Good morning all, today, my colleague, Nitesh Nana, and I, are here to present our project of ***Image Compression based on Non-Parametric Sampling in Noisy Environments***.

In this presentation I will begin with the objectives and specifications of the project. Following this I discuss the overview of the system, and its implementation. My colleague will then discuss the results and give an analysis of the project looking at 4 different test cases. Lastly, giving recommendations for future work and the conclusion.

***Objectives and Specifications:***

* The objectives of the project are to:
  + Create a robust image compression scheme that
    - Creates holes in the image
    - Encodes the image for transmission
    - Introduce random errors through a simulated channel
    - Filling the subsequent holes
* The project must be competed
  + Within a 6 week time frame
  + An allocated budget of R1200
  + And requires ethical clearance from the university

***System Overview:***

* The block diagram presents the overall implementation of our compression schemes
* The green blocks present everything that relates to storing and displaying the image
* The red blocks show the creation of holes and filling of holes.
* The orange blocks indicate the channel where encoding, the introduction of errors and decoding occur.
* The blue block indicates the known technique of DCT that added a higher level of complexity and compression to the compression technique.
* There are 3 compression schemes seen in this slide:
  + The holes only scheme where holes are created in the image, sent through the channel and filled and reconstructed
  + The DCT only scheme where DCT compression is performed on the image, sent through the channel and displayed
  + The combination of holes and DCT which creates holes in the image, compresses using DCT, send the image through the channel, fills the holes and displays the compressed image.

***System Implementation (Initial Steps):***

* All three compression schemes undergo the same initial and ending stages
* Any image that is read into MATLAB has 3 dimensions
  + Height of the image
  + Width of the image
  + Colour map
* All values are 8-bit unsigned integers.
* To reduce complexity all images are converted to grayscale that averages out the colour map, leaving only 2 dimensions:
  + The height
  + The width
* The image is then divided into a domain pool of 8x8 pixel blocks, with each block being put into an array indexing the top left as 1, moving left to right, top to bottom

***System Implementation (Creating the Holes):***

* Each block in the domain pool is indexed as follows.
* According to current literature, no methods are given on how to create holes in an image
* We use DCT and block-based fractal image compression as the foundation of our algorithm
* We start by going to the blue square
  + Calculate the average pixel value of the blue square
  + We then compute the Chebyshev distance of the average to each pixel in the square
  + If the distance for all are less than 6, a hole can be created in the blue square
  + A hole is defined as each pixel in the square having a value of 0
  + Before the hole is created, the same process is done for the orange and green squares to determine the size of the hole created
* The Chebyshev distance of 6 is chosen
  + As 10 simulations were done comparing different values from 1 to 10
  + 6 was chosen as it created a relevant number of holes and the final reconstructed image showed the best results

***System Implementation (Encoding):***

* The encoding scheme utilized is Run-Length encoding
* Our implementation works on each row in the domain pool squares
* Two data sets are saved and transmitted, the pixel value and the corresponding count
* Run length encoding is efficient in repetitive data, such as images with repeating colours
* However very smooth transitions of colour will not work well as pixel values will change by increments of 1.
* Because each pixel will be regarded as different values run length encoding performs poorly
* Along with this, since all pixels are 8-bit unsigned integers, only the relevant number of bits are converted for transmission.
* For example, a pixel with a value of 10 only requires 4 bits as opposed to the 8 allocated.
* Converting the necessary bits is where we see the compression

***System Implementation (Error Introduction):***

* Errors are introduced in one of two ways.
* A bit flip or a ones compliment
* Both introduction methods are randomly chosen based on a probability of an error being introduced
* Along with this, the assumption is made that only the pixel values are affected and not the count
* The bit flip can randomly affect the most significant or least significant or anything in between
* The ones compliment affects the entire number, inverting it completely showing a bigger effect as the value of the pixel is higher.
* The images show the discrepancy of colour in these types of error introduction

***System Implementation (Filling the Holes):***

* Current literature fills holes in the image using pattern matching
  + A similar area of colour matching the surrounding area in the hole is used to fill the holes
* The created technique uses an averaging method instead and the reverse of the creation of holes by checking the Chebyshev distance for the average and each pixel to identify a hole.
* Starting at the top left pixel of the hole, the average pixel of the pixel directly above, directly to the left, and directly above left are calculated and filled into the pixel
* This is done for each pixel in the hole moving left to right, top to bottom

I will now hand over to Nitesh who will discuss the results and give an analysis, the future work and recommendations and concluding arguments.

***Results & Analysis:***

* 3 types of images are tested in the various test cases
  + 3 Pattern images are chosen for their repetitive nature in colour
  + 3 Landscape images are chosen for the details and edges
  + 3 High contrast images are chosen for the repetitive colour, details and edges.
* 4 test cases are done on the different types of images

***Results (Average Compression Ratio):***

* The first test case tests the functionality of the holes only algorithm
* With no errors introduced, the average image resolution is chosen to be around 1200x800 pixels
* This is a pattern image that shows the steps of the holes only algorithm
* As well as the landscape image
* And a high contrast image
* As can be seen from the bar graph, the average compression ratio for the:
  + High contrast images are 2.1
  + Landscape images are 1.7
  + Pattern images are 1.3

***Results (Same Image at Different Resolutions):***

* The second test case tests how effective the hole only algorithm is in terms of compression ratio
* The image size varies as the original image size is approximately 2000x1100 pixels, with a 10% chance of errors being introduced using bit flip introduction
* The idea is that the large the image size, the greater the compression ratio
* However from the line graph, the pattern images do not follow that trend
* The reason for this:
  + Because of the smooth transition of colour, like my colleague mentioned, the run length encoding is not efficient for images with such smooth transitions in colour.
  + The landscape images are shown to have somewhat of a constant compression ratio.
  + The high contrast images are the only images that follow the prediction and it stands to reason as these image types are deemed a combination of the features of landscape and pattern type images.
* This is an example of a pattern image at different sizes
  + As you can see at 20% of the original size, the errors are noticeable
  + As the size of the image increases, the errors are present but less noticeable.

***Results (PSNR & MSE):***

* The third test case test the effect of an increasing error probability on all algorithms
* The image size is approximately 1200x800 pixels, with both bit flip and ones compliment being introduced.
* The prediction for this test case is that as the probability of error increases, so would the mean square error.
* The psnr should decrease as the probability increases, which measured in decibels used to identify perceived errors noticeable by human vision
* Any positive psnr is deemed good as it proves the signal of the original image is still recognizable over the noise.
* These graphs show the trend of pattern images:
  + As the error probability increases above 20%, the holes and holes+dct scheme yield better PSNR results
  + Since ones compliment error introduction affects the number by inverting its binary value, its PSNR is significantly lower and its MSE is significantly higher
* The same trend is seen for landscape images and high contrast image
* A larger discrepancy between the DCT only and schemes involving the holes method is seen in the landscape images.

***Results (Comparison):***

* The final test case is a comparison of the holes only algorithm against DCT only and DWT
* The probability of error being introduced is 10% with bit flip error introduction.
* The images tested are known images:
  + Cameraman
  + Mandrill
  + Peppers
* The table presents average values for all images being tested on each compression scheme
* As one can see the DCT only scheme yields the best result for PSNR
* Although DWT has the best performance for compression, the holes algorithm yields the best results in terms of quality of final compressed image.

***Future Work:***

* In our implementation, the programming was done serially to ensure that the implemented system would work
  + Trade off of this is that the processing time of the implemented system is too long
  + A full hd image takes approximately 1 hour and 10 minutes to compress
* In order to reduce the processing time, parallel programming can be implemented.
* With run length encoding having certain drawbacks, to improve compression a different encoding scheme can be utilized
* Final recommendation would be to train a neural network for the creation and filling holes
  + A trained neural network would be able to determine the distance value for creating and filling the holes.
  + Along with this, a trained neural network for filling holes could have applications similar to that of the GAN concept
  + One could potentially take the trained neural network and maintain image quality by increasing the resolution of lower resolutions images.

***Conclusion:***

* In conclusion, all objectives have been met within the project specification
* The created algorithm proved to be functional
* Able to maintain image quality while still ultimately compressing the image.
* All images were taken from Unsplash.com as this site provides photographs that are free to use and manipulate for any use.