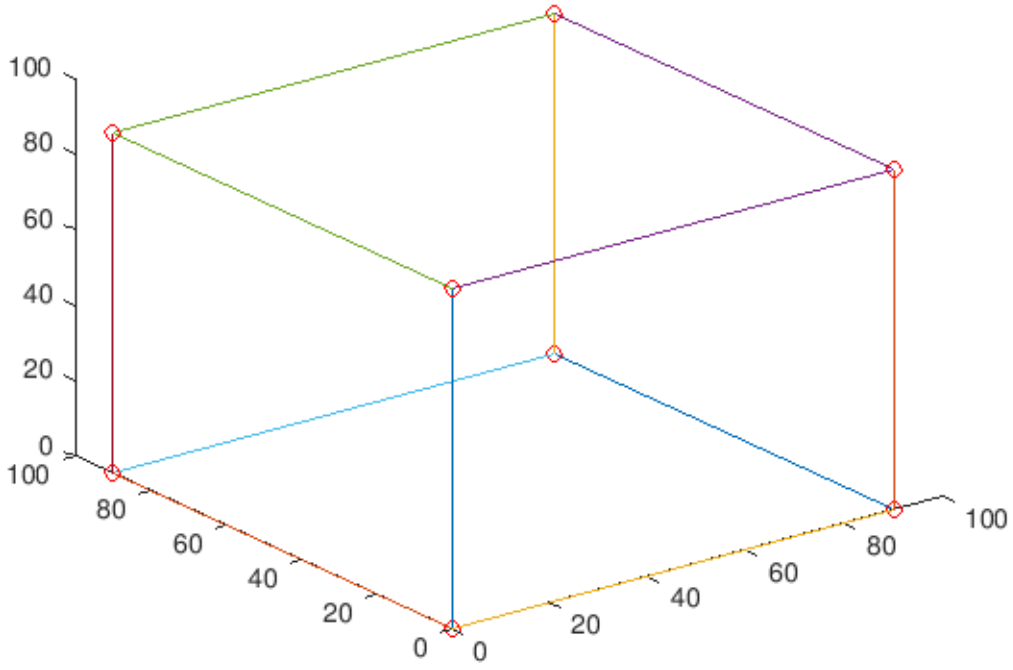
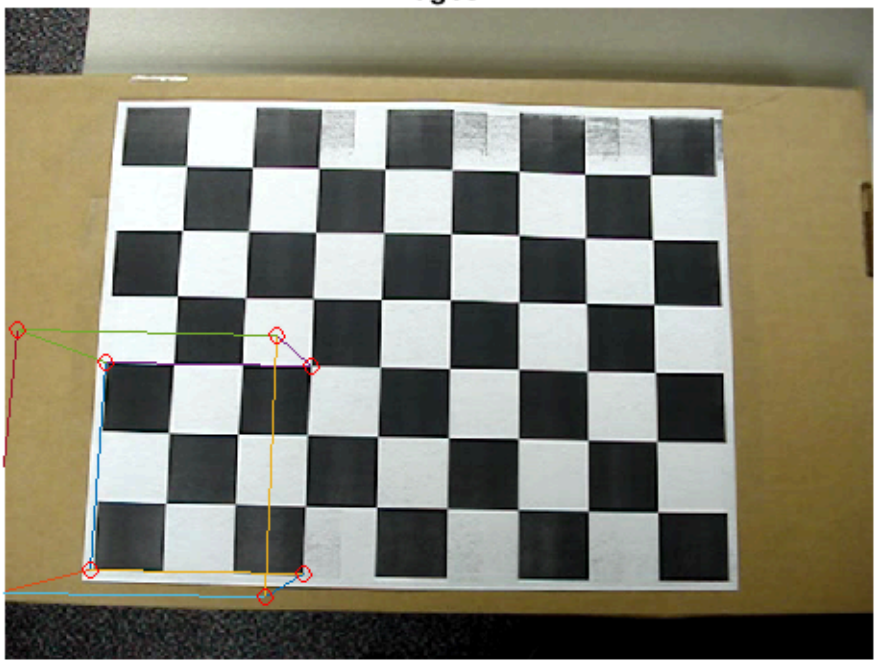
```
%Our first task is to render our cube as a wireframe and have it's
 base
%located at the bottom left corner of our grid
%Our cube will have 8 corners as [X,Y,Z,1]:
% * 4 where 1 at origin and 3 at the extreme of each axis
%       (0,0,0) -- (X,0,0) -- (0,Y,0) -- (0,0,Z)
cube = [0,0,0,1; 90,0,0,1; 0,90,0,1; 0,0,90,1];
% * 4 other corners with combination of values of X,Y,Z
%       (0,Y,Z) -- (X,0,Z) -- (X,Y,0) -- (X,Y,Z)
cube = [cube; 0,90,90,1; 90,0,90,1; 90,90,0,1; 90,90,90,1];

%We take the reverse order of the points as our plotting would be in
%matlab would be in reverse order
%gets all the X coordinates in order of creation
cube_3D(:,1) = cube(:,3);
%gets all the Y coordinates in order of creation
cube_3D(:,2) = cube(:,2);
%gets all the Z coordinates in order of creation
cube_3D(:,3) = cube(:,1);
%gets all last 1's
cube_3D(:,4) = cube(:,4);

%Define the 12 cube edges.
cubeEdges = [cube_3D(1,1:3) cube_3D(2,1:3);
             cube_3D(1,1:3) cube_3D(3,1:3);
             cube_3D(1,1:3) cube_3D(4,1:3);
             cube_3D(2,1:3) cube_3D(6,1:3);
             cube_3D(2,1:3) cube_3D(7,1:3);
             cube_3D(3,1:3) cube_3D(5,1:3);
             cube_3D(3,1:3) cube_3D(7,1:3);
             cube_3D(4,1:3) cube_3D(5,1:3);
             cube_3D(4,1:3) cube_3D(6,1:3);
             cube_3D(8,1:3) cube_3D(5,1:3);
             cube_3D(8,1:3) cube_3D(6,1:3);
             cube_3D(8,1:3) cube_3D(7,1:3);];

%Obtain the edges on the X plane of the 3D cube
edges_Xplane = [cubeEdges(:,1),cubeEdges(:,4)]';
%Obtain the edges on the X plane of the 3D cube
edges_Yplane = [cubeEdges(:,2),cubeEdges(:,5)]';
%Obtain the edges on the X plane of the 3D cube
edges_Zplane = [cubeEdges(:,3),cubeEdges(:,6)]';

figure(), plot3(cube_3D(:,1), cube_3D(:,2), cube_3D(:,3), 'ro')
hold on
plot3(edges_Xplane,edges_Yplane,edges_Zplane);
hold off

for k = 1:4
    img = eval(['img' num2str(k)]);
    H = eval(['Hnew' num2str(k)]);
    %Get the projection defined as A*[R t]
```

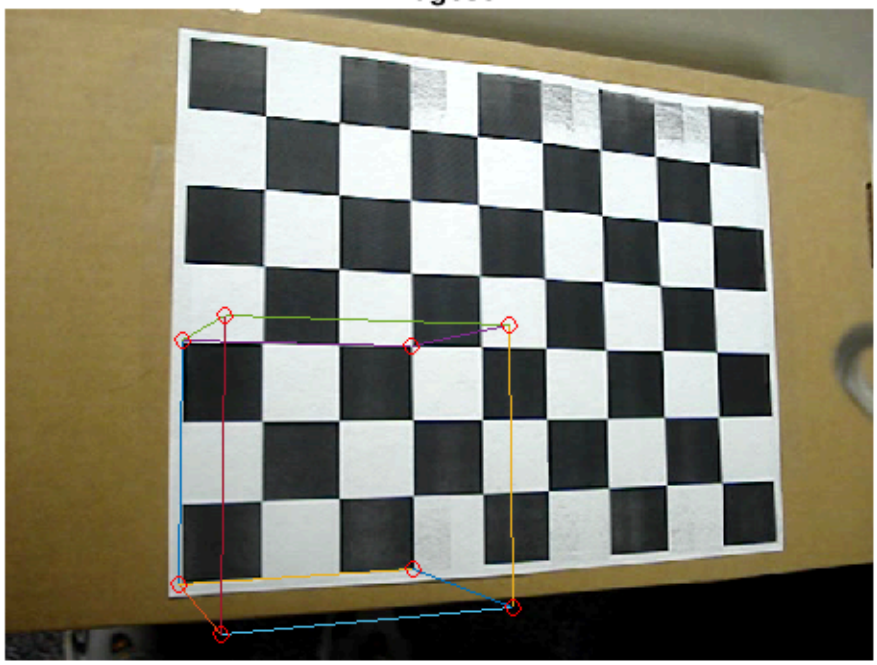
```
    Projection = A*[Rotation(:,:,k), t(:,k)];
    %We use the above computed projection to get the pixel projections
 from
    %the cube points
    pixels = Projection*cube_3D';
    pixels = pixels';
    % Normalize by homogenious coordinate
    for i=1:length(pixels)
        pixels(i,:) = pixels(i,:)/pixels(i,3);
    end

    %Define the 12 cube edges the same as the original 3D cube.
    pixelEdges = [pixels(1,1:3) pixels(2,1:3);
             pixels(1,1:3) pixels(3,1:3);
             pixels(1,1:3) pixels(4,1:3);
             pixels(2,1:3) pixels(6,1:3);
             pixels(2,1:3) pixels(7,1:3);
             pixels(3,1:3) pixels(5,1:3);
             pixels(3,1:3) pixels(7,1:3);
             pixels(4,1:3) pixels(5,1:3);
             pixels(6,1:3) pixels(6,1:3);
             pixels(8,1:3) pixels(5,1:3);
             pixels(8,1:3) pixels(6,1:3);
             pixels(8,1:3) pixels(7,1:3);];
    %Obtain the edges on the X plane of the 3D cube
    projection_edges_X = [pixelEdges(:,1),pixelEdges(:,4)]';
    %Obtain the edges on the Y plane of the 3D cube
    projection_edges_Y = [pixelEdges(:,2),pixelEdges(:,5)]';
    %Obtain the edges on the Z plane of the 3D cube

    figure, imshow(img)
    hold on
    plot(projection_edges_X, projection_edges_Y)
    plot(pixels(:,1), pixels(:,2), 'ro')
    title(['3D cubic grid projected onto images' files(k)])
    hold off
end
```



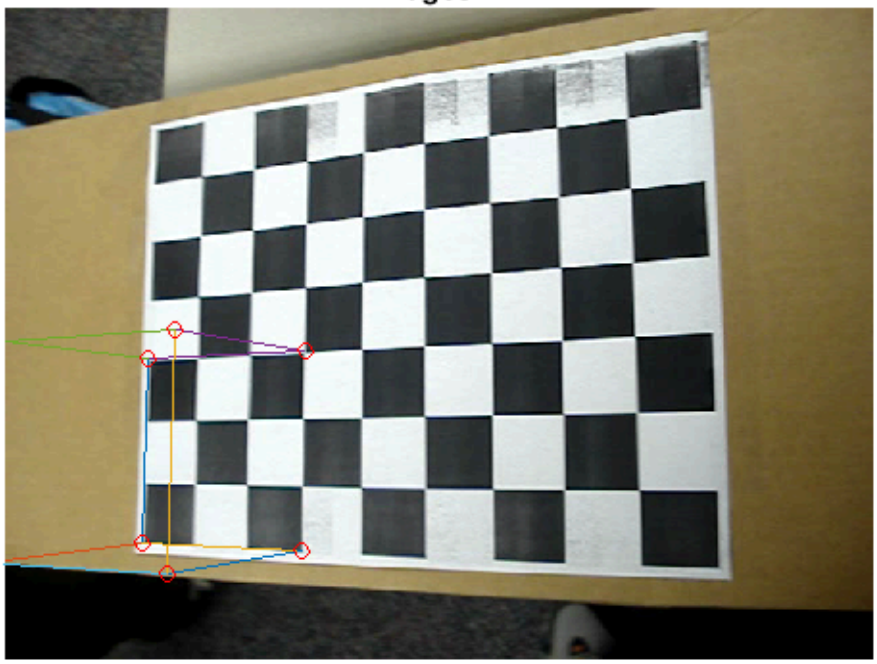
3D cubic grid projected onto images images2

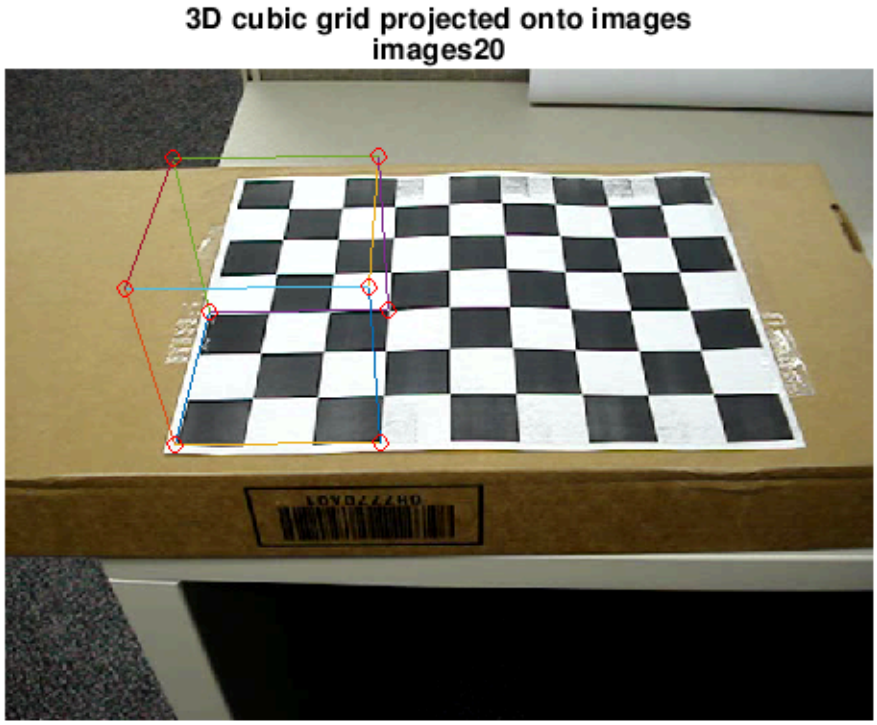


3D cubic grid projected onto images
images9



3D cubic grid projected onto images
images12





Published with MATLAB® R2019a