cols && y < img2.rows) p[j] = img2.ptr<Vec3b>(y)[x];

}

}

Mat affine;

Mat warpMat =(Mat\_<double>(3,3) << 1, 0, 0, 0, 1, 0,0,0,1/distance);

warpPerspective( dst, affine, warpMat, affine.size() );

imshow("affine",affine);

//imwrite("affine.jpg",dst);

imshow("Lena",img);

//imwrite("img.jpg",img);

imshow("img2.jpg",img2);

//imwrite("img2.jpg",img2);

imshow("Lena2",dst);

//imwrite("translation.jpg",dst);

//imwrite("img2.jpg",img2);

//製s作±@mask為¢X了Finpaint local motion

Mat dstcom;

Mat mask=Mat(dst.rows,dst.cols,CV\_8UC1);

float b,g,r;

float current\_y;//作±@灰C階¢D運B送Xe得Óo到Li正D?確T值Ecurrent\_y

int y=0;//四D|捨ÓE五-入J判±P斷\_current\_y與PY值E是O否±\_要n進i位i

if(dx>0){

for(int k=0;k<dst.cols;k++){

for(int m=0;m<dst.rows;m++){

if(k<ddx+1&&m<400)y=255;

else y=0;

mask.at<unsigned char>(m,k)=y;

}}

}

else {

for(int k=0;k<dst.cols;k++){

for(int m=0;m<dst.rows;m++){

if(k>352+ddx&&m<400)y=255;

else y=0;

mask.at<unsigned char>(m,k)=y;

}}

}

inpaint(dst,mask,dstcom,50,INPAINT\_NS );

//記XO住i前e時E刻Le的o位i置Mm資Me訊XT

Mat past=Mat(img.rows,img.cols,CV\_8UC3);

CV\_Assert(img.depth() == CV\_8U);

past.create(img.rows, img.cols, img.type());

for (int i = 0; i < img2.rows; i++){

p = past.ptr<Vec3b>(i);

for (int j = 0; j < img2.cols; j++){

int x,y;

x = j ;

y=i;

if(dx>0){

if (x >= 0 && y >= 0 && x < ddx-4 && y < img.rows)

p[j] = affine.ptr<Vec3b>(y)[x];

else p[j] = dst.ptr<Vec3b>(y)[x]; }

else{

if (x >= 0 && y >= 0 && x > ddx+352 && y < img.rows)

p[j] = img.ptr<Vec3b>(y)[x];

else p[j] = affine.ptr<Vec3b>(y)[x]; }

}

}

imshow("past",past);

// imwrite(cstr3,past);

Mat difference = img - past;

imshow("difference", difference);

// imwrite(cstr4,difference);

imshow("mask",mask);

imshow("inpaint",dstcom);

//imwrite(cstr3,past);

//}

waitKey(0);

return 0;

}