SHIELD Briefing
201100JUN14S

Code Name:
SUMAC14

(Space Unidentified Mass Anomaly Caption 2014)
**Time Line:**

090800JUN14S-SUMAC14 Detected by Hubble Telescope, 3,118,900 miles out, with a heading towards Earth. Late detection due to object shadowing planets and moons.

Action: ***Avengers Activated*** and MATLAB Initiated

**CAPT America** used Edge Detection to some affect.

100630JUN14S-SUMAC14 Advanced with an average speed of 24,000mph.

2,542,900mi. Last image taken by Hubble, tracked by long range laser detectors.

Action: **Ironman** employs If-then statements, Histograms, HSV(Hue, Saturation, Brightness Value) with some progress in color recognition.

111000JUN14S-SUMAC14 1,966,900mi out and closing.

Action: **Black Widow** institutes Noise Reduction with filters. Anomaly image blur gradually being defined.

120945JUN14S-SUMAC14 1,390,900mi out and continued course toward Earth.

Action: **Spiderman** utilizes Distance technique to further attack the issue.

131205JUN14S-SUMAC14 815,000mi out.

Action: **Avengers** combine forces to seek out the anomaly called SUMAC14

141700JUN14S-SUMAC14 held position behind the Moon, 240,000mi.

151300JUN14S-SUMAC14 No Change in position, shadowing the Moon.

161330JUN14S-SUMAC14 continued towards Earth at 6,300mph, 88.800mi out.

171600JUN14S-SUMAC14 slowed to 4,000mph, 40,000mi out.

180700JUN14S-SUMAC14 continued to slow to 2,000mph, 20,000mi out.

192000JUN14S-SUMAC14 orbited Earth at 200mi. above the surface.

201100JUN14S-SUMAC14 entered Earth’s atmosphere, contact made, and Identified.
By: Lily Hernandez, Courtney Pierce, Cameron Pierce, Chandler Neinast
Edge Detection

- Edge Detection is used to find the boundaries of objects within images.
- It works by detecting discontinuities in brightness within the image.

Sobel Filter

Canny Filter
Edge Detection

```matlab
axis equal off;
colormap gray

%% Edge Detection
load('newdeal.mat');
A=I(50);
rgb2gray(A);
B=rgb2gray(A);
Ino=im2double(imnoise(B,'salt & pepper',0));
A2=edge(Ino);
A3=edge(Ino,'canny');
figure(21);
imshow(A)
figure(22);
imshow(A2)
figure(23);
imshow(Ino)
figure(24);
imshow(A3)
```

Figure 21: Original image
Figure 22: Grayscale image
Figure 23: Image with noise
Figure 24: Edge detection with 'canny' method
Edge Detection with Noise

Original Photo

No Noise

Half Noise

Full Noise
Edge Detection Over a Grayscale Image
Color Intensities and Hues

• We used hues and colors to represent pixel intensities in RGB images.
• Experimentation with colors for image histograms and overall image changes.
• Black & White pixels in an RGB system.
• The Red, green, and blue coordinates can reach numbers up to 255 (white).
• These separate color coordinates are used to isolate a certain color to create a histogram. This was needed to see how many pixels held a hue or color compared to how much there was of another color.
Separating the colors is useful because it represents how every color intensity is pieced together to form the image that we see.
A hue histogram is also useful because it represents how the image is made up of not just one color. It shows that each color is split into different hues and saturation to create a new concoction of colors.
load('newdeal.mat');
A=I{8};
figure(1);
imshow(A);

Ir=A(:,:,1);
Ig=A(:,:,2);
Ib=A(:,:,3);

figure(2);
imhist(Ir);
figure(220); imshow(Ir);

figure(3);
imhist(Ig);
figure(33); imshow(Ig);

figure(4);
imhist(Ib);
figure(44); imshow(Ib);
Image Histograms (separate RGB)
Noise Reduction

- **Averaging** - Has the power to reduce noise in an image without compromising any details.

- **Gaussian** - Has the effects of blurring or smoothing.

- **Salt & Pepper** - Causes pixels to ruin.
Noise in Images

- High in intensity (Unwanted intensity)
- Film grain
- Due to corruption in scanning process
- Digital Camera
- Electronic transfer
Load('newdeal.mat');  
-MATLAB reads this certain mat. file, and opens it

A=[8];  
-Tells MATLAB to use this exact Photo

Igray=rgb2gray(Igray)  
-Allows the original image to be gray

Imshow(A)  
-Show the image that MATLAB will focus on

Ino{ind}=im2double(imnoise(Igray,'Gaussian',0.5));  
Ino{ind}=im2double(imnoise(Igray,'salt & pepper',0.3));  
-These add filters to the image MATLAB is focusing on

-The numbers in the commands determines the value of Gaussian, or salt & pepper
Ex:
• Figure(2)
  imshow(A);
  title('Original')

-Three as such

-Allows there to show each of the same image with noise, as an average, and the original.
```matlab
load('newdeal.mat');
A=I(8);
Igray=rgb2gray(A);
imshow(Igray);
m=100;
Aavg=0;

for ind=1:m
    Ino{ind}=im2double(imnoise(Igray,'gaussian',0.5));
    Ino{ind}=im2double(imnoise(Igray,'salt & pepper',.3));
    Aavg=Aavg+Ino{ind}/m
end

Iavg=Iavg/m;

figure(2)
imshow(A);
title('Original')

figure(3)
imshow(Ino{1});
title('With Noise')
```
load('newdeal.mat');
A=I(8);
Igray=rgb2gray(A);
imshow(Igray);

m=100;
Aavg=0;

for ind=1:m
    Ino{ind}=im2double(imnoise(Igray,'gaussian',0.05));
    Ino{ind}=im2double(imnoise(Igray,'salt & pepper',.0));
    Aavg=Aavg+Ino{ind}/m
end

lavg=lavg/m;

figure(2)
imshow(A);
title('Original');

figure(3)
imshow(Ino{1});
title('With Noise');
load('newdeal.mat');
A=I(30);
imshow(A)

m=100;
Aavg=0;

for ind=1:m
    Ino{ind}=im2double(imnoise(A,'gaussian',0.5));
    %Ino{ind}=im2double(imnoise(A,'salt & pepper',0.3));
    Aavg=Aavg+Ino{ind}/m;
end

%Iavg=Iavg/m;

figure(2)
imshow(A);
title('Original')

figure(3)
imshow(Ino{1});
title('With Noise')

figure(4)
imshow(Aavg);
load('newdeal.mat');
A=I{30};
imshow(A)

m=100;
Aavg=0;

for ind=1:m
    Ino{ind}=im2double(imnoise(A,'gaussian',0.05));
    %Ino{ind}=im2double(imnoise(A,'salt & pepper',0.05));
    Aavg=Aavg+Ino{ind}/m
end

%Iavg=Iavg/m;

figure(2)
imshow(A);
title('Original')

figure(3)
imshow(Ino{1});
title('With Noise')
load('newdeal.mat');
A=I(47);
inshow(A)

m=100;
Aavg=0;

for ind = 1:m
    %ino{ind}=im2double(imnoise(A,'gaussian',0.2));
    Ino{ind}=im2double(imnoise(A,'salt & pepper',0.3));

    Aavg=Aavg+Ino{ind}/m;
end

%Iavg=Iavg/m;

figure(2)
inshow(A);
title('Original')

figure(3)
inshow(Ino{1});
title('With Noise')

figure(4)
inshow(Aavg);
load('newdeal.mat');
A=I(47);
imshow(A)

m=100;
Aavg=0;

for ind=1:m
    % Ino{ind}=im2double(imnoise(A,'gaussian',0.05));
    Ino{ind}=im2double(imnoise(A,'salt & pepper',.0));

    Aavg=Aavg+Ino{ind}/m
end

% Iavg=Iavg/m;

figure(2)
imshow(A);
title('Original')

figure(3)
imshow(Ino{1});
title('With Noise')
Distance

- Multiple observations of images are used to measure the distance between images.
- Two types of distance formulas were used in our observations.
  - Euclidean
    - \( d_{\text{e}}(A,B) = \sqrt{\sum \sum (A_{ij} - B_{ij})^2} \)
  - Manhattan
    - \( d_{\text{m}}(A,B) = \sum \sum |A_{ij} - B_{ij}| \)
- We used these equations to show exactly how different the pictures are in comparison.
- Distance is used to clearly define the difference between two images. Everyone has their own opinions on what an image looks like so this allows everyone to more or less see the image in the same way numerically.
Red

Original

With a small amount of noise added

With a lot of noise added

dE = $1.0832 \times 10^4$

dE = $8.3419 \times 10^4$

dM = 83034

dM = 254620
Green

Original

With a small amount of noise added

With a lot of noise added

dE = 8.4850e+03
dM = 72656

dE = 8.2596e+04
dM = 264152
Blue

Original

With a small amount of noise added

With a lot of noise added

\[ dE = 6.3773 \times 10^3 \]

\[ dM = 66194 \]

\[ dE = 8.1628 \times 10^4 \]

\[ dM = 273982 \]
Images

With a lot of noise added

With a small amount of noise added

No noise
Questions?