Rectification of an amplitude image using unit vectors

Table of Contents

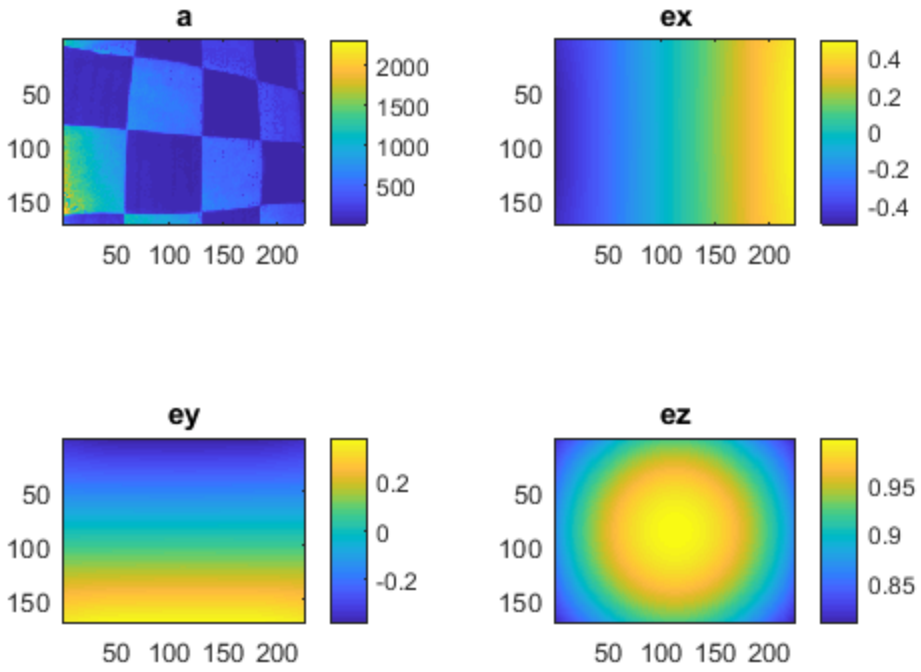
This Matlab code describes how a rectified image is calculated based on the unit vectors (line of sight vectors of each pixel). The process of rectification is seen as filling up a new matrix with amplitude values of an "old" matrix, when only the pixel position has changed. To do this, a set of indices recIdx have to be calculated, that describe which new pixel index corresponds to which old pixel index. Even though we are handling matrices, the indexing array treats the matrix as liniear arrays.

Start empty

```
clear;
clc;
```

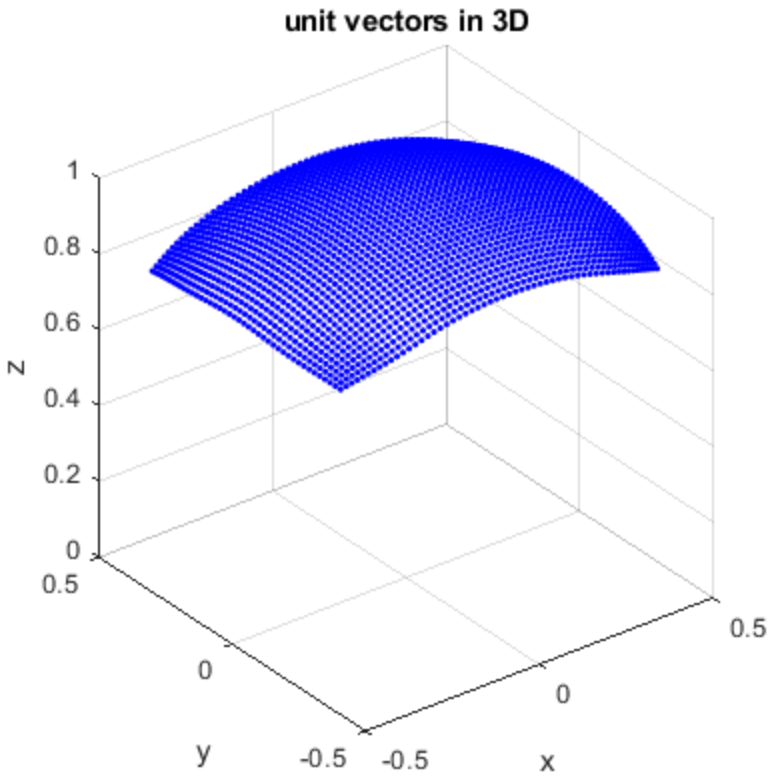
load some sample data, an amplitude image and unit vector matrices

```
load('sampleDat.mat');
%   a         172x224            308224  double  % raw image amplitude
 values
%   ex        172x224            308224  double  % x-coordinates of
 unit-vector
%   ey        172x224            308224  double  % x-coordinates of
 unit-vector
%   ez        172x224            308224  double  % x-coordinates of
 unit-vector

figure(1); clf;
subplot(221); imagesc(a);  daspect([1 1 1]); title('a'); colorbar;
subplot(222); imagesc(ex); daspect([1 1 1]); title('ex'); colorbar;
subplot(223); imagesc(ey); daspect([1 1 1]); title('ey'); colorbar;
subplot(224); imagesc(ez); daspect([1 1 1]); title('ez'); colorbar;

Nx=size(a,2); % size of image along x
Ny=size(a,1); % size of image along y
```



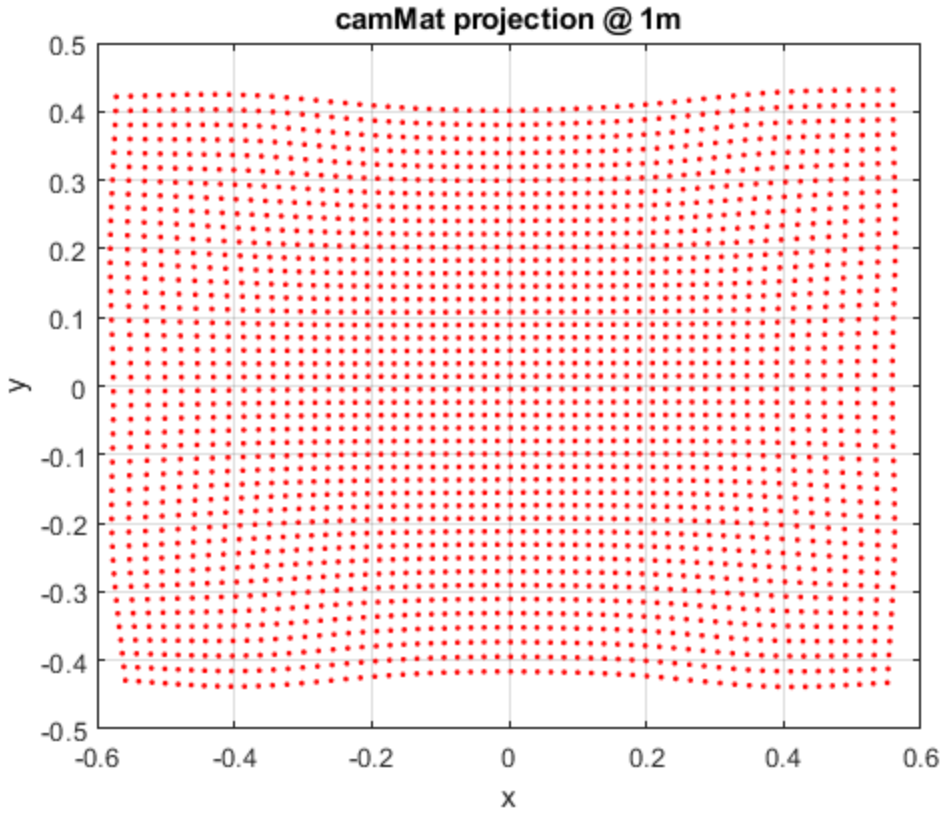
plot of unit vectors

```
figure(2); clf;
plot3(ex(1:4:end,1:4:end),ey(1:4:end,1:4:end),ez(1:4:end,1:4:end),'b.');
grid on; daspect([1 1 1]); zlim([0 1]); title('unit vectors in 3D');
 xlabel('x');ylabel('y');zlabel('z');
```



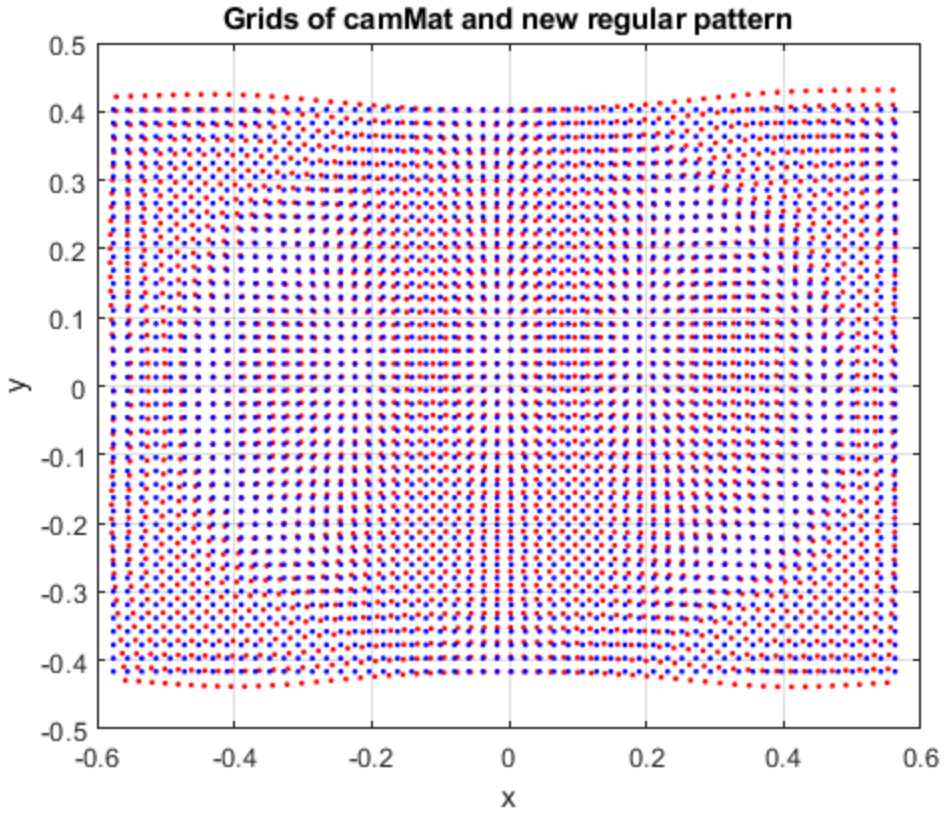
Project vectors on z=1 plane

```
%All unit vectors get stretched by 1/ez, so that the new z-component
 is 1.
%camPos are the new x and y components of these stretched vectors.
camPos=[ ey(:)./ez(:), ex(:)./ez(:)]';
camMat=reshape(camPos,[2,172,224]); % reshape camPos back into matrix
 form

figure(3); clf;
plot(squeeze(camMat(2,1:4:end,1:4:end)),squeeze(camMat(1,1:4:end,1:4:end)),'r.');
grid on; daspect([1 1 1]); title('camMat projection @ 1m');
 xlabel('x');ylabel('y');
```



Build a regularly spaced pattern around the projected and irregularly spaced points of camPos

```
%Create a regularly spaced pattern out of camMat. This will be the
 grid of the rectified picture.
[mx,my] =
 meshgrid(linspace(min(camMat(2,86,:)),max(camMat(2,86,:)),Nx),...
    linspace(min(camMat(1,:,112)),max(camMat(1,:,112)),Ny));

figure(4); clf;
plot(squeeze(camMat(2,1:4:end,1:4:end)),squeeze(camMat(1,1:4:end,1:4:end)),'r.');
hold on;
plot(mx(1:4:end,1:4:end),my(1:4:end,1:4:end),'b.'); grid on;
 daspect([1 1 1]);
title('Grids of camMat and new regular pattern');
 xlabel('x');ylabel('y');
```



calculate indizes recIdx of a simple nearest neighbour rectification only once

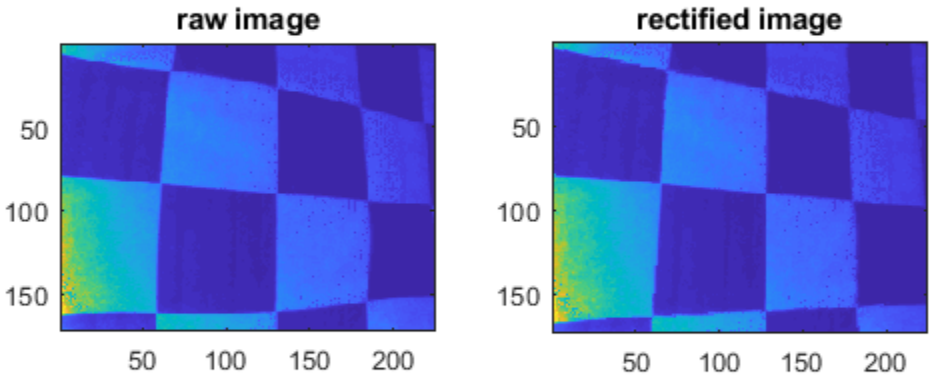
```
recIdx=ones(Ny,Nx); % create new image which will become the rectified
 picture.
searchdist=6; % Half length of the square in which the nearest lying
 point of camMat will be searched.
for idx =1:Nx*Ny

    cent=[mx(idx) my(idx)]; % Current point around which the nearest
 lying neighbor of camMat will be searched.

    sidxarray=[]; % creates empty search index array in which the
 search indices will be saved.
    distarray=[]; % creates empty distance array in which the
 distances between the current point and the search points will be
 saved.
    sidx=idx-searchdist*Ny-searchdist; % Searchindex. Start searching
 searchdist left of current point and searchdist below current point
 (Search then continues up and right of that point (compare Fig.2))

    if sidx <= 0 % Necessary for the first few indices to avoid
 negative indices
        sidx=idx-searchdist;
```

```
        if sidx <= 0
            sidx=idx;
        end
    end

    for g = 1:2*searchdist % This loops through all the points within
 a square of size (2*searchdist)^2 and writes down the search index
 and the corresponding distance of that point to the current center
 point.

        for h=1:2*searchdist

            sidxarray= [sidxarray sidx]; % write down current search
 index
            distarray= [distarray sqrt((mx(idx)-camMat(2,sidx))^2+
(my(idx)-camMat(1,sidx))^2)]; % write down current distance to cent.
            sidx=sidx+1;
            if sidx > Nx*Ny
                break
            end

        end

        sidx=sidx+Ny-2*searchdist; % Jumps one column right (+Ny) and
 starts searching again from the bottom of the square (-2*searchdist)

        if sidx > Nx*Ny % Necessary when the end of the Matrix is
 reached
            break
        end

    end

    [M,I]= min(distarray(:)); % Find the minimum and the corresponding
 search index of distarray
    sidx=sidxarray(I); % This index has the minimal distance to cent

    recIdx(idx)=sidx; % Write the amplitude value of original picture
 at index sidx into rectified picture at current position idx.

end
```

Rectify Image

```
recIm=ones(Ny,Nx); % create an empty image
recIm=a(recIdx);   % calculate the new rectified Image using the
 Indizes previously caluclated

figure(5); clf;
subplot(121); imagesc(a); daspect([1 1 1]); title('raw image');
subplot(122); imagesc(recIm); daspect([1 1 1]); title('rectified
 image');
```

```
%publish('rectExample_annotated','pdf');
```



Published with MATLAB® R2017b