TEST-1 Computer Vision

PAGE No.
DATE / /

Name: Deepak Kumar
Roll No: 21518/SE/051

1) d
2) c
3) d
4) a
5) b
6) b, d
7) a
8) b

(9) Using C++

```
#define pi 3.142857
Cout int m=8, n=8;

int dct_transform (int matrix [] [n])
{
    int i, j, k, l;
    float dct[m][n];

    ~~float ci, cj, dct1, sum;~~
    float x, y, dct1, sum;
for( ~~int~~ i=0; i<m; i++)
    {
        for ( ~~int~~ j=0; j<n; j++) {

if (i==0)   x = 1 / sqrt (m);
else
    x = sqrt (2) / sqrt (m);
if (j==0)
    y = 1/sqrt (n);
else
    y = sqrt (2) / sqrt (n);

sum = 0;
for (k=0; k<m; k++) {
    for (l=0; l<n; l++) {
        dct1 = matrix [k][l] *
            cos ((2*k+1)*i * pi / (2*m)) *
            cos ((2*l+1)*j * pi / (2*n));
        sum = sum + dct1;
    }
}
dct[i][j] = x*y*sum;
}
}
```

```
for (int i=0; i<m; i++)
{
    for (j=0; j<n; j++)
    {
        cout << dct[i][j] << " ";
    }
    cout << "\n";
}
```

PAGE No.
DATE / /

IDCT

⇒ One Dimensional IDCT Equation

$$x_c(n) = \sum_{n=0}^{N-1} c[u]\, x_n \cos\left(\frac{2\pi n}{N}\right)$$

```
int sum = 0;
for (int i=0; i<N-1; i++)
{
    sum = sum + c[n] x x_n x cos(2 x pi x i / N);
}
cout << sum << endl;
```

PAGE No.
DATE / /

IDCT Equation

$$x_c(k) = \sum_{n=0}^{N-1} c[u]\, X_u \cos\left(\frac{k\,2\pi u}{N}\right)$$

where

$k = 0, 1, 2 \ldots N-1$

X_u is the DCT result and

$c[u] = 1, \quad u = 0$

$c[u] = 2$ for $u = 1, 2, 3 \ldots N-1$;

```
#define pi 3.14
int c[N];
for (int i=0; i<N; i++)
{
    if (i==0) c[i]=1;
    else c[i]= 2;
}

int xc[k]=0;
for (int i=0; i<N; i++)
{
    xc[k] = xc[k]+ c[i]*X[i]*cos(k*2*pi*i/N);
}

Cout << xc[k] << endl;
}
```

PAGE No.
DATE / /

10

```
I2 = im2double (Your Image);
[m,n] = size (I2);
Blocks = cell (m/8, n/8);
count i = 0;
for i = 1:8: m-7
    count i = count i + 1;
    count j = 0;
    for (j = 1:8: n-7
        count j = count j + 1;
        Blocks {count i, count j}
            = I2 ( i: i+7, j: j+7);
    end
end
```

```
I = imread ('my Image.png');
I = rgb2gray (I);
[nx, ny] = size (I);

kx = 8; ky = 8;
nxo = nx + (kx - mod (nx, kx)).
nyo = ny + (ky - mod (ny, kx));

I1 = uint8 (zeros (nxo, nyo));
I1 (1:nx, 1:ny) = I;

kxo = nxo / kx;
kyo = nyo / ky;

Y = mat2cell (I1, repmat (
        kxo, [1, size (I1, 1) / kxo],
        repmat (1, size (I1, 2) / kyo)));
```

(12)

- The optical transfer function (OTF) of an optical system such as camera, microscope, human eye, or projector specifies how different spatial frequencies are handled by the system. It is used by optical engineers to describe how the optics project light from the object or scene onto a photographic film, detector array, retina screen, or simple the next item in the optical transmission chain.

MTF

- The resolution and performance of an optical microscope can be characterized by a quantity known as modular transfer function (MTF) which is the ability to transfer contrast from the specimen to the intermediate image plane at a specific resolution.
- Computation of the modulation transfer function is a mechanism that is often used by optical manufacturers to incorporate resolution and contrast data into a single specification.

use: Photolithography
: automatic processing by human observer

using MTF, image quality is measured

- Optical designers and engineers frequently refer to MTF data, especially in applications where success or failure is contingent on how accurately a particular object is imaged.

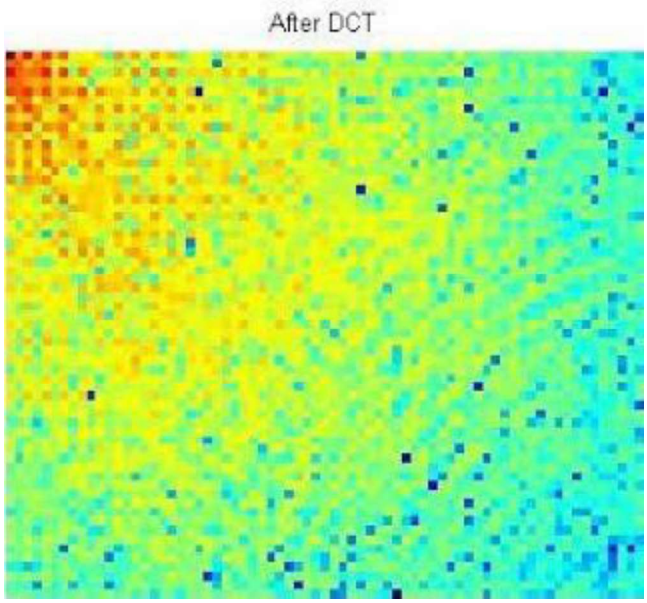
9 b)

MATLAB CODE:

```
%2-D INVERSE DISCRETE COSINE TRANSFORM
%PREALLOCATE THE MATRIX
A=zeros(size(B));
Temp=zeros(size(B));
[M N]=size(B);

x=1:M;
x=repmat(x',1,N);
y=repmat(1:N,M,1);

figure,
imshow(log(abs(B)),[]);colormap(jet);title('After DCT');
```




```
for i=1:M
   for j = 1: N

      if(i==1)
        AlphaP=sqrt(1/M);
      else
```

```
      AlphaP=sqrt(2/M);
    end

    if(j==1)
      AlphaQ=sqrt(1/N);
    else
      AlphaQ=sqrt(2/N);
    end

    cs1=cos((pi*(2*x-1)*(i-1))/(2*M));
    cs2=cos((pi*(2*y-1)*(j-1))/(2*N));
    Temp=B.*cs1.*cs2*AlphaP*AlphaQ;

      A(i,j)=sum(sum(Temp));
  end
end
%OUTPUT
figure,
imshow(abs(A),[0 255]);title('Image after IDCT');
```

11)

```
import numpy as np
import pandas as pd


class ProbabilityVector:
def __init__(self, probabilities: dict):

states = probabilities.keys()
probs = probabilities.values()

assert len(states) == len(probs),
"The probabilities must match the states."
assert len(states) == len(set(states)),
"The states must be unique."
assert abs(sum(probs) - 1.0) < 1e-12,
"Probabilities must sum up to 1."
assert len(list(filter(lambda x: 0 <= x <= 1, probs))) == len(probs), \
"Probabilities must be numbers from [0, 1] interval."

self.states = sorted(probabilities)
self.values = np.array(list(map(lambda x:
probabilities[x], self.states))).reshape(1, -1)

@classmethod
def initialize(cls, states: list):
size = len(states)
rand = np.random.rand(size) / (size**2) + 1 / size
rand /= rand.sum(axis=0)
return cls(dict(zip(states, rand)))

@classmethod
def from_numpy(cls, array: np.ndarray, state: list):
return cls(dict(zip(states, list(array))))

@property
def dict(self):
return {k:v for k, v in zip(self.states, list(self.values.flatten()))}

@property
def df(self):
return pd.DataFrame(self.values, columns=self.states, index=['probability'])

def __repr__(self):
```

```
return "P({}) = {}.".format(self.states, self.values)

def __eq__(self, other):
if not isinstance(other, ProbabilityVector):
raise NotImplementedError
if (self.states == other.states) and (self.values == other.values).all():
return True
return False

def __getitem__(self, state: str) -> float:
if state not in self.states:
raise ValueError("Requesting unknown probability state from vector.")
index = self.states.index(state)
return float(self.values[0, index])

def __mul__(self, other) -> np.ndarray:
if isinstance(other, ProbabilityVector):
return self.values * other.values
elif isinstance(other, (int, float)):
return self.values * other
else:
NotImplementedError

def __rmul__(self, other) -> np.ndarray:
return self.__mul__(other)

def __matmul__(self, other) -> np.ndarray:
if isinstance(other, ProbabilityMatrix):
return self.values @ other.values

def __truediv__(self, number) -> np.ndarray:
if not isinstance(number, (int, float)):
raise NotImplementedError
x = self.values
return x / number if number != 0 else x / (number + 1e-12)

def argmax(self):
index = self.values.argmax()
return self.states[index]
```

Submitted By: