**Dheeraj Agarwal**

**16UCS059**

**Assignment 9 – Blackboard Mosaic**

**CODE :**

//DHEERAJ AGARWAL

//16UCS059

//compilation command: g++ -std=c++11 -g mosac.cpp -o mosac.out -lopencv\_core -lopencv\_imgproc -lopencv\_highgui -lopencv\_imgcodecs -lopencv\_calib3d

//execution command: ./mosac.out

#include <stdio.h>

#include <iostream>

#include "opencv2/highgui/highgui.hpp"

#include "opencv2/imgproc/imgproc.hpp"

#include "opencv2/imgcodecs/imgcodecs.hpp"

#include "opencv2/core/core.hpp"

#include "opencv2/features2d/features2d.hpp"

#include "opencv2/calib3d/calib3d.hpp"

#include "opencv2/opencv.hpp"

using namespace cv;

using namespace std;

//for finding the offset required for each image and also the max and min offset value to decide the dimension of stitched image

void findDim(Mat img,int &gmaxrow,int &gmaxcol,int &gminrow,int &gmincol,Mat homography){

for(int a=0;a<5;a++){

double H[3][3];

//taking the values of homography in a matrix from a Mat object

for(int we=0,k=0;we<3;we++)

for(int j=0;j<3&&k<8;j++)

H[we][j]= homography.at<double>(k++);

H[2][2]=1;

int maxrow=INT\_MIN,maxcol=INT\_MIN,minrow=INT\_MAX,mincol=INT\_MAX;

//finding value of each pixel to get max and min offset value and select the global value

for(int we=0;we<img.rows;we++){

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

double k= H[2][0]\*we + H[2][1]\*j + H[2][2];

int inew= (H[0][0]\*we + H[0][1]\*j + H[0][2])/k;

int jnew= (H[1][0]\*we + H[1][1]\*j + H[1][2])/k;

gmaxrow= max(gmaxrow,inew);

gmaxcol= max(gmaxcol,jnew);

gminrow= min(gminrow,inew);

gmincol= min(gmincol,jnew);

}

}

}

}

//function to remove projective distortion and plotting the points in res accordingly

void removeProjection(Mat img,Mat homography,int maxrow,int maxcol,int minrow,int mincol,Mat &res){

double H[3][3];

//taking the values of homography in a matrix from a Mat object

for(int we=0,k=0;we<3;we++)

for(int j=0;j<3&&k<8;j++)

H[we][j]= homography.at<double>(k++);

H[2][2]=1;

//finding value of each pixel after removing distortion and plotting it in res

for(int we=0;we<img.rows;we++){

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

double k= H[2][0]\*we + H[2][1]\*j + H[2][2];

int inew= (H[0][0]\*we + H[0][1]\*j + H[0][2])/k;

int jnew= (H[1][0]\*we + H[1][1]\*j + H[1][2])/k;

//using reverse mapping

res.at<Vec3b>(inew-minrow,jnew-mincol) = img.at<Vec3b>(we,j);

}

}

}

int main( int argc, const char\*\* argv ){

//corresponding points in reference and given images

vector<Point2d> Ref0 = { Point2d(542.91,87.901), Point2d(547.41,375.99), Point2d(942.03,32.385), Point2d(1035.1,194.43), Point2d(1095.1,718.09), Point2d(868.51,725.59), Point2d( 496.4,680.58), Point2d( 500.9,1048.2), Point2d(545.91,1372.3), Point2d(733.47,701.59), Point2d(976.54,1196.7), Point2d(1044.1,1073.7), Point2d(1120.6,893.64), Point2d(802.49,33.885), Point2d(886.52,592.05), Point2d( 834,260.45), Point2d(520.41,1681.4), Point2d( 514.4,1953 ), Point2d(734.97,1948.5), Point2d(1154.3,1684.3), Point2d(854.25,1666.3), Point2d(1007.3,1702.3), Point2d(794.99,1445.8), Point2d(667.45,1109.7), Point2d(808.49,974.67)};

vector<Point2d> Img0 = { Point2d(899.79,1003.8), Point2d(929.25,1303.3), Point2d(1503.8,964.25), Point2d(1619.3,1153.3), Point2d(1635.8,1651.3), Point2d(1349.3,1637.8), Point2d(882.01,1577.8), Point2d(901.52,1856.9), Point2d(969.04, 2082), Point2d(1174.6,1615.4), Point2d(1454.3,2006.8), Point2d(1544.3,1931.8), Point2d(1655.3,1802.8), Point2d(1285.6,955.16), Point2d(1377.8,1528.3), Point2d(1325.3,1210.3), Point2d(953.25,2267.8), Point2d(956.25,2411.8), Point2d(1173.8,2429.8), Point2d(1616.3,2321.8), Point2d(1296.8,2284.3), Point2d(1455.8,2323.3), Point2d(1234.6,2149.5), Point2d(1098.1, 1917), Point2d(1266.1,1834.4)};

vector<Point2d> Ref1 = { Point2d(553.42,372.99), Point2d(701.96,284.46), Point2d(886.52,589.05), Point2d( 499.4,685.08), Point2d(632.94,718.09), Point2d(793.49,724.09), Point2d(972.04,725.59), Point2d(1104.1,973.17), Point2d(1047.1,1073.7), Point2d(862.51,958.16), Point2d(691.79,999.72), Point2d(592.43,971.67), Point2d( 499.4,1001.7), Point2d(527.91,1163.7), Point2d(622.44,1265.8), Point2d(773.98,1207.2), Point2d(885.02,1243.2), Point2d(1288.6,1685.9), Point2d(994.55,1720.4), Point2d(739.47,1681.4), Point2d(524.91,1826.9), Point2d(526.41,2095.5), Point2d(731.97, 2106), Point2d(1173.1,1861.4), Point2d(957.04,2056.5)} ;

vector<Point2d> Img1 = { Point2d(1347.8,112.25), Point2d(1649.3,125.75), Point2d(1604.3,734.75), Point2d(1014.1,503.53), Point2d(1155.1,671.58), Point2d(1353.2,820.12), Point2d(1583.3,980.75), Point2d(1536.8,1424.8), Point2d(1373.3,1489.2), Point2d(1242.1,1175.7), Point2d(1002.1,1064.7), Point2d(898.52,952.16), Point2d(763.48,904.15), Point2d(682.45,1120.2), Point2d(721.47,1313.8), Point2d(949.54,1369.3), Point2d(1029.1,1522.3), Point2d(1163.3,2336.8), Point2d(815.25,2113.3), Point2d(553.42,1841.9), Point2d(242.82,1804.4), Point2d(101.78,2056.5), Point2d(296.84,2242.5), Point2d(906.02,2398.6), Point2d(542.91,2383.6)};

vector<Point2d> Ref2 = { Point2d(529.41,2092.5), Point2d(518.91,1945.5), Point2d(524.91,1831.4), Point2d( 515.9,1685.9), Point2d(739.47, 2031), Point2d(737.97,1738.4), Point2d(866.25,1856.8), Point2d(996.75,1726.3), Point2d(905.25,2344.3), Point2d(1070.3,2101.3), Point2d(1137.8,1808.8), Point2d(550.42,1376.8), Point2d(616.44,1162.2), Point2d(524.91,899.65), Point2d(686.96,1067.7), Point2d(770.98,974.67), Point2d(872.25,1036.3), Point2d(1002.8,1075.3), Point2d(1089.8,1207.3), Point2d(1137.1,1618.4), Point2d(1210.6,1261.3), Point2d(1252.6,1163.7), Point2d(888.02,1387.3), Point2d(856.51,1661.9), Point2d(999.05,2209.5)};

vector<Point2d> Img2 = { Point2d(764.25,1478.8), Point2d(722.25,1273.3), Point2d(699.75,1103.8), Point2d(675.75,910.25), Point2d(1071.8,1312.3), Point2d(995.25,907.25), Point2d(1209.8,1024.2), Point2d(1374.2,806.62), Point2d(1418.3,1742.8), Point2d(1614.8,1313.8), Point2d(1622.3,874.25), Point2d(655.45,538.04), Point2d(707.96,273.96), Point2d(544.41,32.385), Point2d(785.99,144.92), Point2d(873.01,26.383), Point2d(1017.1,51.89), Point2d(1198.6,57.892), Point2d(1336.7,161.42), Point2d(1536.8,629.75), Point2d(1529.3,182.75), Point2d(1568.3,65.75), Point2d(1118.3,449.75), Point2d(1143.8,787.25), Point2d(1539.8,1496.8)};

vector<Point2d> Ref3 = { Point2d(1310.3,2359.3), Point2d(1274.3,2186.8), Point2d(1262.3,1904.8), Point2d(1293.1,1678.4), Point2d(1113.1,1678.4), Point2d(1104.8,2129.8), Point2d(995.25,2207.8), Point2d(983.25,2444.8), Point2d(867.75,1852.3), Point2d(996.75,1718.8), Point2d(888.75,2336.8), Point2d(737.97,1685.9), Point2d(736.47, 2103), Point2d(520.41,1832.9), Point2d(635.94, 2055), Point2d(1252.6,1166.7), Point2d(1087.6,1204.2), Point2d(1000.6,1078.2), Point2d(883.52,1244.7), Point2d(772.48,1384.3), Point2d(548.91,1376.8), Point2d(619.44,1154.7), Point2d(775.48,1127.7), Point2d(954.75,2021.8), Point2d(1092.8,1951.3)};

vector<Point2d> Img3 = { Point2d(1243.6,1750.4), Point2d(1183.6,1475.8), Point2d(1167.1,1046.7), Point2d(1212.8,739.25), Point2d(948.75,734.75), Point2d(909.75,1375.3), Point2d(737.25,1484.8), Point2d(680.25,1873.3), Point2d(557.92,950.66), Point2d(769.48,782.61), Point2d(540.75,1687.3), Point2d(391.37,721.09), Point2d(329.85,1307.8), Point2d(55.266,908.65), Point2d( 179.8,1231.2), Point2d(1168.6,95.404), Point2d(928.53,143.42), Point2d( 820.5,0.8751), Point2d(644.94,197.43), Point2d( 487.4,347.48), Point2d( 169.3,335.48), Point2d(295.34,90.902), Point2d( 511.4,54.891), Point2d(684.75,1204.3), Point2d(905.25,1112.8)};

vector<Point2d> Ref4 = { Point2d(547.41,80.399), Point2d(379.36,89.402), Point2d(611.93, 404.5), Point2d(692.96,113.41), Point2d( 831,126.91), Point2d(981.04,212.44), Point2d(1128.8,221.75), Point2d(1098.1,716.59), Point2d(897.02, 992.67), Point2d(878.25,725.75), Point2d(755.25,764.75), Point2d(774.75,1124.8), Point2d(618.75,823.25), Point2d(531.75,838.25), Point2d(614.25,1127.8), Point2d(516.75,1105.3), Point2d(1091.3,1198.3), Point2d(1104.8,971.75), Point2d(1253.3,1160.8), Point2d(945.03,26.383), Point2d(1096.6,32.385), Point2d(1364.3,196.25), Point2d(1382.3,745.25), Point2d(876.01,559.04), Point2d(766.48,329.47)};

vector<Point2d> Img4 = { Point2d(232.32,1127.7), Point2d(31.259,1198.2), Point2d(391.37,1498.3), Point2d(418.38,1121.7), Point2d(601.43,1096.2), Point2d(806.99,1148.7), Point2d(992.25,1112.8), Point2d(1111.6,1738.4), Point2d(948.03, 2172), Point2d(836.25,1823.8), Point2d(683.25,1915.3), Point2d(818.25,2396.8), Point2d(522.75,2033.8), Point2d(402.75,2086.3), Point2d(602.25,2453.8), Point2d(470.25,2462.8), Point2d(1272.1,2407.6), Point2d(1216.6,2080.5), Point2d(1492.7,2284.6), Point2d(713.96,950.66), Point2d(892.52,902.65), Point2d(1277.3,1007.8), Point2d(1497.8,1685.8), Point2d(779.98,1618.4), Point2d(575.92,1348.3)};

//vector for storing the homography matrix for each image relative to corresponding image

vector<Mat> homography;

homography.push\_back(findHomography(Img0,Ref0));

homography.push\_back(findHomography(Img1,Ref1));

homography.push\_back(findHomography(Img2,Ref2));

homography.push\_back(findHomography(Img3,Ref3));

homography.push\_back(findHomography(Img4,Ref4));

//reading the reference image

Mat ref = imread("ref.jpg",CV\_LOAD\_IMAGE\_UNCHANGED);

assert(&ref);

//for storing all the given images

Mat img[5];

//will be used to find the dimension of result image

int gmaxrow=INT\_MIN,gmaxcol=INT\_MIN,gminrow=INT\_MAX,gmincol=INT\_MAX;

//reading all given images

for(int we=0;we<5;we++){

string name="m"+to\_string(we)+".jpg";

img[we]= imread(name,CV\_LOAD\_IMAGE\_UNCHANGED);

assert(&img[we]);

}

//creating a window named Image

namedWindow("Image", CV\_WINDOW\_NORMAL);

resizeWindow("Image",1200,1200);

//calling findDim function for each image

for(int a=0;a<5;a++)

findDim(img[a],gmaxrow,gmaxcol,gminrow,gmincol,homography[a]);

//wether global max values are less than reference image row and cols if not then update it

gmaxrow = max(gmaxrow,ref.rows);

gmaxcol = max(gmaxcol,ref.cols);

//object res is created using global max and min values

Mat res(gmaxrow-gminrow+1, gmaxcol-gmincol+1, CV\_8UC3, Scalar(0,0,0));

//loop for all the given images

for(int a=0;a<5;a++){

string name="mnew"+to\_string(a)+".jpg";

//calling removeProjection function

removeProjection(img[a],homography[a],gmaxrow,gmaxcol,gminrow,gmincol,res);

imwrite(name,res);

imshow("Image", res);

waitKey(0);

}

//plotting the points of reference image as it was for better image

for(int we=0;we<ref.rows;we++)

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

res.at<Vec3b>(we-gminrow,j-gmincol) = ref.at<Vec3b>(we,j);

imshow("Image",res);

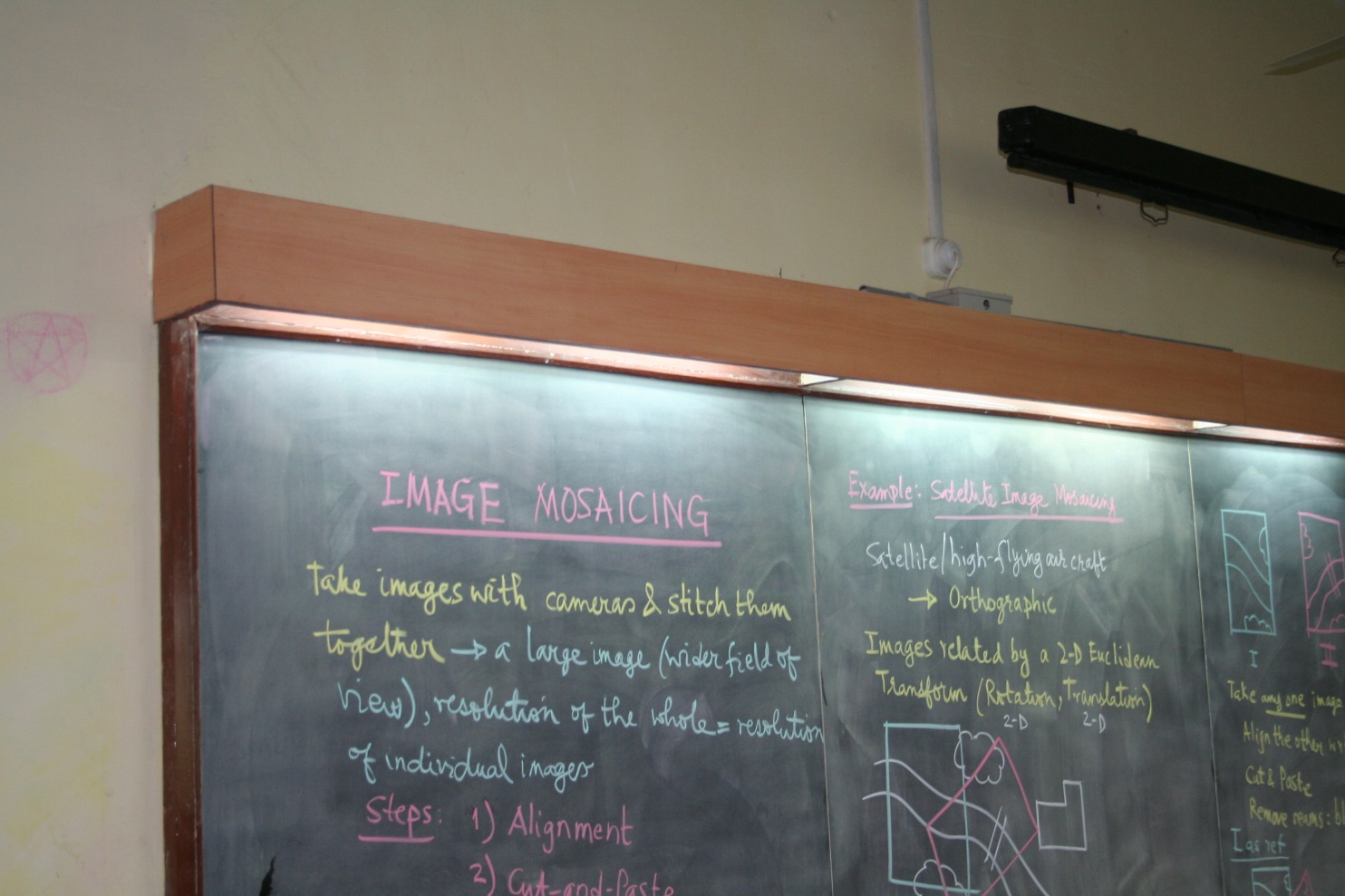
waitKey(0);

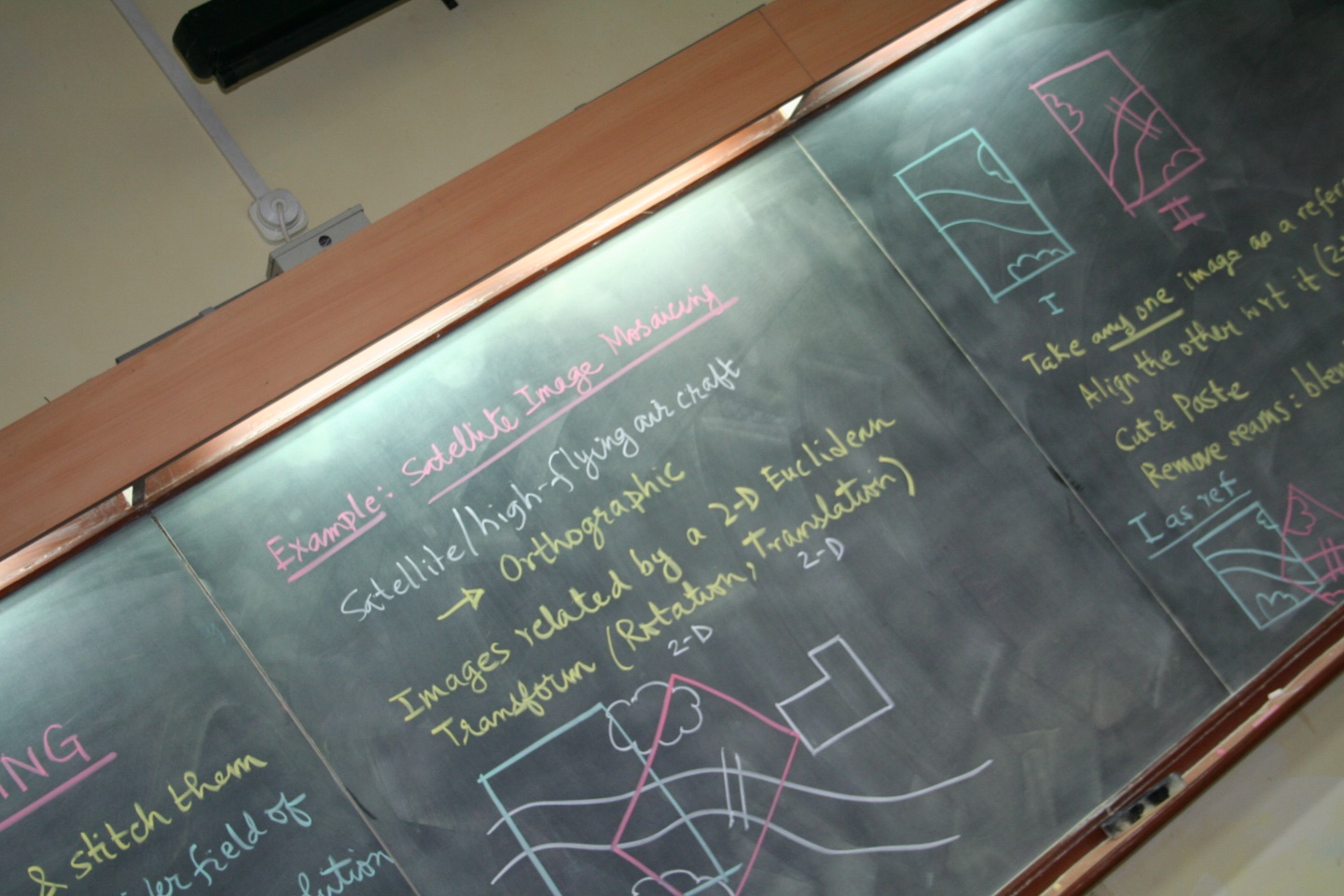
imwrite("result.jpg",res);

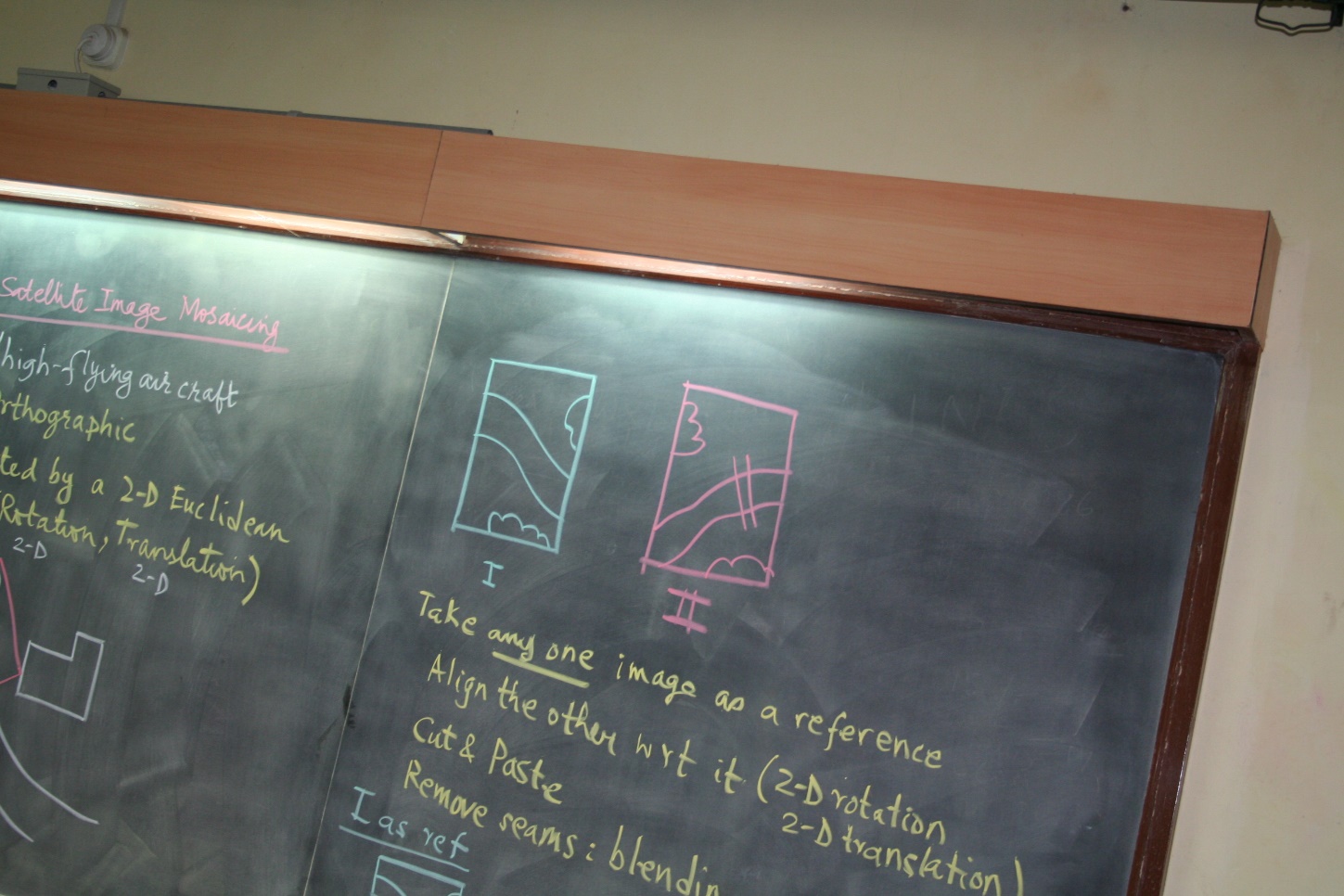
return 0;

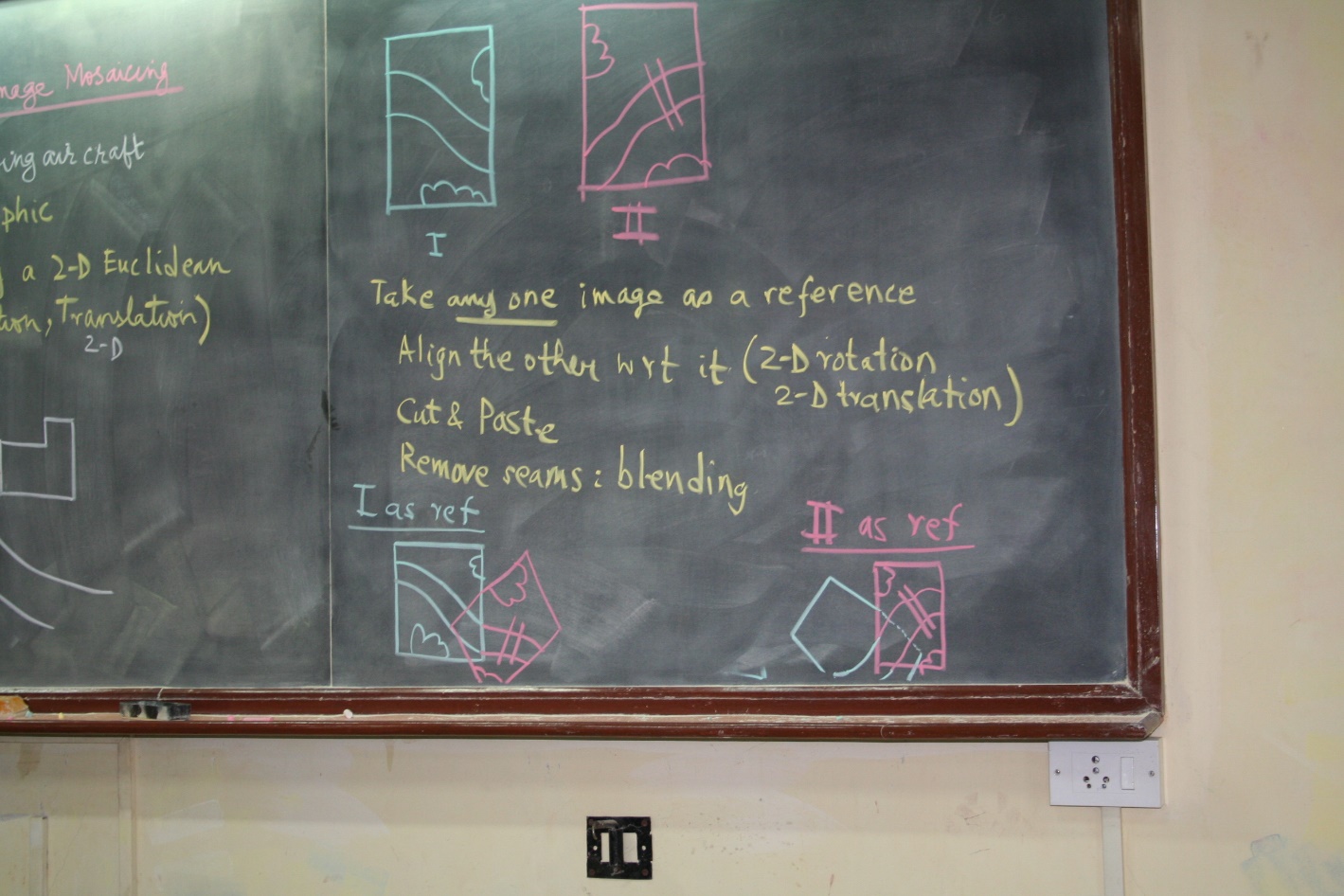
}

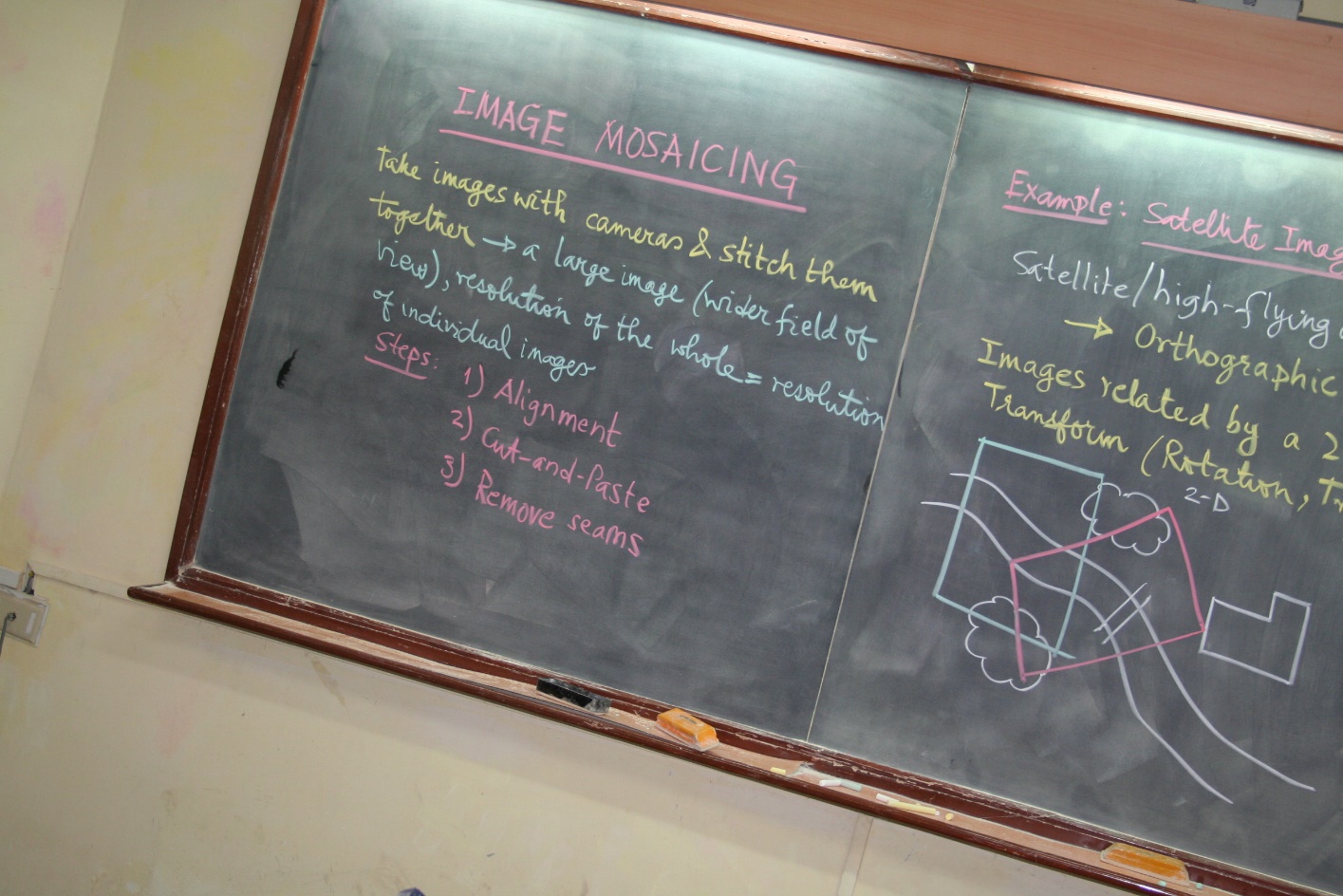
**Input images**



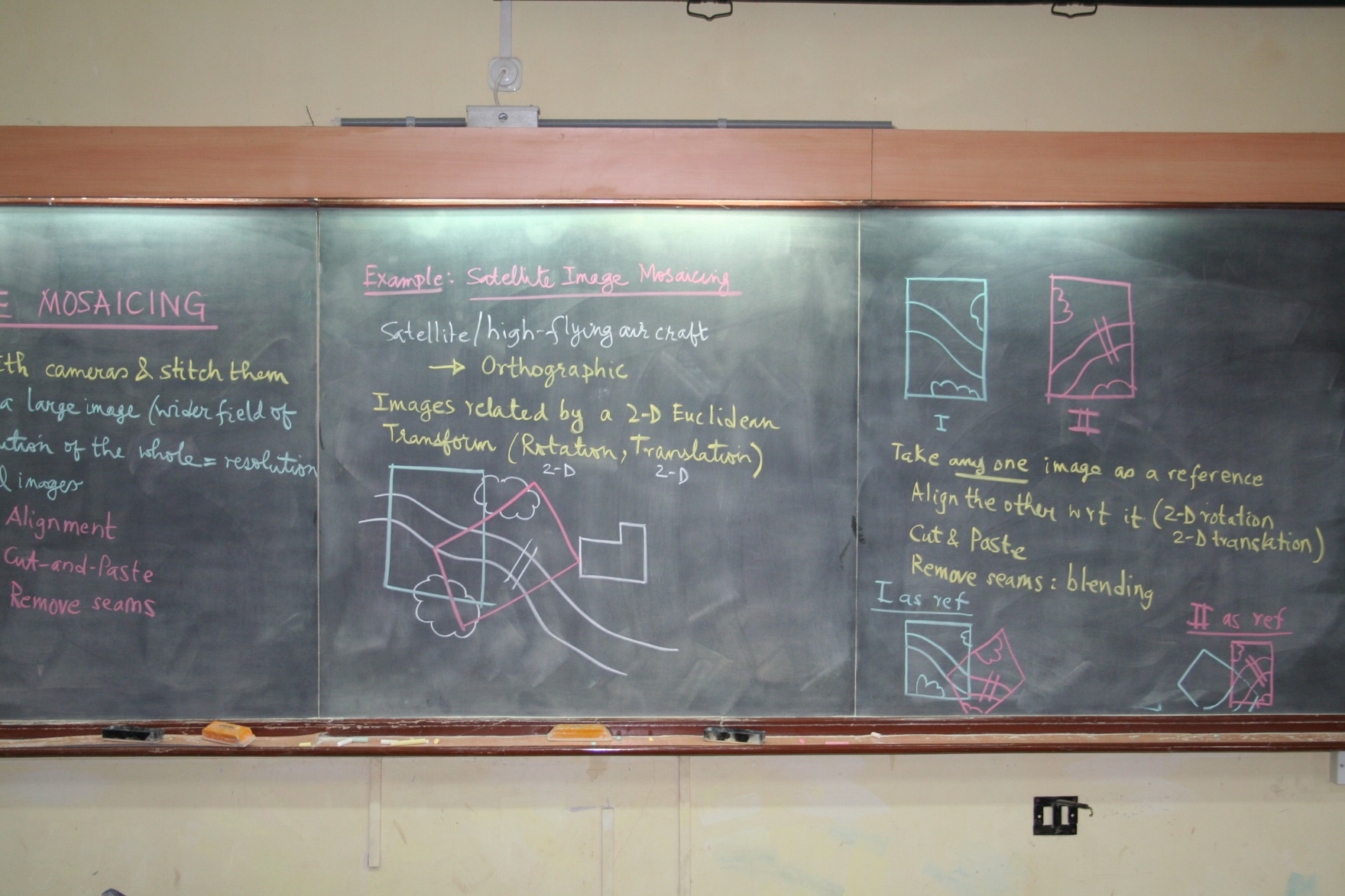






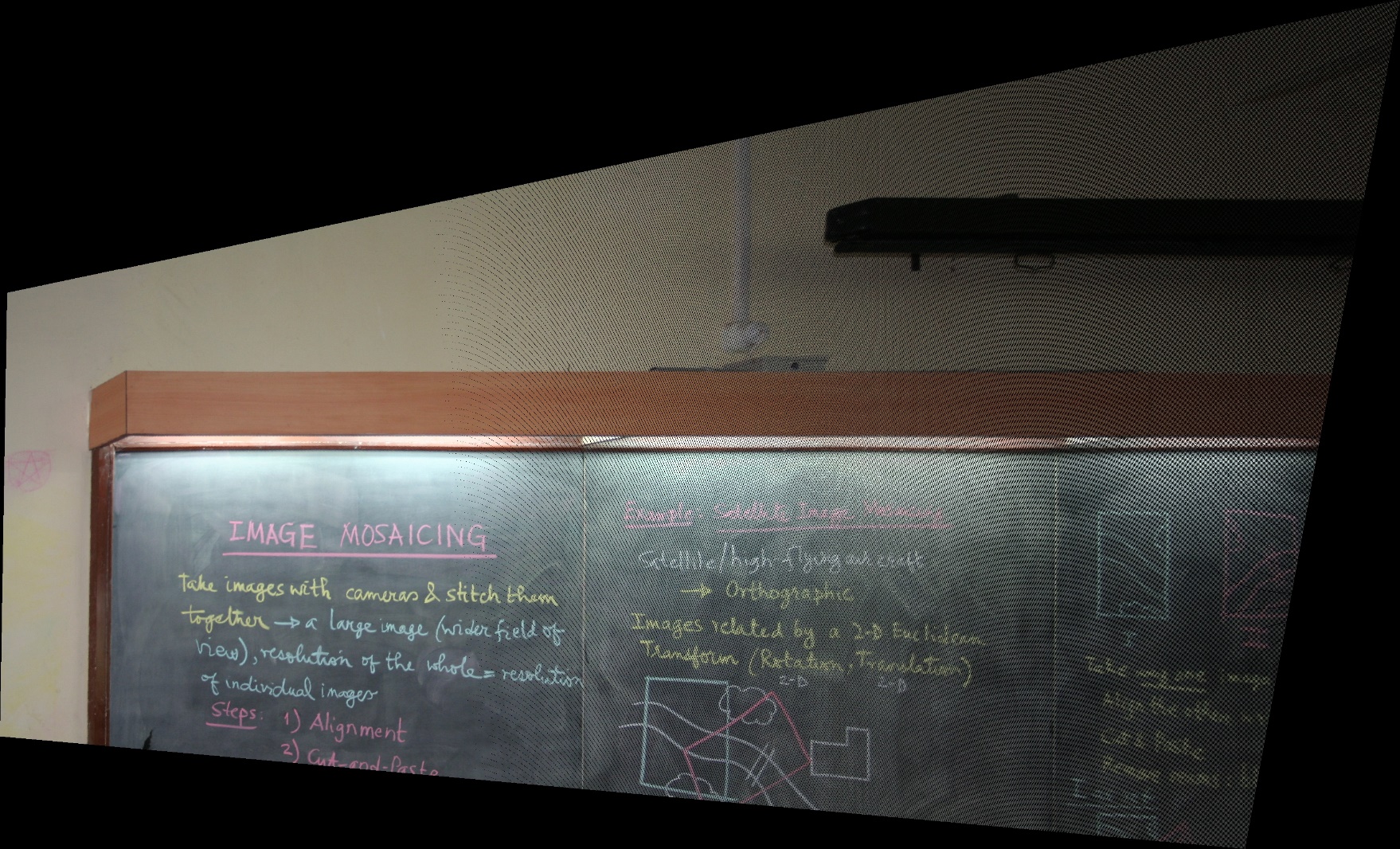


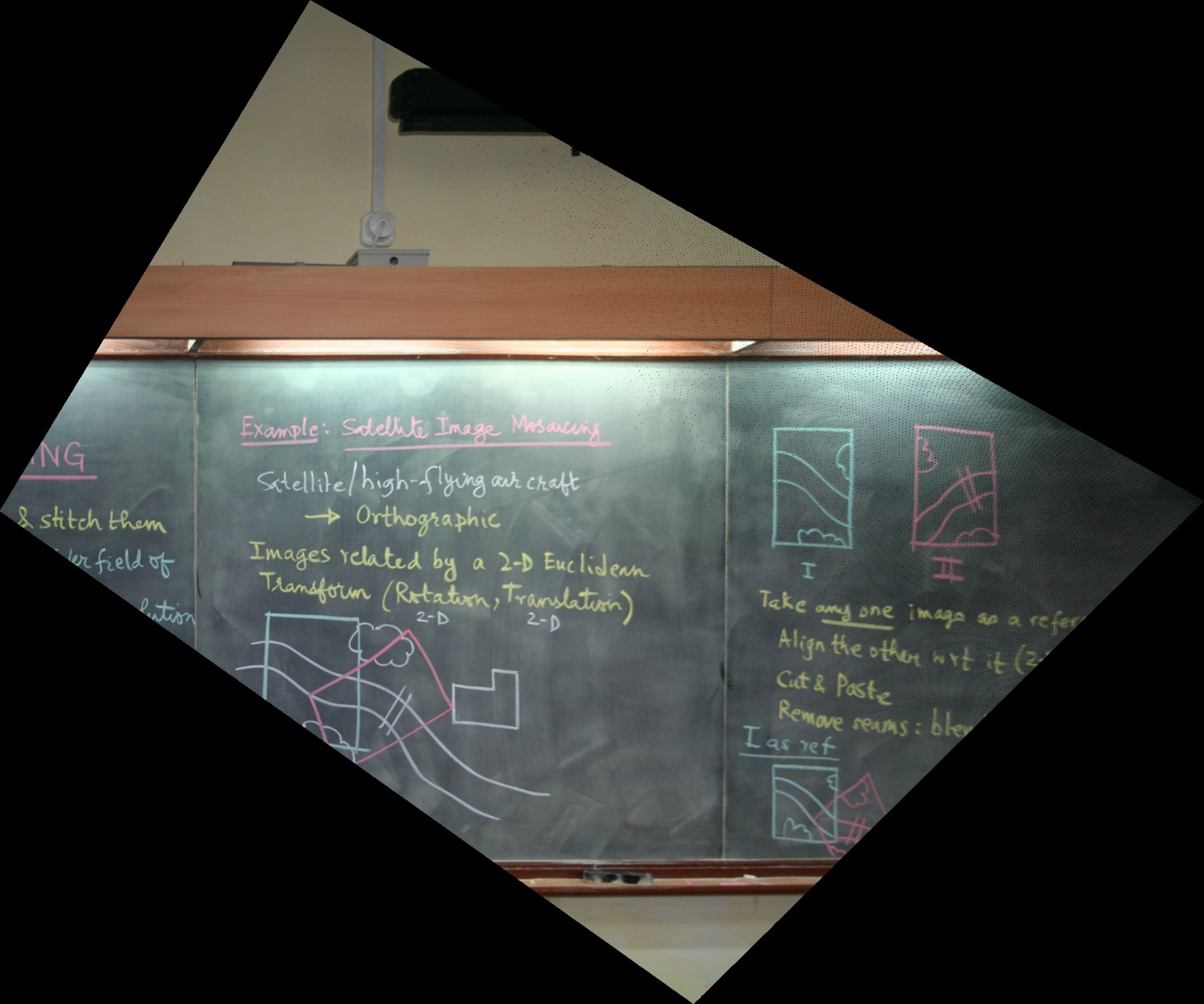
**Reference Image**

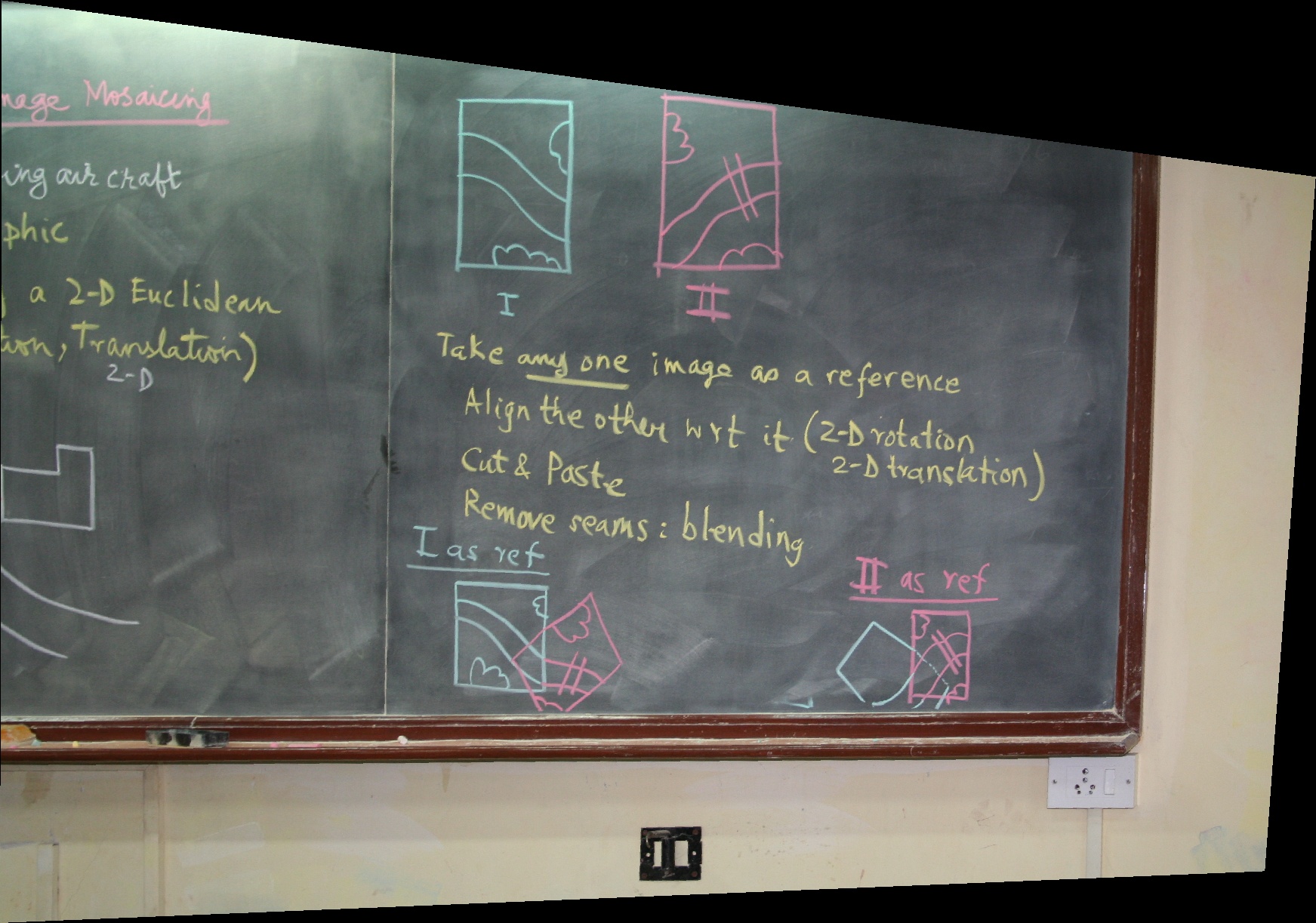


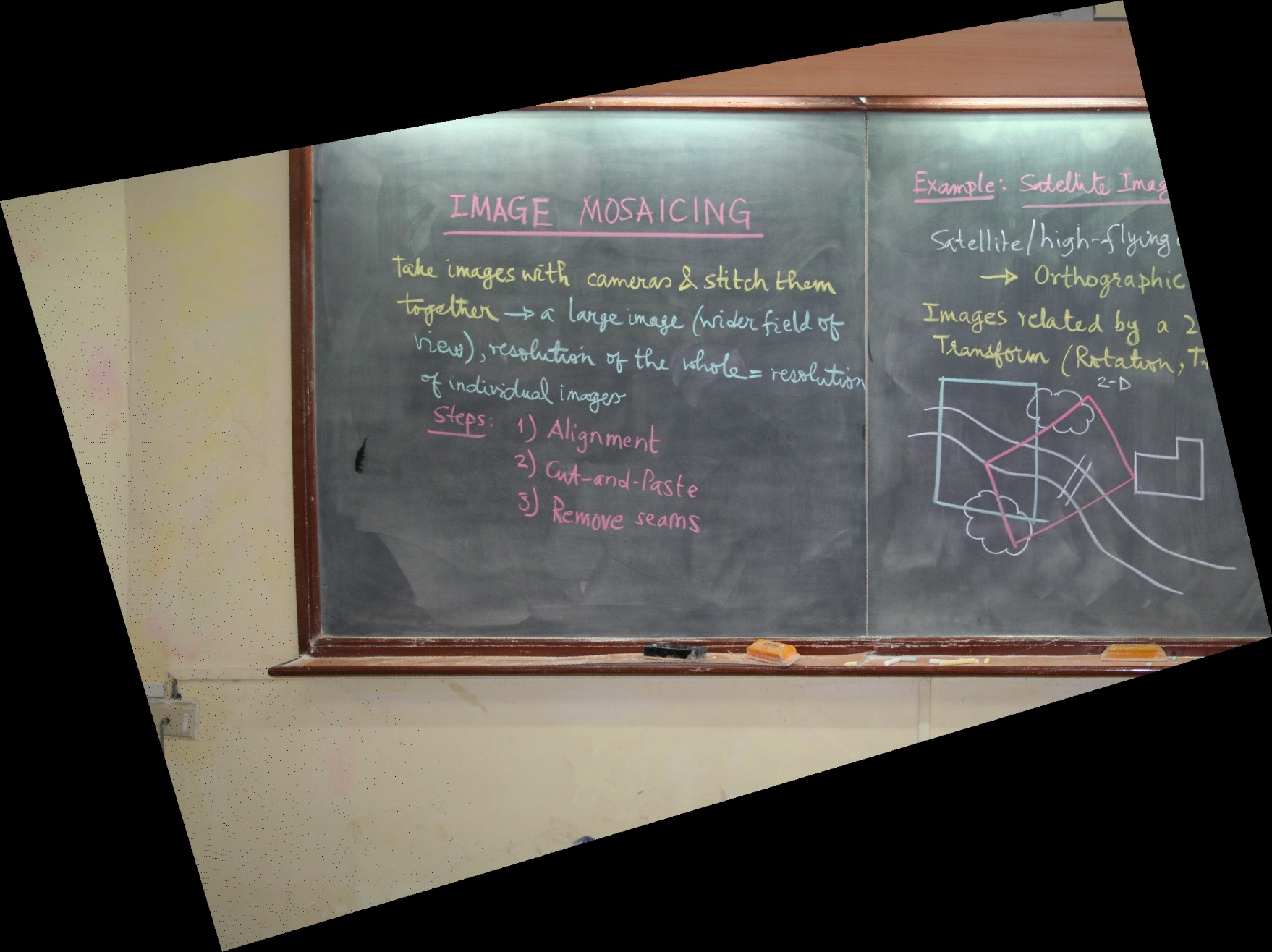
**Images after removing**

**Projective Distortion**

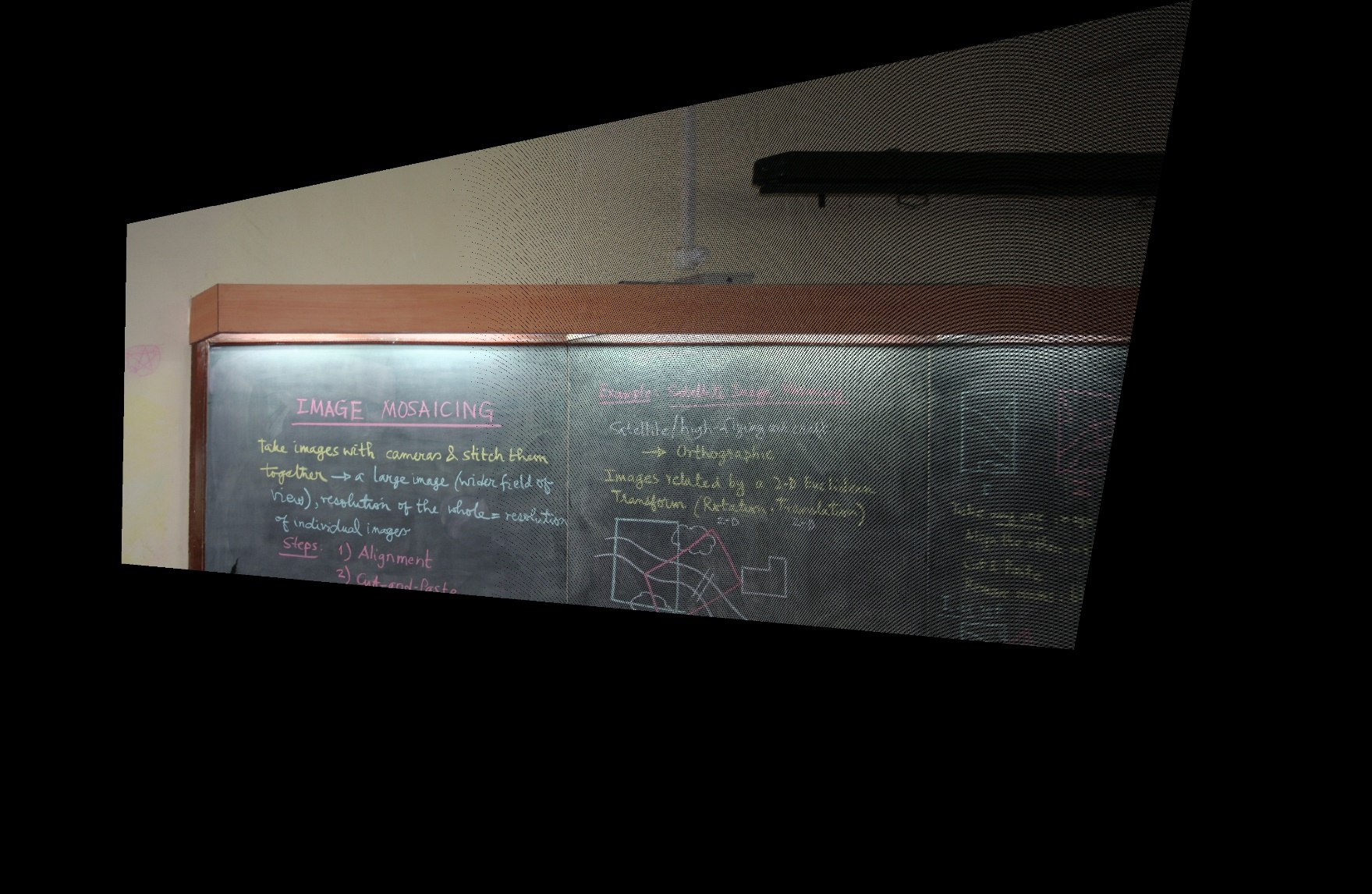


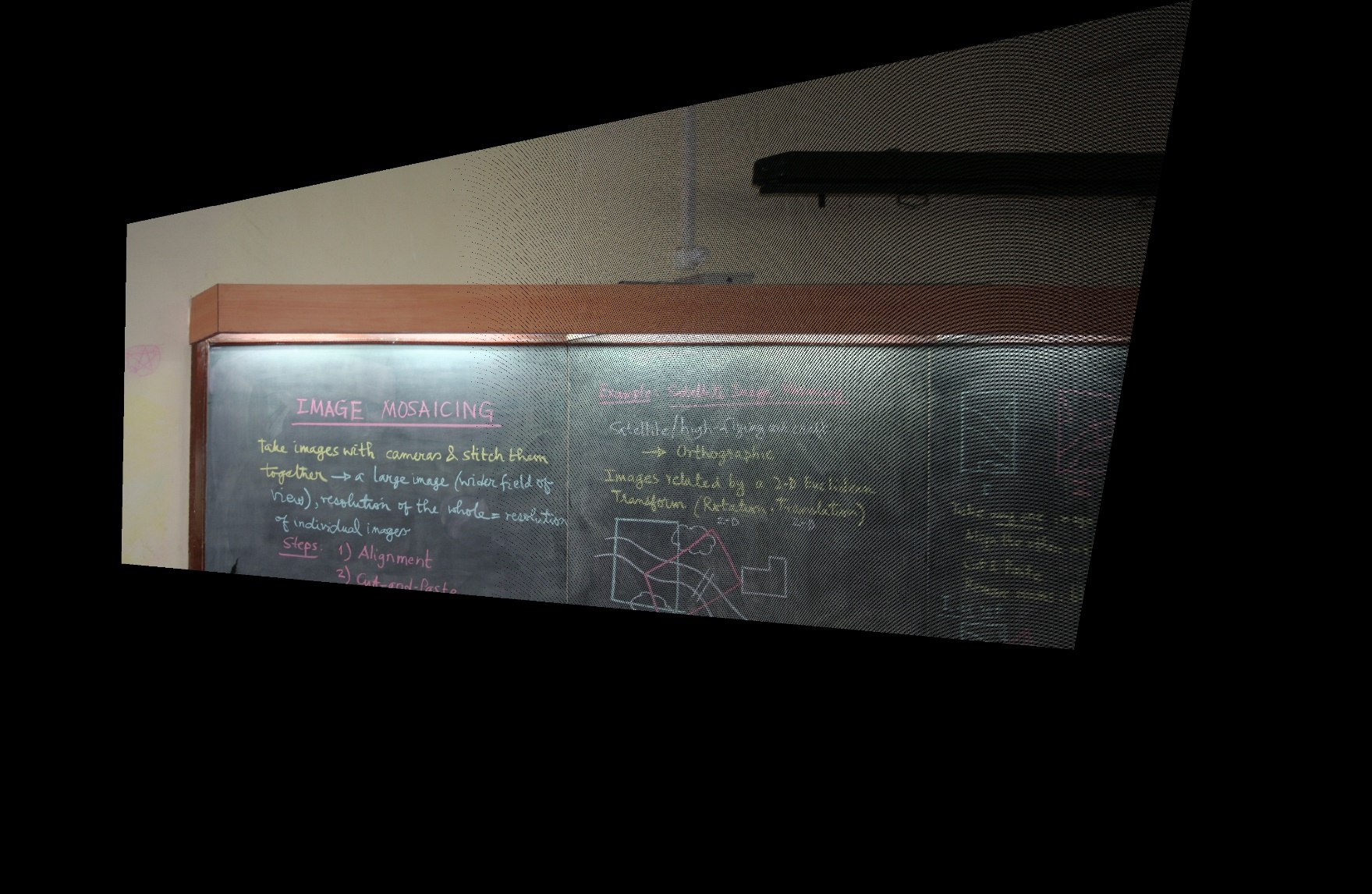


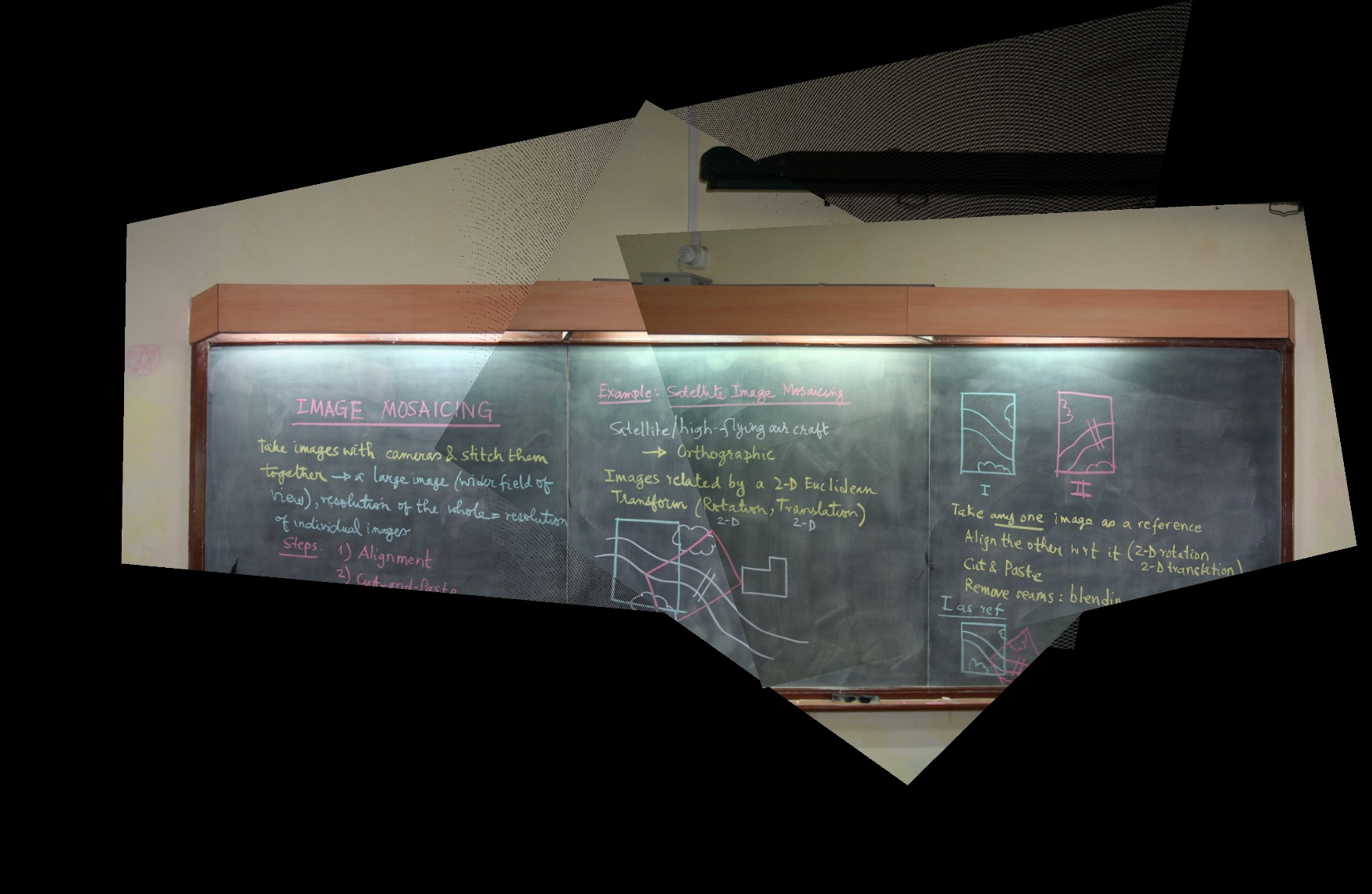




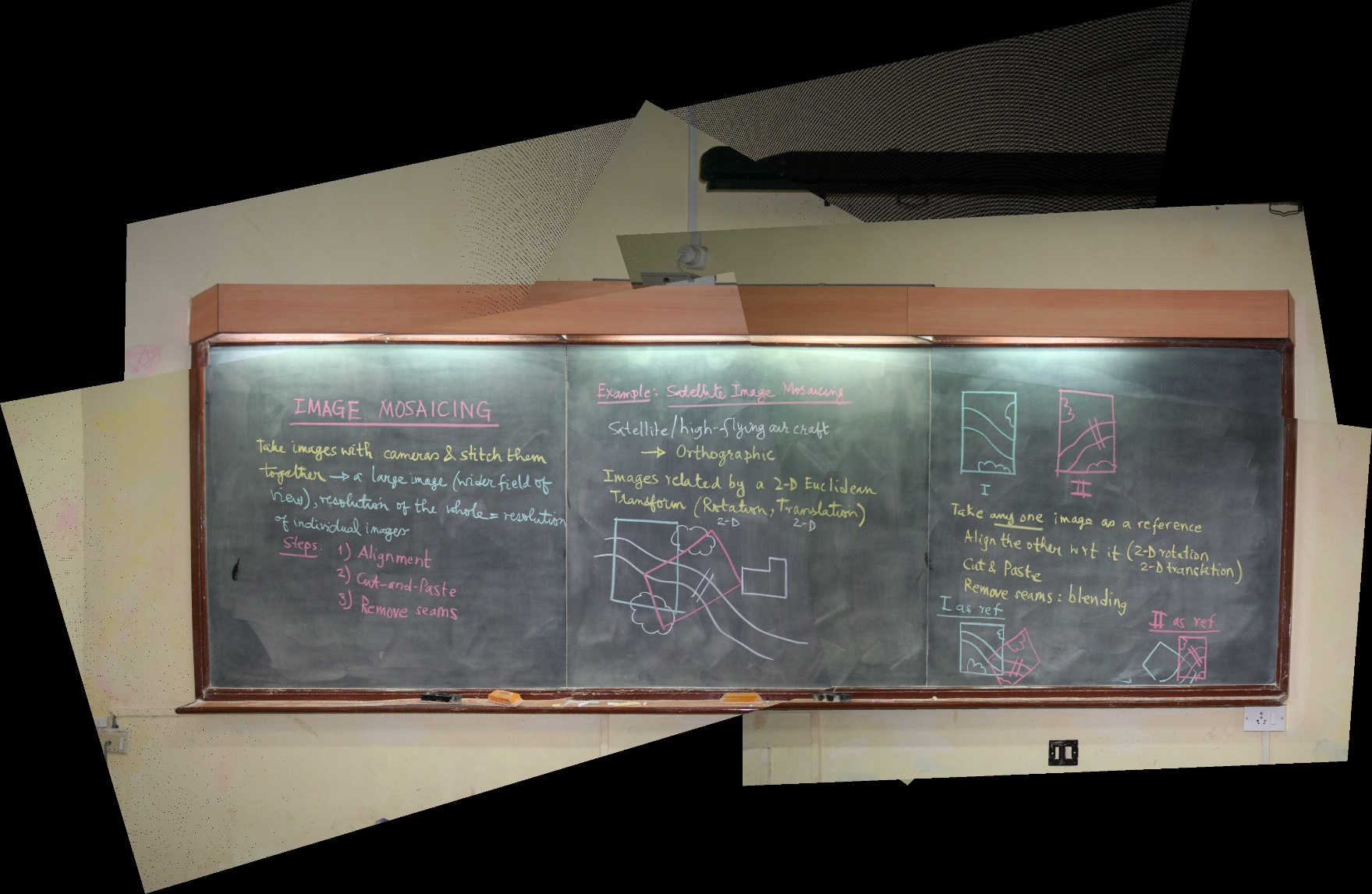
**Output Images**













**Observations:**

* **To remove projective distortion, we first need a reference image and corresponding points of other images to that image.**
* **Then we calculated the homography and find undistorted image using it.**
* **We also needed to find the dimension of the image that is to be produced after stitching these undistorted images to each other. For that we find the global max and min value rows and columns of each image after removing distortion and make a black image with these dimension for result.**
* **Then We mapped the images accordingly using reverse mapping and no point will be out as we have already accounted for the offset. As we plot the image points from which we have removed distortion according to the reference image, the points overlap each other accordingly and a clear image with all the parts of the blackboard is obtained with almost perfect alignment.**
* **We have used 25 reference points in place of 4 to get more accurate homography matrix.**
* **We used Mat objects for storing images and vector<Point2d> to store the points.**