Sample 6-3

標本化

二変量ダウンサンプリング

画像処理特論

村松 正吾

動作確認: MATLAB R2020a

Sampling

Bivariate downsampling

Advanced Topics in Image Processing

Shogo MURAMATSU

Verified: MATLAB R2020a

準備

(Preparation)

```
close all
```

可分離間引き行列の設定

(Setting a separable downsampling factor)

- $\mathbf{M}$: 間引き行列 (downsampling factor)

可分離の場合 (In the separable case)

$$\mathbf{M} = \begin{pmatrix} M_1 & 0 \\ 0 & M_2 \end{pmatrix} \in \mathbb{Z}^{2\times 2}$$

```
% Vertical downsampling ratio
verticalDFactor = 2;
% Horizontal downsampling factor
horizontalDFactor = 3;
```

入力配列の設定

(Setting an input array)

- $\{u[\mathbf{n}]\}_{\mathbf{n}}$: 入力配列 (input array)

```
% Standard deviation
sigma = 2;
```

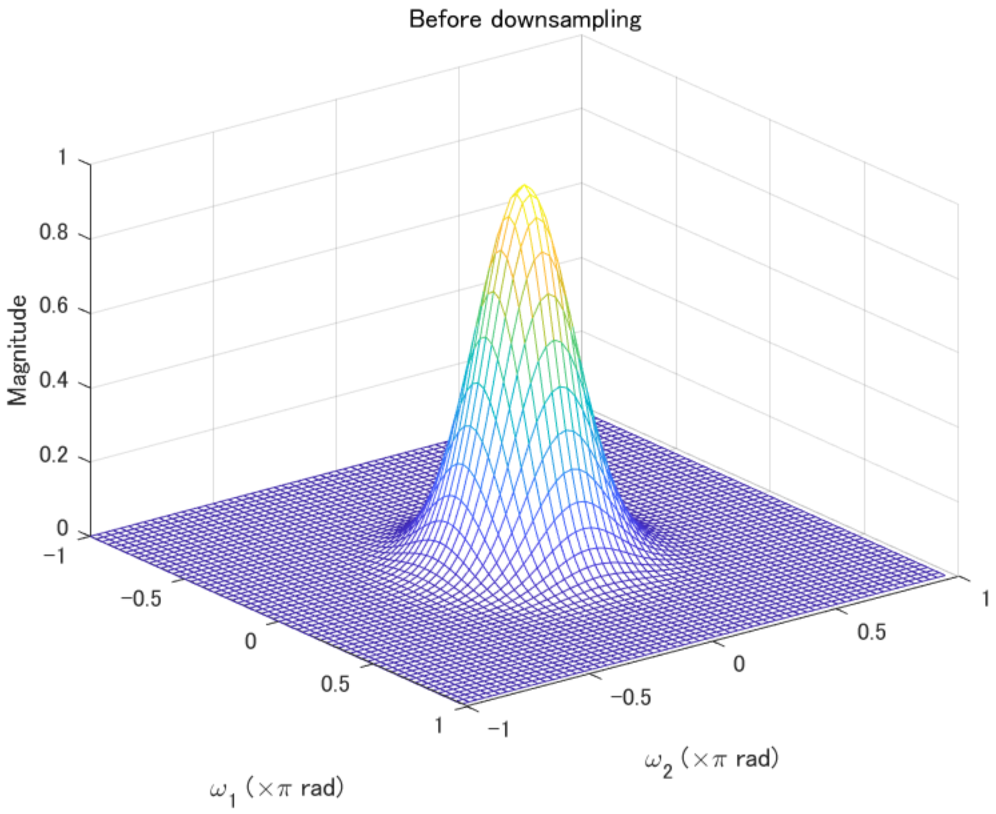
```
% Array size
sizeU = 31;

% Generate an array with bivariate Gaussian function
arrayU = fspecial('gaussian',sizeU,sigma);
```

入力配列のスペクトル

(Spectrum of the input array)

```
% Spectrum of u[n]
figure(1)
freqz2(arrayU)
xlabel('\omega_2 (\times\pi rad)')
ylabel('\omega_1 (\times\pi rad)')
title('Before downsampling')
axis ij
```



出力配列の計算

(Computation of the output array)

- $\{v[\mathbf{m}]\}_{\mathbf{m}}$: 出力数列 (output array)

$$v[\mathbf{m}] = u[\mathbf{M}\mathbf{m}],\ \ \mathbf{m} \in \mathbb{Z}^2$$

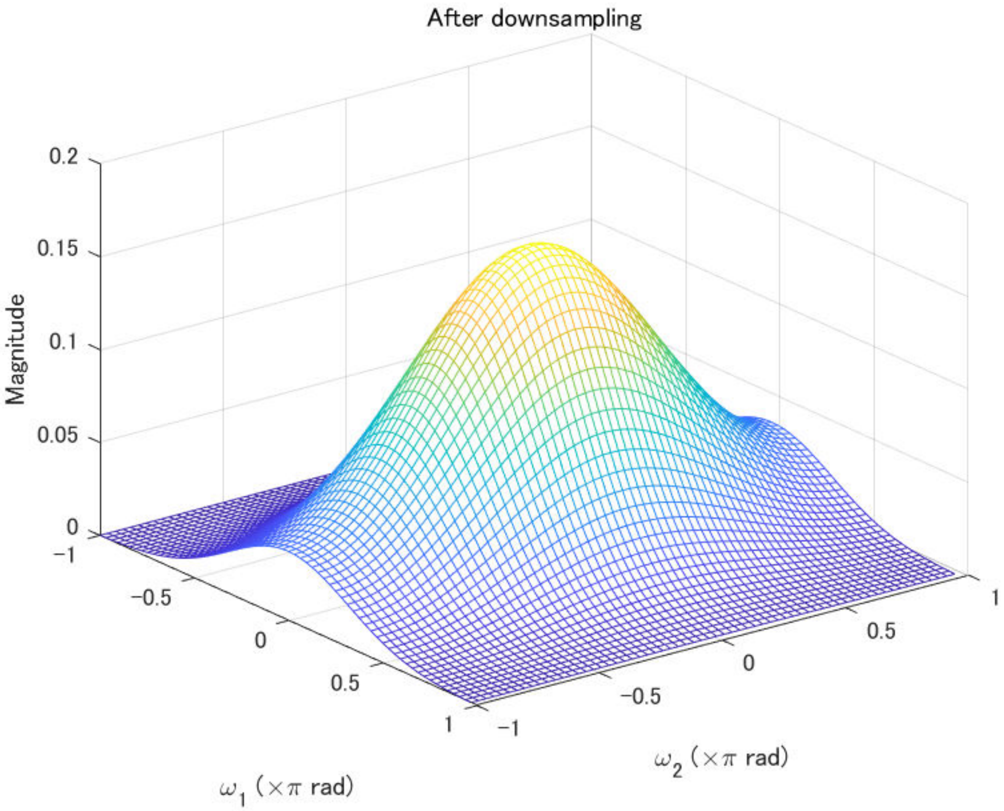
```
% Definition of bivariate separable downsampling
downsample2 = @(x,n) ...
    shiftdim(downsample(...
    shiftdim(downsample(x,...
    n(1)),1),...
    n(2)),1);

% Bivariate separable downsampling
arrayV = downsample2(arrayU, [verticalDFactor horizontalDFactor]);
```

出力配列のスペクトル

(Spectrum of the output array)

```
% Spectrum of v[m]
figure(2)
freqz2(arrayV)
xlabel('\omega_2 (\times\pi rad)')
ylabel('\omega_1 (\times\pi rad)')
title('After downsampling')
axis ij
```



非可分間引き行列の設定

(Setting a non-separable downsampling factor)

- $\mathbf{M}$: 間引き行列 (downsampling factor)

一般的な場合 (In the general case)

$\mathbf{M} = \begin{pmatrix} \mathbf{m}_1 & \mathbf{m}_2 \end{pmatrix} \in \mathbb{Z}^{2\times 2}$

```
% Downsampling factor
downMtx = [ 1 1 ; -1 1 ];
```

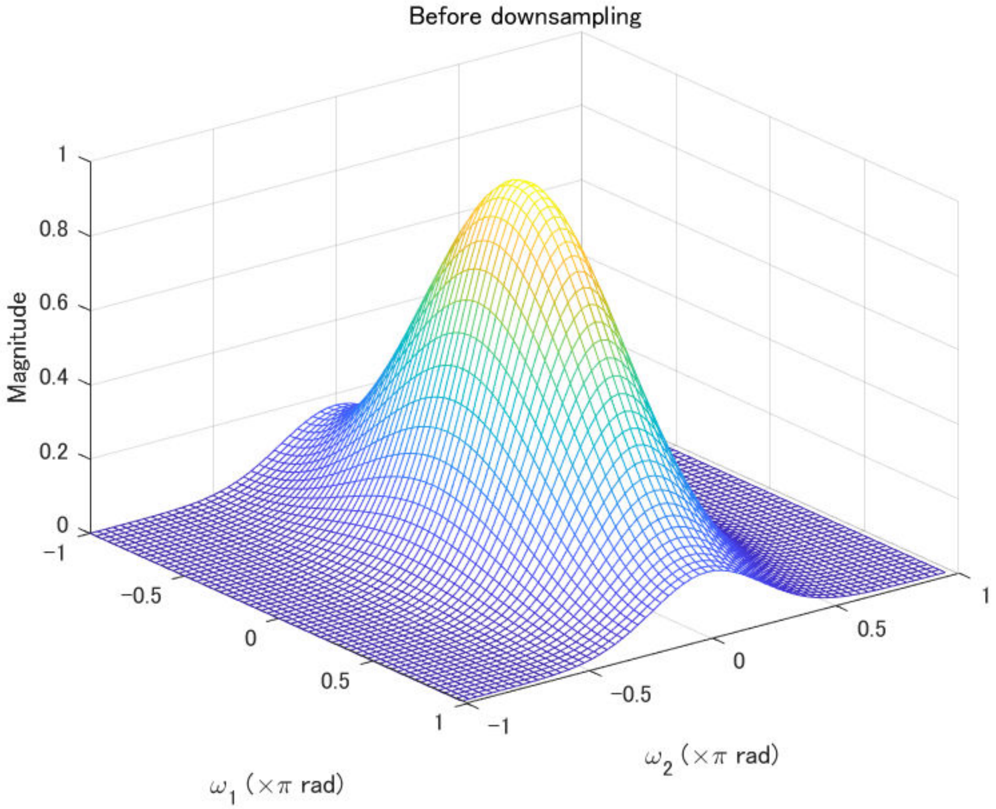
入力配列の設定

(Setting an input array)

```
% Covariance matrix
covMtx  = [2 0 ; 0 1/2];
% Array size
sizeU = 31;

% Generate an array with bivariate Gaussian function
arrayU = mygaussian2(sizeU,covMtx);

% Spectrum of u[n]
figure(3)
freqz2(arrayU)
xlabel('\omega_2 (\times\pi rad)')
ylabel('\omega_1 (\times\pi rad)')
title('Before downsampling')
axis ij
```

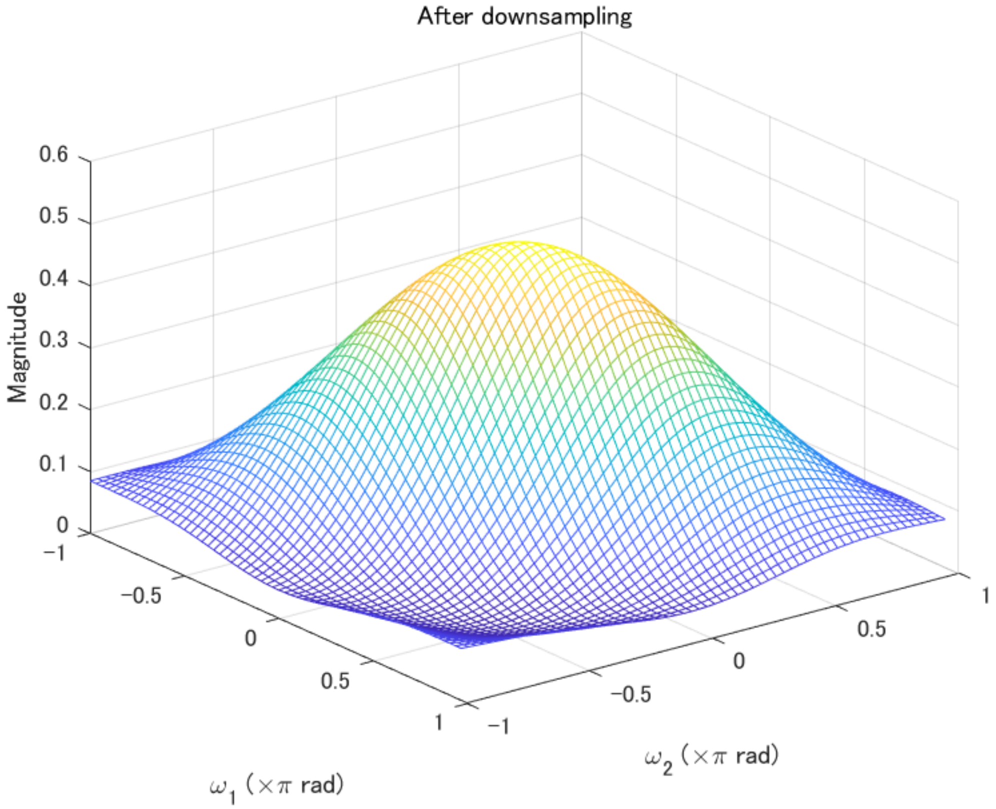


出力配列の計算

(Computation of the output array)

```
% Bivariate non-separable downsampling
arrayV = mydownsample2(arrayU,downMtx);

% Spectrum of v[m]
figure(4)
freqz2(arrayV)
xlabel('\omega_2 (\times\pi rad)')
ylabel('\omega_1 (\times\pi rad)')
title('After downsampling')
axis ij
```



プライベート関数の定義

(Definitions of private functions)

二変量ガウス配列の生成関数　(Generation function of bivariate Gaussian arrays)

```
function x = mygaussian2(sizeX, covMtx)
% MYGAUSSIAN2
%
% Inputs
%
%   sizeX: size of output array
%   covMtx: covariance matrix
%
% Output
%
%   x: array generated by Gauss Func.
%
sizeHalf = floor(sizeX / 2);

if isscalar(covMtx)
    covMtx = covMtx * eye(2);
end

x = zeros(sizeHalf*2);
iRow = 1;
```

```
for n1 = -sizeHalf:sizeHalf
    iCol = 1;
    for n0 = -sizeHalf:sizeHalf
        p = - 0.5 * [ n0 n1 ] * (covMtx\[ n0 n1 ].');
        x(iRow,iCol) = exp(p);
        iCol = iCol + 1;
    end
    iRow = iRow + 1;
end
x = x / sum(sum(x));

end % of mygaussian2
```

二変量非可分ダウンサンプリング関数 (bivariate non-separable downsampling function)

```
function [outputArray, X, Y] = mydownsample2(inputArray,downMtx,phase)
% MYDOWNSAMPLE2
%
% Inputs
%
%   inputArray: input array
%   downMtx:    downsampling matrix
%   phase:      downsampling phase
%
% Outputs
%
%   outputArray: output array
%   X          : horizontal sampling points
%   Y          : vertical sampling points
%
if nargin < 3
    phase = [0 0].';
else
    phase = phase(:);
end

% Size of input array
nRowsInputArray = size(inputArray,1);
nColsInputArray = size(inputArray,2);
nCompInputArray = size(inputArray,3);

% Calculation of the support region after downsampling
vertexPoints(:,1) = [ 0 0 ].';
vertexPoints(:,2) = downMtx \ [ 0 nColsInputArray-1 ].';
vertexPoints(:,3) = downMtx \ [ nRowsInputArray-1 0 ].';
vertexPoints(:,4) = downMtx \ [ nRowsInputArray-1 nColsInputArray-1 ].';
minPoint = floor(min(vertexPoints,[],2));
maxPoint = ceil(max(vertexPoints,[],2));

% Shift the downsampling phase into FPD(downMtx)
```

```
phase = phase - downMtx * floor(downMtx \ phase);

% Downsampling
clear arrayY
iRow = 1;
for m0 = minPoint(1):maxPoint(1)
    iCol = 1;
    for m1 = minPoint(2):maxPoint(2)
        originalPoint = downMtx * [ m0 m1 ].';
        n0 = originalPoint(1) + phase(1);
        n1 = originalPoint(2) + phase(2);
        if n0 >= 0 && n0 < nRowsInputArray  && ...
                n1 >= 0 && n1< nColsInputArray
            outputArray(iRow, iCol, :) = inputArray(n0 + 1, n1 + 1, :);
        else
            outputArray(iRow, iCol, :) = zeros(1,nCompInputArray,...
                'like',inputArray);
        end
       iCol = iCol + 1;
    end
    iRow = iRow + 1;
end

Y = minPoint(1):maxPoint(1);
X = minPoint(2):maxPoint(2);

end % of mydownsample2
```