Hw of CS280. HAOYU CHEN.

1°. a). proof: ∵ vanishing line of a plane is the union of all vanishing points defined by lines on this plane.

∴ Any lines ∈ this plane; its vanishing pnt ∈ vanishing points set of this plane i.e. the vanishing line of this plane.

mathmatically: ∀ a line: $\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} t + \begin{pmatrix} d_1' \\ d_2' \\ d_3' \end{pmatrix}$; ∀ t ∈ R.

if line belong to a plane: $ax+by+cz=d$.

$\Rightarrow \begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} \cdot (a,b,c) = 0 \Rightarrow ad_1 + bd_2 + cd_3 = 0$. — ①.

assume the camera locates at O(0,0,0) the plane is $y = y_0$

Then the vanishing pnt of line can be found:

∀ a pnt ∈ line: i.e. $Q = (d_1t+d_1', d_2t+d_2', d_3t+d_3')$.

$\vec{OQ} = (d_1t+d_1', d_2t+d_2', d_3t+d_3')$ Then intersection of $\vec{OQ}$ & plane $y=y_0$

is $(x\frac{y_0}{y}, y_0, \frac{z}{y}y_0)$ i.e. $\left(\frac{d_1t+d_1'}{d_2t+d_2'}y_0, y_0, \frac{d_3t+d_3'}{d_2t+d_2'}y_0\right)$.

The vanishing pnt is $\lim_{t\to\infty}$ intersection pnt $= \left(\frac{d_1}{d_2}y_0, y_0, \frac{d_3}{d_2}y_0\right)$. — ②

combine ① & ②.

we have the set of any vanishing pnt of lines ∈ a plane (a,b,c,d)

is set $S = \{ \left(\frac{d_1}{d_2}y_0, y_0, \frac{d_3}{d_2}y_0\right)$ and $ad_1+bd_2+cd_3=0$ }.

(here y_0, a,b,c,d is fixed parameter).

∀ pnt $(x,y,z) \in S$.

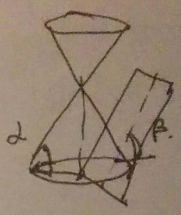
we have: $x = \frac{d_1}{d_2}y_0$; $y = y_0$; $z = \frac{d_3}{d_2}y_0$; ① $\Rightarrow \frac{d_1}{d_2}y_0 = -\left(\frac{cd_3}{d_2}+b\right)\frac{y_0}{a}$.

$\therefore x = -\frac{c}{a}z - \frac{by_0}{a}$, i.e. it's a straight line in the projection plane $y=y_0$. i.e. the vanishing line

∴ the set of any vanishing pnts of lines for a plane belong to the vanishing line of this plane ⑤

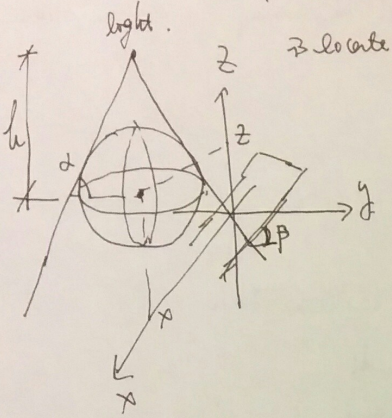
1.(b). Since eccentricity e can be derived as $\frac{\sin\beta}{\sin\alpha}$.

the angle of α & β can be found on the right figure.



For the sphere we can define a light source

$\Rightarrow$ locate at $(X, 0, z+h)$.



Then: $\sin\alpha = \frac{\sqrt{h^2-r^2}}{h}$

$\therefore e = \frac{\sin\beta * h}{\sqrt{h^2-r^2}}$ we can set $h = \sqrt{X^2+z^2}$.

& set $\sin\beta = \frac{X}{\sqrt{X^2+z^2}}$

Then we can get $e = \frac{X}{\sqrt{X^2+z^2-r^2}}$.

i.e we can set the location of light is $(X, 0, z+\sqrt{X^2+z^2})$.

& the angle of projection plane w/ horizontal plane β is $\sin^{-1}\left(\frac{X}{\sqrt{X^2+z^2}}\right)$.

For Parabola. $e = 1$. If we set h

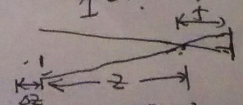
Then $\sin\beta = \frac{\sqrt{h^2-r^2}}{h}$. $\therefore$ we can get Parabola Silhouette by control $\beta = \sin^{-1}\left(\frac{\sqrt{h^2-r^2}}{h}\right)$ for $h > r$.

For Hyperbola: $e > 1$.

Then $\sin\beta > \frac{\sqrt{h^2-r^2}}{h}$.

Similarly we can get hyperbola by controlling $\beta > \sin^{-1}\left(\frac{\sqrt{h^2-r^2}}{h}\right)$ for $h > r$.

1(c). The eye is a camera we define $X_0 Y_0$ the things we see; X_0, Y_0 is their original location.



$\therefore x_0 = \frac{x_0}{z} f$ $y_0 = \frac{y}{z} f$. $\because X \equiv 0$. $\therefore$ we only consider y.

w/ a error Δz of depth: $y_0' = \frac{y}{z+\Delta z} f = \frac{yf}{z(1+\Delta z/z)} \approx \frac{yf}{z}(1-\frac{\Delta z}{z})$. $\therefore \Delta y = y_0 - y_0' = \frac{yf\Delta z}{z^2}$.

$\because \theta = \frac{\Delta y}{f} = \frac{y\Delta z}{z^2}$ $\therefore \Delta z = \frac{\theta z^2}{y}$; here y is the height of a person.

$w/2 = 10m$, $y \approx 1.7m$, $\theta = 1' = \frac{2\pi}{3600} \Rightarrow \Delta z = 0.1027m$

HAOYU CHEN.

hw of CS280

2. 1). assume $\hat{s}=(s_1,s_2,s_3)$; $\vec{b}=(b_1,b_2,b_3)$ $\quad \hat{s}\times\vec{b}=\begin{vmatrix} i & j & k \\ s_1 & s_2 & s_3 \\ b_1 & b_2 & b_3 \end{vmatrix}=(s_2b_3-s_3b_2,\ s_3b_1-s_1b_3,\ s_1b_2-s_2b_1)$.

Then we can write $S=\begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}$

2). $R=\exp(\phi S)=\sum_{k=0}^{\infty}\frac{1}{k!}(\phi S)^k=I+\phi S+\frac{1}{2!}(\phi S)^2+\frac{1}{3!}(\phi S)^3+\cdots$

$S^2=\begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}*\begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}=\begin{bmatrix} -s_3^2-s_2^2 & s_2s_1 & s_1s_3 \\ s_1s_2 & -s_3^2-s_1^2 & s_3s_2 \\ s_1s_3 & s_2s_3 & -s_2^2-s_1^2 \end{bmatrix}$

$S^3=S^2*S=\begin{bmatrix} -s_3^2-s_2^2 & s_2s_1 & s_1s_3 \\ s_1s_2 & -s_3^2-s_1^2 & s_3s_2 \\ s_1s_3 & s_2s_3 & -s_2^2-s_1^2 \end{bmatrix}\begin{bmatrix} 0 & -s_3 & s_2 \\ s_3 & 0 & -s_1 \\ -s_2 & s_1 & 0 \end{bmatrix}=\begin{bmatrix} 0 & s_3 & -s_2 \\ -s_3 & 0 & s_1 \\ s_2 & -s_1 & 0 \end{bmatrix}*(s_3^2+s_2^2+s_1^2)$

$\because s$ is unit vector $\therefore s_1^2+s_2^2+s_3^2=1$ $\quad \therefore S^4=-S^2;\ S^5=S;\ S^6=S^2;\ S^7=-S$

$\therefore S^3=-S$ plug in the equation of R.

$R=I+(\phi-\frac{\phi^3}{3!}+\frac{\phi^5}{5!}-\cdots)S+(\frac{\phi^2}{2!}-\frac{\phi^4}{4!}+\frac{\phi^6}{6!}-\cdots)S^2$

$=I+\sin(\phi)S+(1-\cos(\phi))S^2$. ☑

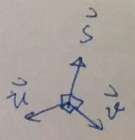
3) See printed pages.

4). $\because \vec{s}$ is the axis to rotate for R. $\therefore$ a rotate for $\vec{s}$ itself has no effects.

$\therefore R\vec{s}=\vec{s}$. $\therefore \vec{s}$ is an eigenvector of R. w/ eigenvalue $\lambda=1$.

5. we first find the eigenvalue of S: $|S-\mu I|=0 \Rightarrow \begin{vmatrix} -\mu & -s_3 & s_2 \\ s_3 & -\mu & -s_1 \\ -s_2 & s_1 & -\mu \end{vmatrix}=0$

$\Rightarrow -\mu^3-\mu(s_1^2+s_2^2+s_3^2)=0 \Rightarrow \mu^2+1=0$. $\therefore \mu=\pm i$.



Then we find the eigenvector for S. assume $\vec{u}, \vec{v}, \vec{s}$ are vertical to each other. Then $Su=\hat{s}\times\vec{u}=v$. $\quad Sv=\hat{s}\times v=-u$.

$\therefore S((-i)*(u+iv))=-i(v+iu)$ $\quad \therefore$ for $\mu=-i$; eigenvalue $\to -i(u+iv)$

for $\mu=i$; — — — $i(u+iv)$

re-define the eigenvector of S as u_s. its eigenvalue as λ_s

Then we have: $R*u_s=u_s+\sin\phi\, S u_s+(1-\cos\phi)S*S*u_s$

$=u_s+\sin\phi\,\lambda_s u_s+(1-\cos\phi)\lambda_s^2 u_s=(1+\sin\phi\lambda_s+(1-\cos\phi)\lambda_s^2)u_s$

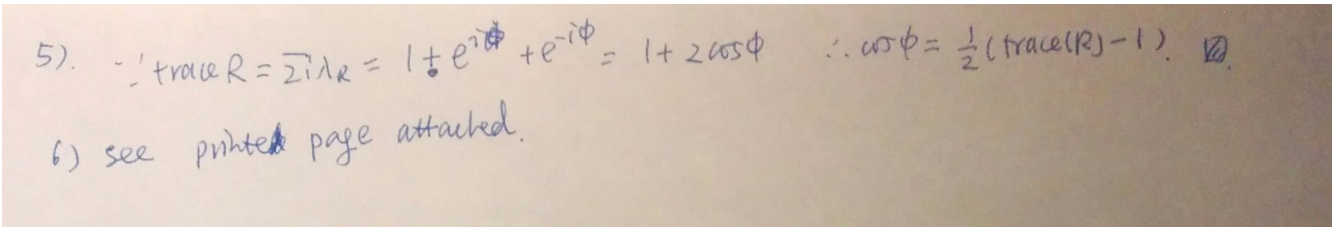
$\therefore$ eigenvalue for R is $1+\sin\phi\lambda_s+(1-\cos\phi)\lambda_s^2$; $\because \lambda_s=\pm i$.

$\therefore \lambda_R=\cos\phi\pm\sin\phi\, i=e^{\pm i\phi}$ & corresponding eigenvector is u_s i.e. $\pm i(u+iv)$

combined the 1st eigenvalue we find $\lambda_R=(1, e^{\pm i\phi})$.

$u_R=(\hat{s}, \pm(v+iu))$. ☑

<3>



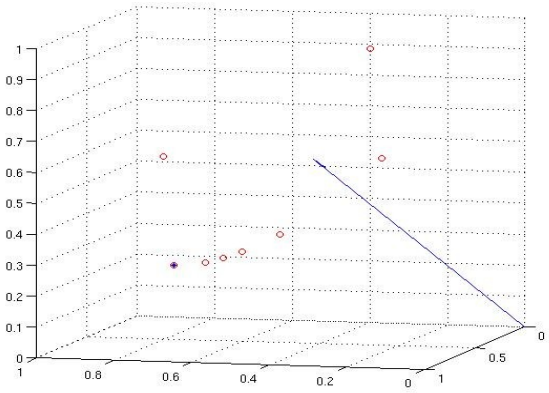
Prob 2.3

```
% main function for problem 2
fry = [0, pi/12, pi/8, pi/6, pi/4, pi/2, pi, 1.5*pi];
s = rand(3,1)
p0 = rand(3,1)

% plot the axis
quiver3(0, 0, 0, s(1), s(2), s(3));
hold on
plot3(p0(1), p0(2), p0(3), '*')

for i = 1:1:8
   R = R_solve(s, fry(i));
   p1 = R*p0;
   plot3(p1(1), p1(2), p1(3), 'ro')

end
```



prob 2.6

```
function [s,phi] = rot_to_ax_phi(R)
[V,D] = eig(R);
tr = 0;
s = zeros(3,1);
for i = 1:1:3
    if (abs(D(i,i)-1.0) < 1e-10)
       s = V(:,i);
    end
    tr = tr + D(i,i);
end
phi = acos(0.5*(tr - 1));
end
```

3. 1) for. affine transform: $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ 0 & 0 & 1 \end{bmatrix}$ for each put v_i & u_i we have

$v_i = H u_i \Rightarrow$ $v_{ix} = h_{11} u_{ix} + h_{12} u_{iy} + h_{13}$

$v_{iy} = h_{21} u_{ix} + h_{22} u_{iy} + h_{23}$.

$\Rightarrow$ Formlate as Linear Regression. $Y = [v_{ix}, v_{iy}]$.

for v_{ix} $X = (1, u_{ix}, u_{iy}, 0, 0, 0)$

for v_{iy} $X = (0, 0, 0, 1, u_{ix}, u_{iy})$.

$\vec{\beta} = (h_{13}, h_{11}, h_{12}, h_{23}, h_{21}, h_{22})$. we can get the $\vec{\beta}^*$ by solving linear regression problem.

for Homography $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix}$. for put v_i, u_i: $w * v_i = H u_i$.

$$\left.\begin{aligned} w * v_{ix} &= h_{11} u_{ix} + h_{12} u_{iy} + h_{13} \\ w * v_{iy} &= h_{21} u_{ix} + h_{22} u_{iy} + h_{23} \\ w &= h_{31} u_{ix} + h_{32} u_{iy} + 1 \end{aligned}\right\} \Rightarrow \left\{\begin{aligned} & h_{31} u_{ix} v_{ix} + h_{32} u_{iy} v_{ix} + v_{ix} \\ &= h_{11} u_{ix} + h_{12} u_{iy} + h_{13}. \\ & h_{31} u_{ix} v_{iy} + h_{32} u_{iy} v_{iy} + v_{iy} \\ &= h_{21} u_{ix} + h_{22} u_{iy} + h_{23} \end{aligned}\right.$$

$\Rightarrow$ $v_{ix} = h_{11} u_{ix} + h_{12} u_{iy} + h_{13} - u_{ix} v_{ix} h_{31} - u_{iy} v_{ix} h_{32}$.

$v_{iy} = h_{21} u_{ix} + h_{22} u_{iy} + h_{23} - h_{31} u_{ix} v_{iy} - u_{iy} v_{iy} h_{32}$.

Thus still a linear regression problem w/

$\beta = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}]$.

2) Since the affine transform is the combination of operations of (translation; scaling; rotation; shearing. It may not be able to transform the points from one image to the other.

& Another reason, since we only choose 4 pairs pairs, which have 8 equations. for affine, we only have 6 unknown params. Thus our *H cannot guarantee that 4 pairs match exactly.

For Homography transform, since we choose 4 pairs pairs, $n = 8$. & our unknow param number is 8 too. Thus for these 4 pairs pairs, our transform can exactly matches points.

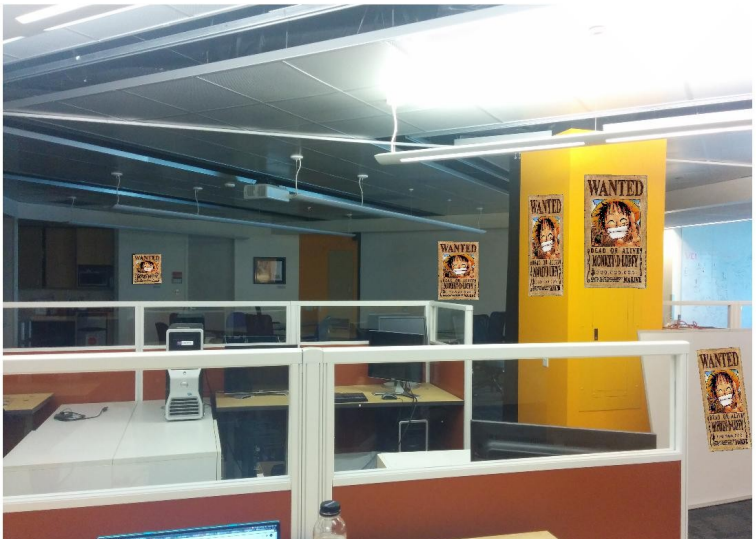
prob 3

Example 1

Affline results

homography results

Example 2



Affline results

Homograph result

problem 3.4

```
function ans = affine_solve(u, v)

% assume v = Hu
N = size(u,1);

X = zeros(2*N, 6);
Y = zeros(2*N, 1);
for i = 1:1:N
    X(2*i - 1, :) = [u(i,1), u(i,2), 1, 0, 0, 0];
    Y(2*i - 1) = v(i, 1);
    X(2*i, :) = [0, 0, 0, u(i, 1), u(i, 2), 1];
    Y(2*i) = v(i, 2);
end

% do Linear regression here
% Y = X*beta
beta = inv(X.'*X)*(X.')*Y;

ans = zeros(3,3);
ans(1, :) = [beta(1), beta(2), beta(3)];
ans(2, :) = [beta(4), beta(5), beta(6)];
ans(3, :) = [0, 0, 1];
end
```

```
function H = homography_solve(u,v)

% v = H u
N = size(u,1);

X = zeros(2*N, 8);
Y = zeros(2*N, 1);
for i = 1:1:N
    X(2*i - 1, :) = [u(i,1), u(i,2), 1, 0, 0, 0, -u(i, 1) * v(i,1), -u(i,2)*v(i,1)];
    Y(2*i - 1) = v(i, 1);
    X(2*i, :) = [0, 0, 0, u(i, 1), u(i, 2), 1, -u(i,1)*v(i,2), -u(i,2)*v(i,2)];
    Y(2*i) = v(i, 2);
end
% do Linear regression here
% Y = X*beta
beta = inv(X.'*X)*(X.')*Y;
H = zeros(3,3);
H(1, :) = [beta(1), beta(2), beta(3)];
H(2, :) = [beta(4), beta(5), beta(6)];
H(3, :) = [beta(7), beta(8), 1];
end

function v = homography_transform(u,H)
N = size(u,2);
u_new = ones(3, N);
u_new(1:2, :) = u;
v_new = H*u_new;

v(1,:) = v_new(1,:)./v_new(3,:);
v(2,:) = v_new(2,:)./v_new(3,:);
```

Appendix of Prob 3

```
% main func for the prob 3

% read the imgs
im1 = imread('../images/onePieceWanted.jpg');
im1 = im2double(im1);
im2 = imread('../images/VCL.jpg');
im2 = im2double(im2);

% for five surface
for s = 1:1:5
    % get the four key pnts

    n = 4;
    pntSet1 = getPntSet(im1, n);
    pntSet2 = getPntSet(im2, n);
    w = size(im2,2);
    h = size(im2,1);
    H = homography_solve(pntSet1, pntSet2);
    % H = affine_solve(pntSet1, pntSet2);
    im_new = imwarped(im1, H,w, h);

    for k = 1:1:3
        for i = 1:1:h
            for j = 1:1:w
                if (im_new(i,j,k)> 1)
                    im_new(i,j,k) = 1;
                elseif (im_new(i,j,k) < 0)
                    im_new(i,j,k) = 0;
                end
                if (im2(i,j,k)> 1)
                    im2(i,j,k) = 1;
                elseif (im2(i,j,k) < 0)
                    im2(i,j,k) = 0;
                end
            end
        end
    end

    im_background = im2;
    im_object = im_new;

    im2 = combineImage(im2, im_new);

    imshow(im2);
end
-----------------------------------------------------------------------------------------
function pntSet = getPntSet(im, n)
disp('choose the key pnt')
figure(), hold off, imshow(im), axis image
for i = 1:1:n
    [x, y, b] = ginput(1);
    if b=='q'
        break;
    end
    pntSet(i,1) = x;
    pntSet(i,2) = y;
```

```
    hold on, plot(pntSet(1:i,1), pntSet(1:i,2), '*-');

end

end
```

```
function res = imwarped(im, H,w,h)

%p' = Hp;
H_re = inv(H);

res = zeros(h,w, size(im,3));
pnt = zeros(size(res,1)*size(res,2),3);
pnt_new = pnt;
k = 1;
for i = 1:1:size(res,1)
    for j = 1:1:size(res,2)
        pnt(k,1) = j;
        pnt(k,2) = i;
        pnt(k,3) = 1;
        k = k+1;
    end
end

pnt_new = H_re*pnt';
pnt_new = pnt_new';
pnt_new(:,1) = pnt_new(:,1)./pnt_new(:,3);
pnt_new(:,2) = pnt_new(:,2)./pnt_new(:,3);

for k = 1:1:size(im,3)
    val = interp2(im(:,:,k),pnt_new(:,1),pnt_new(:,2));
    cont = 1;
    for i = 1:1:size(res,1)
        for j = 1:1:size(res,2)
            if (val(cont) <= 10 && val(cont) >= -10)
                res(i,j,k) = val(cont);
            else
                res(i,j,k) = 0;
            end
            cont = cont+1;
        end
    end
end

end
```

```
function res = combineImage(background, smaller)

%combine 2 ims
res = zeros(size(background,1),size(background,2),size(background,3));


for k = 1:1:size(res,3)
    for i = 1:1:size(res,1)
        for j = 1:1:size(res,2)
            %num = 1;
            %val = im(i,j,k,1);
            if (smaller(i, j, k) ~= 0)
```

```
            res(i,j,k) = smaller(i, j, k);
        else
            res(i,j,k) = background(i,j,k);
    end
  end
end


end
```