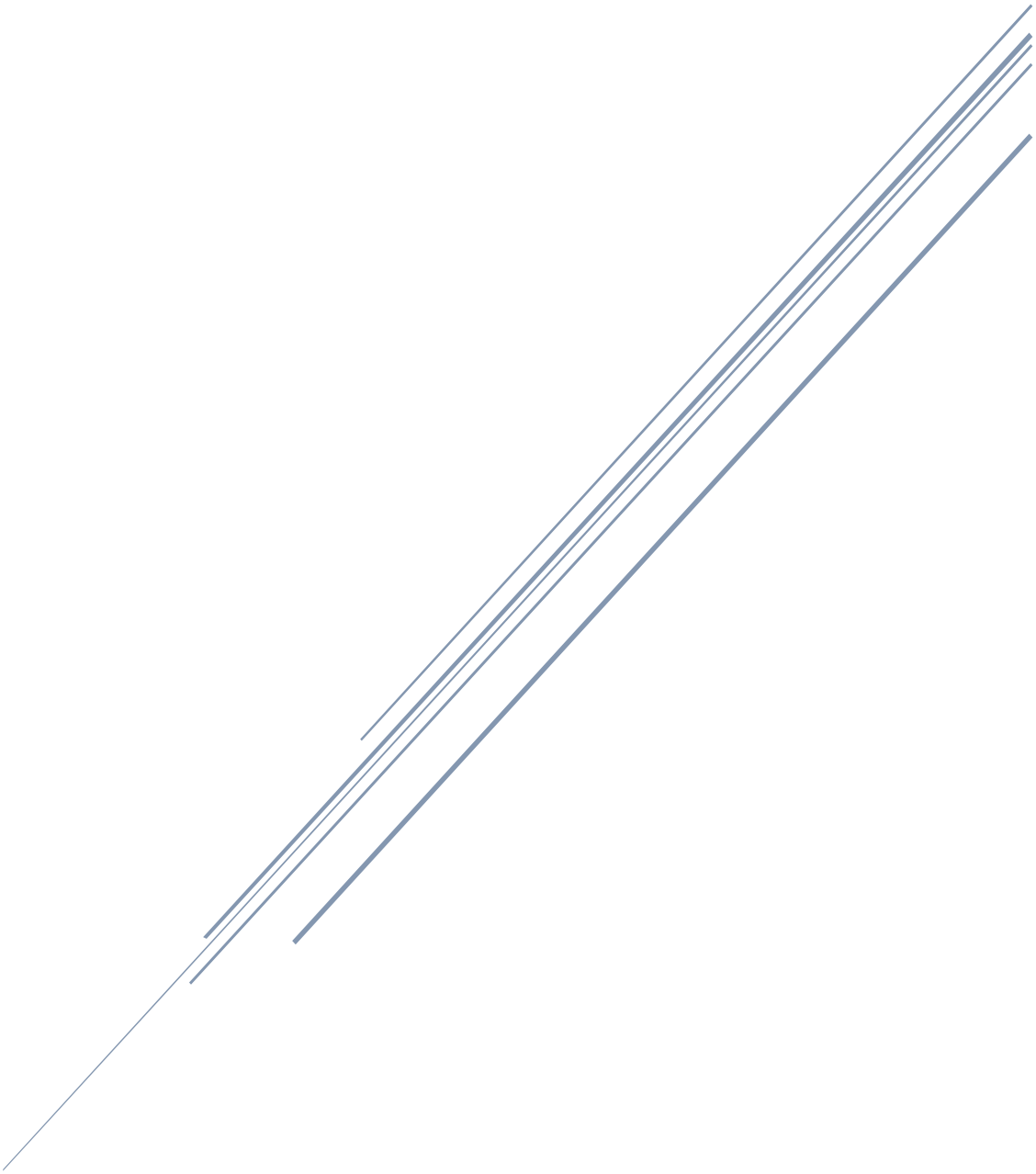
PRÁCTICA 1 – APROXIMACIÓN A LA VISITA VIRTUAL DE UNA PINACOTECA

Sistemas de Percepción – 4 GIERM



Pablo León Barriga
2021/2022

Versión avanzada

Se comienza introduciendo todos los parámetros de la cámara, calculando la matriz de parámetros intrínsecos de la misma.

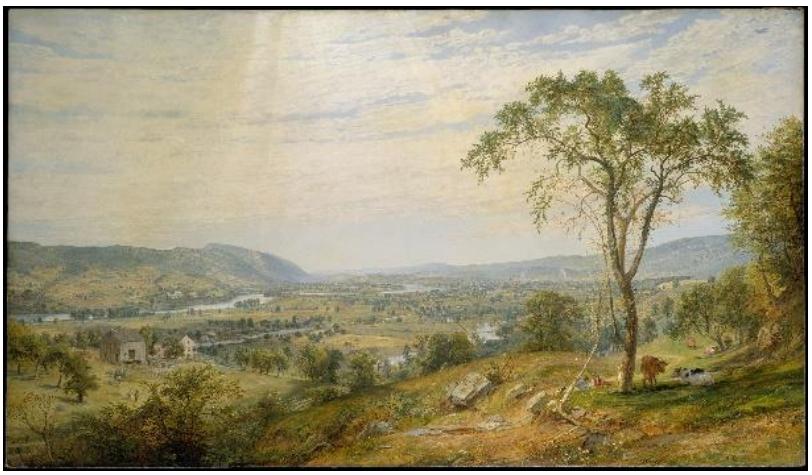
```
%% Parametros de la camara
f = 4.2e-3;                     % Distancia focal: 4.2 mm = 4.2e-3 m
N = 1500;                       % Resolucion imagen, ancho = 1500 pix
M = 1000;                       % Resolucion imagen, alto = 1000 pix
anchoSensor = 4.96e-3;          % Ancho del sensor: 4.96 mm
altoSensor = 3.52e-3;           % Alto del sensor: 3.52 mm
u0 = round(N/2);                % Punto principal de la imagen: u0
v0 = round(M/2);                % Punto principal de la imagen: v0
skew = 0;                       % Coeficiente de no perpendicularidad

% Calculo de los parametros
rho_x = anchoSensor/N;  % Dimension efectiva de pixel, x
rho_y = altoSensor/M;   % Dimension efectiva de pixel, y
fx = f/rho_x;           % Longitud focal efectiva, horizontal [pix]
fy = f/rho_y;           % Longitud focal efectiva, veral [pix]

% Matriz de parametros intrinsecos
K = [fx     skew*fx     u0
    0       fy          v0
    0       0           1 ];

% Vectores con las esquinas de la camara
pXcamara = [1 1 N N];
pYcamara = [1 M M 1];
```

Se pasa ahora a generar la información de cada uno de los cuadros. Se define el alto y ancho, y se indica el origen de coordenadas de cada cuadro, que se encuentra en el centro de estos. Se genera la información de la posición XYZ de cada una de las esquinas de los cuadros, lo que permitirá más adelante tanto mostrarla en una escena 3D como calcular la proyección de los vértices sobre el plano de la cámara. Por último, se definen los ángulos de giro de los ejes de coordenadas respecto al eje de coordenada global {W}, y se calculan todas las matrices de rotación y traslación mundo-cuadro y viceversa.

```
%% Parametros cuadro 1
alto = 1.016;     % Alto de 101'6 cms
ancho = 0.762;    % Ancho de 76'2 cms
cuadro1matriz_dim  =  [ancho/2  0;  0  alto/2];    %  Matriz  para  multiplicar  por
"matriz_esquinas" y obtener las esquinas
cuadro1origen = [1.5, 3, 1.5]; % Origen del cuadro, la altura viene indicada por la 3a
coordenada.
cuadro1X = zeros(1,5);
cuadro1Y = zeros(1,5);
cuadro1Z = zeros(1,5);

% Bucle para calcular las coordenadas 3D del cuadro. En este caso la
% coordenada Y se mantiene constante, puesto que el cuadro esta comprendido
% en el eje XZ.
matriz_esquinasCuadro1 = [-1 1; -1 -1; 1 -1; 1 1; -1 1];   % Matriz para calcular las
esquinas de los cuadros mediante el punto central y el alto/ancho
```

```
for i = 1:5
    cuadro1XZ      =      cuadro1origen(1:2:3)'      +      cuadro1matriz_dim      *
matriz_esquinasCuadro1(i,:)';
    cuadro1X(i) = cuadro1XZ(1);
    cuadro1Y(i) = cuadro1origen(2);
    cuadro1Z(i) = cuadro1XZ(2);
end
cuadro1XYZ_ = [cuadro1X; cuadro1Y; cuadro1Z];
cuadro1XYZ = cuadro1XYZ_(:,1:4);

% Parametros para el calculo de las matrices mundo-cuadro y cuadro-mundo
XYZ_cuadro1 = cuadro1origen;    % Posicion XYZ del sistema de referencia, respecto al
sistema de referencia {W}

% Angulos de giro segun convenio de ejes moviles ZYX
psi_cuadro1 = 0;    % Eje Z
theta_cuadro1 = 0;  % Eje Y
phi_cuadro1 = pi/2; % Eje X

[wtc_cuadro1,  wRc_cuadro1,  wTc_cuadro1,  cTw_cuadro1,  cRw_cuadro1,  ctw_cuadro1]  =
calculo_wRc(XYZ_cuadro1, psi_cuadro1, theta_cuadro1, phi_cuadro1);
```

Este proceso se repite para cada uno de los tres cuadros escogidos, que se muestran a continuación.



Cuadro 2 - The Valley of Wyoming, Jasper Francis Cropsey, 1865

Cuadro 1 - Still Life—Violin and Music, William Michael Harnett, 1888

Cuadro 3 - Paseo a orillas del mar, Joaquín Sorolla, 1909

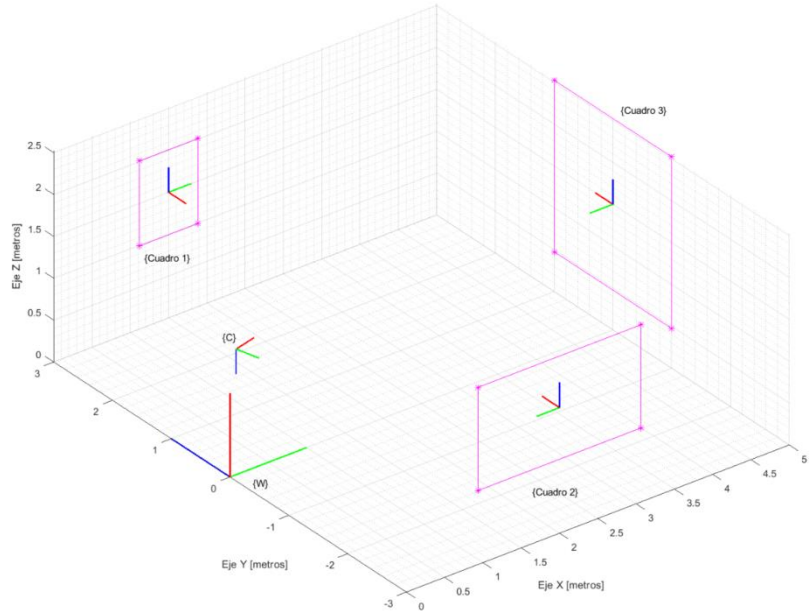
Una vez definidos todos los parámetros de los cuadros, se procede a realizar un bucle for en el que se simulará el movimiento de la cámara, haciendo que los parámetros de las coordenadas XYZ y de los ángulos varíen en función de la evolución del bucle. Para cada bucle se crea la imagen a renderizar, con tres canales de color (RGB). Se le da valor a cada parámetro de movimiento/orientación, y se calculan las matrices rotación y traslación de la misma manera que se hizo con cada uno de los cuadros.

```
for i = 1:nloop

    fRender = 155*ones(M, N, 3);    % Para cada bucle de movimiento, se inicializa la
imagen a renderizar
    tic

    %% Posicion y orientacion relativa de la camara
    % La posicion X de la camara varia segun avanza la simulacion, al igual
    % que la variable correspondiente al angulo de rotacion del eje Z, para
    % simular en movimiento circular de la camara.
    x0_cam = i/nloop * 2;
    y0_cam = 0;
    z0_cam = 1.5;
    XYZ_cam = [x0_cam y0_cam z0_cam];

    % Orientacion {C} respecto a {W}. Convenio de ejes ZYX moviles.
    psi_cam = -1/2*pi + pi/2*4 * (i / nloop);       % Ángulo de rotación respecto a eje
Z (rojo).
    theta_cam = 0.5*0;                              % Ángulo de rotación respecto a eje
Y' (azul).
    phi_cam = -pi/2;                                % Ángulo de rotación respecto a eje
X''(verde).

    [wtc_cam ,wRc_cam, wTc_cam, cTw_cam, cRw_cam, ctw_cam] = calculo_wRc(XYZ_cam,
psi_cam, theta_cam, phi_cam);
```

El siguiente paso es el de calcular la localización de cada uno de los vértices de los cuadros en el plano del sensor de la cámara.

```
%% Calculo de las esquinas de los cuadros en la imagen proyectada
    npuntos = length(cuadrosXYZ(1,:));
    wP1_ = [cuadrosXYZ; ones(1, npuntos)];          % Anadir coordenada homogenea
    pXcuadros = zeros(1, npuntos);
    pYcuadros = zeros(1, npuntos);
    for j = 1:npuntos
        p1_ = K * [cRw_cam ctw_cam] * wP1_(:,j); % Operar para obtener pixeles homogeneos
        p1 = p1_(1:2)/p1_(3);           % Dividir por la variable homogenea, obteniendo
el valor en pixeles del punto
        p1 = round(p1);                   % Redondear puesto los pixeles no pueden ser
flotantes
        pXcuadros(j) = p1(1);
        pYcuadros(j) = p1(2);
    end
    %     return
    pCuadros = [pXcuadros; pYcuadros]';
```

Una vez se tienen los valores en píxeles de cada cuadro, se pasa a dibujar cada una de las gráficas que se mostrarán. Se comienza por la gráfica 3D, que mostrará los ejes correspondientes al origen de coordenadas de cada elemento de la simulación (mundo, cámara y los tres cuadros), así como un rectángulo que representa la posición de cada cuadro.

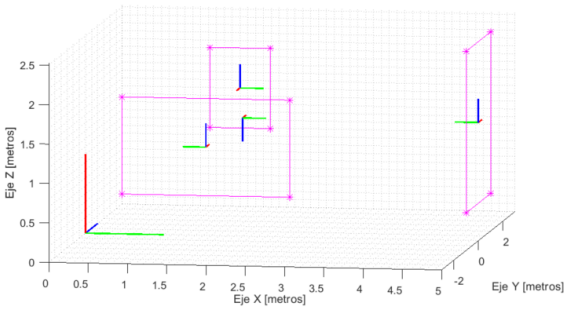


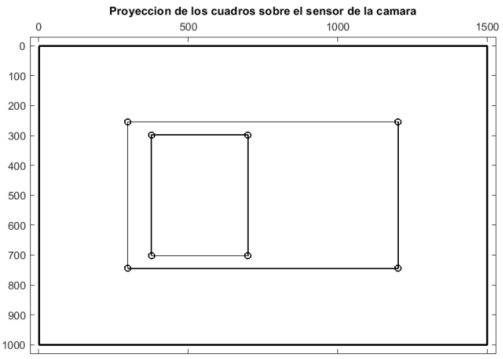

```
%% Graficas
figure(1); title("Imagen 3D");
% Cuadro 1
plot3(cuadro1origen(1), cuadro1origen(2), cuadro1origen(3), '.');
hold on
plot3(cuadro1XYZ_(1,:), cuadro1XYZ_(2,:), cuadro1XYZ_(3,:), '-*','Color','m');

% Cuadro 2
plot3(cuadro2origen(1), cuadro2origen(2), cuadro2origen(3), '.');
hold on
plot3(cuadro2XYZ_(1,:), cuadro2XYZ_(2,:), cuadro2XYZ_(3,:), '-*','Color','m');

% Cuadro 3
plot3(cuadro3origen(1), cuadro3origen(2), cuadro3origen(3), '.');
hold on
plot3(cuadro3XYZ_(1,:), cuadro3XYZ_(2,:), cuadro3XYZ_(3,:), '-*','Color','m');

axis equal
xlabel ("Eje X [metros]"); %xlim([0 8]);
ylabel ("Eje Y [metros]"); %ylim([0 6]);
zlabel ("Eje Z [metros]"); %zlim([-15 3]);

[pf_origen] = dibujo_ejes(zeros(3,1), eye(3), 1);
[pf_cam] = dibujo_ejes(wtc_cam, cRw_cam, long_ejes);
[pf_cuadro1] = dibujo_ejes(wtc_cuadro1, cRw_cuadro1, long_ejes);
[pf_cuadro2] = dibujo_ejes(wtc_cuadro2, cRw_cuadro2, long_ejes);
[pf_cuadro3] = dibujo_ejes(wtc_cuadro3, cRw_cuadro3, long_ejes);
hold off;
```

La siguiente gráfica mostrará las siluetas de los cuadros, según como se verían en la pantalla de la cámara conforme va evolucionando el movimiento de esta. Para ello, se procede a realizar una serie de cálculos que se explican a continuación.

- Detección de cuadros a un lado y otro del plano del sensor:

Realizando únicamente la operación $\begin{bmatrix} u' \\ v' \\ w' \end{bmatrix} = K \cdot [\,{}^{c}\mathrm{R}_{w} \,|{}^{c}\mathrm{t}_{w}\,] \cdot {}^{w}\tilde{\mathrm{P}}$, se obtiene las proyecciones de todos los cuadros que se encuentren a cada lado del sensor, y cuya proyección caiga sobre el mismo, como se muestra a continuación.





Por tanto, es necesario realizar una comprobación para garantizar que el cuadro que se muestra y que posteriormente se renderizará se encuentre en el lado correcto del plano del sensor, es decir, en el mismo plano al que apunta el eje Z de la cámara. Para dar solución al problema, se calcula la ecuación del plano correspondiente al sensor, ya que se tiene el vector normal al mismo (eje Z). Una vez calculado, se comprueba que el signo del valor resultante de sustituir el punto correspondiente a cada esquina de los cuadros en la ecuación del plano es igual al signo del resultado de sustituir un punto conocido a dicho lado del plano, como es el final del eje Z.

```
% Calculo del plano de la imagen para comprobar que puntos se
    % encuentran a un lado y a otro
    vectornormal = pf_cam(3,:);
    puntoperteneciente = wtc_cam;   % El punto perteneciente es el origen de coordenadas
de la camara
    Aplano = vectornormal(1);
    Bplano = vectornormal(2);
    Cplano = vectornormal(3);
    Dplano = -(Aplano * puntoperteneciente(1) + Bplano * puntoperteneciente(2) + Cplano
* puntoperteneciente(3));
    plano = [Aplano Bplano Cplano Dplano];

    % Una vez se tiene todas las variables del plano, basta con sustituir
    % los puntos de los cuadros en la ecuacion del plano, y comprobar el
    % signo con un punto conocido a ese mismo lado del plano (el final del
    % eje Z de la camara, por ejemplo).
    pFinalZ = [x0_cam+pf_cam(3,1) y0_cam+pf_cam(3,2) z0_cam+pf_cam(3,3) 1]; % Se añade
el 1 para multiplicar por Dplano
    resultado_comprobar = sign(plano * pFinalZ');

    % Una vez se tiene todas las variables del plano, basta con sustituir
    % los puntos de los cuadros en la ecuacion del plano, y comprobar el
    % signo con un punto conocido a ese mismo lado del plano (el final del
    % eje Z de la camara, por ejemplo).
    pFinalZ = [x0_cam+pf_cam(3,1) y0_cam+pf_cam(3,2) z0_cam+pf_cam(3,3) 1]; % Se añade
el 1 para multiplicar por Dplano
    resultado_comprobar = sign(plano * pFinalZ');
```

En caso de que los cuatro puntos se encuentren en el lado “correcto” del plano del sensor, se procede a llamar a la función que realiza el renderizado de la imagen, y finalmente se muestran tanto la imagen renderizada, en la que han sido copiados las intensidades de los píxeles de los cuadros originales según la operación de homografía, como la imagen “simulada” de la posición en el sensor de la cámara.

```
    % Una vez se tiene el signo de la operacion anterior, basta con
    % sustituir el punto a comprobar y verificar. En caso de que se
    % encuentre detrás del plano de la camara, no se operará.
    figure(2);
    for k = 0: npuntos/4 - 1            % Por cada cuadro
        todosdentro = 1;
        for l = 4*k + 1 : 4 * (k+1)       % Por cada punto de cada cuadro
            % Comprobar que se encuentren todos los puntos dentro
            pComprobar = wP1_(:, l);
            calculo_plano = sign(plano * pComprobar);
            if (calculo_plano ~= resultado_comprobar)    % Si se encuentran en lados
distintos del plano
                todosdentro = 0;
                break;
            end
        end
        if (todosdentro == 1)   % Solo se mostraran aquellos cuadros cuyos puntos se
encuentren en el mismo lado que el plano
            pXinputrender = pXcuadros(4*k + 1 : 4 * (k+1));
            pYinputrender = pYcuadros(4*k + 1 : 4 * (k+1));
            % Con la siguiente comprobacion unicamente se llama a
            % renderizar para aquellos cuadros que tengan algun pixel
            % dentro de la camara.
            if any(pXinputrender > 1) && any(pXinputrender < N) && any(pYinputrender >
1) && any(pYinputrender < M)
                fRender = renderizadoImagen(4, [pXinputrender; pYinputrender]', M, N,
fRender, k);
                rectangulo(pXcuadros(4*k + 1 : 4 * (k+1)), pYcuadros(4*k + 1 : 4 *
(k+1)), 1); hold on;
            end
        end
    end
    rectangulo(pXcamara, pYcamara, 0);
    % Revertir el eje y, para que comience en la esquina superior izquierda
    set(gca,'YDir', 'reverse');axis equal
    % Situar el eje x en la parte superior del plot, permite facilitar la
    % visualización de la orientación de la cámara
    set(gca,'XAxisLocation', 'top');
    xlim([-29 N+30]);
    ylim([-29 M+30]);
    title("Proyeccion de los cuadros sobre el sensor de la camara")
    hold off;



    figure(3);imshow(uint8(fRender));title("Imagen renderizada");hold off;
%     pause(.5);

    % Revertir el eje y, para que comience en la esquina superior izquierda
    % set(gca,'YDir', 'reverse');
    % Situar el eje x en la parte superior del plot, permite facilitar la
    % visualización de la orientación de la cámara
    set(gca,'XAxisLocation', 'top');
 end
```

- Renderizado de la imagen:

El renderizado de la imagen ocurre en la función “renderizadoImagen.m”, que recibe como parámetros la imagen renderizada hasta el momento, el número de vértices, el valor XY de cada uno de ellos en píxeles y el número de cuadro que se trata. La función comienza con la lectura del cuadro que se va a tratar y el cálculo de la matriz de homografía. Se calculan también los máximos y mínimos de cada eje a buscar, para evitar alargar la búsqueda innecesariamente.

```
function fRender = renderizadoImagen(nvertices, vertXY, M, N, fRender, ncuadro)
%renderizadoImagen Funcion que construye la renderizacion de la imagen

% Obtener los vectores de coordenadas X Y.
vectorXesquinas = vertXY(:,1);
vectorYesquinas = vertXY(:,2);
vectorXY = [vectorXesquinas vectorYesquinas]';

% Almacenar el cuadro del que se copiaran los valores de intensidad
imagen_a_leer = sprintf('im%d.jpg', ncuadro + 1);
fCuadroOriginal = imread(imagen_a_leer);

% Obtener los parametros del cuadro original para calcular la homografia
[M_fCuadroOriginal, N_fCuadroOriginal, C_fCuadroOriginal] = size(fCuadroOriginal);
vIN = vectorXY;
vOUT = [1 1 N_fCuadroOriginal N_fCuadroOriginal; 1 M_fCuadroOriginal M_fCuadroOriginal
1];
matriz_homografia = HomographySolve(vIN, vOUT);

% Obtener minimos y maximos en eje X e Y, para limitar la busqueda de
% puntos posibles en el interior del poligono.
xMinCuadro = max(min(vertXY(:,1)), 1);
xMaxCuadro = min(max(vertXY(:,1)), N);
yMinCuadro = max(min(vertXY(:,2)), 1);
yMaxCuadro = min(max(vertXY(:,2)), M);
```

Se procede a realizar dos bucles for anidados, que van recorriendo cada píxel de la imagen, comprobando si se encuentra dentro de los límites del trapecio formado por la proyección del cuadro en el sensor. Para comprobar si un determinado punto se encuentra dentro del trapecio formado por los cuatro vértices de la proyección, se programa en MATLAB el algoritmo encontrado en el siguiente enlace: *https://wrf.ecse.rpi.edu/Research/Short_Notes/pnpoly.html*. El algoritmo simula un rayo semi-infinito horizontal desde el punto que se está comprobando hacia la derecha, y comprueba el número de veces que atraviesa alguna arista del trapecio. Si ese número es impar, el punto se encuentra dentro del trapecio. Si resulta par, el punto no pertenece al trapecio.

```
% Bucle que recorre cada punto dentro de los limites calculados,
% comprobando si se encuentra dentro del poligono, y creando el render.
for a = yMinCuadro : yMaxCuadro
    for b = xMinCuadro : xMaxCuadro
        % Para cada punto dentro de los limites:

        dentro = -1;         % Si dentro es -1, se encuentra fuera del poligono. Si
dentro es 1, se encuentra dentro.
        coordXpunto = b;    % El punto a verificar
        coordYpunto = a;
```

```
        % El algoritmo para comprobar que un punto se encuentra de un
        % trapecio se obtiene de la siguiente web:
        % https://wrf.ecse.rpi.edu/Research/Short_Notes/pnpoly.html
        % A continuacion se ha "traducido" a matlab el codigo original C.
        j = nvertices;
        for i = 1: nvertices
            if  (  ((vectorYesquinas(i)  >  coordYpunto)  ~=  (vectorYesquinas(j)  >
coordYpunto)) &&...
                    (coordXpunto < (vectorXesquinas(j)-vectorXesquinas(i)) * ...
                    (coordYpunto-vectorYesquinas(i))      /      (vectorYesquinas(j)-
vectorYesquinas(i)) + vectorXesquinas(i)) )
                dentro = -dentro;
            end
            j = i;
        end
```

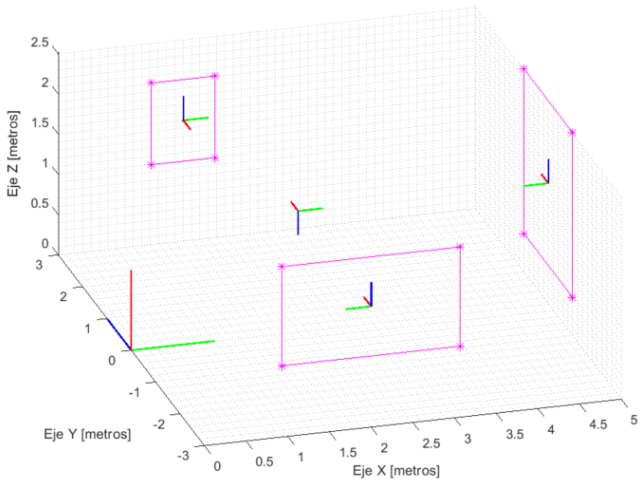
Una vez confirmado que cierto punto se encuentra dentro del trapecio, se procede a multiplicar por la matriz de homografía, eliminando la normalización de las coordenadas y consiguiendo finalmente el píxel del cuadro original del que copiar los valores de intensidad a los canales RGB de la imagen renderizada.

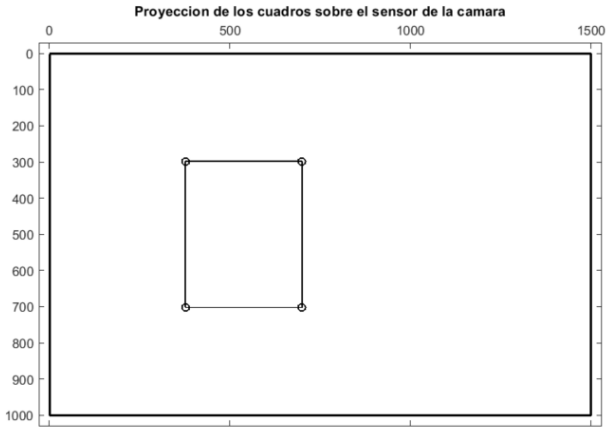
```
% Si se confirma que el pixel comprobado se encuentro dentro del
        % poligono de la imagen, se pasa a operar, realizando la operacion
        % de homografia y copiando los valores de la imagen original en la
        % vista de la camara.
        if (dentro == 1)
            for z = 1:3
                % Calcular que punto del cuadro original se corresponde con
                % el comprobado
                pCuadroOriginal = matriz_homografia * [b a 1]';
                pCuadroOriginal = pCuadroOriginal(1:2) / pCuadroOriginal(3);
                pCuadroOriginal = round(pCuadroOriginal);
                % Saturar el indice de busqueda en el cuadro original
                if pCuadroOriginal(1) > N_fCuadroOriginal
                    pCuadroOriginal(1) = N_fCuadroOriginal;
                elseif pCuadroOriginal(1) < 1
                    pCuadroOriginal(1) = 1;
                end
                if pCuadroOriginal(2) > M_fCuadroOriginal
                    pCuadroOriginal(2) = M_fCuadroOriginal;
                elseif pCuadroOriginal(2) < 1
                    pCuadroOriginal(2) = 1;
                end
                % Copiar los valores del cuadro original al pixel de la
                % imagen de la camara
                fRender(a,     b,     z)     =     fCuadroOriginal(pCuadroOriginal(2),
pCuadroOriginal(1), z);
            end
        end
    end
end
end
```

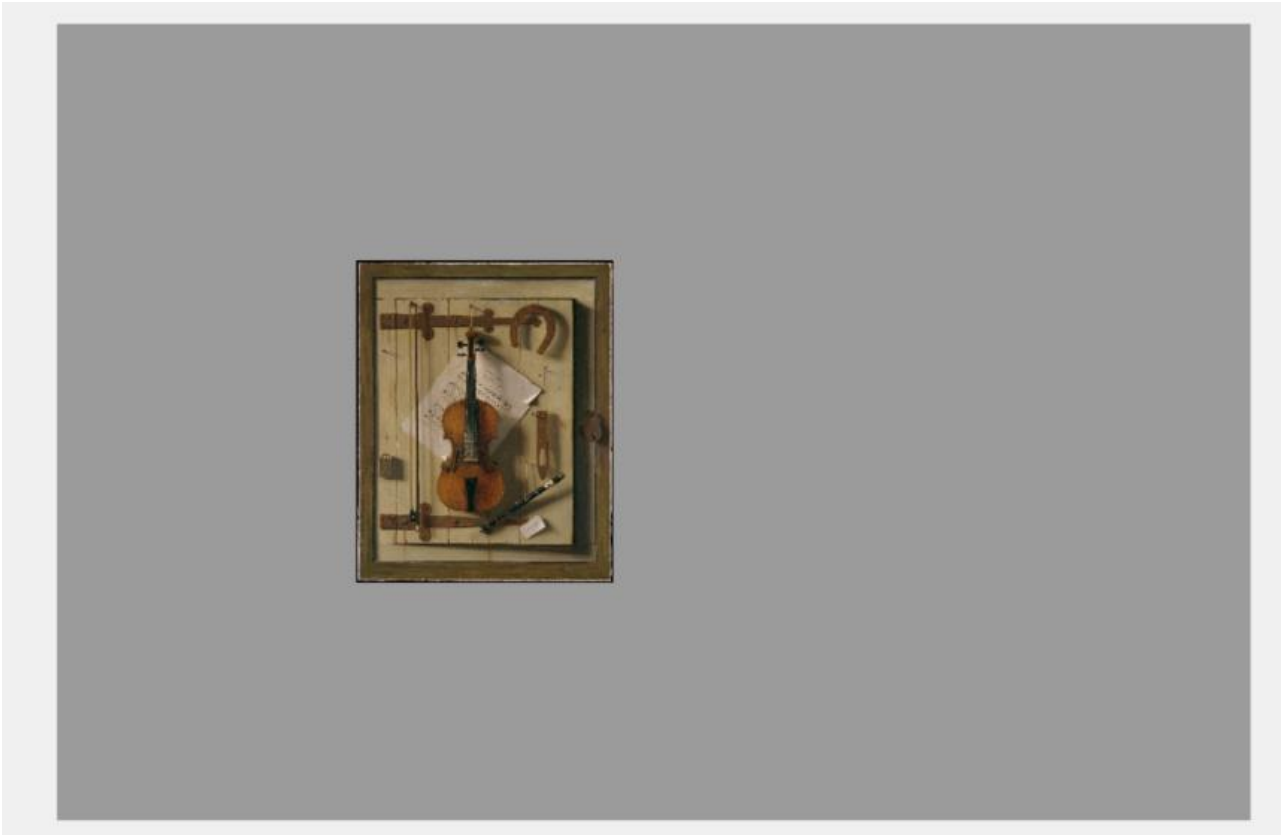
Resultado final

A continuación, se muestran algunas capturas de los resultados que devuelve el programa.

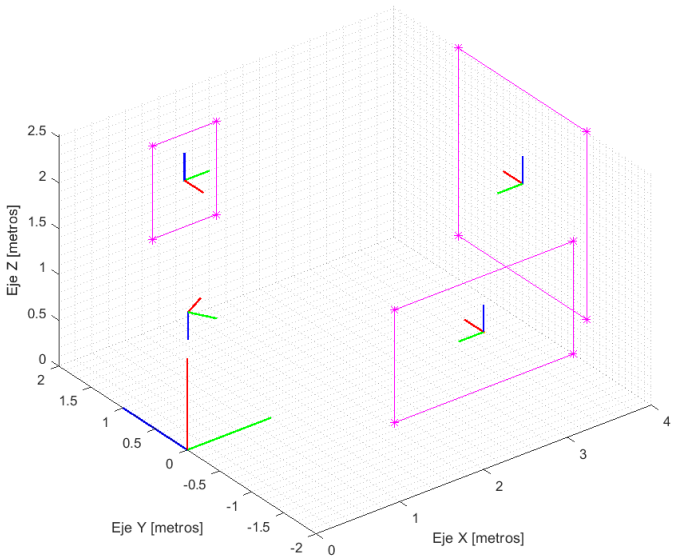
1) Cámara frente a cuadro 1, con cuadro 2 detrás.

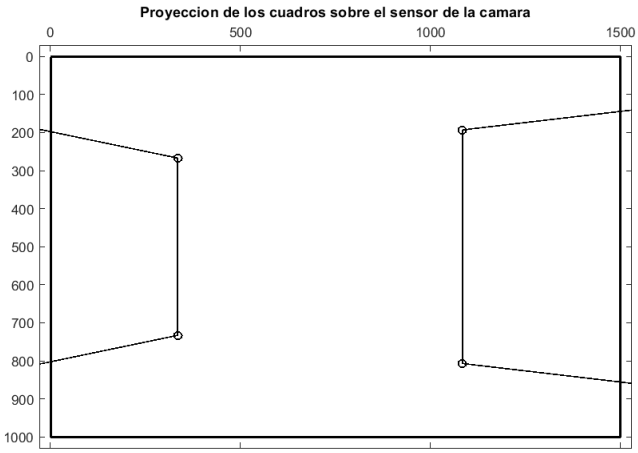


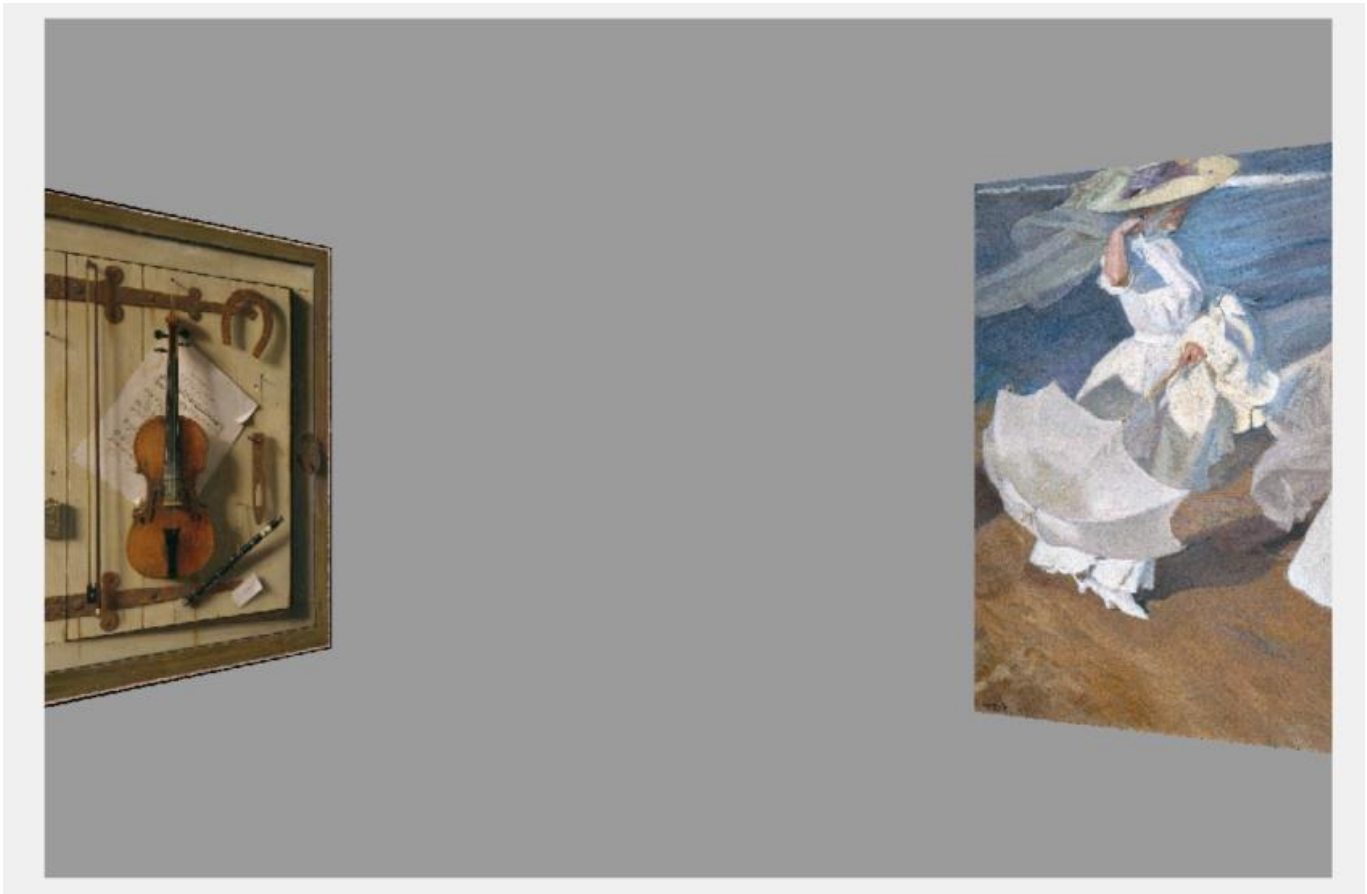




2) Cámara con vista simultánea de cuadros 1 y 3







3) Cámara con vista del cuadro 2 y 3 simultáneamente