THIS IS THE SOURCE CODE OF TOUCH PAD PROJECT BY AAKASH DABAS.
YOU CAN REUSE BUT WITHOUT CLAIMING ANY OF ITS RIGHTS.(DATED:20-07-2015)

```
#include "opencv2/highgui/highgui.hpp"
#include <fstream>
#include <iostream>
#include<stdio.h>
#include<Windows.h>
#include"timer.h"

using namespace cv;
using namespace std;

int w, h,lmt=50,lmt2=100,*fBgk,xT=0,yT=0,*mat,*matCmp,*matDet,touch=0;
float speed = 0,f=1,density=0;
bool fDetected = false, calib = false, l_Click = false,
r_Click=false,d_Click=false,drag=false,scroll=false,allowedC = false, allowedRC = false;
timer cntrl_1,cntrl_2;
Mat frameC,frameTmp;
POINT cP1, cP2;

struct bigPt
{
	double x = 0, y = 0;
}p1,p2;

float mod(float x)
{
	if (x > 0)
		return x;
	else return -x;
}

void circle(Mat frame, int x, int y, float rad, int r = 150, int g = 0, int b = 0)
{
	for (float agl = 0; agl <= (2 * 22) / 7.0; agl += 0.01)
	{
		Vec3b p;
		p.val[0] = b;
		p.val[1] = g;
		p.val[2] = r;
		int xC = x + rad*sin(agl), yC = y + rad*cos(agl);
		if (xC>0 && xC<w&&yC>0 && yC<h)
			frame.at<Vec3b>(Point(xC, yC)) = p;
```

```
	}
}

float distnc(bigPt pt1, bigPt pt2)
{
	return sqrt((pt1.x - pt2.x)*(pt1.x - pt2.x) + (pt1.y - pt2.y)*(pt1.y - pt2.y));
}

void line(Mat frame, float x1, float y1, float x2, float y2, int r = 0, int g = 200, int b = 0)
{
	Vec3b p;
	p.val[0] = b;
	p.val[1] = g;
	p.val[2] = r;
	if (x1 == x2)
	{
		for (int i = y1; i != y2&&i>0 && i<h&&x1>0 && x1<w; (y1<y2 ? i++ : i--))
			frame.at<Vec3b>(Point(x1, i)) = p;
	}
	else
	{
		float slope = (y1 - y2) / (x1 - x2);
		float intcpt = y2 - slope*x2;
		if (mod(slope)<1)
			for (float i = x1; i != x2&&i>0 && i<w; (x1 < x2 ? i++ : i--))
			{
				int j = slope*i + intcpt;
				if (i>0 && i < w&&j>0 && j < h)
					frame.at<Vec3b>(Point(i, j)) = p;
			}
		else
			for (float i = y1; i != y2&&i>0 && i<h; (y1 < y2 ? i++ : i--))
			{
				int j = (i - intcpt) / slope;
				if (i>0 && i < w&&j>0 && j < h)
					frame.at<Vec3b>(Point(j, i)) = p;
			}
	}
}

void rect(Mat frame, int x1, int y1, int lnth, int brth, int r = 255, int b = 0, int g = 0)
{
	Vec3b p;
	p.val[0] = b;
	p.val[1] = g;
```

```
	p.val[2] = r;
	for (int i = 0; i < lnth || i<brth; i++)
	{
		if (i < lnth&&i + x1<w&&i + x1>0)
		{
			if (y1>0 && y1<h)
				frame.at<Vec3b>(Point(i + x1, y1)) = p;
			if (y1 + brth<h&&y1 + brth>0)
				frame.at<Vec3b>(Point(i + x1, y1 + brth)) = p;
		}
		if (i < brth&&i + y1<h&&i + y1>0)
		{
			if (x1>0 && x1<w)
				frame.at<Vec3b>(Point(x1, y1 + i)) = p;
			if (x1 + lnth<w&&x1 + lnth>0)
				frame.at<Vec3b>(Point(x1 + lnth, y1 + i)) = p;
		}
	}
}

void compControl()
{
	POINT p;
	GetCursorPos(&p);
	if (fDetected&&cntrl_1.elapsed()>50)
	{
		if (distnc(p1, p2) > 2)
		{
			if (distnc(p1, p2) < 10)
				f = 2;
			else if (distnc(p1, p2) < 20)
				f = 2.5;
			else
				f = 3.5;
		}
		else
			f = 0;
		float xDif = p1.x - p2.x, yDif = p1.y - p2.y;
		if (mod(xDif) < 50 && mod(yDif) < 50)
		{
			xDif *= -f;
			yDif *= f;
		}
		else
			xDif = yDif = 0;
```

```
            int xNew = p.x + xDif, yNew = p.y + yDif * 2;
            if (xNew > 0 && yNew > 0 && yNew < 1080 && xNew < 1920 &&
xDif&&!scroll&&(drag?cntrl_1.elapsed()>200:1))
                  SetCursorPos(xNew, yNew);
            p2 = p1;
      }

      if (l_Click&&cntrl_1.elapsed()>200&&!r_Click)
      {
            l_Click = false;
            d_Click = false;
            INPUT input = { 0 };
            input.type = INPUT_MOUSE;
            input.mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
            ::SendInput(1, &input, sizeof(INPUT));
            ::ZeroMemory(&input, sizeof(INPUT)); // why zeroMemory? removing this code changes
nothing that i can tell
            input.type = INPUT_MOUSE; // why reset this variable? is it not already set?
            input.mi.dwFlags = MOUSEEVENTF_LEFTUP;
            ::SendInput(1, &input, sizeof(INPUT));
      }
      else if (d_Click&&cntrl_1.elapsed() > 200&&!drag)
      {
            d_Click = false;
            for (int i = 0; i < 2; i++)
            {
                  l_Click = false;
                  INPUT input = { 0 };
                  input.type = INPUT_MOUSE;
                  input.mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
                  ::SendInput(1, &input, sizeof(INPUT));
                  ::ZeroMemory(&input, sizeof(INPUT)); // why zeroMemory? removing this code changes
nothing that i can tell
                  input.type = INPUT_MOUSE; // why reset this variable? is it not already set?
                  input.mi.dwFlags = MOUSEEVENTF_LEFTUP;
                  ::SendInput(1, &input, sizeof(INPUT));
            }
      }
      else if (r_Click)
      {
            r_Click = false;
            INPUT input = { 0 };
            input.type = INPUT_MOUSE;
            input.mi.dwFlags = MOUSEEVENTF_RIGHTDOWN;
            ::SendInput(1, &input, sizeof(INPUT));
```

```
            ::ZeroMemory(&input, sizeof(INPUT)); // why zeroMemory? removing this code changes
nothing that i can tell
            input.type = INPUT_MOUSE; // why reset this variable? is it not already set?
            input.mi.dwFlags = MOUSEEVENTF_RIGHTUP;
            ::SendInput(1, &input, sizeof(INPUT));
      }
}

int main(int argc, char* argv[])
{
      bool access = false, access2 = false;
      int camNo = 0;
      cvNamedWindow("Output", CV_WINDOW_AUTOSIZE);
      if (!access)
      {
            system("cls");
            int no_of_cam = -1;
            VideoCapture cap(0);
            int i;
            for (i = 0; cap.open(i)&&i<9; i++);
            cout << "        SELECT THE CAM\n\n";
            if (i > 1)
            {
                  cout << i << " cams are detected. You are required to choose the one inside the
touch pad. Look in the webcam feed window.";
                  cout << "\nYou can change the cam using the left and right arrow keys. When you
select the right cam press enter.";
                  cout << "(Before this please click on the cam window)";
                  while (!access)
                  {
                        int keyIn = waitKey(1);
                        static int tmp=-1;
                        if (keyIn == 2424832 && camNo > 0)
                              camNo--;
                        if (keyIn == 2555904 && camNo < 9 && camNo<i-1)
                              camNo++;
                        if (tmp!=camNo)
                        cap.open(camNo);
                        tmp = camNo;
                        Mat frame;
                        cap >> frame;
                        imshow("Output",frame);
                        if (keyIn == 13)
                              access = true;
                  }
```

```
		}
	}
	VideoCapture cap(camNo); // open the video camera no. 0
	w = cap.get(CV_CAP_PROP_FRAME_WIDTH); //get the width of frames of the video
	h = cap.get(CV_CAP_PROP_FRAME_HEIGHT); //get the height of frames of the video
	timer t1;
	t1.start();
	fBgk = new int[w*h];
	memset(fBgk, 0, sizeof(int)*w*h);
	cntrl_1.start();
	cntrl_2.start();
	mat = new int[w*h];
	matCmp = new int[w*h];
	matDet = new int[w*h];
	memset(mat, 0, sizeof(int)*w*h);
	memset(matCmp, 0, sizeof(int)*w*h);
	memset(matDet, 0, sizeof(int)*w*h);
	system("cls");
	cout << "Wait till the cam is calibrated";
	while (1)
	{
		memset(matDet, 0, sizeof(int)*w*h);
		Mat frame;
		double nPix = 0;
		cap >> frame;
		for (int i = 1; i < w; i++)
			for (int j = 1; j < h; j++)
			{
				Vec3b p = frame.at<Vec3b>(Point(i, j));
				mat[i+j*w]=p.val[0] = p.val[1] = p.val[2] = (p.val[0] + p.val[1] + p.val[2] )
/ 3;
				frame.at<Vec3b>(Point(i, j)) = p;
			}
		frameC = frame.clone();
		if (t1.elapsed()>3000)  //Uses to identify the outof view region
		{
			if (t1.elapsed() > 3500)
			{
				t1.stop();
				calib = true;
				memcpy(matCmp, mat, sizeof(int)*w*h);
				system("cls");
				cout << "R E A D Y -  C A L I B E R A T E D\n\n";
				cout << "CONTROLS:\n";
				cout << "Move Finger:  To Move Cursor\n";
```

```
                        cout << "Left Single Click: Tap for once\n";
                        cout << "Left Double Click: Double Tap\n";
                        cout << "Right click: Tap twice and hold the finger down\n";
                        cout << "Left click and drag: Tap thrice and hold finger down\n";
                        cout << "\n\n\n\nYou can change the threshold of finger detection using '+'
and '-' keys\n";
                        cout << "\n\n\n\nTo exit press 'ESC'";
                  }
                  for (int i = 1; i < w; i++)
                        for (int j = 1; j < h; j++)
                              if (mat[i+j*w] < lmt)
                                    fBgk[i + j*w] = 1;
            }

            //To color the detected hand region
            for (int j = 1; j < h&&calib; j++)
                  for (int i = 1; i < w; i++)
                  {
                        Vec3b p = frame.at<Vec3b>(Point(i, j));
                        int lmtT=matCmp[i+j*w]/255.0*lmt;
                        if (fBgk[i + j*w] != 1 && mat[i + j*w] < lmtT)
                        {
                              p.val[1] = 200;
                              matDet[i + j*w]=1;
                        }
                        else
                              matDet[i + j*w]=0;
                        frame.at<Vec3b>(Point(i, j)) = p;
                  }

            //To indentify the forward most set of pixels
            bool cont = true, det = false,detected=false;
            for (int j = 1; j < h&&cont&&calib; j++)
                  for (int i = 1; i < w&&cont; i++)
                  {
                        if (matDet[i+j*w] ==1)
                        {
                              detected = false;
                              nPix =0;
                              p1.x = p1.y = 0;
                              int s = 60;
                              float area = 0,blank=0;
                              for (int l = 0; l <= s; l++)  //To find the mid of the finger tip
                              {
                                    int a=-1, b=100;
```

```
                        for (int k = -s / 2; k <= s / 2; k++)
                        {
                              if (i + k > 0 && i + k < w&&j + l>0 && j + l < h)
                              {
                                    int lmtT = matCmp[i + k + (j + l)*w] / 255.0*lmt;
                                    if (mat[k + i + (l + j)*w] < lmtT && fBgk[i + k + (j
+ l)*w] != 1)
                                    {
                                          nPix++;
                                          p1.x += i + k;
                                          p1.y += j + l;
                                          if (a == -1)
                                                a = i + k;
                                          b = i + k;
                                    }
                                    else if (a != -1)
                                          blank++;
                              }
                        }
                        if (a!=-1)
                        area += (b-a)+1;
                  }

                  density = nPix / area;
                  if (nPix > 100)
                        cont = false;
                  if (nPix > 100 && density > 0.85&&density < 1)
                  {
                        p1.x /= nPix;
                        p1.y /= nPix;
                        int a = 0, b = 0;
                        int lmtT = matCmp[i + j*w] / 255.0*lmt;
                        bool flag = false;
                        for (int k = 1; k < w&&!flag; k++)
                              for (int l = p1.y - 10; l < p1.y + 50 && l < h&& !flag;
l++)
                              {
                                    if (l > 0 && matDet[k + l*w] == 1)
                                    {
                                          bool flag2 = true;
                                          int m;
                                          for (m = 0; m < 5; m++)
                                                if (m + l < h&&matDet[k + (m + l)*w] ==
0)
                                                      flag2 = false;
```

```
                                    if (m == 5 && flag2)
                                    {
                                          flag = true;
                                          a = k;
                                    }
                              }
                        }
                  flag = false;
                  for (int k = w - 2; k > 1 && !flag; k--)
                        for (int l = p1.y - 10; l < p1.y + 30 && l < h&& !flag;
l++)
                        {
                              if (l > 0 && matDet[k + l*w] == 1)
                              {
                                    bool flag2 = true;
                                    int m;
                                    for (m = 0; m < 5; m++)
                                          if (m + l < h&&matDet[k + (m + l)*w] ==
0)
                                                flag2 = false;
                                    if (m == 5 && flag2)
                                    {
                                          flag = true;
                                          b = k;
                                    }
                              }
                        }

                  if (b - a < 80)
                  {
                        if (!fDetected&&cntrl_1.elapsed() < 200)
                        {
                              allowedRC = true;
                        }
                        if (!fDetected&&cntrl_1.elapsed() > 50)
                        {
                              p2 = p1;
                              GetCursorPos(&cP1);
                              cntrl_1.reset();
                        }
                        fDetected = true;
                        cntrl_1.start();
                        circle(frame, p1.x, p1.y, 15, 255);
                        circle(frame, p1.x, p1.y, 16, 255);
                        circle(frame, p1.x, p1.y, 17, 255);
```

```
                                    detected = true;
                                }
                            }
                        }
                    }
            if (!detected&&fDetected)
            {
                fDetected = false;
                allowedC = true;
            }
            GetCursorPos(&cP2);
            if (!fDetected&&cntrl_1.elapsed() < 500 && mod(cP1.x - cP2.x) < 10 && mod(cP1.y - cP2.y)
< 10&&allowedC&&touch==0)
            {
                l_Click = true;
                allowedC = false;
                drag = false;
                scroll = false;
                cntrl_1.reset();
                touch++;
            }
            if (fDetected&&cntrl_1.elapsed() > 250 && mod(cP1.x - cP2.x) < 10 && mod(cP1.y - cP2.y) <
10 && allowedRC&&touch==1&&!drag)
            {
                l_Click = false;
                r_Click = true;
                allowedRC = false;
            }
            if (!fDetected&&cntrl_1.elapsed() < 500 && mod(cP1.x - cP2.x) < 10 && mod(cP1.y - cP2.y)
< 10 && allowedC&&touch == 1&&!drag)
            {
                l_Click = false;
                d_Click = true;
                allowedC = false;
                cntrl_1.reset();
                touch++;
            }
            if (fDetected&&cntrl_1.elapsed() > 100 && mod(cP1.x - cP2.x) < 10 && mod(cP1.y - cP2.y) <
10 && allowedRC&&touch >= 2)
            {
                allowedRC = false;
                l_Click = false;
                r_Click = false;
                d_Click = false;
                INPUT input = { 0 };
```

```
		input.type = INPUT_MOUSE;
		input.mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
		::SendInput(1, &input, sizeof(INPUT));
		drag = true;
	}
	if (drag)
	{
		d_Click = false;
		INPUT input = { 0 };
		input.type = INPUT_MOUSE;
		input.mi.dwFlags = MOUSEEVENTF_LEFTDOWN;
		::SendInput(1, &input, sizeof(INPUT));
	}
	if (cntrl_1.elapsed() > 500)
		touch = 0;
	if (calib)
	{
		rect(frame, 20, 20, 50, 50, 50, 200, 50);
		rect(frame, 21, 21, 50, 50, 50, 200, 50);
		rect(frame, 19, 19, 50, 50, 50, 200, 50);
		line(frame, 25, 25, 65, 25);
		line(frame, 25, 26, 65, 26);
		line(frame, 25, 25, 25, 65);
		line(frame, 26, 25, 26, 65);
		line(frame, 25, 65, 65, 65);
		line(frame, 25, 64, 65, 64);
	}
	else
	{
		rect(frame, 20, 20, 50, 50, 200);
		rect(frame, 21, 21, 50, 50, 200);
		rect(frame, 19, 19, 50, 50, 200);
	}
	imshow("Output", frame);
	compControl();
	int keyIn = waitKey(1);
	if (keyIn == 27)
	{
		break;
	}
	if (keyIn == 13)
	{
	}
	if (keyIn == '+')
		lmt++;
```

```
        else if (keyIn == '-'&&lmt>0)
              lmt--;
    }
    return 0;
}
```