DIGITAL IMAGE COMPRESSION TECHNIQUE

IRINA RABEJA
MEE DCSc

REFERENCES

AN ADAPTIVE ZONAL FILTER FOR DATA COMPRESSION BY L.M.CHENG, A.S. HO, R.E.BURGE

DIGITAL IMAGING BY M. GALER, L. HORVAT

DIGITAL IMAGE PROCESSING BY K. R. CASTLEMAN

DIGITAL IMAGE COMPRESSION TECHNIQUES BY M. RABANI, P. W. JONES

PRINCIPLES OF DIGITAL IMAGE SYNTHESIS BY A. S. GLASSEN

TEORIA TRANSMISIUNII INFORMATIEI BY A. SPATARU

THE COMPLETE GUIDE TO DIGITAL GRAPHIC DESIGN BY B. GORDON, M. GORDON

MATLAB PROGRAMMING LANGUAGE

WIKIPEDIA

WIKIMEDIA

CONTENT

DIGITAL IMAGE

COMPRESSION TECHNIQUE

DIGITAL IMAGE

THE INTERNET IS THE LARGEST COMPUTER NETWORK IN THE WORLD WITH BILLIONS USERS WORLD WIDE ALLOWING PEOPLE TO ACCESS LARGE VOLUMES OF DATA AND EXCHANGE IDEAS FAST.
DISPLAYING PHOTOGRAPHS OR IMAGES ON A COMPUTER SCREEN IS A PROCESS INVOLVING THAT THE FILES OF THE PHOTOGRAPHS OR IMAGES ARE TRANSMITTED THROUGH PHONE AND CABLE LINES, VIA INTERNET, AND ASSEMBLED BY THE BROWSER SOFTWARE LIKE INTERNET EXPLORER, SAFARI, FIREFOX, NETSCAPE NAVIGATOR INTO AN IMAGE ON THE COMPUTER SCREEN.
THE PHOTOGRAPHS/IMAGES HAVE COMPLEX COMPUTER FILES THAT TAKE MORE TIME TO TRANSMIT THAN THE WORDS WRITTEN ON THE COMPUTER MONITOR.
THE FILE SIZE AND THE TRANSMISSION TIME OF AN IMAGE ARE KEY ASPECTS OF UNDERSTANDING HOW TO MAKE AND USE IMAGES ON COMPUTERS.
THE TRANSMISSION TIME OF AN IMAGE OR PHOTOGRAPH DEPENDS ON THE NUMBER OF BITS/SECOND RATE, A CHARACTERISTIC OF THE INTERNET.
IT IS WELL TO CREATE AN IMAGE TRANSMITTED IN 15 SECONDS OR LESS STILL LOOKING GOOD.

THE COMPUTER IMAGES ARE MADE UP OF HUNDREDS TO MILLIONS OF SMALL SQUARES CALLED "PIXELS".
AN IMAGE OF SIZE MXN COMPOSED BY PIXELS IS SHOWN IN FIG 1.

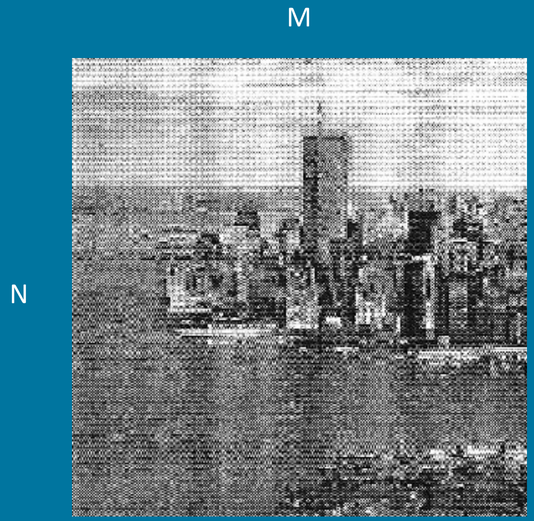


FIG 1

THE PROCESS OF TRANSFORMING AN IMAGE IN A BUNCH OF PIXELS WITH INFORMATION ABOUT THE PIXELS - LIKE THEIR NUMBER, POSITION AND COLOR - IN A DIGITAL FORM IN A FILE IS CALLED DIGITISING. BY DIGITISING, THE IMAGES/PHOTOGRAPHS TAKE A DIGITAL FORM OR ARE CONVERTED TO DIGITAL IMAGES/DIGITAL PHOTOGRAPHS.
TO BE TRANSMITTED OVER THE INTERNET AND APPEAR ON THE COMPUTER SCREEN AN IMAGE MUST BE CONVERTED IN ITS DIGITAL FORM.
DIGITAL IMAGES HAVE MANY ADVANTAGES OVER STANDARD STATIC IMAGES:
CAN BE MANIPULATED WITH IMAGING SOFTWARE IN ORDER TO ZOOM TO AREAS OF INTEREST FOR DETAILED EXAMINATION, CAN BE ROTATED TO ALTER THE VIEWER'S PERSPECTIVE, CAN BE ATTACHED TO THEM AUDIO AND TEXT FILES AND FINALLY CAN BE ENCRYPTED TO ENSURE THEY ARE NOT STOLEN IN TRANSIT.
IN COMPUTING, AN IMAGE COMPOSED OF PIXELS IS KNOWN AS A BITMAPPED IMAGE OR A RASTER IMAGE. IF THE IMAGE IS 300 PIXELS WIDE AND 500 PIXELS HIGH, IT IS SAID THAT IT HAS A SIZE OF 300X500=150,000 PIXELS. THE EYE/BRAIN SYSTEM SEES THE SQUARES AS AN IMAGE THAT LOOKS LIKE THE REAL THING EVEN THOUGH WE REALLY LOOK AT A TOTAL OF 150,000 TINY SQUARES.
EACH OF 150,000 PIXELS HAS A DEGREE OF GRAY/COLOR ASSIGNED TO IT.

ON THE SCREEN AN IMAGE IS MADE BY THE COMPUTER BROWSER SOFTWARE THAT ASSIGNS A DEGREE OF COLOR TO EACH PIXEL AND DISPLAYS IT.
THE NUMBER OF DISTINCT COLORS THAT CAN BE REPRESENTED BY A PIXEL DEPENDS ON THE NUMBER OF BITS PER PIXEL (BPP).
THE SIMPLEST PIXEL REPRESENTATION IS A BLACK AND WHITE MONOCHROME IMAGE IN WHICH ONE BIT REPRESENTS ONE PIXEL. MONOCHROME CRTS USE WHITE, GREEN OR AMBER PHOSPHORS AS A SINGLE COLOR OVER A GREY/BLACK SCREEN BACKGROUND.
SOME MONITORS HAVE THE ABILITY TO VARY THE BRIGHTNESS OF INDIVIDUAL PIXELS, THEREBY CREATING THE ILLUSION OF DEPTH AND COLOR, EXACTLY LIKE A BLACK-AND-WHITE TELEVISION.
SO, A 1 BPP IMAGE USES 1-BIT FOR EACH PIXEL, SO EACH PIXEL CAN BE EITHER ON OR OFF.
EACH ADDITIONAL BIT DOUBLES THE NUMBER OF COLORS AVAILABLE, SO A 2 BPP IMAGE CAN HAVE 4 COLORS, A 3 BPP IMAGE CAN HAVE 8 COLORS AND SO ON.
THE IMAGE FILE SIZE DEPENDS ON THE NUMBER OF PIXELS COMPOSING AN IMAGE AND THE COLOR DEPTH OF THE PIXELS; IT IS EXPRESSED AS A NUMBER OF BYTES, A BYTE BEING BASICALLY EQUAL WITH EIGHT BITS. THE GREATER THE NUMBER OF PIXELS (IMAGE RESOLUTION), THE LARGER THE FILE.
ALSO, EACH PIXEL OF AN IMAGE INCREASES IN SIZE WHEN ITS COLOR DEPTH INCREASES —
AN 8 BPP IMAGE STORES 256 COLORS, A 24 BPP IMAGE STORES 16.8 MILLION COLORS (TRUECOLOR IMAGE).

1 BPP	= 2 COLORS (MONOCHROME)
2 BPP	= 4 COLORS
3 BPP	= 8 COLORS
....	
8 BPP	= 256 COLORS
16 BPP	= 65,536 COLORS ("HIGHCOLOR")
24 BPP	≈ 16.8 MILLION COLORS ("TRUECOLOR")

PIXEL STRUCTURES

1-BIT MONOCHROME (black & white): 2 COLORS

0	1
●	○

8-BIT GREYSCALE (grey range) : 256 COLORS/GREY SHADES

00000000		00010000		11111111
◆	………..	◆	………..	◇

24-BIT COLOR (three 8-bit subpixels): 16.8 mil COLORS - TRUECOLOR

RED	00000000	11111111	00000000	00000000	11111111	11111111	00000000	11111111	
GREEN	00000000	00000000	11111111	00000000	00000000	11111111	11111111	11111111	
BLUE	00000000	00000000	00000000	11111111	11111111	00000000	11111111	11111111	…………..
	⬛	🟥	🟩	🟦	🟪	🟨	🟦	⬜	

• •

• •

EXAMPLES

1.
A SMALL IMAGE OF SIZE 300 X 450 PIXELS HAS 135,000 PIXELS
FOR 1 BYTE DEGREE OF GRAY ASSIGNED PER PIXEL THE
IMAGE FILE SIZE =135,000 BYTES (8X135000=1,080,000 BITS)
AT A MODEM RATE OF 512 KBITS/SEC THE
IMAGE TRANSMISSION TIME = 2.109 SEC [LESS THAN 15 SEC]

2.
A LARGER IMAGE OF SIZE 900 X1500 PIXELS HAS 1,350,000 PIXELS
FOR 1 BYTE DEGREE OF GRAY ASSIGNED PER PIXEL THE
IMAGE FILE SIZE=1,350,000 BYTES (8X1,350,000=10,800,000 BITS)
AT A MODEM RATE OF 512 KBITS/SEC THE
IMAGE TRANSMISSION TIME = 21.093SEC [MORE THAN 15 SEC]

COMPRESSION TECHNIQUE

THE IMAGES CAN BE USED IN THEIR DIGITAL FORM, STORED AND TRANSMITTED AS FILES. DEPENDING ON THE SIZE OF IMAGE AND ON THE NECESSARY RESOLUTION, THE AMOUNT OF DATA GENERATED BY THE DIGITAL IMAGES MAY BE SO GREAT THAT RESULTS IN IMPRACTICAL STORAGE, PROCESSING AND COMMUNICATION REQUIREMENTS.
A SOLUTION WOULD BE TO REDUCE THE REDUNDANCY OF THE IMAGE DATA IN ORDER TO BE ABLE TO STORE OR TRANSMIT DATA IN AN EFFICIENT FORM. THE REDUCING OF THE AMOUNT OF DATA REQUIRED TO REPRESENT A DIGITAL IMAGE IS ACCOMPLISHED BY THE IMAGE COMPRESSION. IMAGE COMPRESSION IS THE APPLICATION OF DATA COMPRESSION ON DIGITAL IMAGES. IMAGE COMPRESSION CAN BE LOSSY OR LOSSLESS.
LOSSY COMPRESSION METHODS ARE SUITABLE FOR APPLICATIONS WHERE MINOR LOSS OF FIDELITY IS ACCEPTABLE TO ACHIEVE A SUBSTANTIAL REDUCTION IN BIT RATE (NUMBER OF BITS CONVEYED OR PROCESSED PER UNIT OF TIME).
AN INTERESTING METHOD FOR LOSSY COMPRESSION IS THE *TRANSFORM CODING* THAT TRANSMITS INFORMATION ABOUT THE COMPRESSED OR CODED FOURIER TRANSFORM OF THE ORIGINAL IMAGE.
AT THE RECEPTION IT IS OBTAINED THE COMPRESSED IMAGE BY DOING THE INVERSE FOURIER TRANSFORM OF THE RECEIVED CODED/COMPRESSED FOURIER TRANSFORM.

IN MATHEMATICS THE FOURIER TRANSFORM **FT** OF A FUNCTION F(X) THE **FT**[F(X)] IS CALLED THE FREQUENCY DOMAIN REPRESENTATION OF F(X). IT SHOWS/DESCRIBES THE FREQUENCIES OF THE SINE AND COSINE WAVES, WHICH ARE PRESENT IN THE FUNCTION F(X).
THE FOURIER TRANSFORM **FT** IS NAMED IN THE HONOUR OF THE FRENCH MATHEMATICIAN AND PHYSICIST **JEAN-BAPTISTE JOSEPH FOURIER** (1768 –1830).
JEAN-BAPTISTE JOSEPH FOURIER IS BEST KNOWN FOR INITIATING THE INVESTIGATION OF THE FOURIER SERIES AND THEIR APPLICATIONS TO PROBLEMS OF HEAT TRANSFER AND VIBRATIONS, FIRST USED FOR THE PURPOSE OF SOLVING THE HEAT EQUATION IN A METAL PLATE.
HE MADE IMPORTANT CONTRIBUTIONS TO THE STUDY OF TRIGONOMETRIC SERIES, AFTER PRELIMINARY INVESTIGATIONS BY **MADHAVA**, **NILAKANTHA SOMAYAJI**, **JYESTHADEVA**, **LEONARD EULER**, **JEAN LE ROND D’ALEMBERT** AND **DANIEL BERNOULLI**. HE APPLIED THIS TECHNIQUE TO FIND THE SOLUTION OF THE HEAT EQUATION, PUBLISHING HIS INITIAL RESULTS IN 1807 AND 1811 YEARS, AND PUBLISHING HIS ‘THEORIE ANALYTIQUE DE LA CHALEUR’ IN 1822 YEAR.
THE FOURIER SERIES HAS MANY APPLICATIONS IN ELECTRICAL ENGINEERING, VIBRATION ANALYSIS, ACOUSTICS, OPTICS, SIGNAL PROCESSING, IMAGE PROCESSING ETC.
THE STUDY OF FOURIER SERIES IS A BRANCH OF FOURIER ANALYSIS.

JOSEPH FOURIER

IN MATHEMATICS, A FOURIER SERIES IS A SUM THAT REPRESENTS A PERIODIC FUNCTION OR PERIODIC SIGNAL AS A SUM OF SIMPLE OSCILLATING FUNCTIONS, SINE WAVES SIN(X) AND COSINE WAVES COS(X) OR EQUIVALENT COMPLEX EXPONENTIALS $E^{IX} = COS(X) + ISIN(X)$ [EULER'S FORMULA].

$F(X) = \sum_N [A_N\, COS(NX) + B_N\, SIN(NX)]$ $\qquad$ $F(X) = \sum_N C_N E^{INX}$.

THE FOURIER TRANSFORM **FT** OF F(X) IS:

FT$[F(X)] = F(X)\, E^{-INX} = \sum_N C_N E^{INX} E^{-INX} = \sum_N C_N$

MANY FUNCTIONS OR SIGNALS ARE DEFINED IN THE TWO-DIMENSIONAL SPACE OR X-Y PLANE AS FUNCTION F(X,Y):

$F(X,Y) = \sum_M \sum_N C_M C_N E^{INX} E^{IMY} = \sum_M \sum_N C_M C_N E^{I(NX+MY)}$

THEN THE TWO DIMENSIONAL FOURIER TRANSFORM FT OF F(X,Y) IS:

FT$[F(X,Y)] = F(X,Y)\, E^{-I(NX+MY)} = \sum_M \sum_N C_M C_N$

FOR MOST REAL IMAGES, VIZUALIZING THE FOURIER SPECTRUM IT IS OBSERVED THAT THE COMPONENTS OF LARGER ENERGY CLUSTER IN THE LOW SPATIAL FREQUENCY REGION AND THE MOST OF THE ENERGY DISTRIBUTION OF THE IMAGE IS ABOUT THE CENTER OF THE IMAGE IN THE TRANSFORM SPACE.

THOSE COMPONENTS HAVE THE MOST INFORMATION ABOUT THE IMAGE AND THOSE SHOULD BE PRESERVED IN CASE OF COMPRESSION. THEY CAN BE SEPARATED WITH A FILTER WHICH CAN FOLLOW THE SHAPE OF THE MAXIMUM ENERGY DISTRIBUTION. SUCH FILTERS WOULD HAVE DIFFERENT SHAPES: CIRCLE, SQUARE, STARS. SEE FIG 2

THE CLOSEST FILTERS MAY BE DESCRIBED BY A HYPOCYCLOID/ASTEROID SHAPE.

(ASTEROID MEANS STARLIKE IN GREEK.)

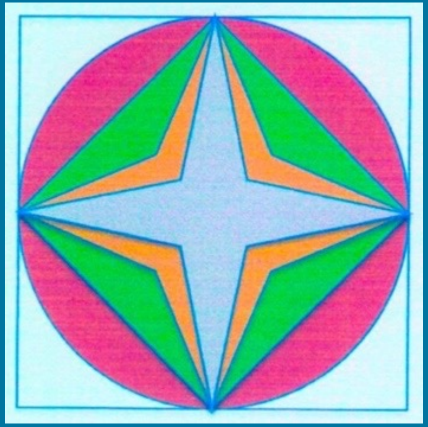


FIG 2

THE FOURIER TRANSFORM IS AN IMPORTANT TOOL IN IMAGE PROCESSING USED TO EXPRESS AN IMAGE FUNCTION THROUGH ITS SINE & COSINE COMPONENTS.
IN IMAGE PROCESSING IS USED THE TWO-DIMENSIONAL FOURIER TRANSFORM, MORE PRECISELY THE DISCRETE TWO-DIMENSIONAL FOURIER TRANSFORM **DFT** BECAUSE THE IMAGE IS COMPOSED BY DISCRETE PIXELS.
IN THE FOURIER DOMAIN IMAGE EACH POINT REPRESENTS A PARTICULAR FREQUENCY CONTAINED IN THE SPATIAL DOMAIN IMAGE. DFT IS A SAMPLED FOURIER TRANSFORM AND THEREFORE DOES NOT CONTAIN ALL FREQUENCIES FORMING AN IMAGE, BUT ONLY A SET OF SAMPLES WHICH IS LARGE ENOUGH TO FULLY DESCRIBE THE SPATIAL DOMAIN IMAGE.
THE NUMBER OF FREQUENCIES CORRESPONDS TO THE NUMBER OF PIXELS IN THE SPATIAL DOMAIN IMAGE, SO THE IMAGE IN THE SPATIAL AND FOURIER DOMAINS ARE OF THE SAME SIZE.

THE INTERACTIVE PROGRAM, WHICH MANIPULATES MATRICES AS ITS FUNDAMENTAL OBJECTS - **MATLAB** OR **MATRIX LABORATORY -** CALCULATES THE FOURIER TRANSFORMS WITH A FASTER ALGORITHM CALLED FAST FOURIER TRANSFORM **FFT** IF THE NUMBER OF PIXELS IS A POWER OF 2.

THE FOURIER TRANSFORM PRODUCES A COMPLEX NUMBER OUTPUT EXPRESSED BY MAGNITUDE/SPECTRUM AND PHASE.

FOR THE IMAGE COMPRESSION IS USED AN ASTEROID ZONAL FILTER IN THE FOURIER TRANSFORM SPACE. THE FILTER IS A SIMPLE ADAPTIVE ZONAL FILTER WITH SHAPE ADJUSTMENT TO MAXIMIZE THE ENERGY STORED IN THE TRANSFORM COEFFICIENTS.

THE GENERAL CARTESIAN COORDINATES REPRESENTATION OF AN ASTEROID CURVE IS GIVEN BY THE FUNCTION:

$$U^E + V^E = R^E \qquad \text{FOR } 0 < E < 2$$

WHERE U,V ARE THE COORDINATES IN THE FOURIER TRANSFORM SPACE AND R IS THE MAXIMUM HALF LENGTH.

THE CORRESPONDING ZONAL FILTER H(U,V) IS GIVEN BY :

$$H(U,V) = 1 \text{ FOR } | U^E + V^E | \leq R^E$$

$$H(U,V) = 0 \text{ FOR } | U^E + V^E | > R^E$$

THE REPRESENTATION OF THE ASTEROID CURVES FOR FOUR VALUES OF E [E=2, 1, 0.75, 0.5] GIVES THE FOUR SHAPES OF CIRCLE, SQUARE (SHIFTED Π/4) AND STARS:

FOR:	EQUATION:	SHAPE:
E=2	$U^2 + V^2 = R^2$	CIRCLE
E=1	$U + V = R$	SQUARE
E=0.75	$U^{0.75} + V^{0.75} = R^{0.75}$	STAR1
E= 0.5	$U^{0.5} + V^{0.5} = R^{0.5}$	STAR2

TO ANALYSE THE COMPRESSION PROCESS IT IS USED THE *COMPRESSION FACTOR* **C**, WHICH IN THE PRESENT TOPIC IT IS DEFINED AS THE RATIO OF THE NUMBER OF PIXELS OF THE ORIGINAL IMAGE TO THE NUMBER OF PIXELS OF THE COMPRESSED IMAGE.

TO ANALYSE THE QUALITY OF THE COMPRESSED IMAGES IN THE PRESENT TOPIC, IT IS USED THE NORMALISED *MEAN SQUARE ERROR* **MSE** OF THE COMPRESSED IMAGE WITH RESPECT TO THE ORIGINAL IMAGE:

$$\mathrm{MSE} = \sum_{j=0}^{M-1}\sum_{i=0}^{M-1} (F^*(i,j)-F(i,j))^2 \,/\, \sum_{j=0}^{M-1}\sum_{i=0}^{M-1} F^2(i,j)$$

i,j = SAMPLE COORDINATES IN REAL SPACE (SQUARE IMAGE MxM)

$F(i,j)$ = SAMPLED SPATIAL INTENSITIES OF THE ORIGINAL IMAGE

$F^*(i,j)$ = SAMPLED SPATIAL INTENSITIES OF THE COMPRESSED IMAGE

FURTHER WILL BE DESCRIBED THE WAY TO REALIZE THE COMPRESSION OF AN IMAGE/PHOTOGRAPH FILE USING THE INTERACTIVE PROGRAM MATLAB. THE PROCESS OF COMPRESSION IS DONE BY MATLAB PROGRAM, FOLLOWING MORE STEPS:

- IT IS TAKEN A DIGITAL IMAGE OF SIZE 128X128 PIXELS BY SCANNING A PHOTOGRAPH.

THE DIGITAL IMAGE IS PRESENTED IN FIG 3. THE IMAGE HAS BLACK AND WHITE HORIZONTALLY & VERTICALLY DIRECTED FEATURES.

- IT IS OBTAINED THE DISCRETE FOURIER TRANSFORM DFT OF THE RESPECTIVE DIGITAL IMAGE.

THE DFT SPECTRUM IS PRESENTED IN FIG 4. DFT IMAGE HAS ALSO 128X128 PIXELS AND ITS CENTRE IS THE POINT (64,64). IT IS OBSERVABLE THAT THE DFT IMAGE IS BRIGHTER AROUND ITS CENTRE MEANING THAT THE MAIN ENERGY DISTRIBUTION IN THE FREQUENCY DOMAIN IS AROUND IMAGE CENTRE. THAT ARIA SHOULD BE PRESERVED.

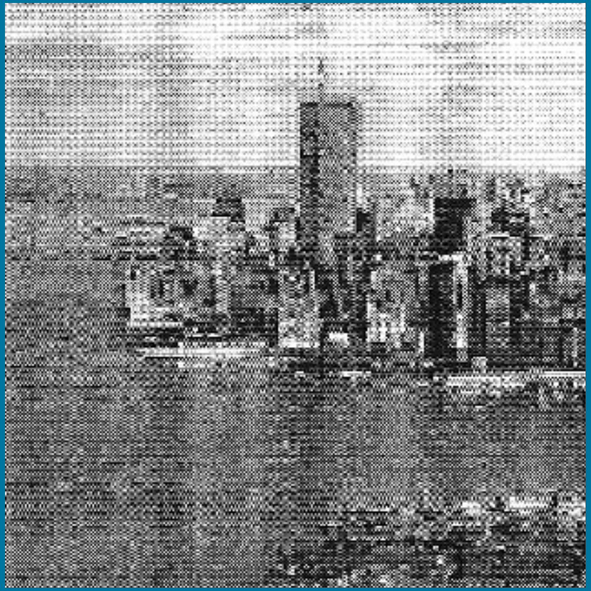


FIG 3

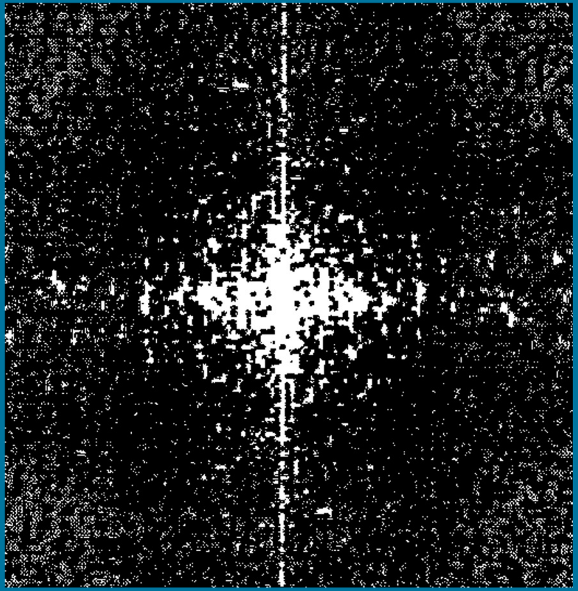


FIG 4

- ARE DONE THE FILTERS AS 128X128 HILBERT MATRIX HJ OF ONES 1S AND ZEROES 0S WITH ONES 1S IN THE INTERIOR OF FIGURES WHICH HAVE TO FOLLOW THE SHAPE OF THE MAXIMUM ENERGY DISTRIBUTION IN THE FOURIER DOMAIN. THE FIGURES ARE CIRCLE, SQUARE, STAR1 AND STAR2 WITH THE CENTRE POINT (64,64) GIVEN BY THE ASTEROID EQUATION:
$(X-64)^E + (Y-64)^E = R^E$ FOR **E** VALUES OF 2, 1, 0.75, 0.5.

SKETCHES FOR HILBERT MATRIX HJ SHAPES

$$H1 \quad \begin{bmatrix} 0&0&0&0&1&1&0&0&0&0 \\ 0&0&1&1&1&1&1&1&0&0 \\ 0&1&1&1&1&1&1&1&1&0 \\ 1&1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1&1 \\ 0&1&1&1&1&1&1&1&1&0 \\ 0&0&1&1&1&1&1&1&0&0 \\ 0&0&0&0&1&1&0&0&0&0 \end{bmatrix}$$

$$H2 \quad \left[\begin{matrix} 0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&1&1&1&1&0&0&0 \\ 0&0&1&1&1&1&1&1&0&0 \\ 0&1&1&1&1&1&1&1&1&0 \\ 1&1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1&1 \\ 0&1&1&1&1&1&1&1&1&0 \\ 0&0&1&1&1&1&1&1&0&0 \\ 0&0&0&1&1&1&1&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0 \end{matrix} \right.$$

$$H3 \quad \begin{bmatrix} 0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&1&1&1&1&0&0&0 \\ 0&0&1&1&1&1&1&1&0&0 \\ 1&1&1&1&1&1&1&1&1&1 \\ 1&1&1&1&1&1&1&1&1&1 \\ 0&0&1&1&1&1&1&1&0&0 \\ 0&0&0&1&1&1&1&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0 \\ 0&0&0&0&1&1&0&0&0&0 \end{bmatrix}$$

$$H4 \quad \begin{bmatrix} 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&1&1&1&0&0&0&0 \\ 0&0&1&1&1&1&1&0&0&0 \\ 1&1&1&1&1&1&1&1&1&1 \\ 0&0&1&1&1&1&1&1&0&0 \\ 0&0&0&1&1&1&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \end{bmatrix}$$

THE FILTERS ARE DONE FOR TWO CASES:

A. CONSTANT FREQUENCY BANDWIDTH IN THE FOURIER DOMAIN: R = CT = 64

B. CONSTANT COMPRESSION FACTOR BETWEEN THE ORIGINAL AND COMPRESSED IMAGE IN THE FOURIER DOMAIN: C = 128X128/PJ = CT = 6

CASE **A** USES THE EQUATIONS:

E=2	$(X-64)^2+(Y-64)^2=64^2$
E=1	X+Y=64 X-Y=64 X-Y=-64 X+Y=192
E=0.75	$(X-64)^{0.75}+(Y-64)^{0.75}=64^{0.75}$
E=0.5	$(X-64)^{0.5}+(Y-64)^{0.5}=64^{0.5}$

CASE **B** USES THE EQUATIONS:

E=2	$(X-64)^2+(Y-64)^2=29^2$
E=1	X+Y=92 X-Y=36 X-Y=-36 X+Y=165
E=0.75	$(X-64)^{0.75}+(Y-64)^{0.75}=44^{0.75}$
E=0.5	$(X-64)^{0.5}+(Y-64)^{0.5}=63^{0.5}$

- IT IS DONE THE FILTERING BY MULTIPLYING THE DISCRETE FOURIER TRANSFORM DFT WITH THE HILBERT MATRIX FILTER (4 VARIANTS) FOR CASE **A**.

SEE FIG 5

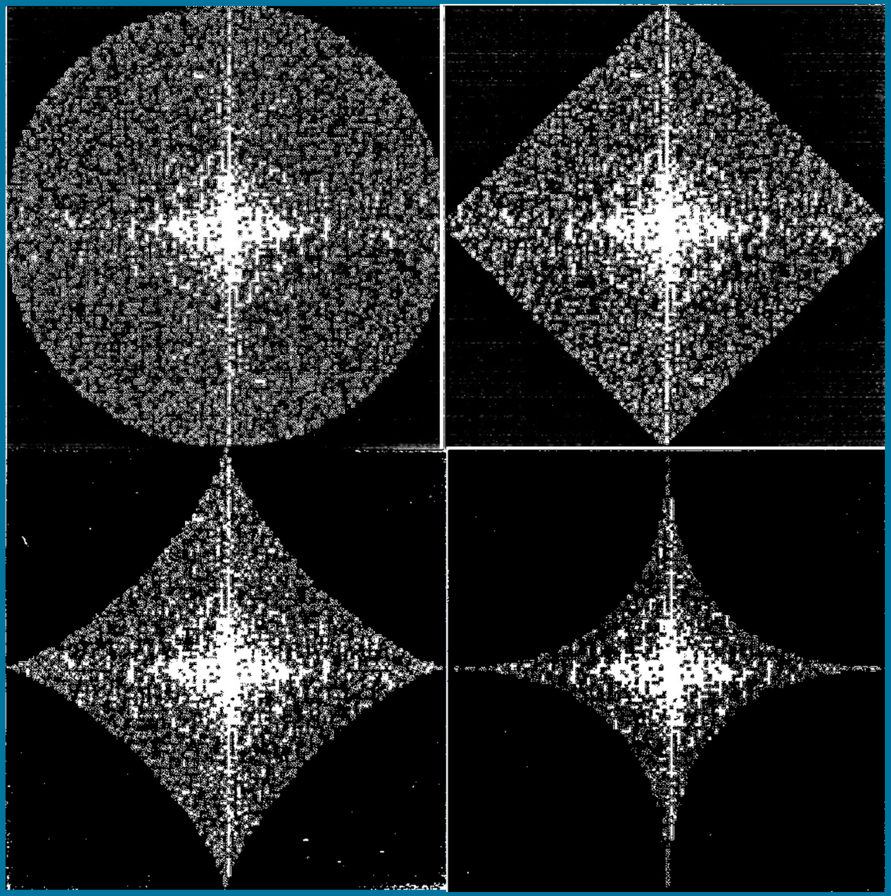


FIG 5

- IT IS DONE THE FILTERING BY MULTIPLYING THE DISCRETE FOURIER TRANSFORM DFT WITH THE HILBERT MATRIX FILTER (4 VARIANTS) FOR CASE B.

SEE FIG 6

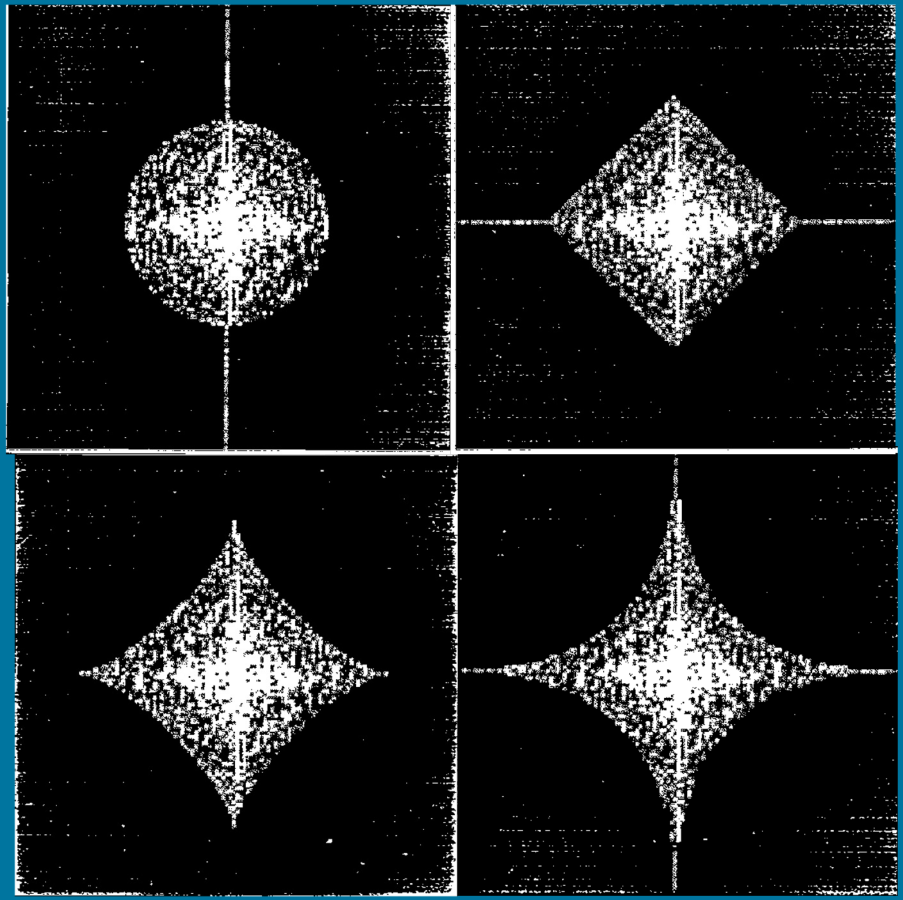


FIG 6

- IT IS VISUALIZED THE COMPRESSED IMAGE BY TAKING THE INVERSE DISCRETE FOURIER TRANSFORM IDFT OF THE FILTERED DFT FOR CASE A. SEE FIG 7

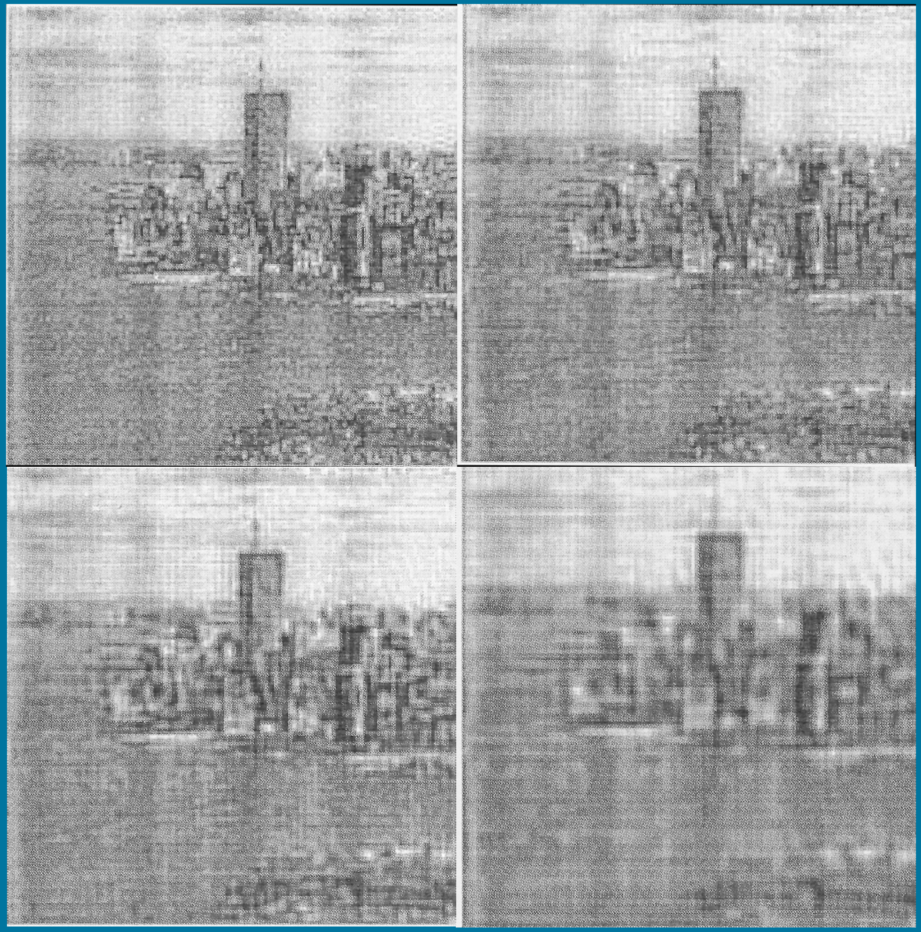


FIG 7

- IT IS VISUALIZED THE COMPRESSED IMAGE BY TAKING THE INVERSE DISCRETE FOURIER TRANSFORM IDFT OF THE FILTERED DFT FOR CASE B. SEE FIG 8

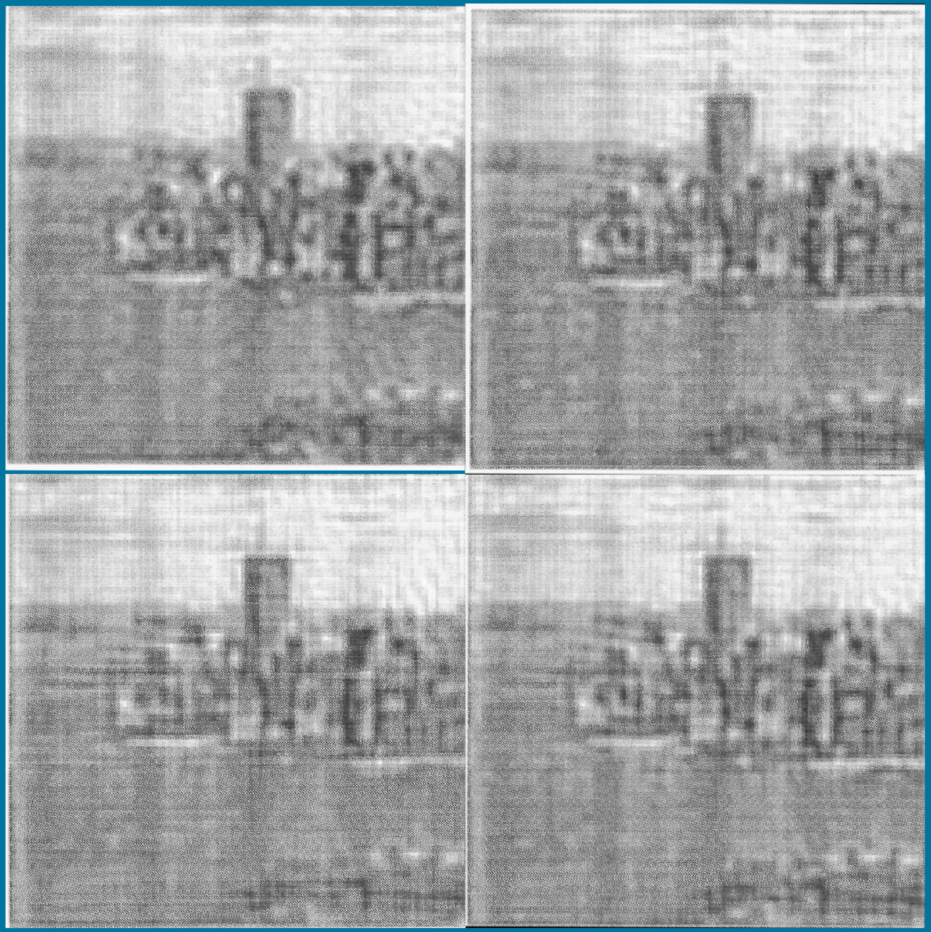


FIG 8

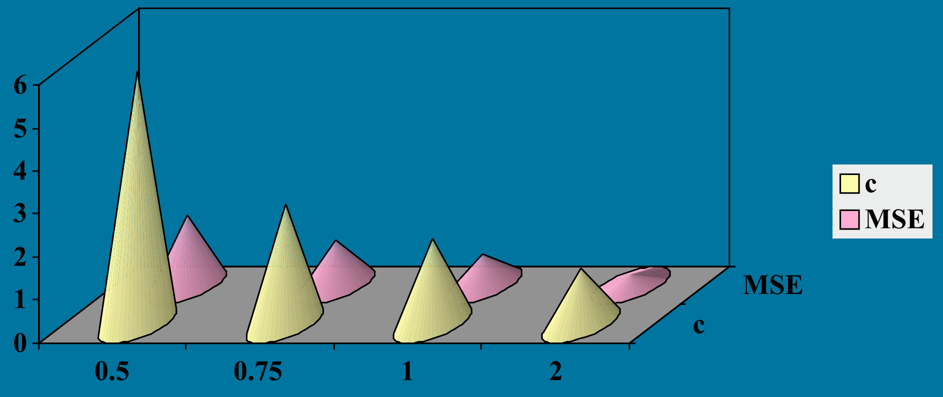
- CALCULATE THE REALIZED COMPRESSION FACTOR C(e) [CASE A]

- CALCULATE THE ERROR OF FILTERING MSE(e) [CASES A&B]

- DO GRAPHIC PRESENTATION FOR C(e) & MSE(e) [CASES A&B]

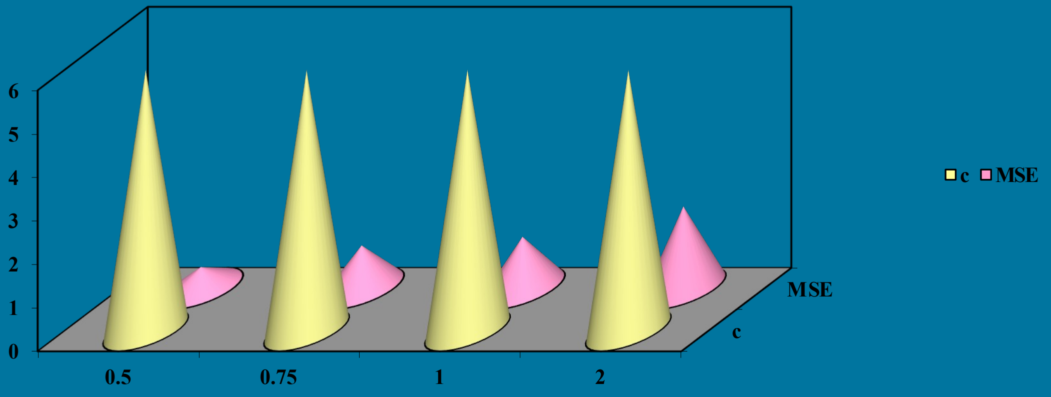
CASE A

E	2	1	0.75	0.5
C	1.3	2	2.8	5.9
MSE	0.0027	0.0073	0.0107	0.0163



CASE B

E	2	1	0.75	0.5
C	6	6	6	6
MSE	0.0177	0.01730	0.0168	0.0165



CONCLUSIONS & COMMENTS

CASE A - CONSTANT FREQUENCY BANDWIDTH OF THE IMAGE

THE SMALLEST NORMALISED *MEAN SQUARE ERROR* MSE=0.0027 IS OBTAINED WITH CIRCULAR FILTER FOR E=2.

BUT FOR E=2 THE COMPRESSION FACTOR IS THE SMALLEST C=1.3.

THE RESULTS SHOW THAT THE COMPRESSED IMAGES HAVE BETTER QUALITY (SMALLER ERROR) FOR CIRCULAR FILTERS BUT HAVE ALSO THE SMALLEST COMPRESSION FACTOR.

CASE B - CONSTANT COMPRESSION RATIO OF THE IMAGE

THE SMALLEST NORMALISED *MEAN SQUARE ERROR* MSE=0.0165 IS OBTAINED WITH STAR FILTER FOR E=0.5.

THE RESULTS SHOW THAT THE COMPRESSED IMAGES HAVE BETTER QUALITY (SMALLER ERROR) FOR STAR FILTERS WITH SMALLER EXPONENTS E.

THE EDGES AND OTHER SHARP TRANSITIONS IN THE GREY LEVELS OF AN IMAGE CONTRIBUTE SIGNIFICANTLY TO THE HIGH-FREQUENCY CONTENT OF ITS FOURIER TRANSFORM.

ALL IMAGES SHOWED IN THE PRESENTATION SUFFER AFTER DECOMPRESSION OF BLURRING OR SMOOTHING BECAUSE OF THE REDUCTION OF HIGH-FREQUENCY CONTENT.

FOR COMPRESSED IMAGES WITH LOW CONTRAST IT IS NECESSARY A FURTHER IMPROVEMENT, AN ENHANCEMENT, A CONTRAST STRETCHING.

MATLAB PROGRAM FOR STUDY CASE A

```
path('h:\',PATH)
a=loadbmp('h:\newyork');
viewim(a)
ap=a(1:128,1:128);                                        %square image
P=128*128                                                     %number of pixels of the digital image
viewim(ap)

b=fft2(ap);                                                   %Fourier transform of the image
bs=fftshift(b);
c=abs(bs);
m=max(c(:));
n=300*255/m;                                                  %normalization
c=n*c;
viewim(c)
bss=fftshift(bs);
a0=ifft2(bss);
a0=abs(a0);
viewim(a0)
D=sum(a0(:).^2)

for x=1:128                                                   %circular filter realized with a Hilbert Matrix
for y=1:128                                                   %[e=2  r=64]
H1(x,y)=1;
if y>sqrt(64.^2-(x-64).^2+64;
H1(x,y)=0;
end
if y>-sqrt(64.^2-(x-64).^2+64;
H1(x,y)=0;
end
end
end

d1=bs*H1;                                                     %filtering in frequency domain
d1a=n*abs(d1);
viewim(d1a)
d1s=fftshift(d1);
a1=ifft2(d1s);                                                %compressed image
a1=abs(a1);
viewim(a1)

N1=sum((a0(:)-a1(:)).^2);
MSE1=N1/D                                                     %normalised mean square error
                                                              %after filtering with the circular filter
P1=sum(H1(:));                                                %number of pixels in filter
C1=P/P1                                                       %compression factor (circle-filter  r=64)
```

```
for x=1:128                          %square filter realized      with a Hilbert Matrix[e=1  r=64]
for y=1:128
H2(x,y)=1;
if x<64;
if y<64-x;
H2(x,y)=0;
end
end
if x>64;
if y<x-64;
H2(x,y)=0;
end
end
if x<64;
if y>x+64;
H2(x,y)=0;
end
end
if x>64;
if y>192-x;
H2(x,y)=0;
end
end
end
end

d2=bs*H2;                            %filtering in frequency domain
d2a=n*abs(d2);
viewim(d2a)
d2s=fftshift(d2);
a2=ifft2(d2s);                       %compressed image
a2=abs(a2);
viewim(a2)

N2=sum((a0(:)-a2(:)).^2);
MSE2=N2/D                            %normalised mean square error
                                                 %after filtering with the square-filter
P2=sum(H2(:));                       %number of pixels in filter
C2=P/P2                              %compression factor (square-filter r=64)
```

```
for x=1:128                                          %star filter realized with a Hilbert Matrix[e=0.75  r=64]
for y=1:128
H3(x,y)=1;
if y<-(64.^0.75-(x-64).^0.75).^(4/3)+64;
H3(x,y)=0;
end
if y>(64.^0.75-(x-64).^0.75).^(4/3)+64;
H3(x,y)=0;
end
if y<-(64.^0.75-(64-x).^0.75).^(4/3)+64;
H3(x,y)=0;
end
if y>(64.^0.75-(64-x).^0.75).^(4/3)+64;
H3(x,y)=0;
end
end
end

d3=bs*H3;                                            %filtering in frequency domain
d3a=n*abs(d3);
viewim(d3a)
d3s=fftshift(d3);
a3=ifft2(d3s);                                       %compressed image
a3=abs(a3);
viewim(a3)

N3=sum((a0(:)-a3(:)).^2);
MSE3=N3/D                                            %normalised mean square error
                                                     %after filtering with the star filter
P3=sum(H3(:));                                       %number of pixels in filter
C3=P/P3                                              %compression factor (star-filter  e=0.75  r=64)
```

```
for x=1:128                                  %star filter realized with a Hilbert Matrix[e=0.5  r=64]
for y=1:128
H4(x,y)=1;
if y<-(64.^0.5-(x-64).^0.5).^2+64;
H4(x,y)=0;
end
if y>(64.^0.5-(x-64).^0.5).^2+64;
H4(x,y)=0;
end
if y<-(64.^0.5-(64-x).^0.5).^2+64;
H4(x,y)=0;
end
if y>(64.^0.5-(64-x).^0.5).^2+64;
H4(x,y)=0;
end
end
end

d4=bs*H4;                                    %filtering in frequency domain
d4a=n*abs(d4);
viewim(d4a)
d4s=fftshift(d4);
a4=ifft2(d4s);                               %compressed image
a4=abs(a4);
viewim(a4)

N4=sum((a0(:)-a4(:)).^2);
MSE4=N4/D                                    %normalised mean square error
                                             %after filtering with the star-filter
P4=sum(H4(:));                               %number of pixels in filter
C4=P/P4                                      %compression factor (star-filter  e=0.5  r=64)
```

MATLAB PROGRAM FOR STUDY CASE B

```
path(‘h:\’,PATH)
a=loadbmp(‘h:\newyork’);
viewim(a)
ap=a(1:128,1:128);                    %square image
P=128*128                             %number of pixels of the digital image
viewim(ap)

b=fft2(ap);                           %Fourier transform of the image
bs=fftshift(b);
c=abs(bs);
m=max(c(:));
n=300*255/m;                          %normalization
c=n*c;
viewim(c)
bss=fftshift(bs);
a0=ifft2(bss);
a0=abs(a0);
viewim(a0)
D=sum(a0(:).^2)

for x=1:128                           %circular filter realized with a Hilbert Matrix[e=2  c=6]
for y=1:128
H1(x,y)=1;
if y>sqrt(29.^2-(x-64).^2+64;
H1(x,y)=0;
end
if y<-sqrt(29.^2-(x-64).^2+64;
H1(x,y)=0;
end
end
end

d1=bs*H1;                             %filtering in frequency domain
d1a=n*abs(d1);
viewim(d1a)
d1s=fftshift(d1);
a1=ifft2(d1s);                        %compressed image
a1=abs(a1);
viewim(a1)

N1=sum((a0(:)-a1(:)).^2);
MSE1=N1/D                             %normalised mean square error
                                      %after filtering with the circular filter
P1=sum(H1(:));                        %number of pixels in filter
C1=P/P1                               %compression factor (circle-filter  c=6)
```

```
for x=1:128                                      %square filter realized with a Hilbert Matrix[e=1  c=6]
for y=1:128
H2(x,y)=1;
if x<64;
if y<92-x;
H2(x,y)=0;
end
end
if x>64;
if y<x-36;
H2(x,y)=0;
end
end
if x<64;
if y>x+36;
H2(x,y)=0;
end
end
if x>64;
if y>165-x;
H2(x,y)=0;
end
end
end
end

d2=bs*H2;                                        %filtering in frequency domain
d2a=n*abs(d2);
viewim(d2a)
d2s=fftshift(d2);
a2=ifft2(d2s);                                   %compressed image
a2=abs(a2);
viewim(a2)

N2=sum((a0(:)-a2(:)).^2);
MSE2=N2/D                                        %normalised mean square error
                                                 %after filtering with the square-filter
P2=sum(H2(:));                                   %number of pixels in filter
C2=P/P2                                          %compression factor (square-filter  c=6)
```

```
for x=1:128                                        %star filter realized with a Hilbert Matrix[e=0.75  c=6]
for y=1:128
H3(x,y)=1;
if y<-(44.^0.75-(x-64).^0.75).(4/3)+64;
H3(x,y)=0;
end
if y>(44.^0.75-(x-64).^0.75).(4/3)+64;
H3(x,y)=0;
end
if y<-(44.^0.75-(64-x).^0.75).(4/3)+64;
H3(x,y)=0;
end
if y>(44.^0.75-(64-x).^0.75).(4/3)+64;
H3(x,y)=0;
end
end
end

d3=bs*H3;                                          %filtering in frequency domain
d3a=n*abs(d3);
viewim(d3a)
d3s=fftshift(d3);
a3=ifft2(d3s);                                     %compressed image
a3=abs(a3);
viewim(a3)

N3=sum((a0(:)-a3(:)).^2);
MSE3=N3/D                                          %normalised mean square error
                                                   %after filtering with the circular filter
P3=sum(H3(:));                                     %number of pixels in filter
C3=P/P3                                            %compression factor (star-filter  e=0.75  c=6)
```

```
for x=1:128                                   %star filter realized with a Hilbert Matrix[e=0.5  c=6]
for y=1:128
H4(x,y)=1;
if y<-(63.^0.5-(x-64).^0.5).^2+64;
H4(x,y)=0;
end
if y>(63.^0.5-(x-64).^0.5).^2+64;
H4(x,y)=0;
end
if y<-(63.^0.5-(64-x).^0.5).^2+64;
H4(x,y)=0;
end
if y<-(63.^0.5-(64-x).^0.5).^2+64;
H4(x,y)=0;
end
end
end

d4=bs*H4;                                     %filtering in frequency domain
d4a=n*abs(d4);
viewim(d4a)
d4s=fftshift(d4);
a4=ifft2(d4s);                                %compressed image
a4=abs(a4);
viewim(a4)

N4=sum((a0(:)-a4(:)).^2);
MSE4=N4/D                                     %normalised mean square error
                                              %after filtering with the star-filter
P4=sum(H4(:));                                %number of pixels in filter
C4=P/P4                                       %compression factor (star-filter  e=0.5  c=6)
```