

【2次元離散コサイン変換 2D Discrete Cosine Transform】

■ 2D DCT

$$F(u, v) = \frac{2C(u)C(v)}{\sqrt{N_x N_y}} \sum_{x=0}^{N_x-1} \sum_{y=0}^{N_y-1} f(x, y) \cos\left(\frac{(2x+1)u}{2N_x} \pi\right) \cos\left(\frac{(2y+1)v}{2N_y} \pi\right)$$

$$C(u), C(v) = \begin{cases} \frac{1}{\sqrt{2}} & (u, v = 0) \\ 1 & (u, v \neq 0) \end{cases}$$

Example)

$$N_x = N_y = 2$$

$f(x, y) : f(0, 0) = 200, f(0, 1) = 150, f(1, 0) = 100, f(1, 1) = 70$ とする.

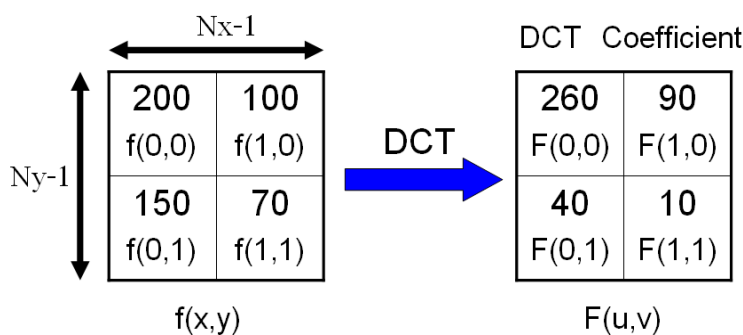
$$F(u, v) = C(u)C(v) \sum_{x=0}^1 \sum_{y=0}^1 f(x, y) \cos\left(\frac{(2x+1)u}{4} \pi\right) \cos\left(\frac{(2y+1)v}{4} \pi\right)$$

$$F(0, 0) = \frac{1}{2} \sum_{x=0}^1 \sum_{y=0}^1 f(x, y) = \frac{1}{2}(200 + 100 + 150 + 70) = 260$$

$$\begin{aligned} F(0, 1) &= \frac{1}{\sqrt{2}} \sum_{x=0}^1 \sum_{y=0}^1 f(x, y) \cos\left(\frac{(2y+1)}{4} \pi\right) \\ &= \frac{1}{\sqrt{2}} [f(0, 0) \cos\left(\frac{\pi}{4}\right) + f(0, 1) \cos\left(\frac{3\pi}{4}\right) + f(1, 0) \cos\left(\frac{\pi}{4}\right) + f(1, 1) \cos\left(\frac{3\pi}{4}\right)] = 40 \end{aligned}$$

$$\begin{aligned} F(1, 0) &= \frac{1}{\sqrt{2}} \sum_{x=0}^1 \sum_{y=0}^1 f(x, y) \cos\left(\frac{(2x+1)}{4} \pi\right) \\ &= \frac{1}{\sqrt{2}} [f(0, 0) \cos\left(\frac{\pi}{4}\right) + f(0, 1) \cos\left(\frac{\pi}{4}\right) + f(1, 0) \cos\left(\frac{3\pi}{4}\right) + f(1, 1) \cos\left(\frac{3\pi}{4}\right)] = 90 \end{aligned}$$

$$\begin{aligned} F(1, 1) &= \sum_{x=0}^1 \sum_{y=0}^1 f(x, y) \cos\left(\frac{(2x+1)}{4} \pi\right) \cos\left(\frac{(2y+1)}{4} \pi\right) \\ &= f(0, 0) \cos\left(\frac{\pi}{4}\right) \cos\left(\frac{\pi}{4}\right) + f(0, 1) \cos\left(\frac{\pi}{4}\right) \cos\left(\frac{3\pi}{4}\right) + f(1, 0) \cos\left(\frac{3\pi}{4}\right) \cos\left(\frac{\pi}{4}\right) + f(1, 1) \cos\left(\frac{3\pi}{4}\right) \cos\left(\frac{3\pi}{4}\right) = 10 \end{aligned}$$



8 × 8 画素 DCT

$$F(u, v) = \frac{1}{4} C(u)C(v) \sum_{x=0}^7 \sum_{y=0}^7 f(x, y) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

$$f(x, y) = \frac{1}{4} \sum_{u=0}^7 \sum_{v=0}^7 C(u)C(v) F(u, v) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right)$$

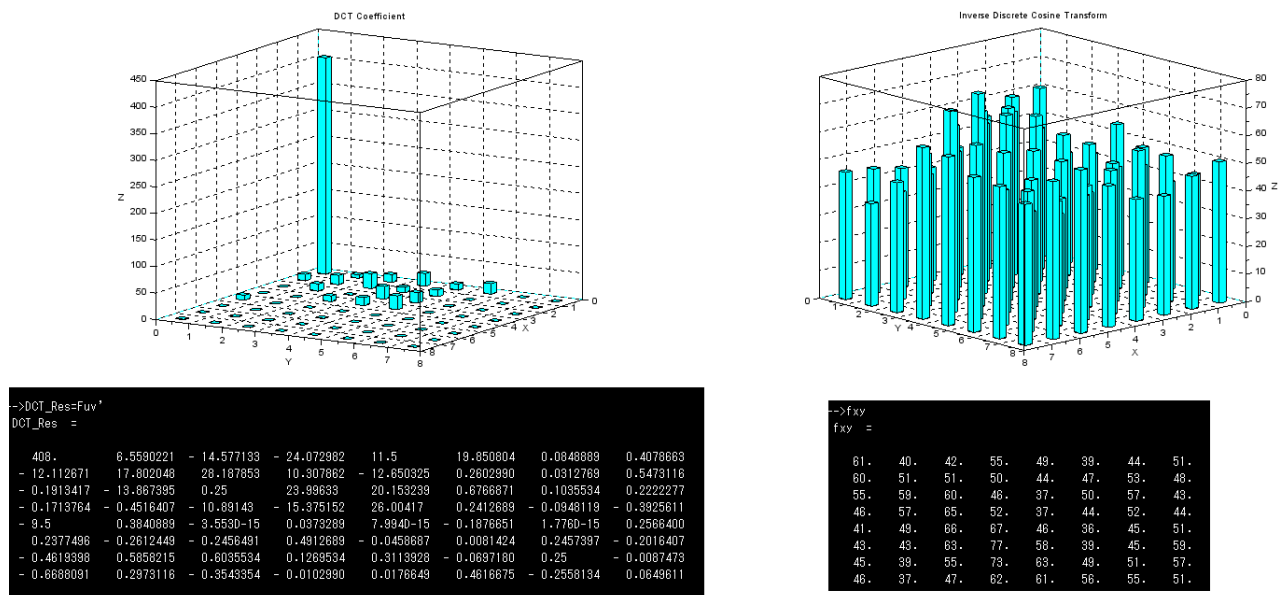


Figure 1: Scilab 実行結果

Source Code 1: Scilab

```

////////////////////////////////////
// 2次元離散コサイン変換
// Discrete Cosine Transform
// Inverse Discrete Cosine Transform
// 8×8Pixel
//                                     M.Tsutsui
////////////////////////////////////
clear all;

pi=%pi;

funcprot(0) //cosファクター用関数
function [cos_vec]=dct_cos(N);

cos_vec=[];

for u=0:N-1;
    if(u==0) then ,
        C_u=1/sqrt(2);
    else C_u=1;
    end

    for v=0:N-1;
        if (v==0) then,
            C_v=1/sqrt(2);
        else C_v=1;
        end

        for y=0:N-1;
            for x=0:N-1;
                cos_vec=[cos_vec,1/4*C_u*C_v*cos((2*x+1)*u/16*pi)*cos((2*y+1)*v/16*pi)];
            end
        end
    end
end
endfunction

```

```

funcprot(0) //IDFT用 cosファクター用関数
function [i_cos_vec]=idct_cos(N);
    i_cos_vec=[];

    for x=0:N-1;
        for y=0:N-1;
            for v=0:N-1;
                if(v==0) then ,
                    C_v=1/sqrt(2);
                else C_v=1;
                end
                for u=0:N-1;
                    if (u==0) then,
                        C_u=1/sqrt(2);
                    else C_u=1;
                    end
                    i_cos_vec=[i_cos_vec,1/4*C_u*C_v*cos((2*x+1)*u/16*pi)*cos((2*y+1)*v/16*pi)];
                end
            end
        end
    end
endfunction

funcprot(0)
function [squ_mr]=sq_mx_func(size,target);//正方形列 変換関数
    squ_mr=[];
    for K=1:1:size;
        squ_mr=[squ_mr;target(size*K-(size-1):size*K)];
    end
endfunction

N=8;

//-----f(x,y) 任意の明るさ-----//
f00=61 ; f10=40 ; f20=42 ; f30=55 ; f40=49 ; f50=39 ; f60=44 ; f70=51;
f01=60 ; f11=51 ; f21=51 ; f31=50 ; f41=44 ; f51=47 ; f61=53 ; f71=48;
f02=55 ; f12=59 ; f22=60 ; f32=46 ; f42=37 ; f52=50 ; f62=57 ; f72=43;
f03=46 ; f13=57 ; f23=65 ; f33=52 ; f43=37 ; f53=44 ; f63=52 ; f73=44;
f04=41 ; f14=49 ; f24=66 ; f34=67 ; f44=46 ; f54=36 ; f64=45 ; f74=51;
f05=43 ; f15=43 ; f25=63 ; f35=77 ; f45=58 ; f55=39 ; f65=45 ; f75=59;
f06=45 ; f16=39 ; f26=55 ; f36=73 ; f46=63 ; f56=49 ; f66=51 ; f76=57;
f07=46 ; f17=37 ; f27=47 ; f37=62 ; f47=61 ; f57=56 ; f67=55 ; f77=51;

//-----Pixel_Vector-----//
Pix_vec=[f00, f10,f20,f30,f40,f50,f60,f70,f01,f11,f21,f31,f41,f51,f61,f71,f02,f12,f22,f32,f42,f52,f62,f72,f03,f13,f23,f33,f43,f53,f63,f73,f04
,f14,f24,f34,f44,f54,f64,f74,f05,f15,f25,f35,f45,f55,f65,f75,f06,f16,f26,f36,f46,f56,f66,f76,f07,f17,f27,f37,f47,f57,f67,f77];
//※折り返しなし

Pix_vec2=[f00, f10,f20,f30,f40,f50,f60,f70;
    f01,f11,f21,f31,f41,f51,f61,f71;
    f02,f12,f22,f32,f42,f52,f62,f72;
    f03,f13,f23,f33,f43,f53,f63,f73;
    f04,f14,f24,f34,f44,f54,f64,f74;
    f05,f15,f25,f35,f45,f55,f65,f75;
    f06,f16,f26,f36,f46,f56,f66,f76;
    f07,f17,f27,f37,f47,f57,f67,f77];//8×8

//-----DCT-----//

PV_R=[];
for G=1:1:N*N;

```

```

    PV_R=[PV_R,Pix_vec]; //Pix_vec拡張(次元合わせ)
end

DCT=dct_cos(N).*PV_R; //Pix_vecとcos_vecの加算

DCT_sum=sum(sq_mx_func(N^2,DCT),2); //列方向に加算

Fuv=sq_mx_func(N,DCT_sum '); //F(u,v) ※DCT_sumは転置

//-----DCT-----//

//-----IDCT-----//

Fuv_vec=[];

for i=1:1:N;
    Fuv_vec=[Fuv_vec,Fuv(i,:)]; //加算用に行列のサイズ変換
end

I_PV_R=[];
for J=1:1:N^2;
    I_PV_R=[I_PV_R,Fuv_vec]; //Fuv拡張(icos_vecと加算可能にするため,次元合わせ)
end

IDCT=idct_cos(N).*I_PV_R;

IDCT_a=[];

IDCT_sum=sum(sq_mx_func(N^2,IDCT),2); //列方向に加算

fxy=[];
fxy=sq_mx_func(N,IDCT_sum '); //f(x,y)
//-----IDCT-----//

//_Plot_//////////
x1_p=linspace(1,N,N);
x2_p=x1_p;

subplot(1,3,1)
Sgrayplot(x1_p,x2_p,Pix_vec2, 'strf="041"')
set(gcf(),'color_map',jetcolormap(256));
xgrid();
title("f(x,y)", 'fontsize', 5);

subplot(1,3,2)
hist3d(abs(Fuv));
xgrid();
title("F(u,v)", 'fontsize', 5);

subplot(1,3,3)
Sgrayplot(x1_p,x2_p,abs(fxy), 'strf="041"')
set(gcf(),'color_map',jetcolormap(256));
xgrid();
title("F(u,v)→IDFT", 'fontsize', 5);

```

Source Code 2: Python

```

#-----
# Module_Name:Discrete Cosine Transform/

```

```

# Inverse Discrete Cosine Transform
# Author:m_tsutsui
#-----
#Library_Import-----
from numpy import*
import math, numpy as np
import matplotlib.pyplot as plt
#Library_Import-----

def Pix_vec_sqr_mx(N):#Pix_vec square matrix
    Pix_vec_sqmx=[]
    for i in range(1,N+1,1):
        Pix_vec_sqmx.append((Pix_vec[N*i-N:N*i]))

    return array(Pix_vec_sqmx)

def cos_vec(N):#_Cos_Vector(DCT)
    cos_vec=[]
    for u in range(0,N,1):
        if(u==0):
            C_u=1/sqrt(2)
        else: C_u=1

        for v in range(0,N,1):
            if (v==0):
                C_v=1/sqrt(2)
            else: C_v=1
            for y in range(0,N,1):
                for x in range(0,N,1):
                    cos_vec.append(1/4*C_u*C_v*cos((2*x+1)*u/16*pi)*cos((2*y+1)*v/16*pi))

    cos_vec_A=np.array(cos_vec)

    return cos_vec_A

def i_cos_vec(N):#_Cos_Vector(IDCT)
    cos_vec_i=[]
    for x_i in range(0,N,1):
        for y_i in range(0,N,1):
            for v_i in range(0,N,1):
                if(v_i==0):
                    C_v_i=1/sqrt(2)
                else: C_v_i=1
                for u_i in range(0,N,1):
                    if(u_i==0):
                        C_u_i=1/sqrt(2)
                    else: C_u_i=1
                    cos_vec_i.append(1/4*C_u_i*C_v_i*cos((2*x_i+1)*u_i/16*pi)*cos((2*y_i+1)*v_i/16*pi))

    cos_vecI_A=np.array(cos_vec_i)

    return cos_vecI_A

if __name__ == '__main__':

#-----f(x,y)_table-----
f00=61 ; f10=40 ; f20=42 ; f30=55 ; f40=49 ; f50=39 ; f60=44 ; f70=51;
f01=60 ; f11=51 ; f21=51 ; f31=50 ; f41=44 ; f51=47 ; f61=53 ; f71=48;
f02=55 ; f12=59 ; f22=60 ; f32=46 ; f42=37 ; f52=50 ; f62=57 ; f72=43;
f03=46 ; f13=57 ; f23=65 ; f33=52 ; f43=37 ; f53=44 ; f63=52 ; f73=44;

```

```

f04=41 ; f14=49 ; f24=66 ; f34=67 ; f44=46 ; f54=36 ; f64=45 ; f74=51;
f05=43 ; f15=43 ; f25=63 ; f35=77 ; f45=58 ; f55=39 ; f65=45 ; f75=59;
f06=45 ; f16=39 ; f26=55 ; f36=73 ; f46=63 ; f56=49 ; f66=51 ; f76=57;
f07=46 ; f17=37 ; f27=47 ; f37=62 ; f47=61 ; f57=56 ; f67=55 ; f77=51;

#-----Pixel_Vector-----
Pix_vec=[f00, f10,f20,f30,f40,f50,f60,f70,f01,f11,f21,f31,f41,f51,f61,f71,f02,f12,f22,f32,f42,f52,f62,f72,f03,f13,f23,f33,f43,f53,f63,f73,
        f04,f14,f24,f34,f44,f54,f64,f74,f05,f15,f25,f35,f45,f55,f65,f75,f06,f16,f26,f36,f46,f56,f66,f76,f07,f17,f27,f37,f47,f57,f67,f77]

N=int (sqrt(np.size(Pix_vec))) #size

#-----DCT-----
PV_R=Pix_vec*(N**2)
PV_R_A=np.array(PV_R)

DCT=cos_vec(N)*PV_R_A

DCT_a=[]
for K in range(1,N*N+1,1):
    DCT_a.append(DCT[N**2*K-N**2:N**2*K])

DCT_sum=sum(DCT_a,1)
DCT_sum_M=matrix(DCT_sum)
DCT_sum_t=DCT_sum.M.T

Fuv=[]
for L in range(1,N+1,1):#square matrix
    Fuv.append(DCT_sum_t[N*L-N:N*L])

Fuv_A=array(Fuv)
Fuv_M=np.matrix(Fuv_A)

DCT_Res=Fuv_M.T
print(DCT_Res) #check
#-----DCT-----

#-----IDCT-----
Fuv_vec=list(DCT_sum) #cast matrix -> list
PV_R_I=Fuv_vec*(N**2)
PV_R_I_A=np.array(PV_R_I)

IDCT=i.cos_vec(N)*PV_R_I_A

IDCT_a=[]
for K_i in range(1,N*N+1,1):
    IDCT_a.append(IDCT[N**2*K_i-N**2:N**2*K_i])

IDCT_sum_i=sum(IDCT_a,1)
IDCT_sum_i_M=matrix(IDCT_sum_i)
IDCT_sum_i_t=IDCT_sum_i.M.T

fxy=[]
for L_i in range(1,N+1,1):#square matrix
    fxy.append(IDCT_sum_i_t[N*L_i-N:N*L_i])

fxy_A=array(fxy)
fxy_M=np.matrix(fxy_A)

IDCT_Res=fxy_M
print(IDCT_Res) #check
#-----IDCT-----

```

```
#-----plot-----
x1_p=np.linspace(0,N,N)
x1,x2=np.meshgrid(x1_p,x1_p)

plt.figure(facecolor='w')
plt.subplot(131)
plt.contourf(x1,x2,Pix_vec_sqr_mx(N),100)
plt.grid()
plt.title('f(x,y)')

plt.subplot(132)
plt.contourf(x1,x2,array(DCT_Res),100)
plt.grid()
plt.title('DCT')

plt.subplot(133)
plt.contourf(x1,x2,array(IDCT_Res),100)
plt.grid()
plt.title('IDCT')

plt.show()
#-----plot-----
```