REMOVING PROJECTIVE AND AFFINE DISTORTIONS

Fan Zhang
Ben Overzat

COMPUTER VISION

Contents

1. ABSTRACT

This report will discuss the removal of affine and projective distortions from images. Projective distortion is the distortion of parallel lines in the real world when projected onto an image plane, which causes them to appear to intersect at a vanishing point. The removal of this distortion results in real-world parallel lines appearing parallel in an image. Affine distortion is skewing, or the distortion of angles between sets of parallel lines in an image. When affine distortion is rectified, orthogonal lines on a real-world plane appear orthogonal in an image. There are direct and indirect methods of estimating the projective and affine distortion of an image. Once the distortion parameters are estimated, the distortion can be rectified by transforming the image in a way inverse to the distortion. Removing project and affine distortion from an image leaves the real world and the image geometrically similar to each other, meaning that ratios and angles agree between the image and real-world planes. Then, knowledge of the size of a known object in the image can help to measure the lengths of other objects in the image. This report will demonstrate the removal of projective and affine distortion for this purpose using both direct and indirect methods on a set of images.

2. EXPERIMENT AND ANALYSIS

2.1 Direct Approach

In a pinhole projection model, the transformation between a real-world plane and an image plane can be represented by a matrix transformation of coordinate vectors. Homogeneous image coordinates (x_1,x_2,x_3) map to homogeneous coordinates on a real-world plane $(X,Y,1)$ by a matrix transformation as follows:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}$$

The matrix elements h encode the distortion parameters for the image projection of the plane.

Because the coordinates are homogeneous, the matrix elements need only to be proportional to each other. Therefore the value of one of the elements can be fixed, and the rest can be solved for. This gives eight degrees of freedom, and can be solved for using eight equations. These come from the selection of four points on an image and their coordinates on the real-world plane. Each point provides two equations, setting up a system as follows:

$$\underbrace{\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \end{bmatrix}_{2\text{N}\times 9}}_{\text{A}} \underbrace{\begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ \vdots \\ . \\ h_{31} \\ h_{32} \\ h_{33} \end{bmatrix}}_{\mathbf{h}} = \mathbf{0}$$

$$\text{A} \quad \mathbf{h} = \mathbf{0}$$

This system can be solved through the singular value decomposition of the matrix A. Once the h values are obtained, the inverse of the matrix can be applied to the set of image points to remove the distortions in the image, including projective and affine distortion.

2.2 Indirect Approach

When real-world coordinates are not known, indirect methods can be used to remove projective and affine distortions through identification of parallel and orthogonal lines.

2.2.1 Remove Projective Distortion

Since parallel lines are still parallel under affine transformation, we can use the transformation matrix H1 to map the vanishing line to the line at infinity l^{∞}, thus recovering the affine properties of the image.

Given two pairs of image lines m1, m2, m3 and m4, where m1 | | m2 and m3 | | m4. In a projective distorted image, we can find the intersection of m1 and m2, the vanishing point p1, and the intersection m3 and m4, the vanishing point p2. The line formed by the vanishing point p1 and p2 is our vanishing line $l = (l1, l2, l3)^T$. So, in our homogeneous coordinate system, we have:

$$p1 = m1 \times m2,$$
$$p2 = m3 \times m4,$$
$$l = p1 \times p2.$$

Therefore, by applying the transformation H1, where

$$H1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l1 & l2 & l3 \end{bmatrix}$$

We know that the vanishing line will be mapped to the line at infinity $l^{\infty} = (0, 0, 1)^T$.

Therefore, $(0, 0, 1)^T = H_1^{-T} l$.

$$H_1^{-T} = \begin{bmatrix} 1 & 0 & -l1/l3 \\ 0 & 1 & -l2/l3 \\ 0 & 0 & 1/l3 \end{bmatrix}$$

Anyway, all we need to do now is to find the vanishing point, then find the vanishing line, figure out our matrix H1, and use the matrix H1 to recalculate the Xc and obtain the image after removing projective distortion.

2.2.2 Remove Affine Distortion

Affine distortion can be rectified by the identification of two sets of orthogonal lines in an image. In general, an affine transformation of image coordinates can be defined by a transformation matrix H_{affine}:

$$H_{affine} = \begin{bmatrix} K & 0 \\ 0^T & 1 \end{bmatrix}; \quad K = \begin{bmatrix} a & b \\ 0 & 1/a \end{bmatrix}$$

The parameters a and b can be solved for by defining a matrix S and vector s as follows:

$$S = KK^T = \begin{bmatrix} a^2 + b^2 & b/a \\ b/a & 1/a^2 \end{bmatrix} \rightarrow s = (a^2 + b^2, b/a, 1/a^2)$$

The vector s can be found as the cross product of two vectors $\boldsymbol{u}$ defined from two sets of orthogonal lines. For a pair of orthogonal lines $\boldsymbol{m}$ and $\boldsymbol{l}$, the corresponding vector $\boldsymbol{u}$ is defined as follows:

$$\boldsymbol{u} = (l_1 m_1, l_1 m_2 + l_2 m_1, l_2 m_2)$$

Once s is found, K can be found through the Cholesky decomposition of S. The inverse of the affine transformation can be applied to the set of image coordinates to rectify the distortion.

2.3 Results Of Direct Approach

The expected outcome of this direct approach is that if we remove the projective and affine distortion, the rectangular surface of multiple objects on the same plane, its edges will revert to parallelism, and its angles will revert to right angles. Projective transformations include rotation, translation, scaling and shear. The invariants of projective transformation are coincidence relation and cross ratio of length. So after removing the projective distortion, the objects in the picture will first recover their parallelism. Affine transformations include rotation, translation, scaling and shear. The invariants of affine transformation are: parallel lines, the ratio of the length of parallel lines, the ratio of the area. So after removing the affine distortion, the objects in the picture will recover their verticality of the angles.
Our original image is shown in the figure 2.1.

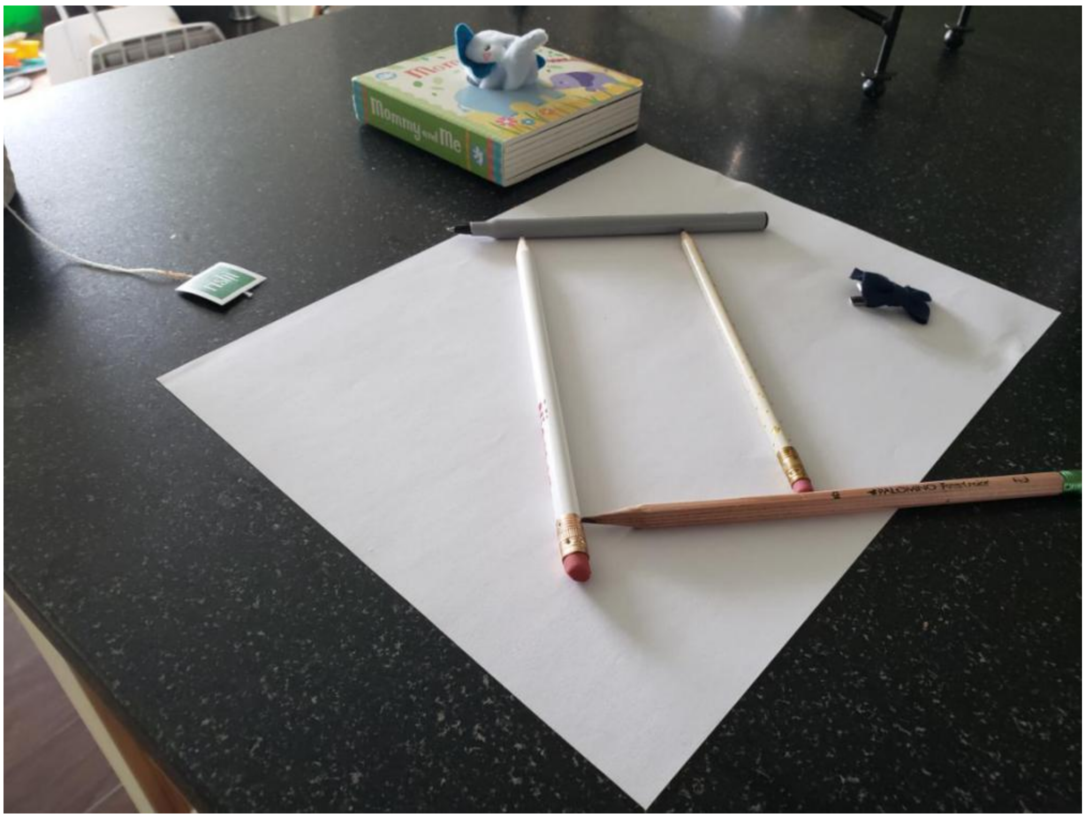

Fig 2.1 The original image

Then, we already know that the paper in the image is 8.5 inches x11 inches. So we can figure out where this piece of paper is in the real world coordinate system, and we're assuming that the point in bottom middle off the image is (0,0). Then we calculate the coordinates of the four vertices on this piece of paper. The red coordinates are the world coordinates that is scaled to an image coordinate system, and the blue coordinates are the coordinates in the image, as is shown in the figure 2.2.

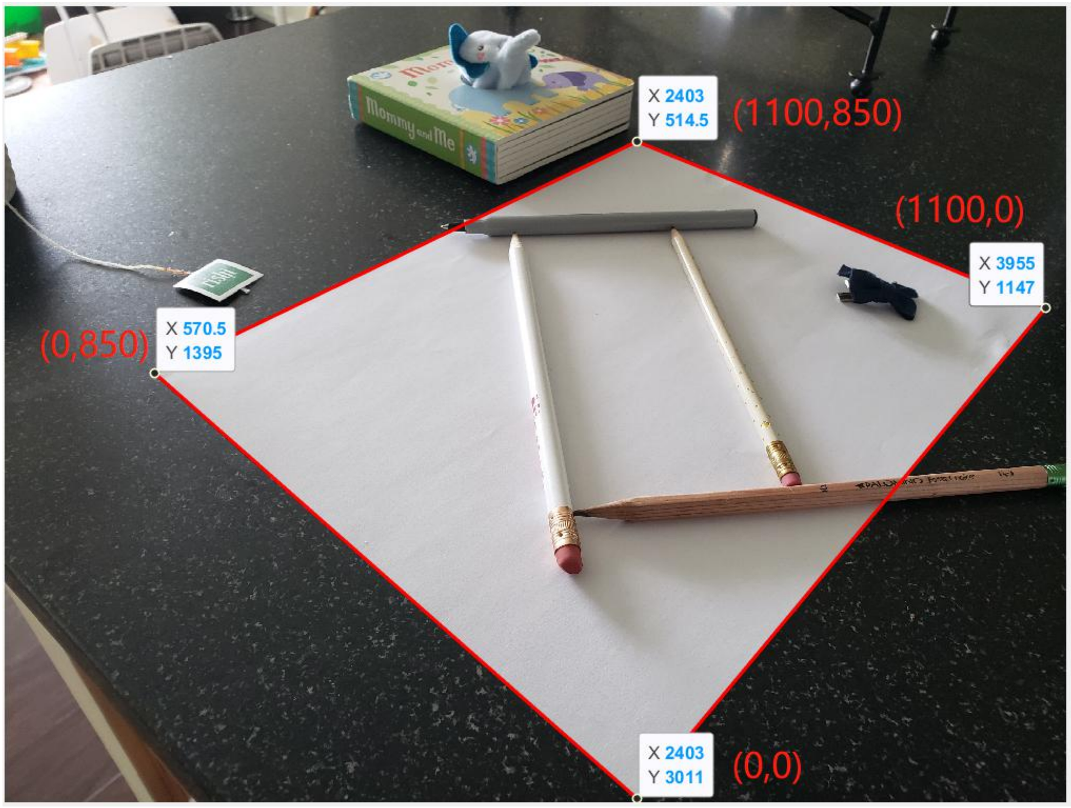

Fig 2.2 The vertices of the paper

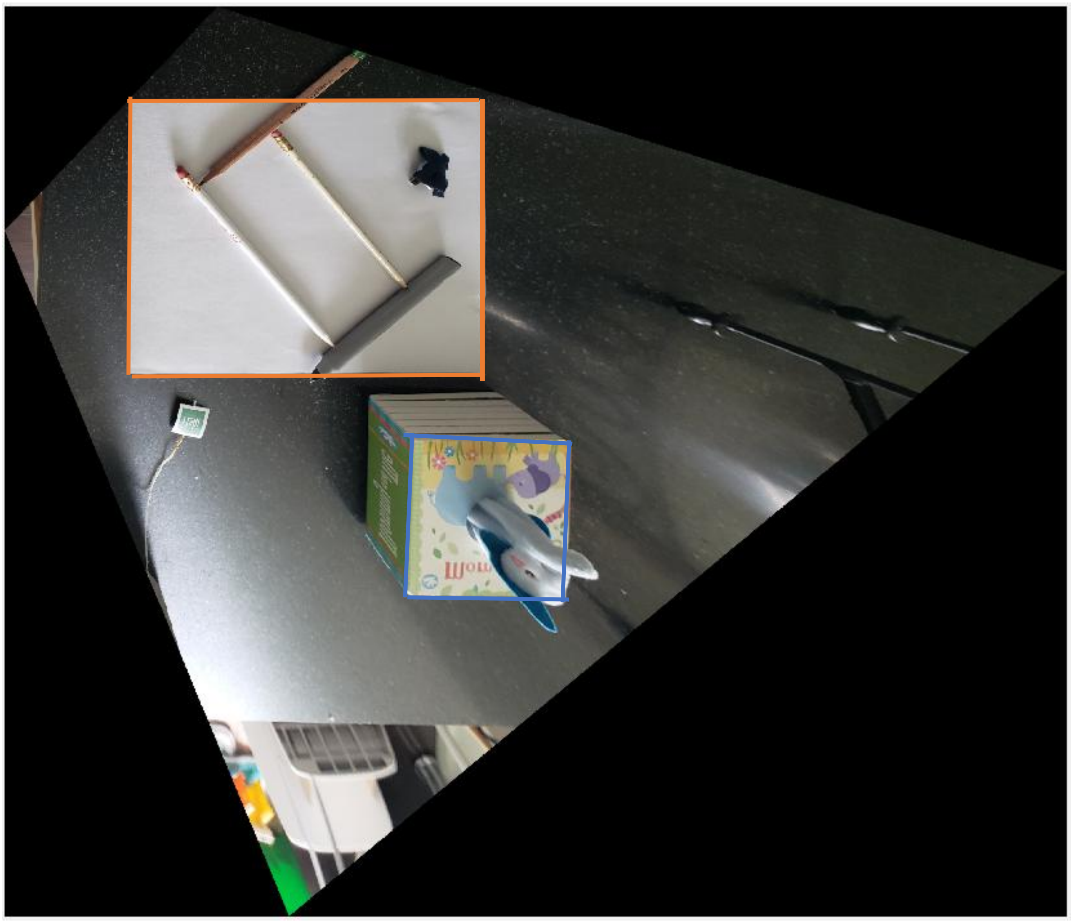

Fig 2.3 The image after removing the distortions

Finally, we get the image after removing the distortions, as is shown in the figure 2.3. We can clearly observe that the orange lines form a rectangle, which means that we have successfully removed the distortions in this plane. We also can observe other surfaces that are on the same plane with the paper, such as the top of the tissue box, are also rectified. Other planes, however, such as the sides of the tissue box, remain distorted.

For the figure 2.2, we also can figure out the coordinates of the black object relative to the paper, the lengths of the pens and the positions of the pens on the paper, because we know the true length of the paper, we can calculate it by scale, the actual length of the object / the length of the object on the image = the actual length of paper / the length of paper on the image, as is shown in the figure 2.3a.

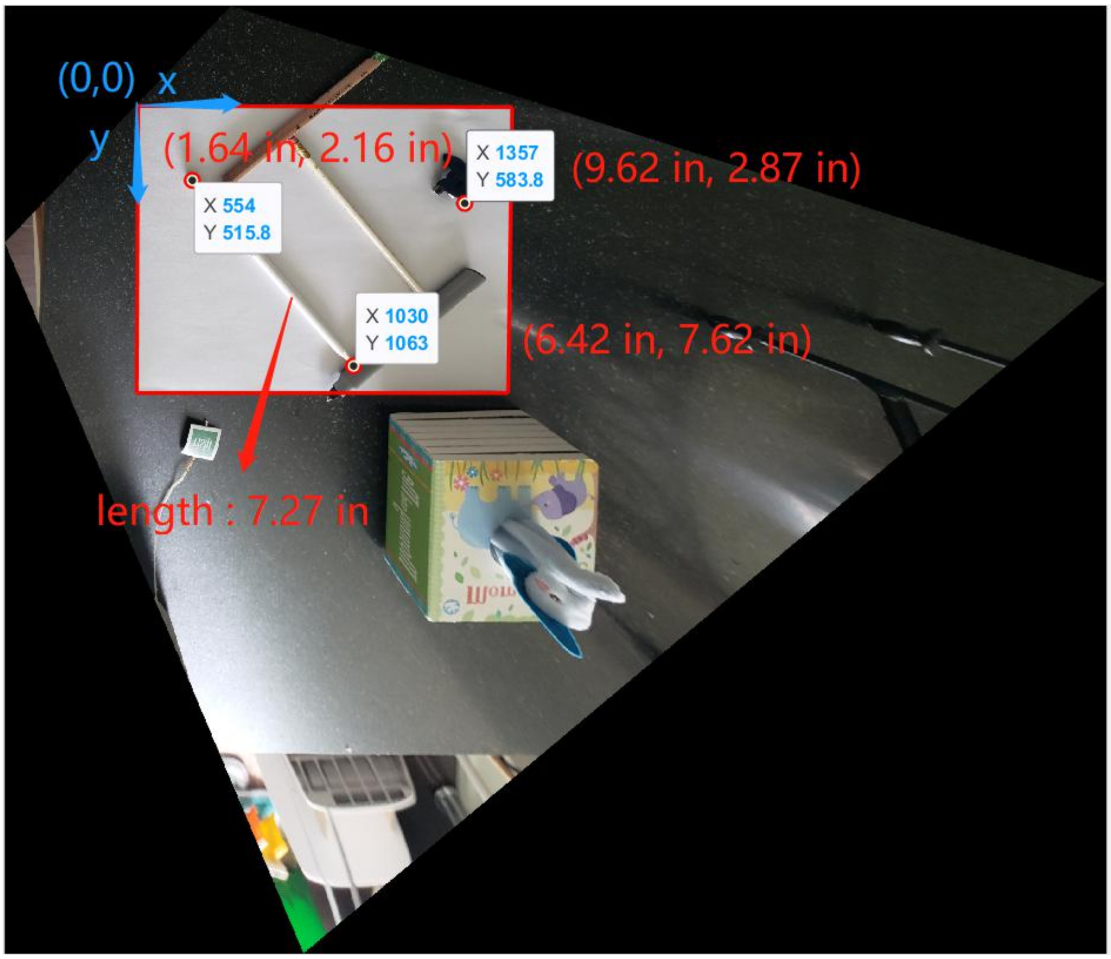

Fig 2.3a Size and position of objects

The blue coordinates are the coordinates in the image, and the red coordinates are the actual coordinates (take the top left corner of the paper as the origin point), in centimeters, and calculate the length of the pencil.

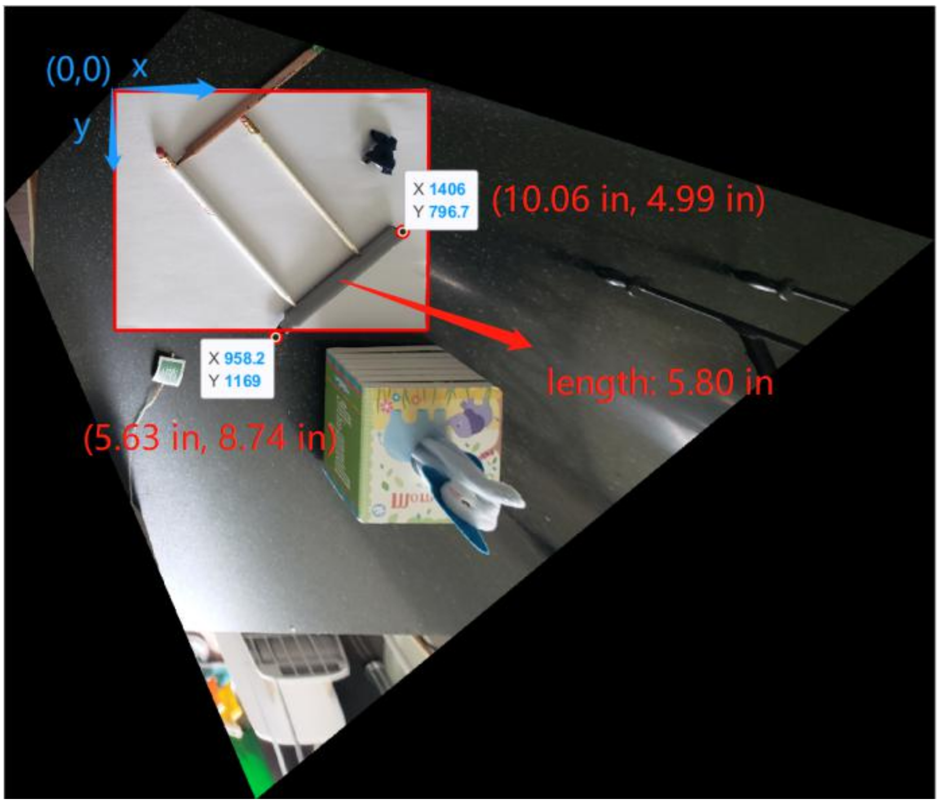

Fig 2.3b Size and position of objects

We got the endpoint positions and the length of this pen, as is shown in the figure 2.3b.

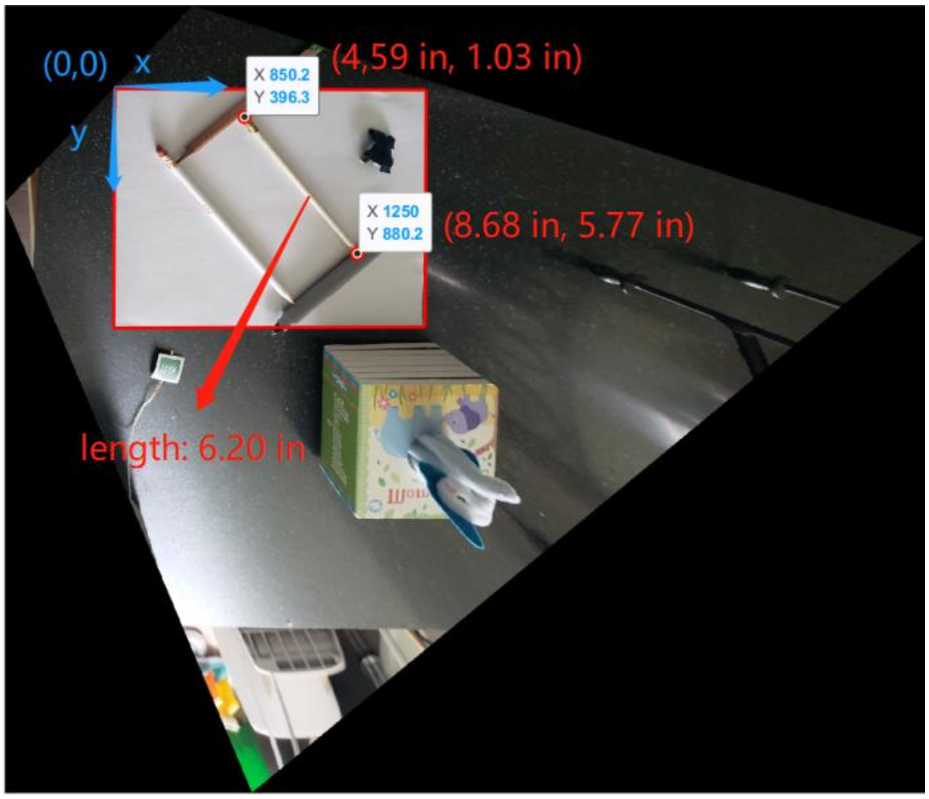

Fig 2.3c Size and position of objects

We have the other pencil's endpoint position and its length, which seems to be a little shorter than the previous pencil, as is shown in the figure 2.3c

2.4 Results Of Indirect Approach

The expected outcome of this indirect approach is that when we remove the projective distortion first , the rectangular surface of multiple objects on the same plane, its edges will revert to parallelism. And after removing the affine distortion, its angles will revert to right angles.

First of all, our original image is shown in the figure 2.4.

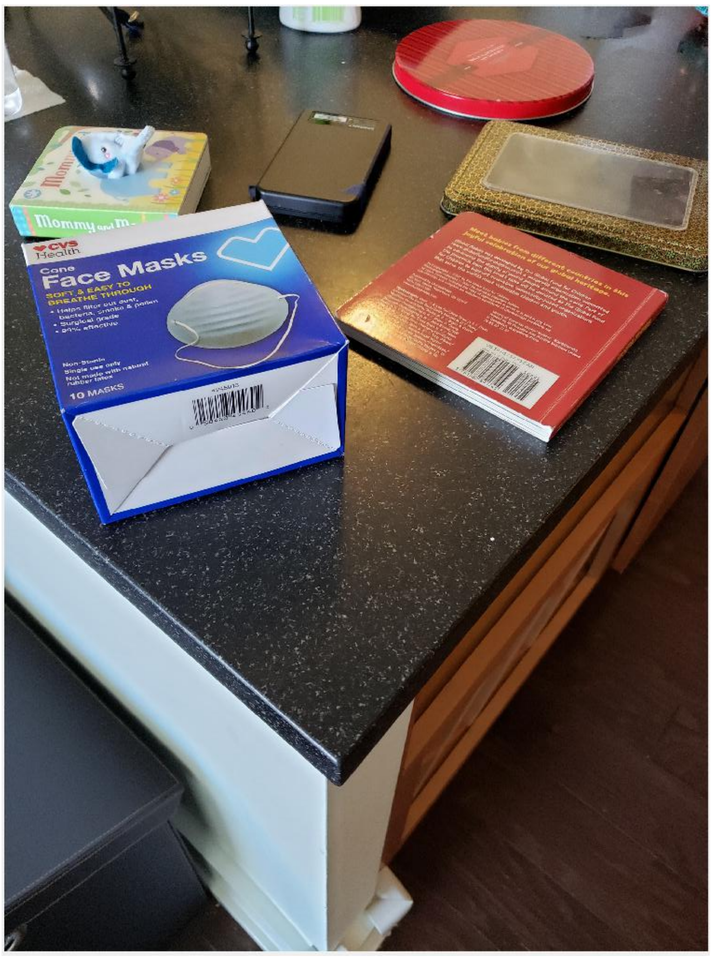

Fig 2.4 The original image

Then, we get two pairs of parallel lines by getting the four vertices of the rectangle, as is shown in the figure 2.5.

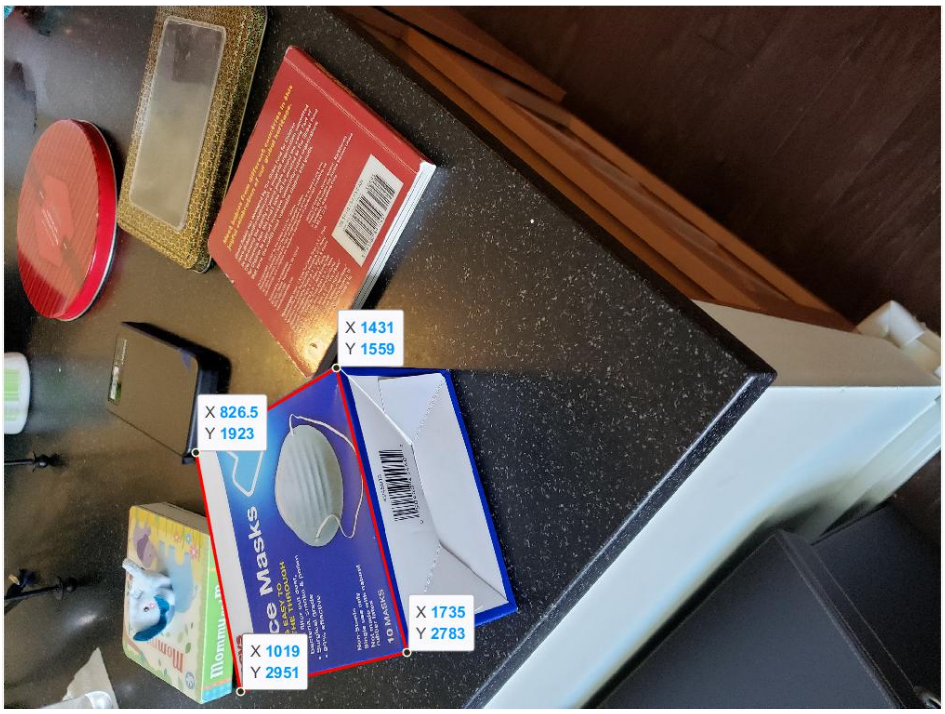

Fig 2.5 Get two pairs of parallel lines by four points

Then, we can obtain the affinely rectified image as is shown in the figure 2.6.

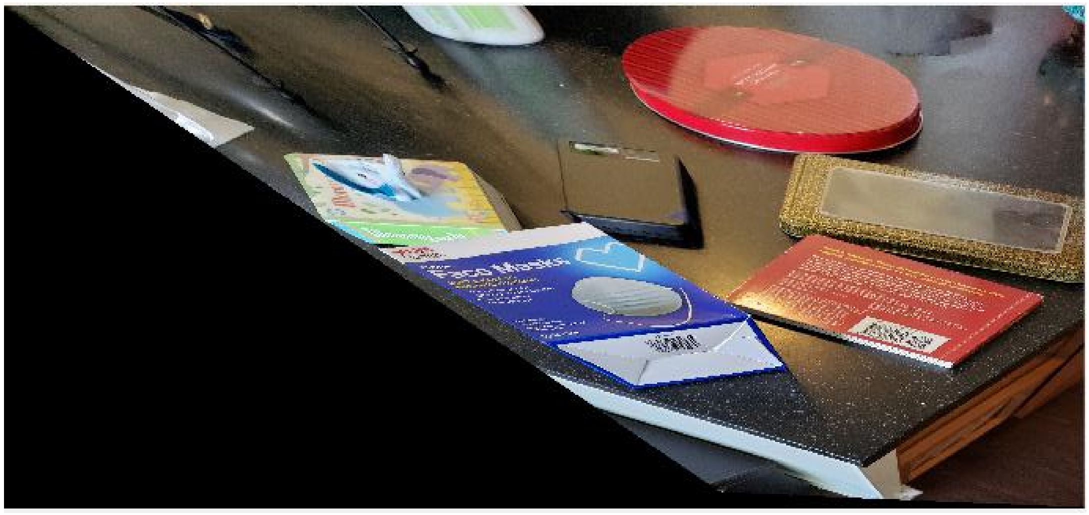

Fig 2.6 The image after removing projective distortion

Then get two pairs of physically orthogonal lines, as is shown in the figure 2.7.

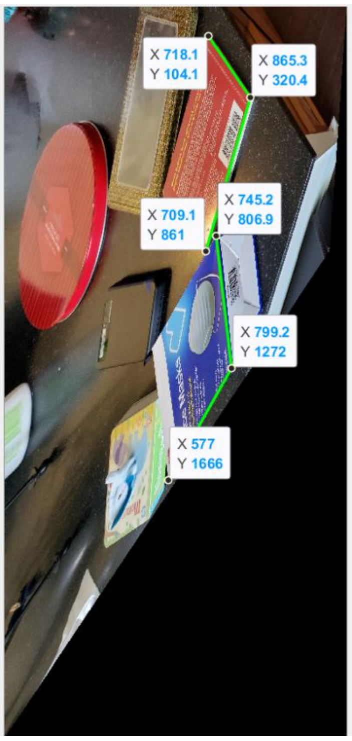

Fig 2.7 Get two pairs of orthogonal lines by six points

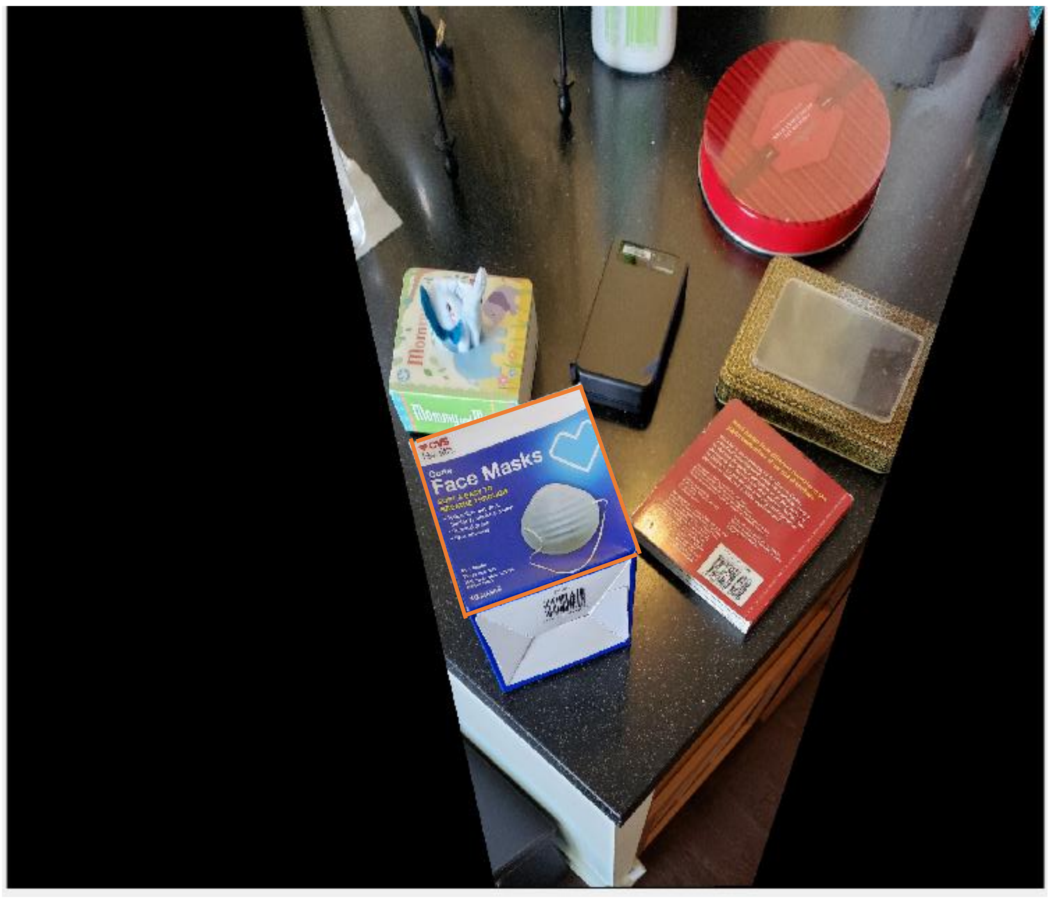

Fig 2.8 The image after removing affine distortion

Finally, we get the image after removing affine distortion, as is shown in the figure 2.8. We can clearly observe that the orange lines form a rectangle. And the surfaces of other objects on this plane are also rectangular, indicating that the distortions in this plane have been removed.

For the same image, the distortions on a different plane can be rectified as shown in the figures below.

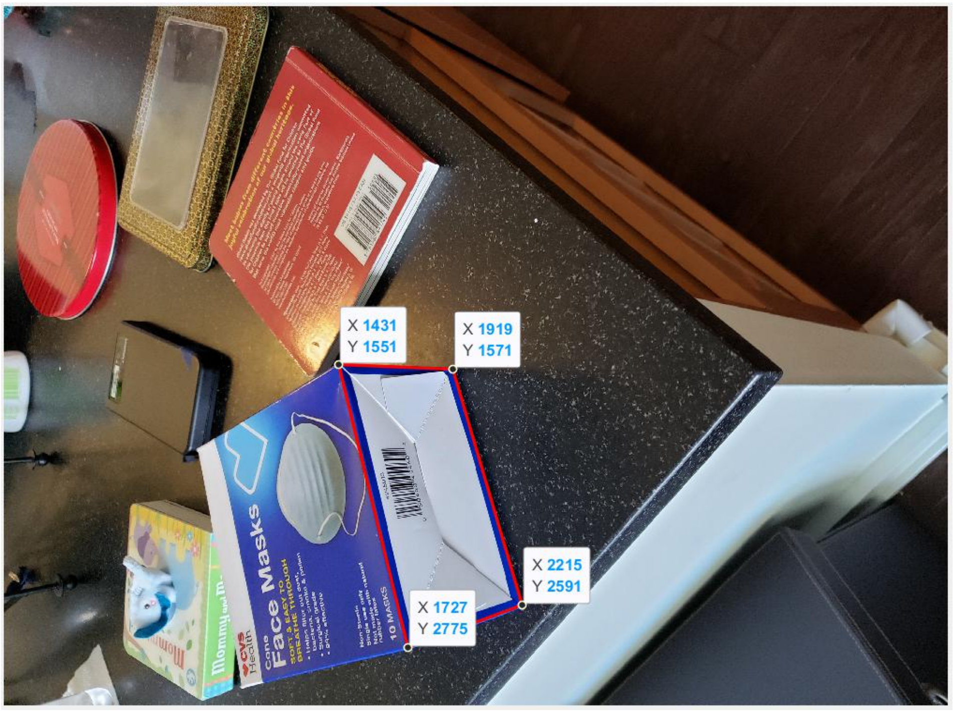

Fig 2.9 Get two pairs of parallel lines by four points

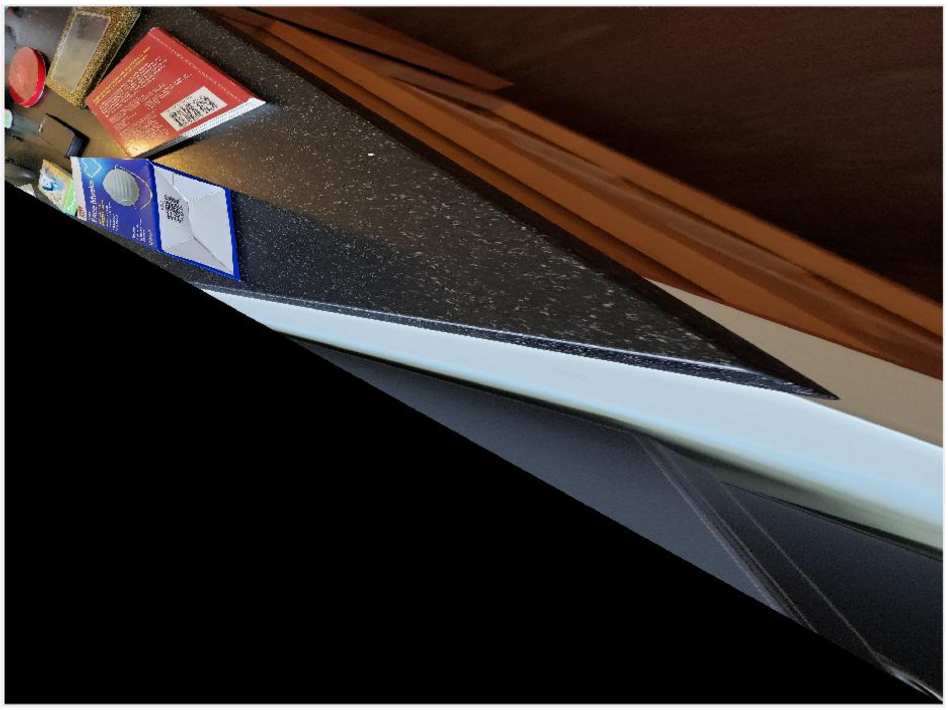

Fig 2.10 The image after removing projective distortion

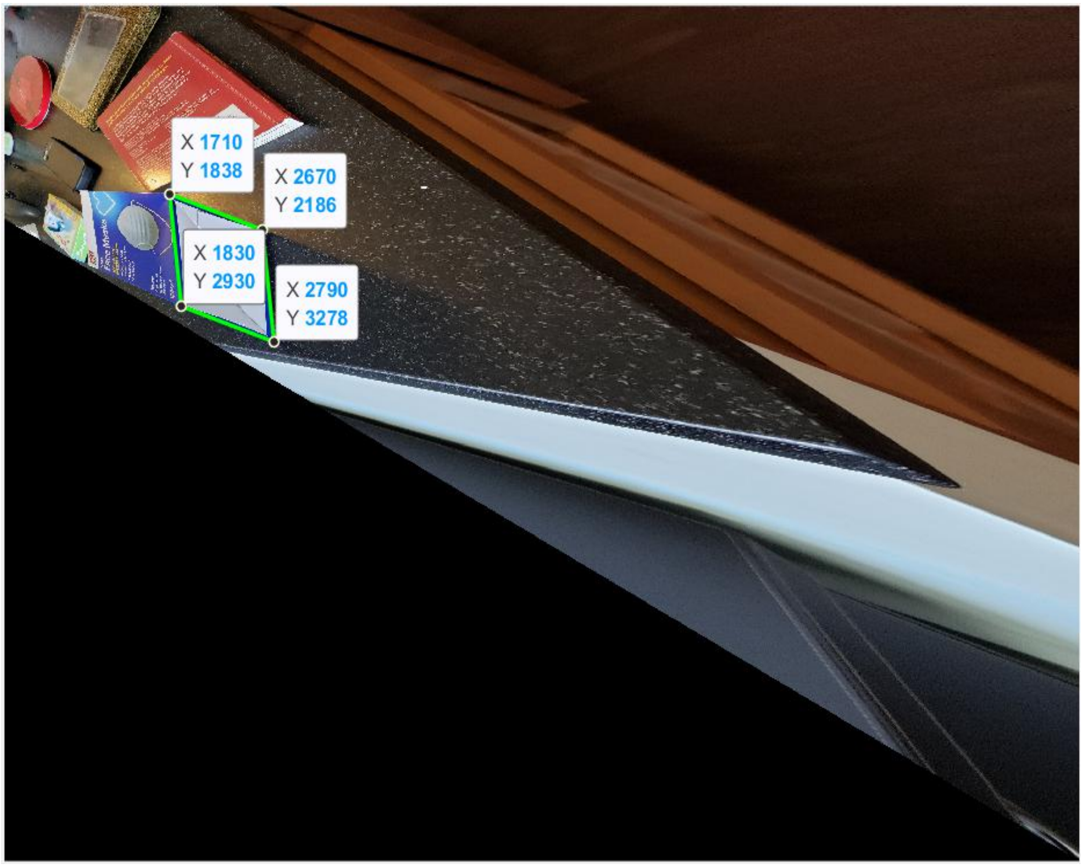

Fig 2.11 Get two pairs of orthogonal lines by four points

Here in the figure 2.11, the reason why we used four points here is because I can only find these four points in this plane, so I used four points to get two pairs of vertical lines . But when programming, we encountered a problem. If we used two pairs of vertical lines parallel to each other, we need to use another method to solve the matrix K. If we still use Cholesky decomposition to decompose the matrix S, matlab will report an error (the matrix is not a positive definite matrix). We can use this method to solve this problem: Since the symmetric matrix S can be written as $S = UDU^T$, where $U^{-1} = U^T$, we can get $A = U\ \mathrm{sqrt}(D)\ U^T$.

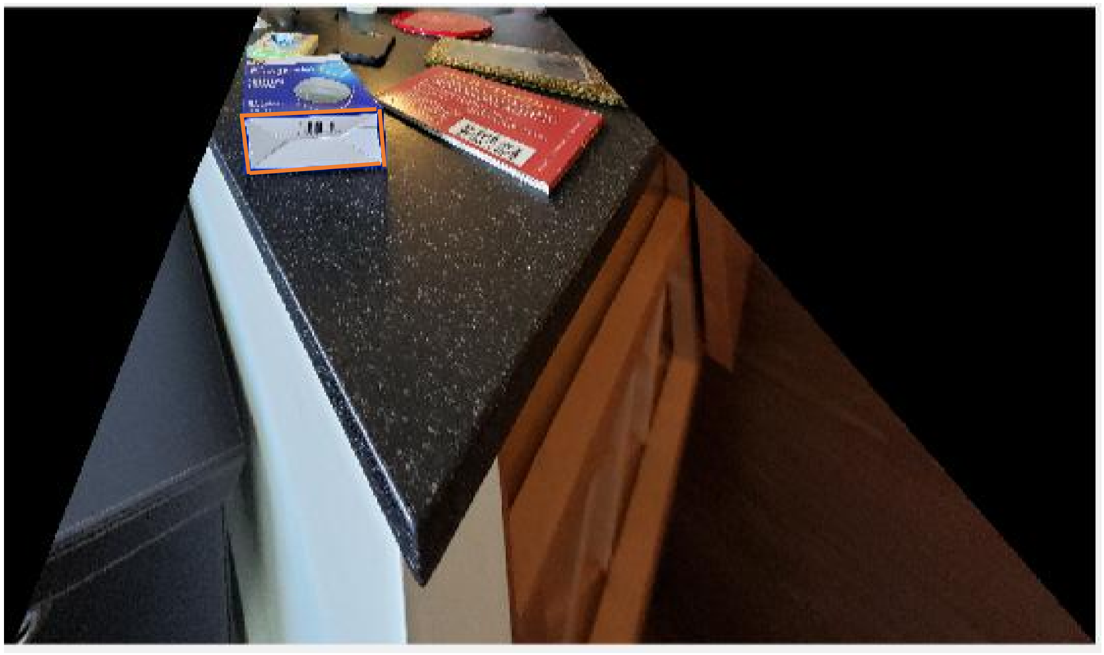

Fig 2.12 The image after removing affine distortion

As we can see in the figure 2.12, although this is a picture, the result is completely different because of the different planes used, but the orange lines form a rectangle, which shows that the distortions in this plane are removed.

Below is the rectification of another image:

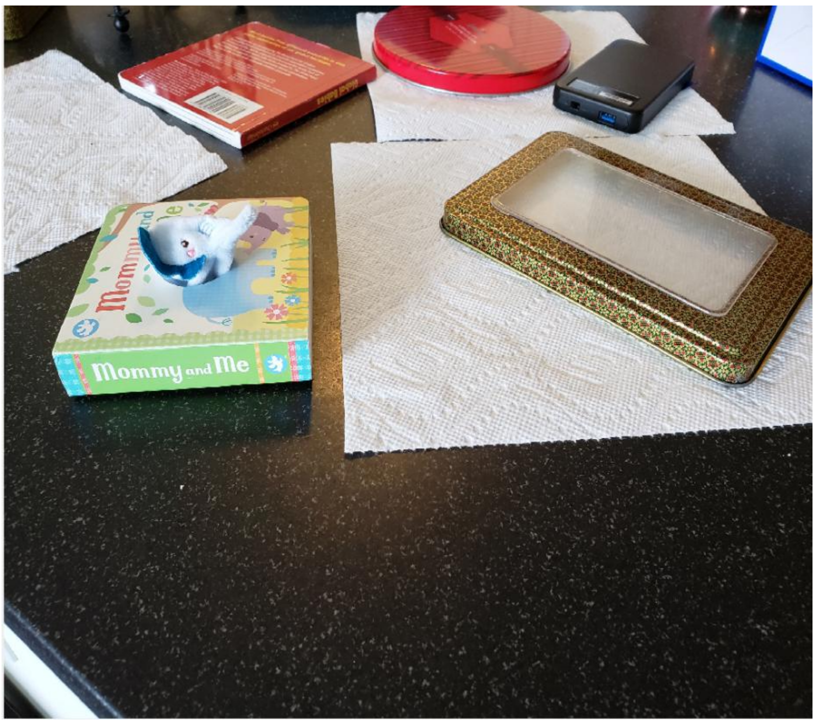

Fig 2.13 The original image

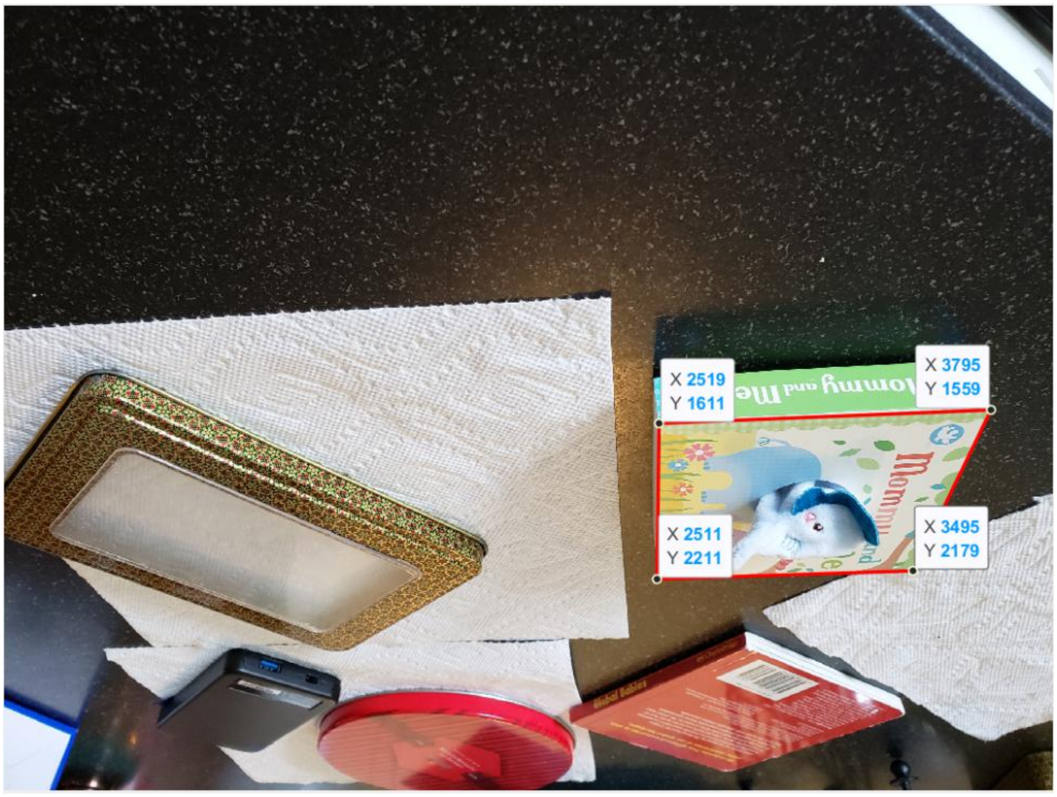

Fig 2.14 Get two pairs of parallel lines by four points

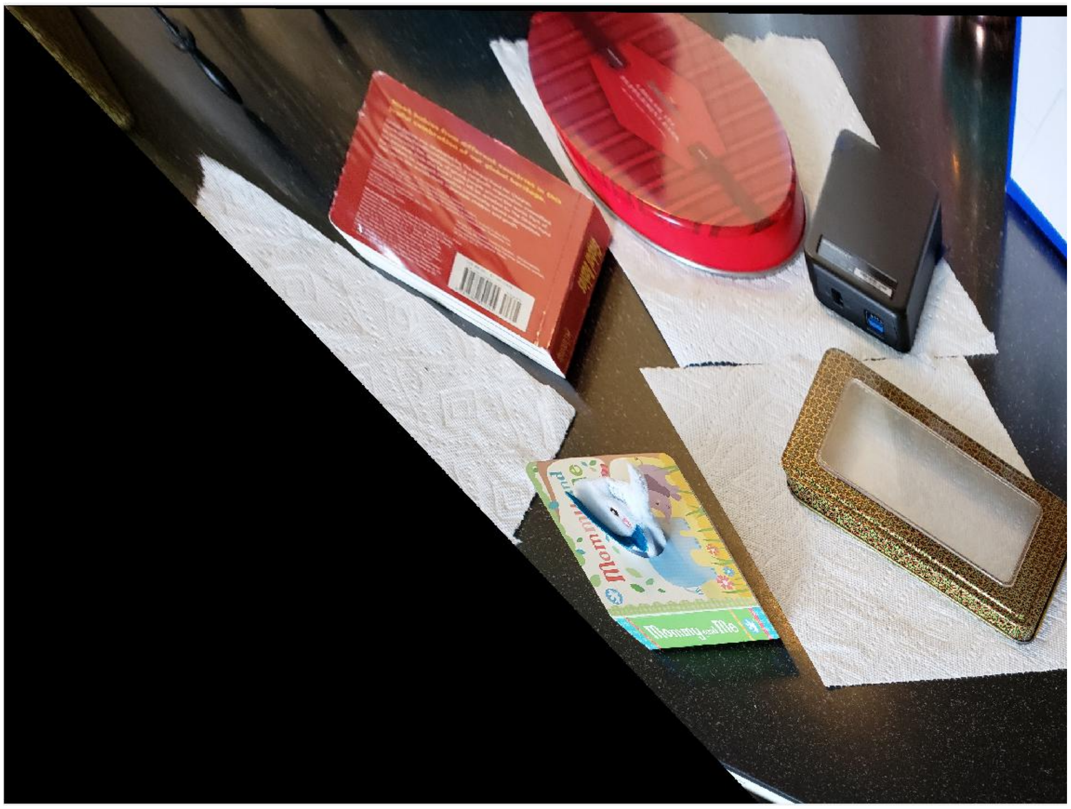

Fig 2.15 The image after removing projective distortion

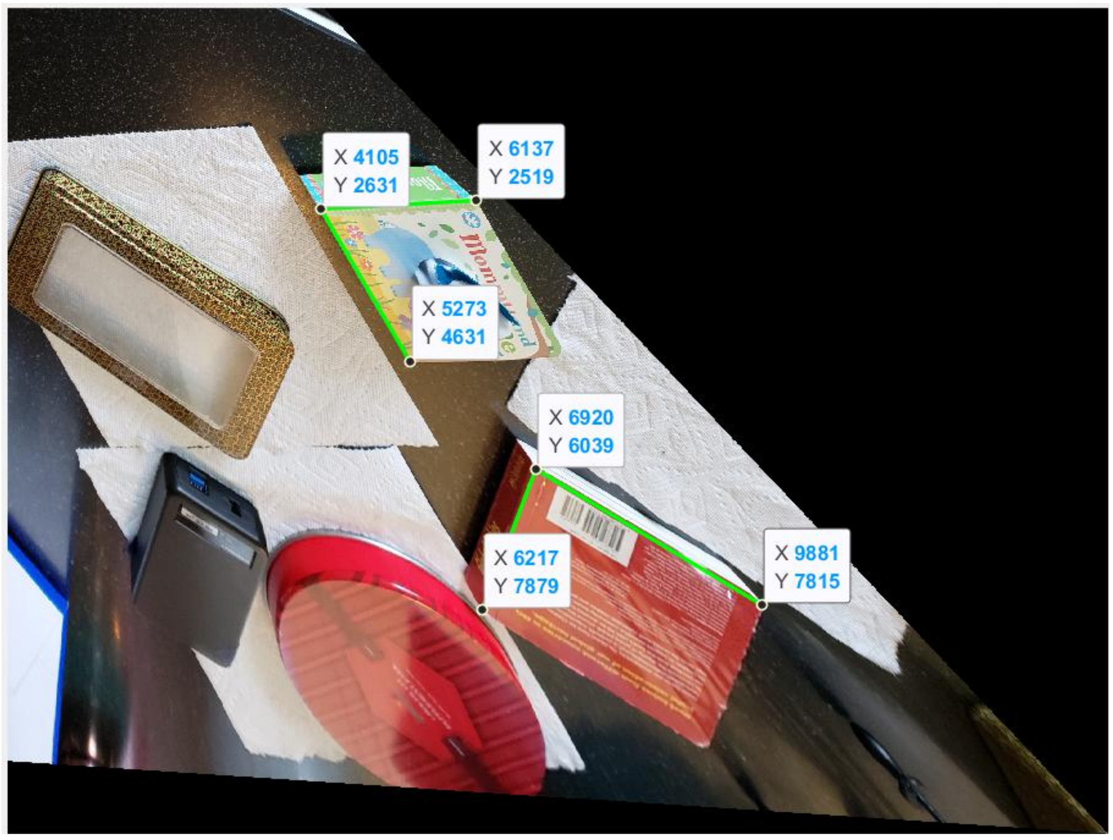

Fig 2.16 Get two pairs of orthogonal lines by six points

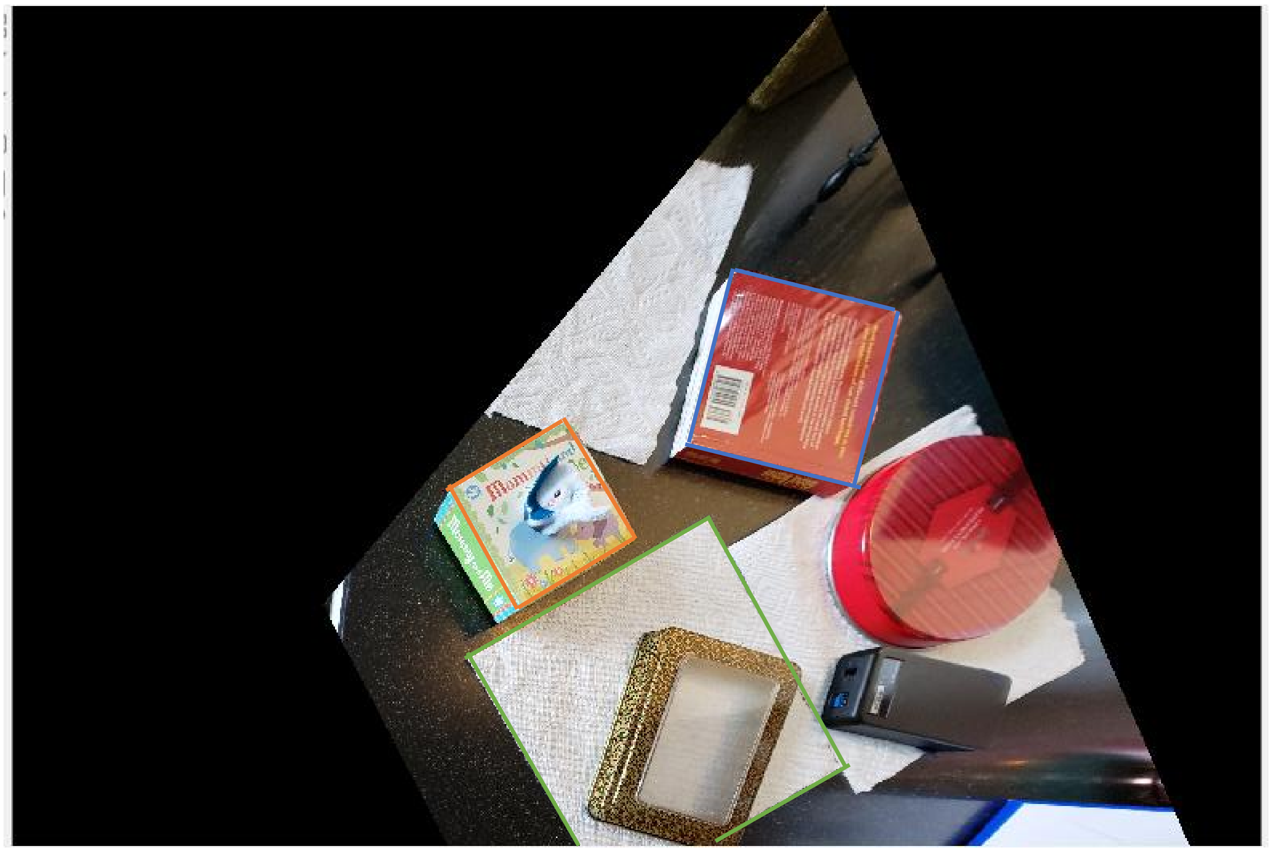

Fig 2.17 The image after removing affine distortion

Below is the rectification process for another image:

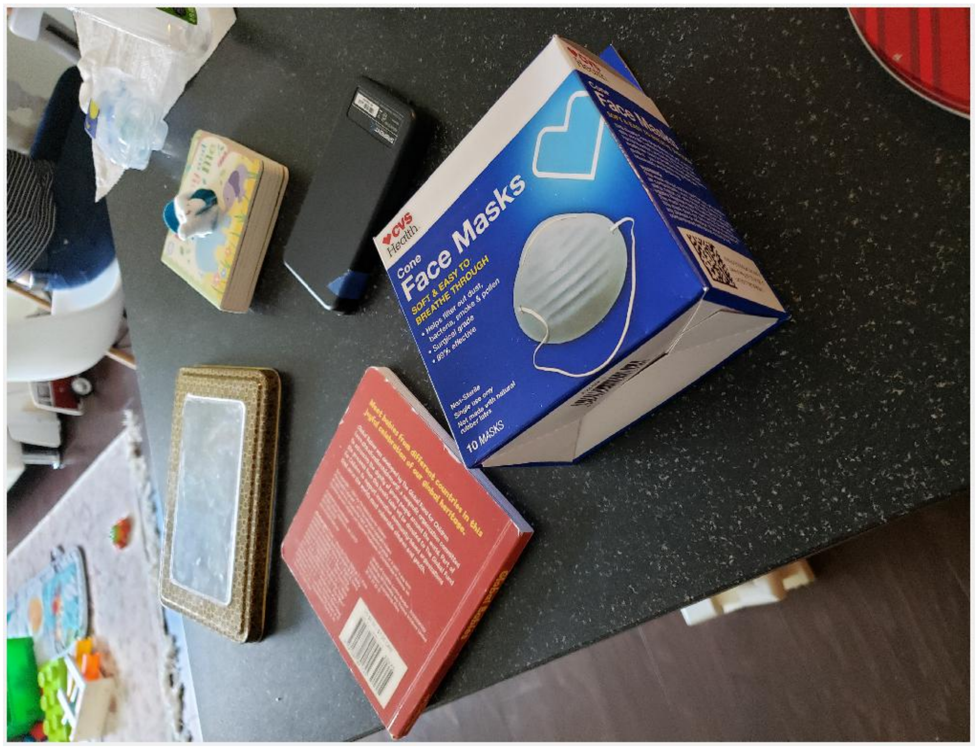
Fig 2.18 The original image

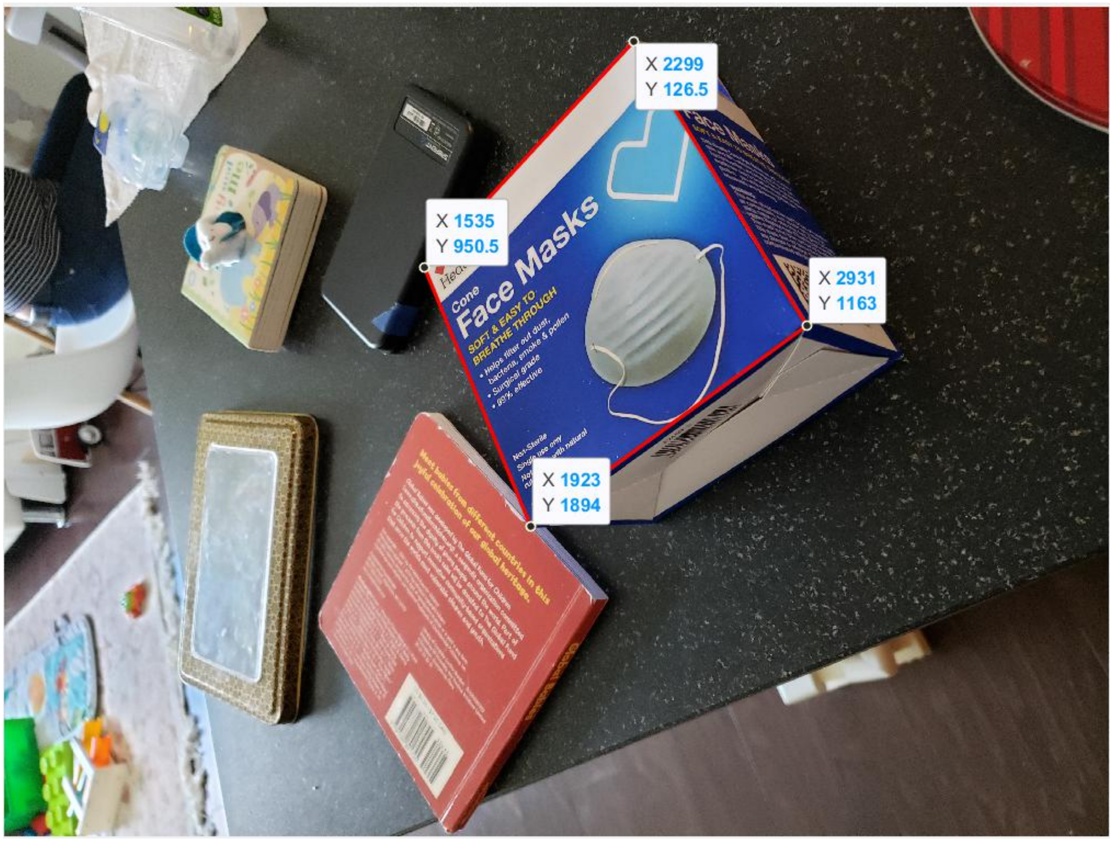

Fig 2.19 Get two pairs of parallel lines by four points

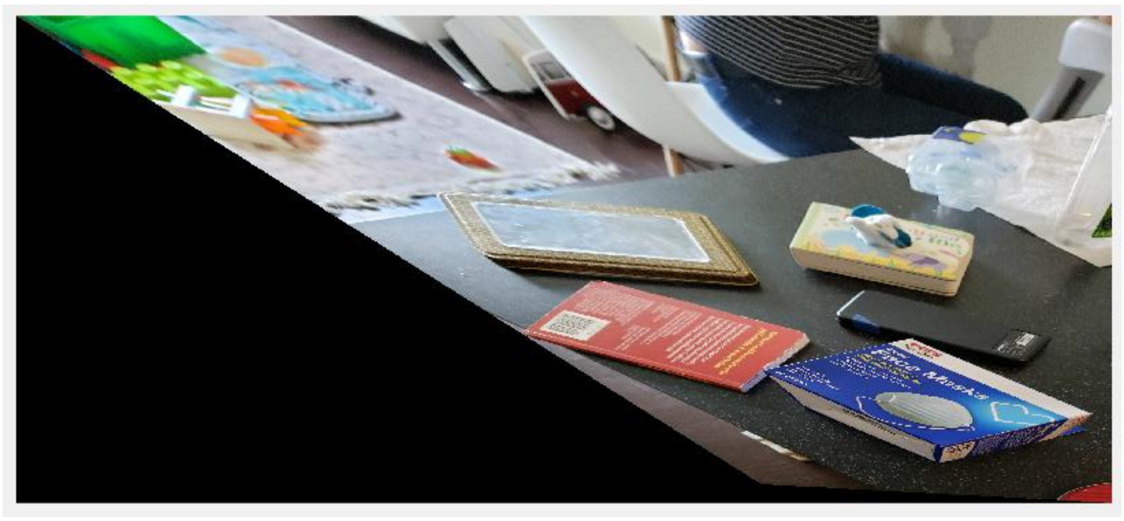
Fig 2.20 The image after removing projective distortion

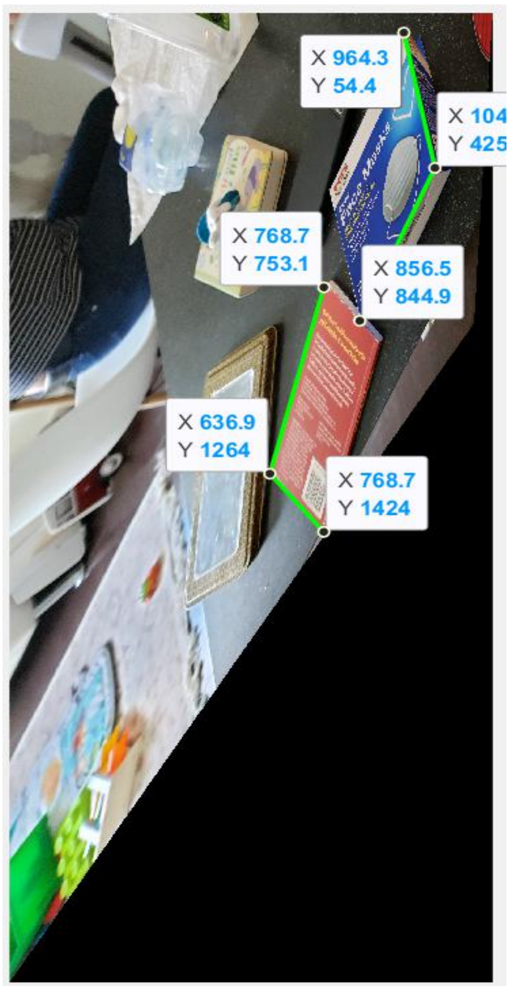

Fig 2.21 Get two pairs of orthogonal lines by six points

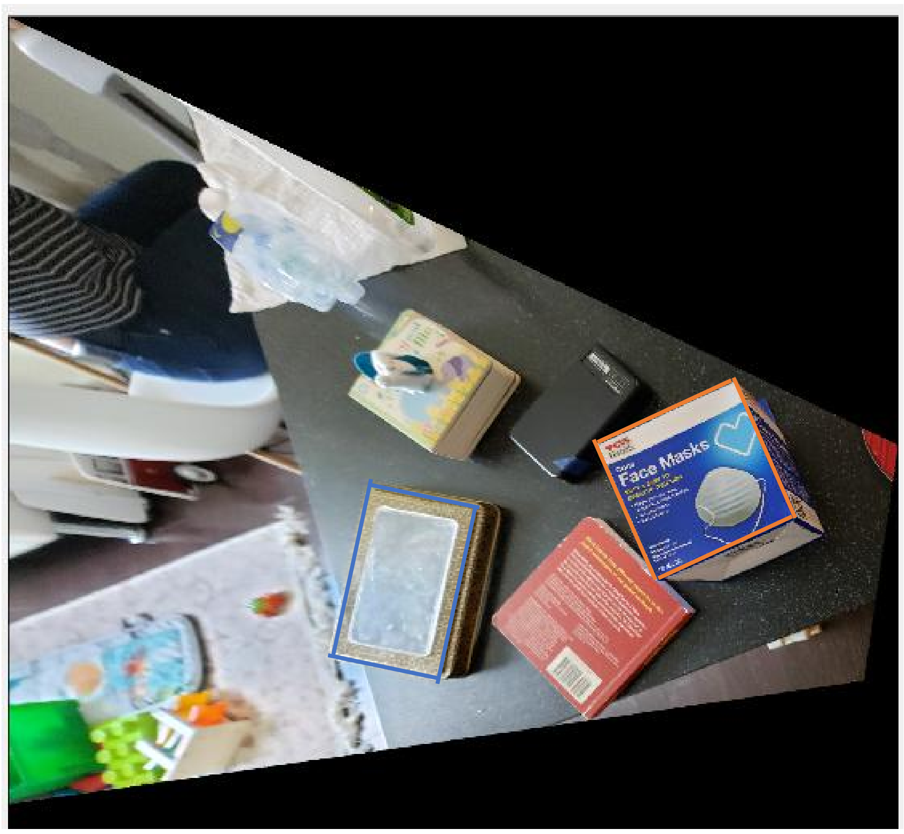

Fig 2.22 The image after removing affine distortion

Finally, one more image rectification is shown below.

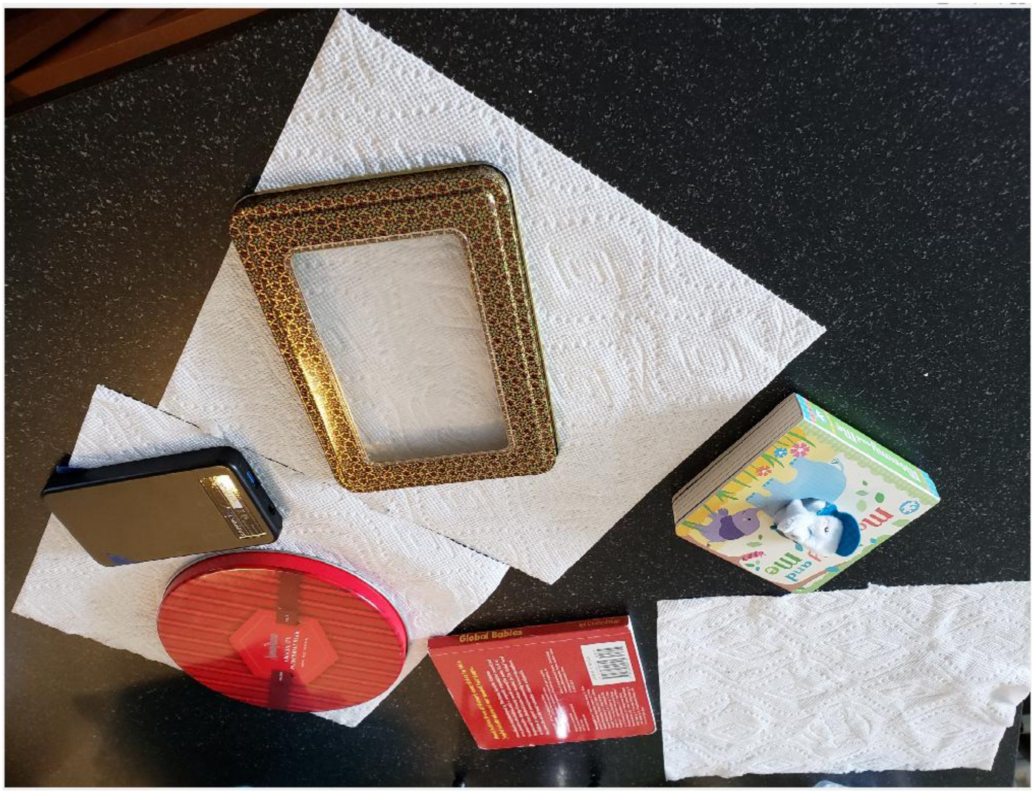

Fig 2.23 The original image

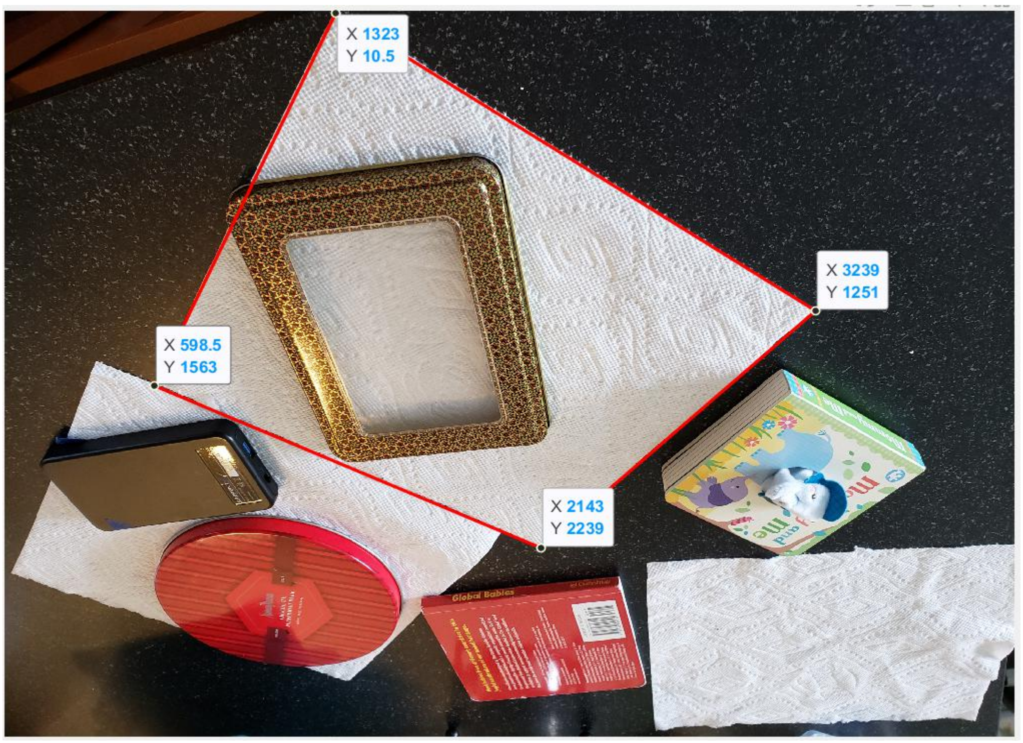

Fig 2.24 Get two pairs of parallel lines by four points

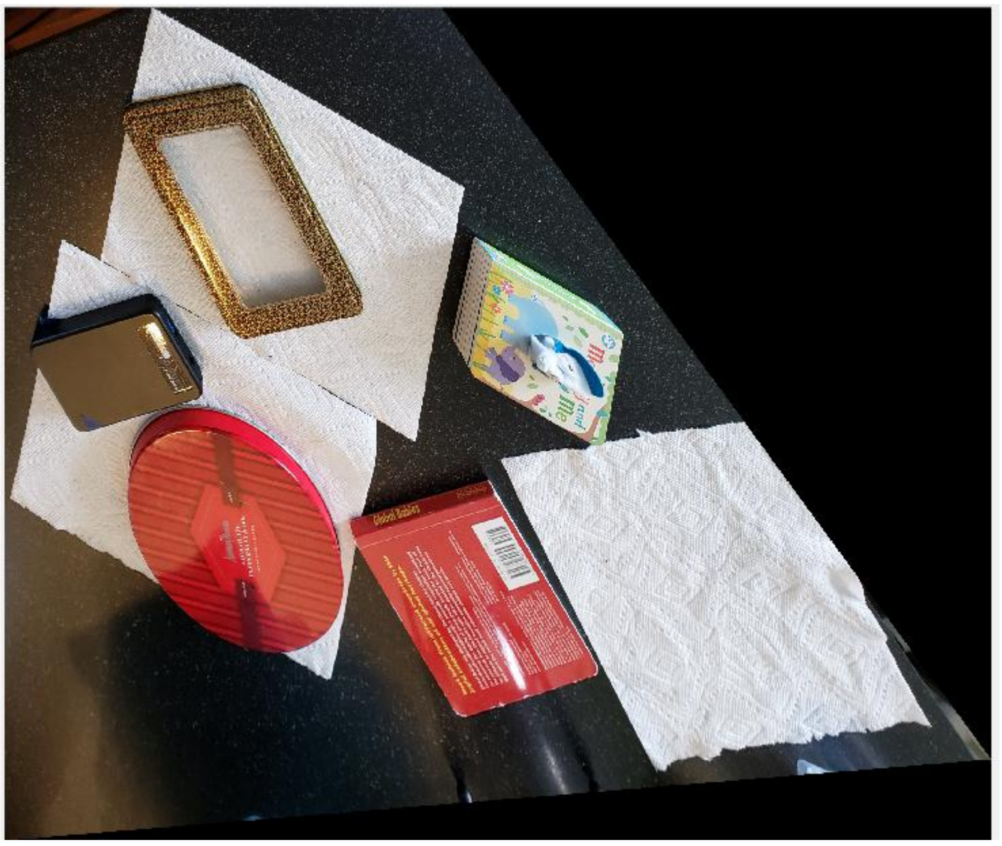

Fig 2.25 The image after removing projective distortion

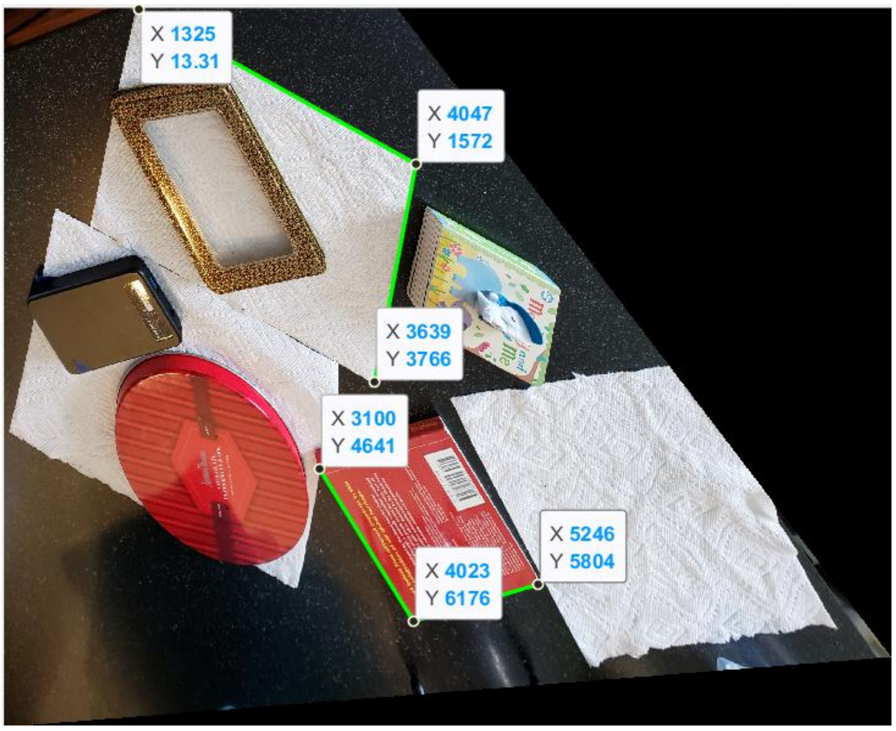

Fig 2.26 Get two pairs of orthogonal lines by four points

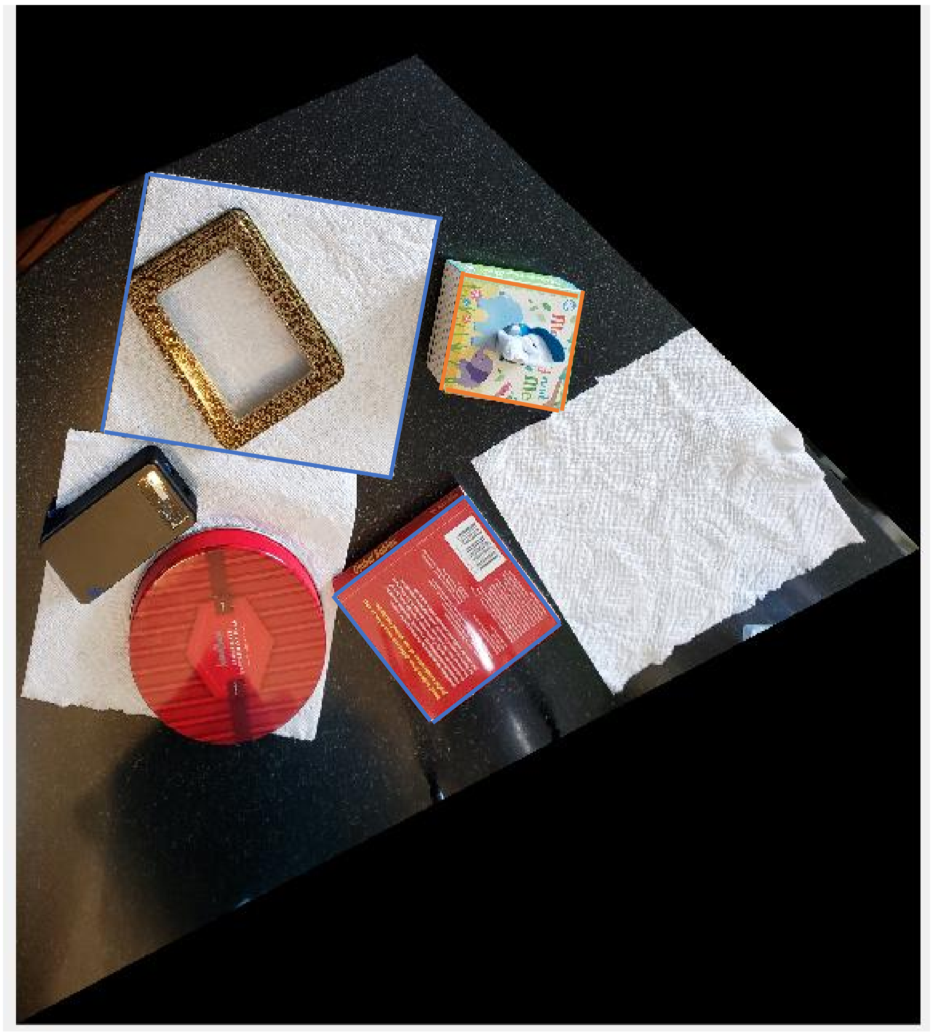

Fig 2.27 The image after removing affine distortion

3. CONCLUSION

This project involved the rectification of affine and projective distortion in an image to reconstruct real-world planes. This was done using both the direct method of identifying coordinates on the real-world plane of interest and the indirect method of identifying sets of parallel and orthogonal lines. Rectification of this distortion allows us to reconstruct scene geometry and measure ratios and angles accurately. Knowledge of the size of one scene object can then be used to measure the sizes of other objects on the same plane. Implementation was done in MATLAB to estimate the parameters of affine and projective distortion on a set of images, and then transform coordinates to rectify this distortion. The success of the rectification is evident in the results, where rectangular objects have orthogonal and parallel lines. This project gave practical insight into the relationship between scene geometry and image transformations.

CODE FOR 2.1

```
clear;close all;clc
syms h11 h12 h13 h21 h22 h23 h31 h32
frame = 'myimg.jpg';
img = imread(frame);
imshow(img);
[xx,yy]=ginput(4);
p11 = [xx(1),yy(1),1]';
p21 = [xx(2),yy(2),1]';
p31 = [xx(3),yy(3),1]';
p41 = [xx(4),yy(4),1]';
s = size(img);

    figure();
    imshow(img);
    hold on;

plot([p11(1),p41(1)],[p11(2),p41(2)],'Color','r','Lin
eWidth',2)

plot([p21(1),p31(1)],[p21(2),p31(2)],'Color','r','Lin
eWidth',2)

plot([p11(1),p21(1)],[p11(2),p21(2)],'Color','r','Lin
eWidth',2)

plot([p31(1),p41(1)],[p31(2),p41(2)],'Color','r','Lin
eWidth',2)

real1=[0,0,1]';
real2=[0,850,1]';
real3=[1100,850,1]';
real4=[1100,0,1]';

f1=sym(h11*real1(1)+h12*real1(2)+h13-
p11(1)*(h31*real1(1)+h32*real1(2)+1));
f2=sym(h21*real1(1)+h22*real1(2)+h23-
p11(2)*(h31*real1(1)+h32*real1(2)+1));
f3=sym(h11*real2(1)+h12*real2(2)+h13-
p21(1)*(h31*real2(1)+h32*real2(2)+1));
f4=sym(h21*real2(1)+h22*real2(2)+h23-
p21(2)*(h31*real2(1)+h32*real2(2)+1));

f5=sym(h11*real3(1)+h12*real3(2)+h13-
p31(1)*(h31*real3(1)+h32*real3(2)+1));
```

```
f6=sym(h21*real3(1)+h22*real3(2)+h23-
p31(2)*(h31*real3(1)+h32*real3(2)+1));
f7=sym(h11*real4(1)+h12*real4(2)+h13-
p41(1)*(h31*real4(1)+h32*real4(2)+1));
f8=sym(h21*real4(1)+h22*real4(2)+h23-
p41(2)*(h31*real4(1)+h32*real4(2)+1));
[h11,h12,h13,h21,h22,h23,h31,h32]=solve([f1,f2,f3,f4,
f5,f6,f7,f8]);
[h11,h12,h13,h21,h22,h23,h31,h32]=deal(double(h11),do
uble(h12),double(h13),double(h21),double(h22),double(
h23),double(h31),double(h32));

H=[h11,h12,h13;h21,h22,h23;h31,h32,1];

H_P=projective2d(inv(H)');
newimg=imwarp(img,H_P);
figure,imshow(newimg);
```

CODE FOR 2.2

```
clear;close all;clc

% remove the projective distortion
frame = 'myimg4.jpg';
img = imread(frame);
imshow(img);
[xx,yy]=ginput(4);

p11 = [xx(1),yy(1),1]';
p21 = [xx(2),yy(2),1]';
p31 = [xx(3),yy(3),1]';
p41 = [xx(4),yy(4),1]';

    figure();
    imshow(img);
    hold on;

plot([p11(1),p41(1)],[p11(2),p41(2)],'Color','r','Lin
eWidth',2)

plot([p21(1),p31(1)],[p21(2),p31(2)],'Color','r','Lin
eWidth',2)

plot([p11(1),p21(1)],[p11(2),p21(2)],'Color','r','Lin
eWidth',2)

plot([p31(1),p41(1)],[p31(2),p41(2)],'Color','r','Lin
eWidth',2)

% get four lines
l1 = cross(p11, p21);
l2 = cross(p31, p41);
l3 = cross(p11, p41);
l4 = cross(p21, p31);
s = size(img);

% get our two vanishing points
p_inf1 = cross(l1,l2);
p_inf2 = cross(l3,l4);
% infinit line
l_inf = cross(p_inf1, p_inf2);
l_inf = l_inf / l_inf(3);

H = [1, 0, 0; 0, 1, 0];
H(3,:) = l_inf';
```

```matlab
H_P=projective2d(H');
final_img=imwarp(img,H_P);
figure,imshow(final_img);


% remove the affine distortion
[xx1,yy1]=ginput(6);

p1 = [xx1(1),yy1(1), 1]';
p2 = [xx1(2),yy1(2), 1]';
p3=[xx1(3),yy1(3), 1]';

p4 = [xx1(4),yy1(4), 1]';
p5 = [xx1(5),yy1(5), 1]';
p6= [xx1(6),yy1(6), 1]';

    figure;
    imshow(final_img);
    hold on;

plot([p1(1),p2(1)],[p1(2),p2(2)],'Color','g','LineWid
th',2)

plot([p2(1),p3(1)],[p2(2),p3(2)],'Color','g','LineWid
th',2)

plot([p4(1),p5(1)],[p4(2),p5(2)],'Color','g','LineWid
th',2)

plot([p5(1),p6(1)],[p5(2),p6(2)],'Color','g','LineWid
th',2)


pl1 = cross(p1,p2);
pm1 = cross(p2,p3);

pm2 = cross(p4,p5);
pl2 = cross(p5,p6);

l11 = pl1(1);
l12 = pl1(2);
m11 = pm1(1);
m12 = pm1(2);

l21 = pl2(1);
l22 = pl2(2);
```

```
m21 = pm2(1);
m22 = pm2(2);

orth1 = [l11*m11, l11*m12 + l12*m11, l12*m12];
orth2 = [l21*m21, l21*m22 + l22*m21, l22*m22];

syms s11 s12

f1=sym(s11*orth1(1)+s12*orth1(2)+orth1(3));
f2=sym(s11*orth2(1)+s12*orth2(2)+orth2(3));
[s11,s12]=solve([f1,f2]);
[s11,s12]=deal(double(s11),double(s12));

S = [s11, s12; s12, 1];
% [U, D, V] = svd(S);
% A = U'*sqrt(D)*U;
K = chol(S, 'lower');
A=K;
H2 = [A, [0;0]; [0,0,1]];

H_A=affine2d(inv(H2)');
fnewimg=imwarp(final_img,H_A);
figure,imshow(fnewimg);
```