

# Automated Identification of Potholes Using Remotely Sensed Data

MEGHAN KULKARNI

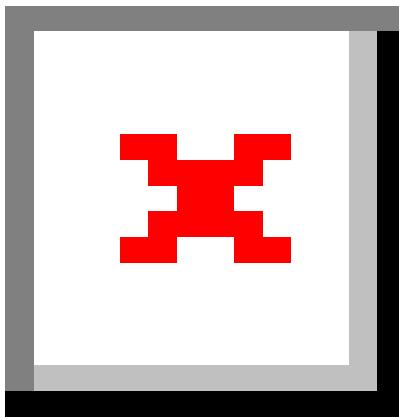
# Introduction

Transportation infrastructure is a key element driving economic growth in developing countries. World-class expressways and highways make transportation efficient. However over time this infrastructure requires maintenance. Potholes or cracks cause safety hazards and present dangerous conditions for motorists. UAV technology can be utilized to remotely capture real time road condition data. This project aims to investigate digital processing alternatives for feature identification.



# Purpose

Evaluate automated identification of roadway damage to increase efficiency of UAV employment using nearest neighbor algorithm to distinguish target features.



# Rationale

Methods must be developed to coordinate UAV flight paths pre-programmed with GPS routes according to roadway infrastructure. This aspect poses no particular issues given the reliability of conventional route programming with GPS enabled UAV units.

This proposal utilizes static images as an alternative methodology to live video for the purpose of increasing automated efficiency. In other words, this proposal intends to eliminate the need for active monitoring by a UAV pilot.

# Background

Potholes present transportation challenges often requiring constant need for repair and maintenance. Proliferation of UAV technology across a variety of applications has emerged as an economical and robust platform with which to increase roadway maintenance efficiency as well as public safety.

The present challenge concerns reliable automated identification of distressed areas of roadway infrastructure. In methods proposed by Lokeshwor Huidrom, et.al:

“pothole, cracks and patches from real life video clips of highways are detected and quantified automatically using various image processing techniques supported by heuristically derived decision logic.”

# Hypothesis

Digital still frame images in the visual spectrum captured with UAV's can be processed using the Nearest Neighbor (8 pixel) formula to distinguish between pixel values for the purpose of identifying particular features – in this case areas of roadway in need of maintenance or repair.

# Methodology

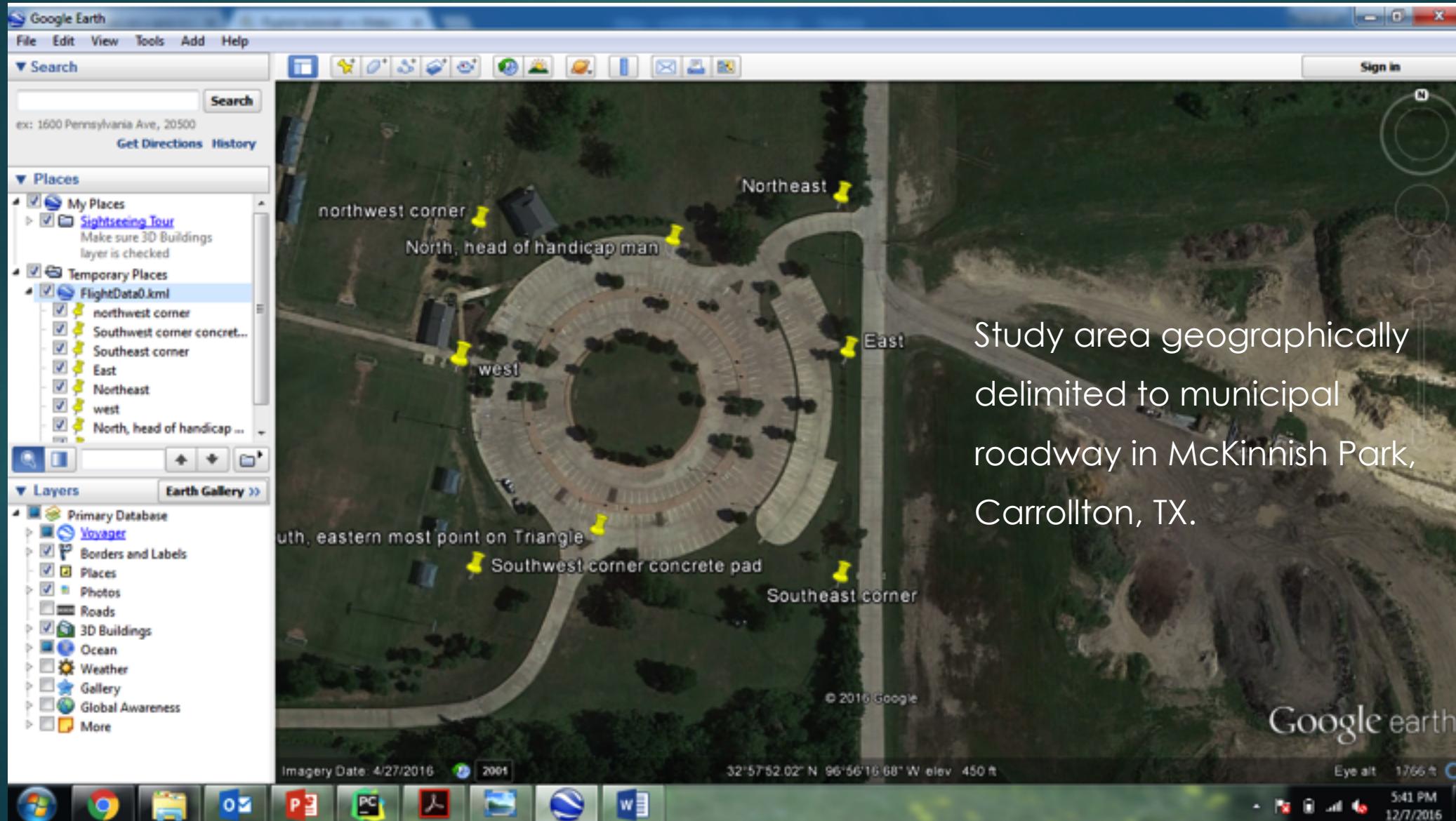
## Image Acquisition

- ▶ Acquire digital image of target feature – in this case Google Earth imagery
- ▶ Convert radiance data band (BGR) values to digital number format (DN)
- ▶ Process data in Python script via OpenCV library

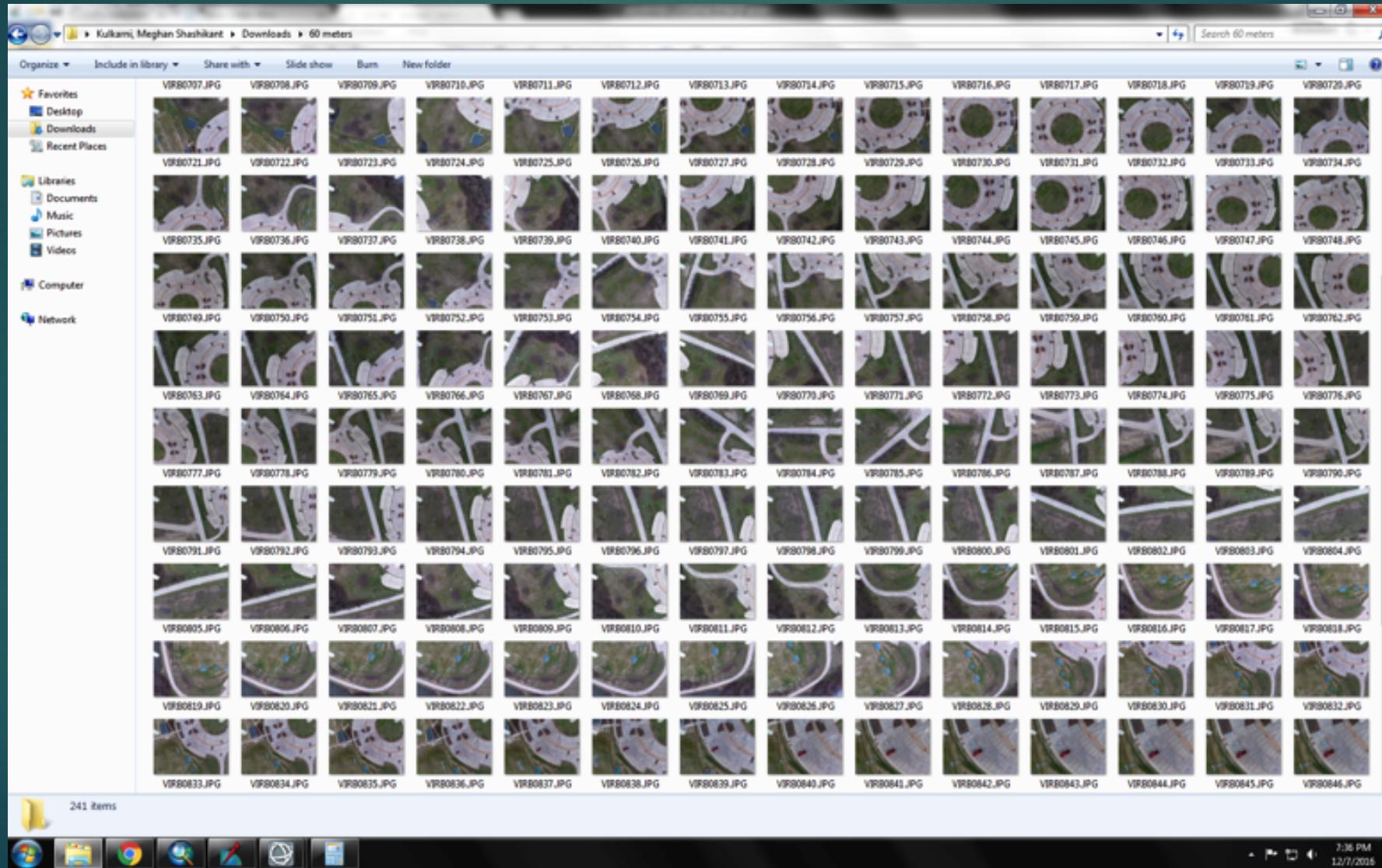
## Statistical Modeling

- ▶ Generate histogram by plotting BGR DN values
- ▶ Analyze histogram for max/min DN values
- ▶ Compare/contrast multiple digital image samples

# Methodology – Image Acquisition



# Methodology – Image Acquisition



# Methodology – Coding & Analysis

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** RSProject - [C:\Users\Tulip\PycharmProjects\RSProject] - ImageStudy2.py - PyCharm 2016.2.3
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** Standard toolbar with icons for file operations.
- Project View:** Shows the project structure under "RSProject": ImageStudy.py, ImageStudy2.py, Lab9.py, lab14.py, MatplotlibTest.py, VideoStudy.py, and External Libraries.
- Code Editor:** The "ImageStudy2.py" file is open, displaying Python code for image processing using cv2 and numpy. The code reads a JPEG image, prints its dimensions, and extracts a region of interest (ROI) from row 2800 to 3303 and column 300 to 803. It then prints the maximum and minimum values of the ROI.

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread(r"D:\GIS-MS\Fall 2016\Remote Sensing Fundamentals\Project\60 meters\VIRB0704.JPG", cv2.IMREAD_COLOR)

# data contains 3456 rows and 4608 columns and 3 bands Blue, Green and Red
rows, cols, channels = img.shape

print(rows)
print(cols)
print(channels)

# roi = img[0:rows, 0:cols, 0:channels]
# roi = img[0:rows, 0:cols, 0:channels]
roi = img[2800:3303, 300:803, 0:channels]

max_value = roi.max()
min_value = roi.min()

print(max_value)
print(min_value)
```

- Run Tab:** Shows the command "C:\Users\Tulip\Anaconda2\python.exe C:/Users/Tulip/PycharmProjects/RSProject/ImageStudy2.py" and the resulting output:
  - 3456
  - 4608
  - 3
  - 255
  - 29
  - 0,145,0
  - 0,145,1
  - 0,145,2
  - 0,146,0
  - 0,146,1
- Bottom Status Bar:** Platform and Plugin Updates: PyCharm is ready to update. (today 1:56 PM)
- Bottom Right:** Encoding: UTF-8, Line Endings: CRLF.

# Methodology – Coding & Analysis

The screenshot shows the PyCharm IDE interface with the following details:

- Title Bar:** RSProject - [C:\Users\Tulip\PycharmProjects\RSProject] - ImageStudy2.py - PyCharm 2016.2.3
- Menu Bar:** File Edit View Navigate Code Refactor Run Tools VCS Window Help
- Toolbars:** Standard toolbar with icons for file operations.
- Project View:** Shows the project structure under "RSProject":
  - ImageStudy.py
  - ImageStudy2.py
  - Lab9.py
  - Lab14.py
  - MatplotlibTest.py
  - VideoStudy.py
- Code Editor:** The "ImageStudy2.py" file is open, displaying Python code for image processing. The code includes nested loops to iterate over a region of interest (ROI) and conditional logic to check pixel values. A yellow highlight covers the bottom portion of the loop body.

```
# Clipping small cross section of image to optimize image processing in below code
img2 = roi[0:503, 0:503]

for i in range(0, 500):
    for j in range(0, 500):
        for k in range(0,channels):
            # Nearest Neighbor 8 pixel checking code below
            px1 = roi[i, j]
            px2 = roi[i, j-1]
            px3 = roi[i, j+1]
            px4 = roi[i+1, j]
            px5 = roi[i-1, j]
            px6 = roi[i-1, j-1]
            px7 = roi[i+1, j+1]
            px8 = roi[i-1, j+1]
            px9 = roi[i+1, j-1]
            if px1[0] > 200 and px1[1] > 200 and px1[2] > 200 and \
               px2[0] > 200 and px2[1] > 200 and px2[2] > 200 and \
               px3[0] > 200 and px3[1] > 200 and px3[2] > 200 and \
               px4[0] > 200 and px4[1] > 200 and px4[2] > 200 and \
               px5[0] > 200 and px5[1] > 200 and px5[2] > 200 and \
               px6[0] > 200 and px6[1] > 200 and px6[2] > 200 and \
               px7[0] > 200 and px7[1] > 200 and px7[2] > 200 and \
               px8[0] > 200 and px8[1] > 200 and px8[2] > 200 and \
               px9[0] > 200 and px9[1] > 200 and px9[2] > 200 and \
```
- Run Tab:** Shows the command: C:\Users\Tulip\Anaconda2\python.exe C:/Users/Tulip/PycharmProjects/RSProject/ImageStudy2.py. The output window displays:

```
3456
4608
3
255
29
0,145,0
0,145,1
0,145,2
0,146,0
0,146,1
```
- Bottom Status Bar:** Platform and Plugin Updates: PyCharm is ready to update. (today 1:56 PM)

# Limitations/Delimitations

## Limitations

- ▶ Employment of reliable UAV platform for adequate digital image acquisition is assumed, but not employed due to availability.

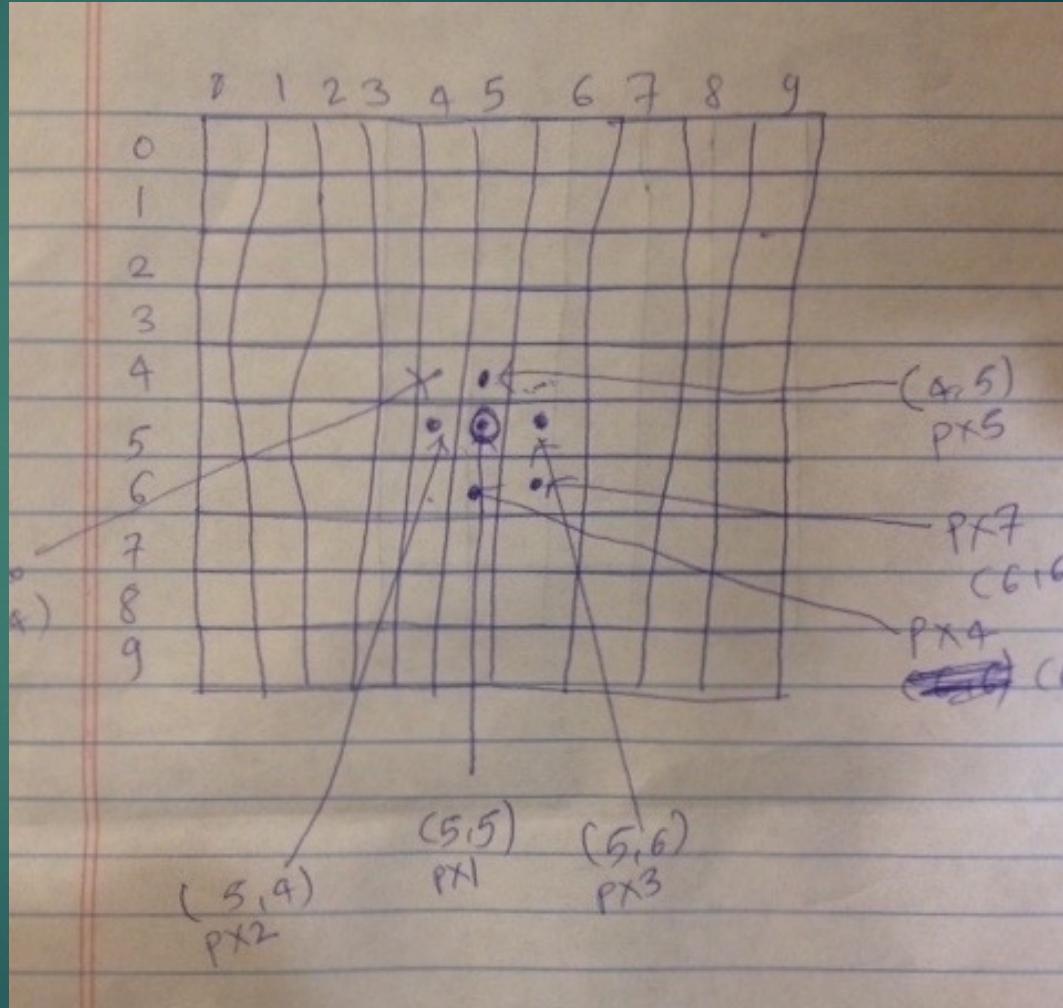
## Delimitations

- ▶ Data corresponds to municipal roadway in McKinnish Park, Carrollton, TX.
- ▶ Limiting study to concrete (high reflectance) roads, pavement/asphalt roadways not included for preliminary findings.

# Definitions (Key Terms)

## Nearest Neighbor (8 pixel) Formula

Nearest neighbor interpolation is the simplest approach to interpolation. Rather than calculate an average value by some weighting criteria or generate an intermediate value based on complex rules, this method simply determines the “nearest” neighboring pixel, and assumes the intensity value of it.



# Tools

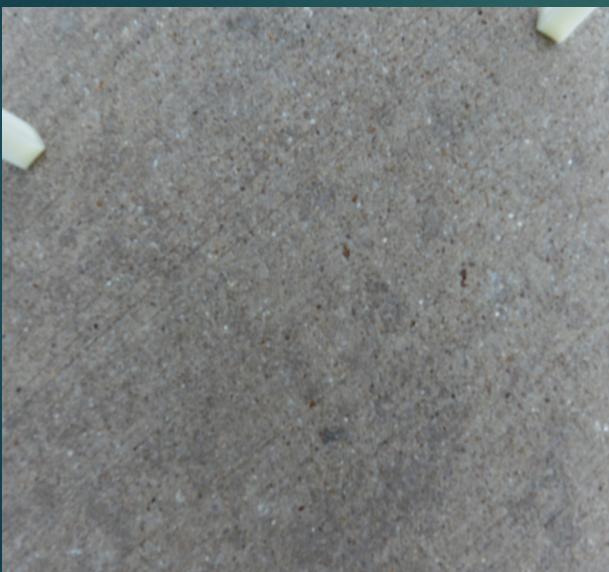
1. OpenCV (Open Source Image Processing Library)
2. Python - NumPy Library (a part of SciPy Toolkit)
3. Google Earth for input images

# Preliminary Results

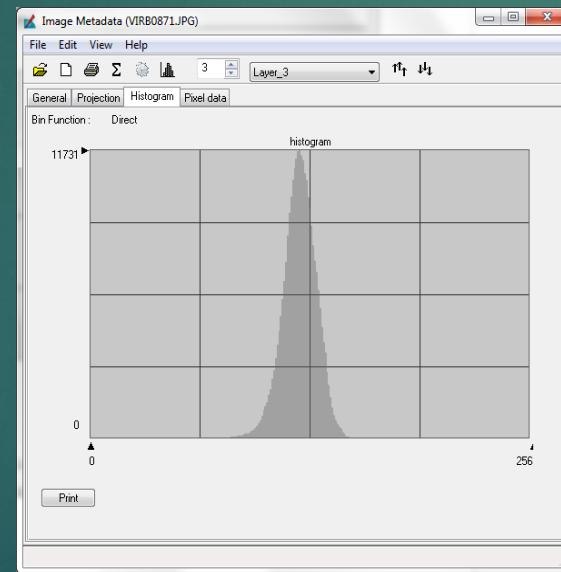


Preliminary analytical results render homogenous target features relatively indistinguishable in all three bands (RGB).

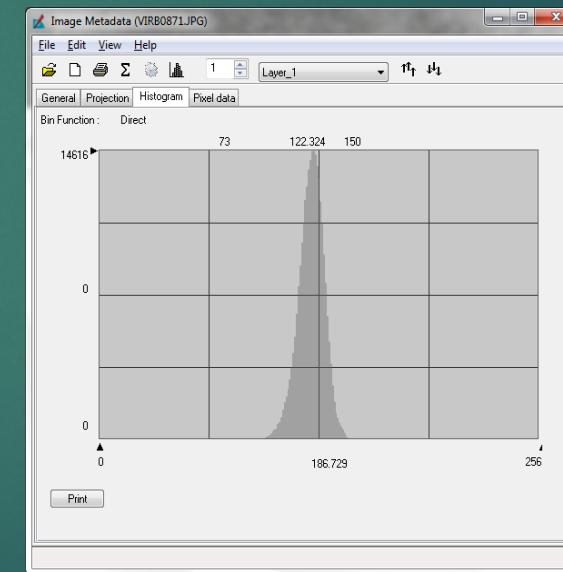
Image Data File  
VIRB0871.JPG



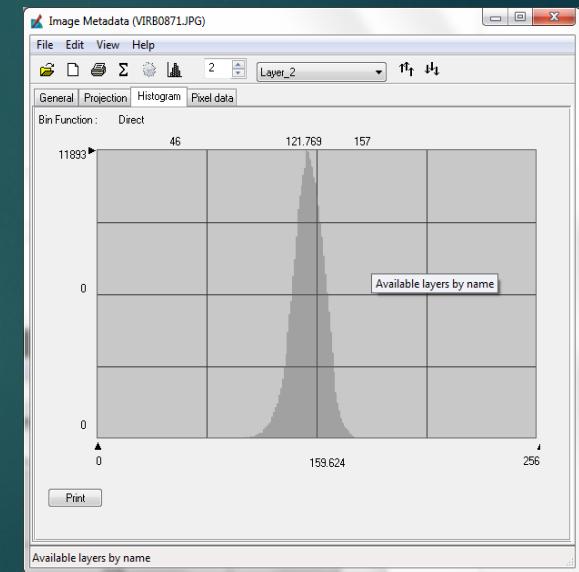
Band 1 (Blue)  
Histogram



Band 2 (Green)  
Histogram



Band 3 (Red)  
Histogram

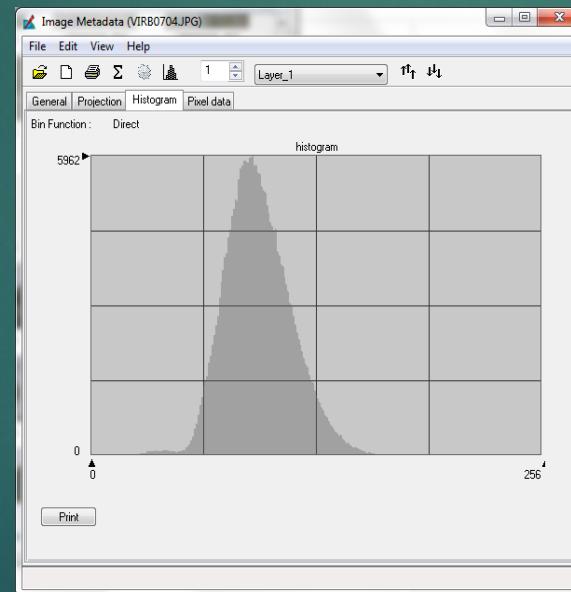


Whereas, procedures applied to contrasting pixel sets exhibit differing histogram widths according to Digital Number. Simply stated, our analytical procedure identifies target based on bands.

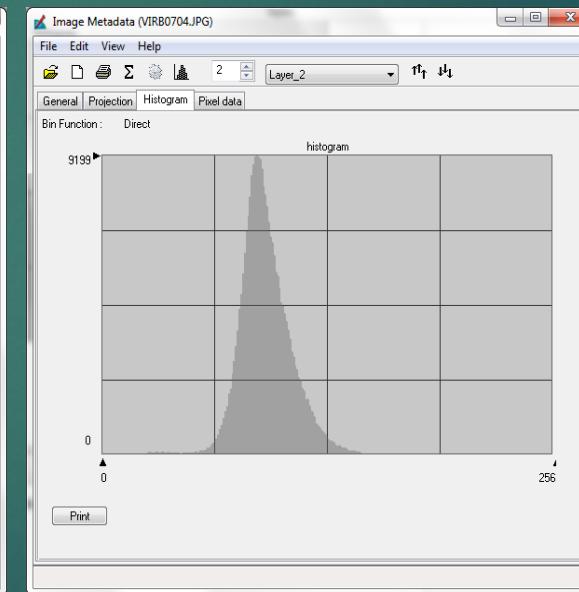
Image Data File  
VIRB0704.JPG



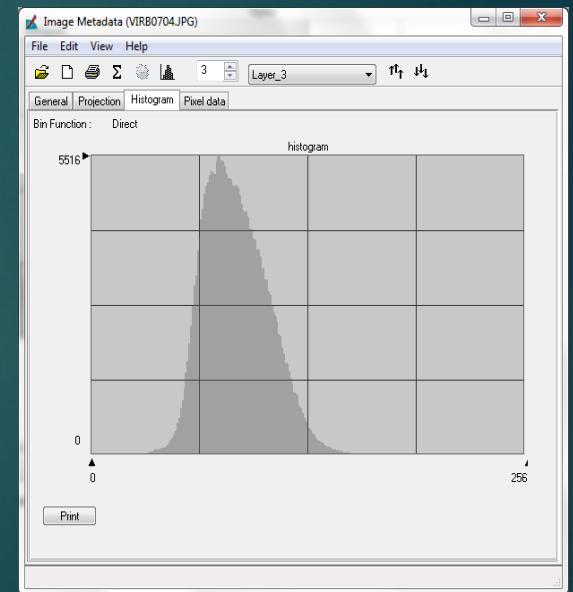
Band 1 (Blue)  
Histogram



Band 2 (Green)  
Histogram



Band 3 (Red)  
Histogram



# References

## ► References:

1. L Huidrom et. al, “*Method for automated assessment of potholes, cracks and patches from road surface video clips*”, *2<sup>nd</sup> Conference of Transportation Research Group of India*
2. OpenCV with Python for Image and Video Analysis

<https://www.youtube.com/watch?v=Z78zbnLIPUA&list=PLQVvvaaoQuDttJXILtAJxJetJcqmqlQq>

3. UAV Data provided by Dr. Anthony Cummings