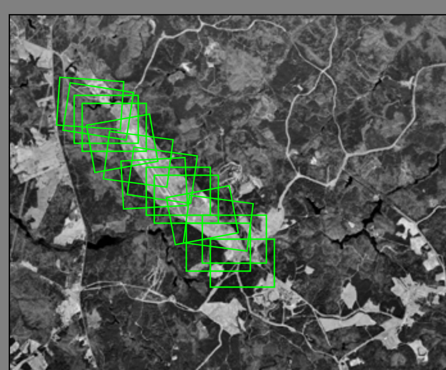
Lecture 100

Georegistration of Aerial Video Frames



Computer Vision
University of Central Florida

Image Registration

- Video frame to video frame registration
  - Direct method
    - Estimate transformation (e.g. affine, homography…)
  - Feature based method
    - Detect features
    - Find feature correspondences
    - Fit transformation
- Wide baseline matching (two views of the same scene)
    - Detect features
    - Find feature correspondences
    - Transformation (Fundamental matrix constraint)

Geo Registration

- Register a video frame (mission image) with the reference image.
- Reference image is geodetically accurate.
  - Each pixel has corresponding GPS coordinates (long, lat).
- Reference image can be 5 to 10 years older than the mission image.
- Reference image is normally orhtorectified image (e.g. Digital Ortho Quad (DOQ), Google Earth).
- Reference and mission images can be from very different viewpoints.
- Geo registration transfers geodetic accuracy of reference image to the mission image.

Applications

- Targeting
- Map generation
- Rescue and relief
- Road construction
- All applications of Google earth

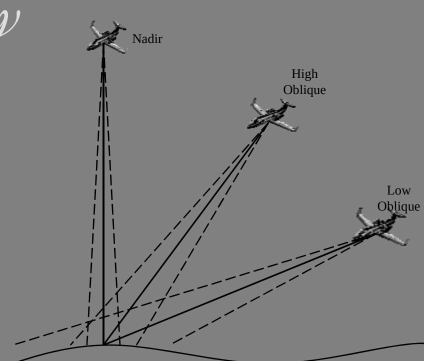
*geo*Registration

Problem Overview

Unmanned Aerial Vehicle

Sensor Setup & Imaging

Reference Imagery



Computer Vision
University of Central Florida

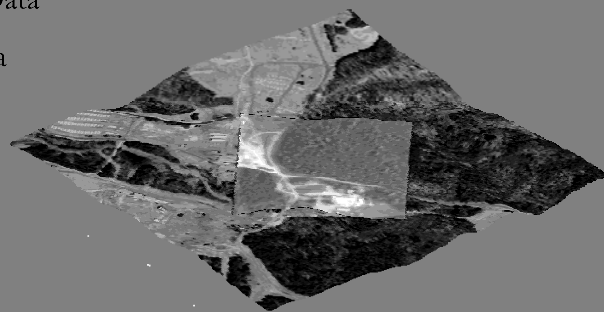
*geo*Registration

Data Overview

Aerial Video Data

Reference Data

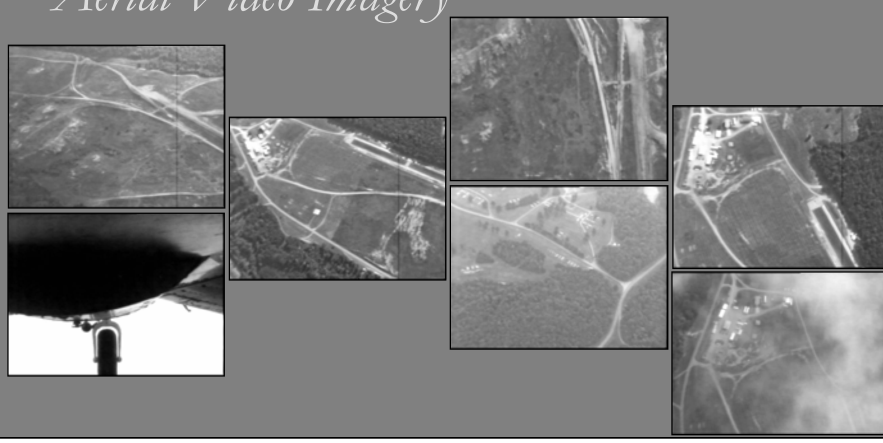
Telemetry



Computer Vision
University of Central Florida

*geo*Registration

Aerial Video Imagery



Computer Vision
University of Central Florida

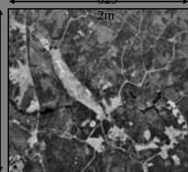
*geo*Registration

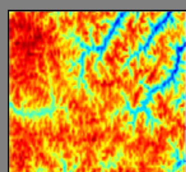
Reference Data

829
2m

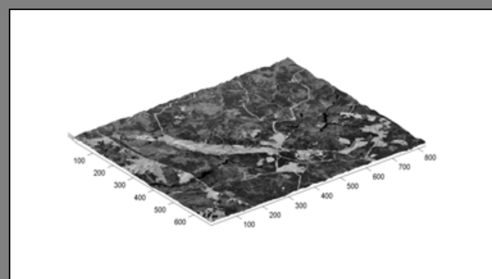
DOQ

685
6 m





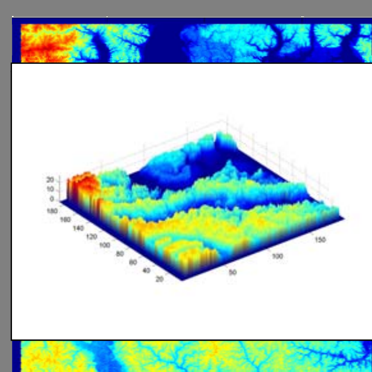
DEM



Computer Vision
University of Central Florida

Data I/O

- **Digital Elevation Map**
  - **Information**
    - Standards – DTED1, DTED2…
    - Posts
    - Accuracy



Computer Vision
University of Central Florida

Data I/O

Telemetry

Gimbal

Directions of Rotation

Converting Latitude and Longitude into distances

Computer Vision
University of Central Florida

Data I/O

- ## Reference Image
  - **Information**
    - Longitude and Latitude
    - High Resolution
    - Orthographic Image

Computer Vision
University of Central Florida

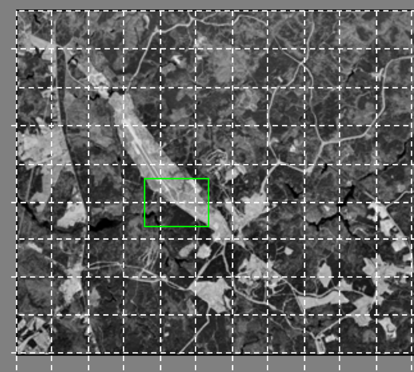
Data I/O

Reference Image

Size : 8292 x 6856 @ GSD 1

Broke it into manageable chunks

Retrieval and Pasting based on LatLong Axis



*geo*Registration

Telemetry Data

Vehicle Longitude

Vehicle Latitude

Vehicle Height

Vehicle Heading

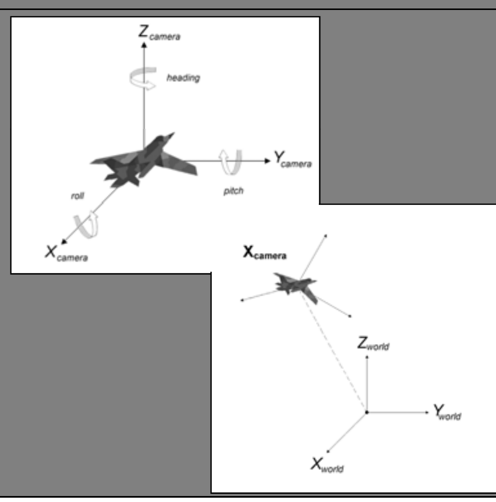
Vehicle Roll

Vehicle Pitch

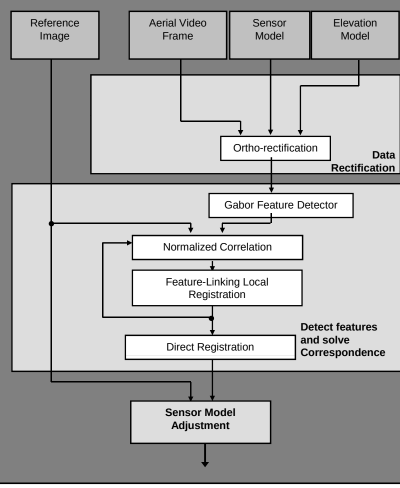
Camera Elevation Angle

Camera Scan Angle

Camera Focal Length



*geo*Registration



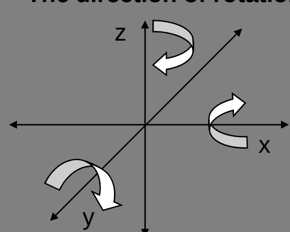
Computer Vision
University of Central Florida

*ortho*Rectification

Generating a Sensor Model

SENSORMODEL returns the 4x4 camera transformation matrix from the calibration parameters in a .geom file

The direction of rotation



Roll, pitch, heading?

Computer Vision
University of Central Florida

*ortho*Rectification

Generating a Sensor Model

$$\mathbf{X}_{\mathbf{camera}} = [X_{camera} \quad Y_{camera} \quad Z_{camera}]^{\mathrm{T}}$$

$$\mathbf{X}_{\mathbf{world}} = [X_{world} \quad Y_{world} \quad Z_{world}]^{\mathrm{T}}$$

$$\mathbf{X}_{\mathbf{camera}} = \Pi_t \mathbf{X}_{\mathbf{world}}$$

$$\Pi_t = f \text{ (camera elevation, camera scan, vehicle pitch, vehicle roll, vehicle heading, vehicle elevation)}$$

$$\Pi_t = \begin{bmatrix} \cos\omega & 0 & -\sin\omega & 0 \\ 0 & 1 & 0 & 0 \\ \sin\omega & 0 & \cos\omega & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\tau & \sin\tau & 0 & 0 \\ -\sin\tau & \cos\tau & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi & 0 & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\beta & \sin\beta & 0 \\ 0 & -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 & 0 \\ -\sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & \Delta T_x \\ 0 & 1 & 0 & \Delta T_y \\ 0 & 0 & 1 & \Delta T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Computer Vision
University of Central Florida

*ortho*Rectification

Generating a Sensor Model

$$\Pi_c = P\Pi_t$$ P is Perspective matrix

$$\Pi_c = PG_y G_z R_y R_x R_z T$$

$$\mathbf{x}_{\mathbf{ortho}} = [x_{ortho} \quad y_{ortho} \quad z_{elev} \quad 1]^{\mathrm{T}}$$

$$\mathbf{x}_{\mathbf{video}} = [x_{video} \quad y_{video} \quad 0 \quad 1]^{\mathrm{T}}$$

$$\mathbf{x}_{\mathbf{video}} = \Pi_c \mathbf{x}_{\mathbf{ortho}}$$

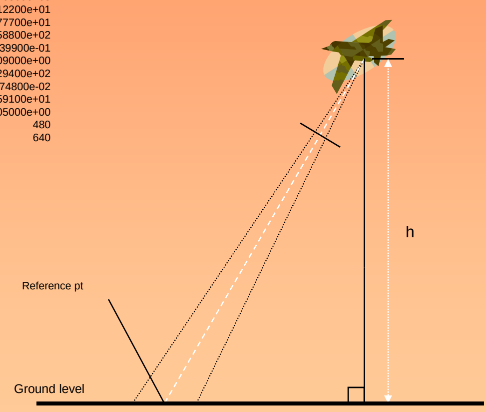
Computer Vision
University of Central Florida

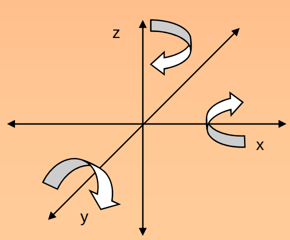
```
c(1,1) = (cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*cos(v_hdg)-sin(c_scn)*cos(v_pch)*sin(v_hdg);
c(1,2) = -(cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*sin(v_hdg)-sin(c_scn)*cos(v_pch)*cos(v_hdg);
c(1,3) = -cos(c_scn)*sin(v_rll)-sin(c_scn)*sin(v_pch)*cos(v_rll);
c(1,4) = -((cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*cos(v_hdg)-sin(c_scn)*cos(v_pch)*sin(v_hdg))*vx-(-(cos(c_scn)*cos(v_rll)-
         sin(c_scn)*sin(v_pch)*sin(v_rll))*sin(v_hdg)-sin(c_scn)*cos(v_pch)*cos(v_hdg))*vy-(-cos(c_scn)*sin(v_rll)-sin(c_scn)*sin(v_pch)*cos(v_rll))*vz;

c(2,1) = (-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(-
         sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*sin(v_hdg);
c(2,2) = -(-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(-
         sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*cos(v_hdg);
c(2,3) = sin(c_elv)*sin(c_scn)*sin(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*cos(v_rll);
c(2,4) = -((-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(-
         sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-(-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-
         sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(-sin(c_elv)*cos(c_scn)*cos(v_pch)-
         cos(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(sin(c_elv)*sin(c_scn)*sin(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*cos(v_rll))*vz;

c(3,1) = (cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-
         sin(c_elv)*sin(v_pch))*sin(v_hdg);
c(3,2) = -(cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-
         sin(c_elv)*sin(v_pch))*cos(v_hdg);
c(3,3) = -cos(c_elv)*sin(c_scn)*sin(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*cos(v_rll);
c(3,4) = -
         ((cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-
         sin(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-
         (cos(c_elv)*sin(c_scn)*cos(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(cos(c_elv)*cos(c_scn)*cos(v_pch)-
         sin(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(-cos(c_elv)*sin(c_scn)*sin(v_rll)+(cos(c_elv)*cos(c_scn)*sin(v_pch)+sin(c_elv)*cos(v_pch))*cos(v_rll))*vz;

c(4,1) =
         (1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(1/fl*cos(c_elv)*cos(c_s
         cn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg);
c(4,2) = -
         (1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(1/fl*cos(c_elv)*cos(c_sc
         n)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg);
c(4,3) = -1/fl*cos(c_elv)*sin(c_scn)*sin(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll);
c(4,4) = -
         ((1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(1/fl*cos(c_elv)*cos(c_s
         cn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-
         (1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(1/fl*cos(c_elv)*cos(c_sc
         n)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(-
         1/fl*cos(c_elv)*sin(c_scn)*sin(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll))*vz+1;
```

ORTHORECTIFICATION

Camera Information

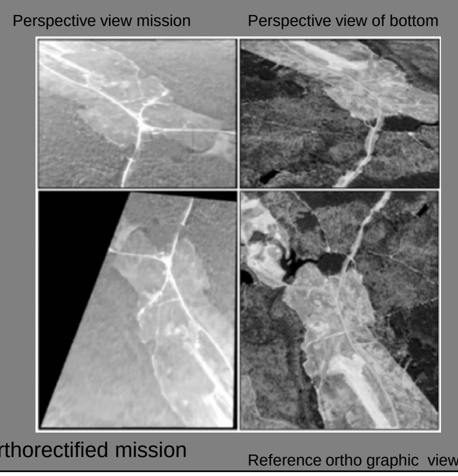
image_time	9.400008152666640300e+08
vehicle_latitude	3.813193746469612200e+01
vehicle_longitude	-7.734523185193877700e+01
vehicle_height	9.949658409987658800e+02
vehicle_pitch	9.995171174441039900e-01
vehicle_roll	1.701626418113209000e+00
vehicle_heading	1.207010551753029400e+02
camera_focal_length	1.658968732990974800e-02
camera_elevation	-5.361314389557259100e+01
camera_scan_angle	-7.232969433546705000e+00
number_image_lines	480
number_image_samples	640





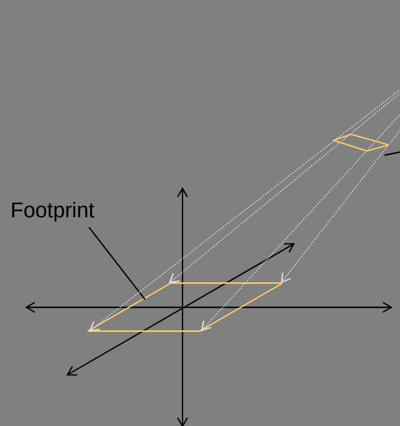
*ortho*Rectification

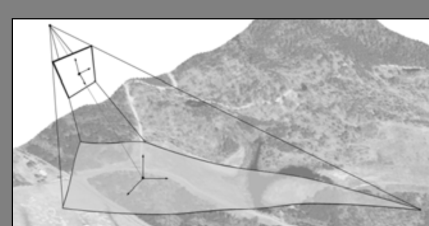
- Bringing both imageries onto a common view projection
- Not accurate due to telemetry noise
- Perspective Projection vs. Orthographic Projection



Computer Vision
University of Central Florida

*ortho*Rectification



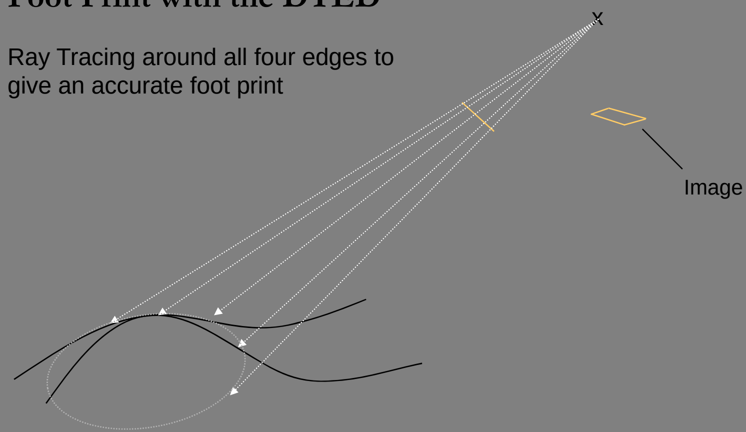


Computer Vision
University of Central Florida

*ortho*Rectification

Foot Print with the DTED

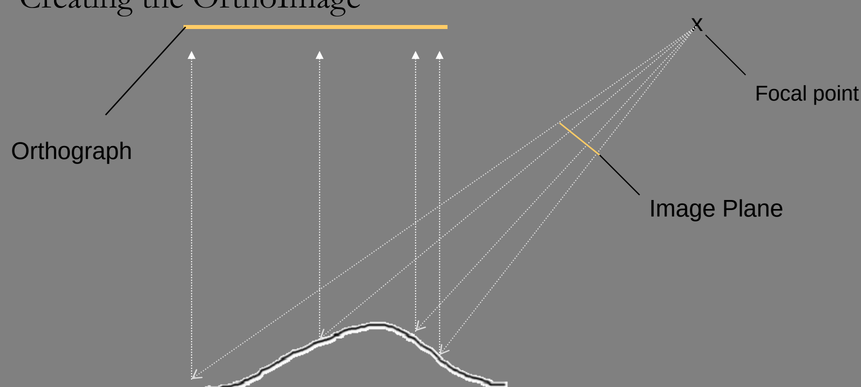
Ray Tracing around all four edges to give an accurate foot print



Computer Vision
University of Central Florida

*ortho*Rectification

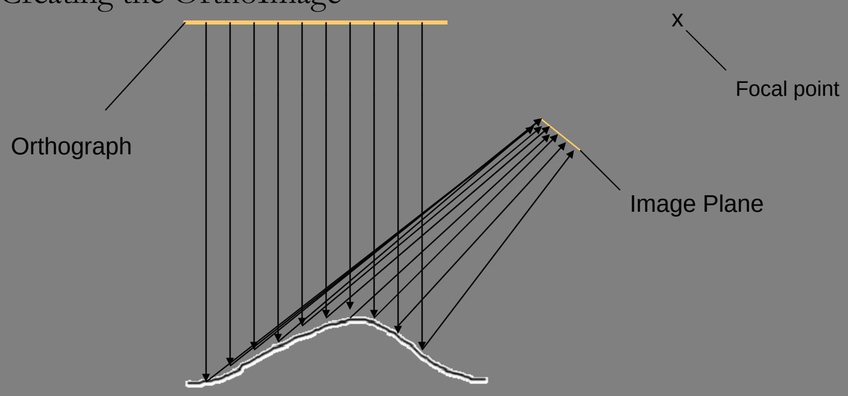
Creating the OrthoImage



Computer Vision
University of Central Florida

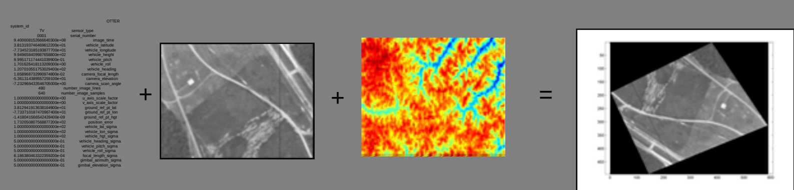
*ortho*Rectification

Creating the OrthoImage

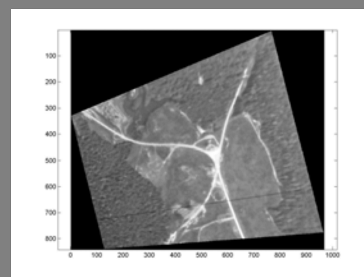


*ortho*Rectification

Sensor File + Mission Images + DEM. = Orthorectification

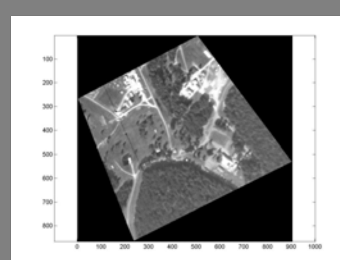


*ortho*Rectification



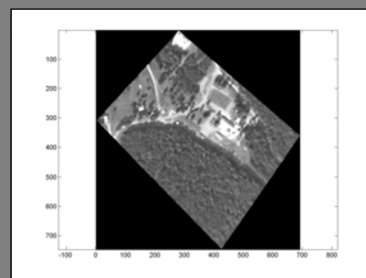
Computer Vision
University of Central Florida

*ortho*Rectification



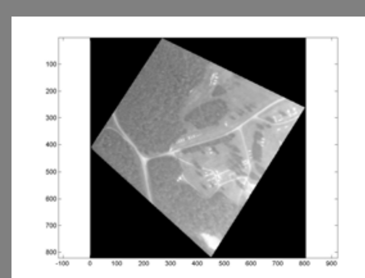
Computer Vision
University of Central Florida

*ortho*Rectification



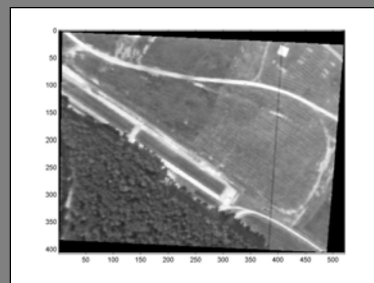
Computer Vision
University of Central Florida

*ortho*Rectification



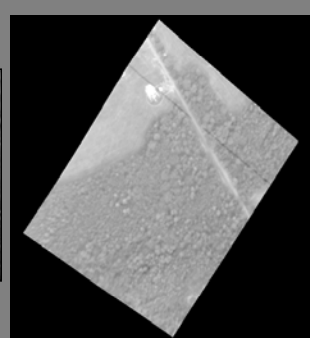
Computer Vision
University of Central Florida

*ortho*Rectification



Computer Vision
University of Central Florida

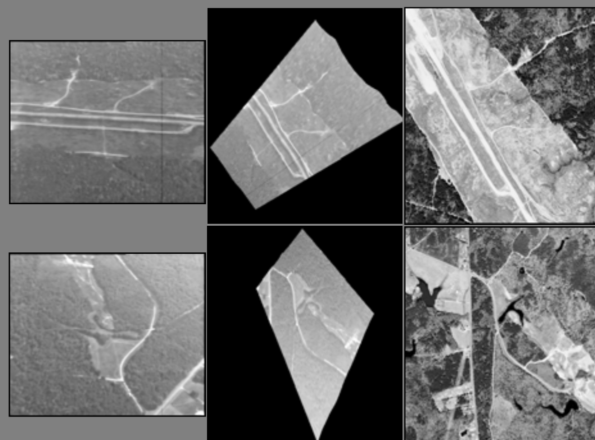
*ortho*Rectification



Reference image

Computer Vision
University of Central Florida

*ortho*Rectification

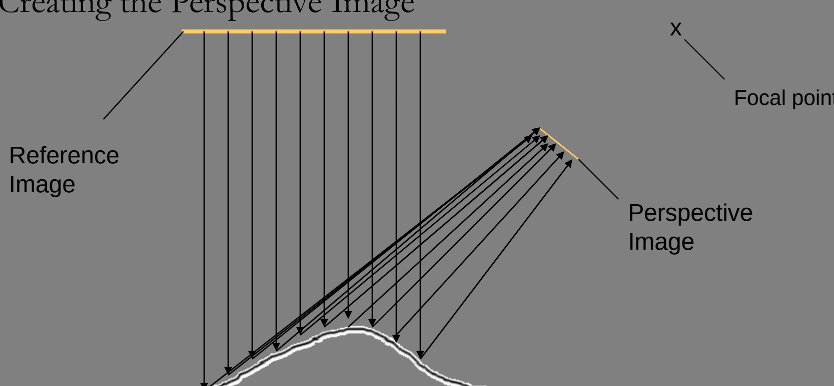


Reference image

Reference image

*perspective*Projection

Creating the Perspective Image



Computer Vision
University of Central Florida

*perspective*Projection

Computer Vision
University of Central Florida

*perspective*Projection

Computer Vision
University of Central Florida

*feature*Extraction

Selecting Features

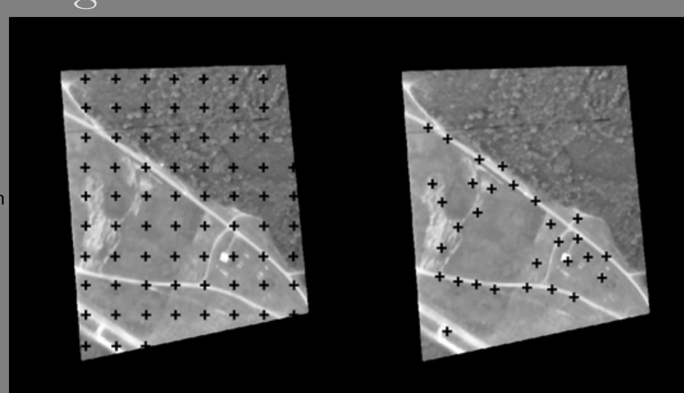
- Uniform selection of salient patches is undesirable
- Ideally, window around match point should have high energy features
- High 'self-correlation' in terrain imageries (buildings, roads, rivers, forests, plains etc)

Computer Vision
University of Central Florida

*feature*Extraction

Selecting Features

Uniform selection



Computer Vision
University of Central Florida

*feature*Extraction

Gabor Features

$$G(x,y,\theta,f) = e^{(i(f_x x+f_y y))} \cdot e^{(-(f_x^2+f_y^2)(x^2+y^2)/2\sigma^2)} \qquad f_x = f\cos\theta, \quad f_y = f\sin\theta$$

Generate Gabor feature image

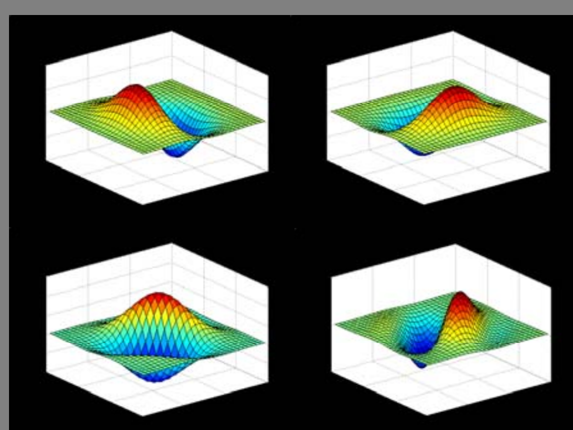
- Convolve with 4 Gabor masks
- Take absolute value
- Multiply 4 Gabor outputs together

Pick the highest peak as one feature point

Discard a window around the selected peak and pick the next highest peak

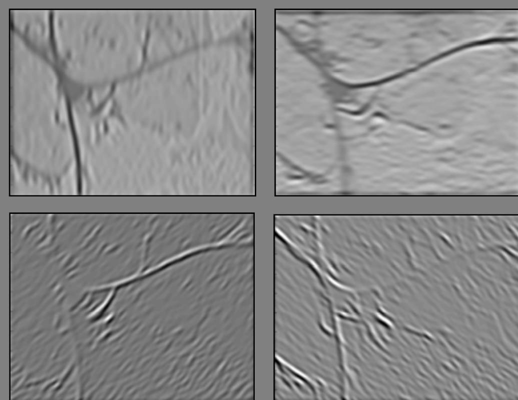
Computer Vision
University of Central Florida

*feature*Extraction

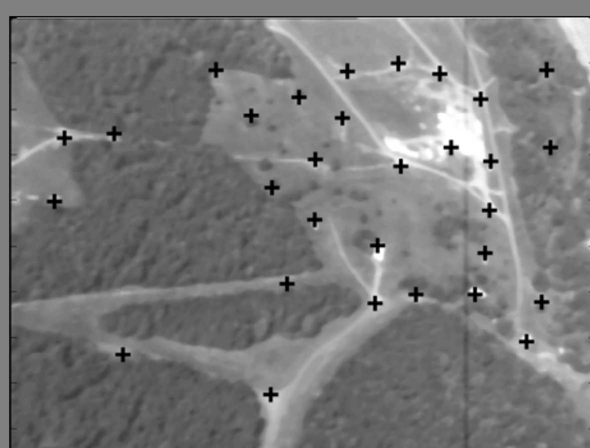


Computer Vision
University of Central Florida

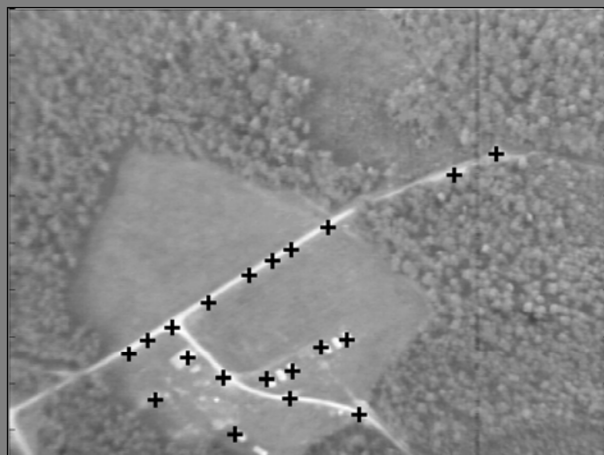
*feature*Extraction



*feature*Extraction

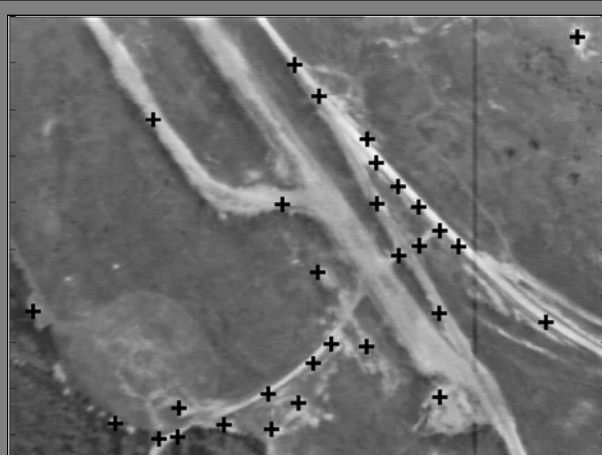


*feature*Extraction



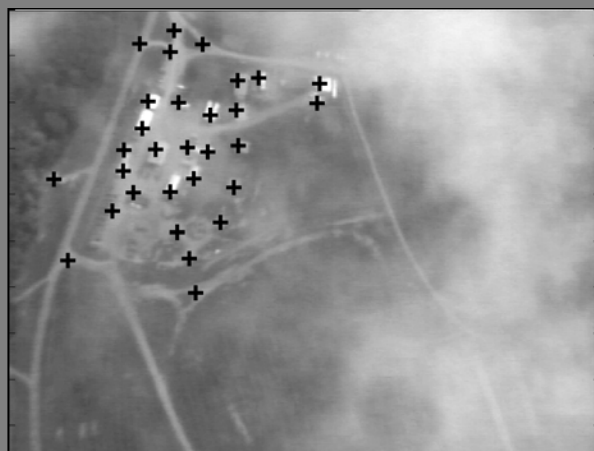
Computer Vision
University of Central Florida

*feature*Extraction



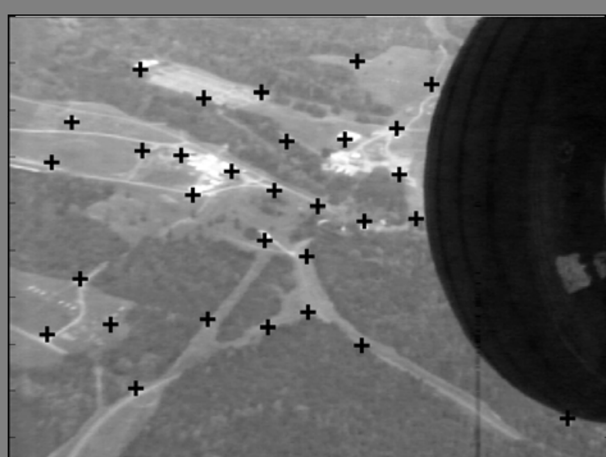
Computer Vision
University of Central Florida

*feature*Extraction



Computer Vision
University of Central Florida

*feature*Extraction

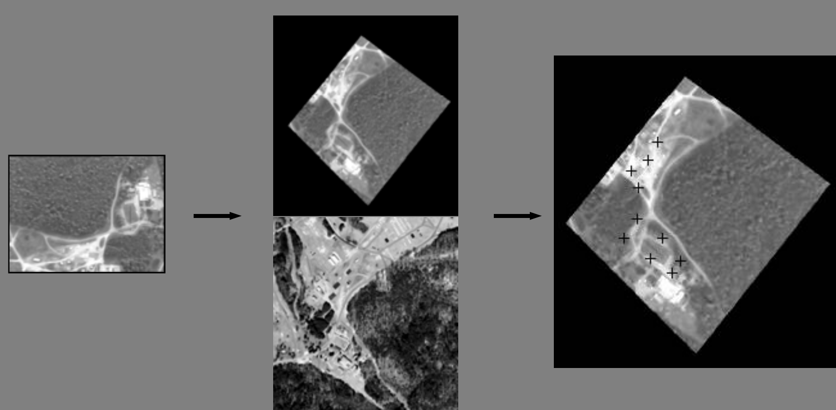


Computer Vision
University of Central Florida

*feature*Extraction

Computer Vision
University of Central Florida

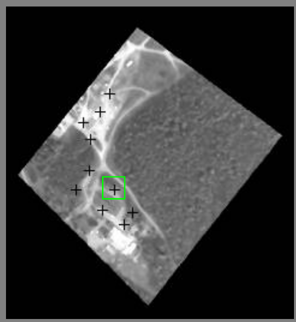
*quick*Review



Computer Vision
University of Central Florida

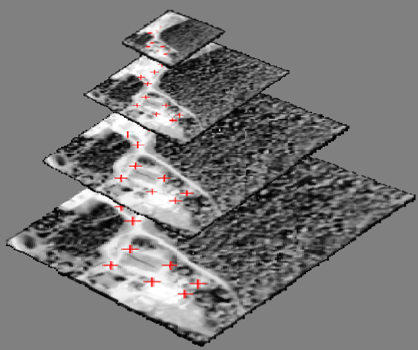
Correlation

A Patch is taken around each of the *n* feature point and correlated with a window of the reference image



Correlation

Hierarchical Implementation



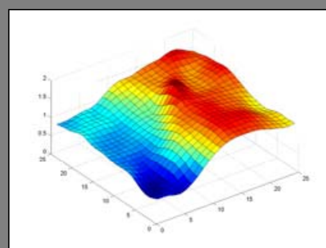
Correlation is performed in a hierarchical fashion to efficiently account for large displacements.

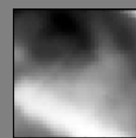
A 4-level Gaussian Pyramid System was implemented

*surface*Binding

Normalized Cross Correlation is performed between both windows

$$\eta_i(u,v) = \frac{\sum_x \sum_y T_i(x,y) I(x-u, y-v)}{[\sum_x \sum_y I^2(x-u, y-v)]^{1/2}}$$

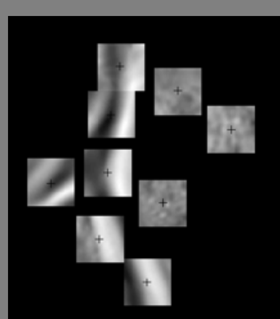




Computer Vision
University of Central Florida

*surface*Binding

Correlation Surfaces



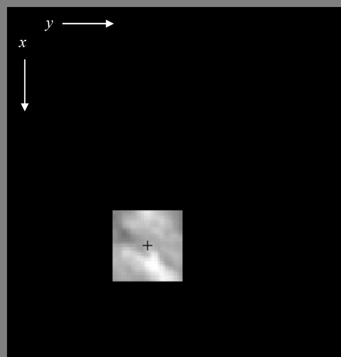
A patch around each of the *n* feature point is correlated to yield *n* feature surfaces.

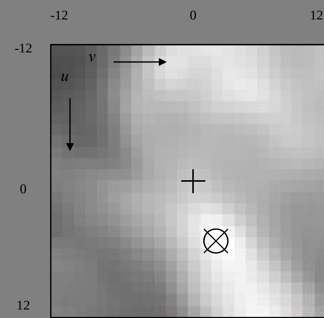
Due to the challenging nature of the images, a distinct consensus among the correlation surfaces is difficult to find.

Computer Vision
University of Central Florida

*surface*Binding

Relative & Absolute Coordinates

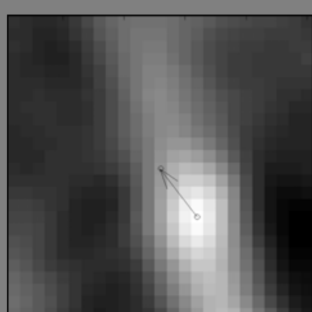




Computer Vision
University of Central Florida

*surface*Binding

Translational Model



The 'sum-surface' is defined as

$$\eta(u,v) = \sum_{i=1}^{n} \eta_i(u,v)$$

If the vector (u_{peak}, v_{peak}) is applied to the correlation surface, $\eta(0,0)$ would be maximized for

$$\eta(u,v) = \sum_{i=1}^{n} \eta_i(u',v') \quad \text{where} \quad \begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} u_{peak} \\ v_{peak} \end{bmatrix}$$

Computer Vision
University of Central Florida

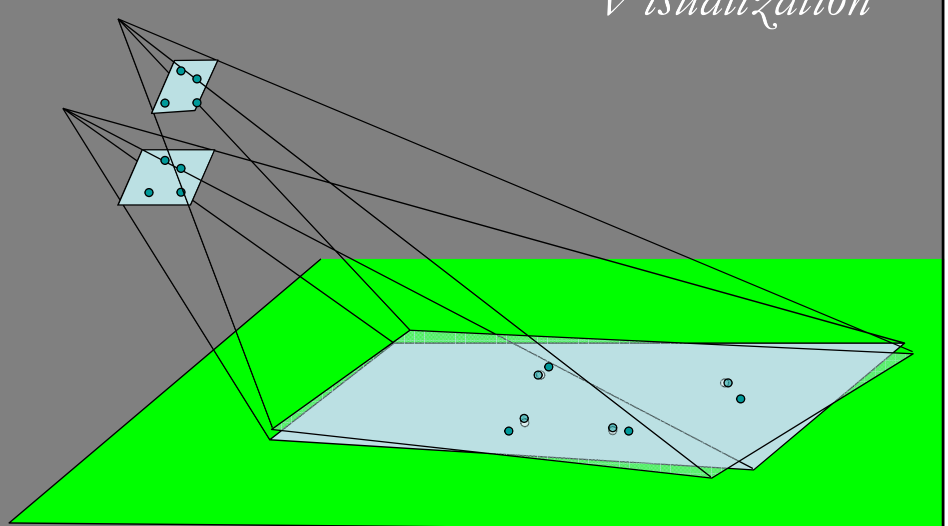
*sensor*Adjustment

Minimization

- Inputs:
  - Sensor Geometry File
  - Points on Orthorectified mission image and corresponding points on reference image
  - DEM or Equation of Plane
- Output
  - Adjusted Sensor Geometry

Computer Vision
University of Central Florida

Visualization



*sensor***Adjustment**

Steps in Minimization

•INITIALIZATION

–Conversion from image coordinates to world coordinates

Applied to Ortho-Mission points and Reference points

$\mathbf{X}_r = (X_R, Y_R, Z_R)^T \qquad \mathbf{X}_m = (X_M, Y_M, Z_M)^T$

X and Y coordinates are simply scaled

Z coordinate is read from the DEM or from the Plane Equation

–Project Ortho-Mission Points to Image Plane

$$\Pi_c = PG_yG_zR_yR_xR_zT$$

Read Sensor Model, S_0

–Specify ground plane or DEM, D

*sensor***Adjustment**

Steps in Minimization

• ITERATIONS

–Compute projection of mission points (x_m, y_m) to world (using DEM or Ground Plane) - (X_M, Y_M, Z_M)

– Error Term:

$$E = \mathbf{\Sigma} \mid\mid [X_R, Y_R, Z_R]^T - [X_M, Y_M, Z_M]^T \mid\mid$$

–Adjust Sensor Model to minimize this error

*sensor*Adjustment

Methods Used

–BFGS-Quasi Newton with Finite Difference Derivates

–Quasi Newton with Analytical Derivatives

–Levenberg Marquardt with Finite Difference Derivatives

*sensor*Adjustment

Analytical Error Term vs Computed Error

Analytical Error term directly relates the 9 sensor parameters to error

Allows computation of exact analytical derivatives

However, it is long and cumbersome

Takes longer to compute on a MATLAB implementation

```
curErr = (xm-(((cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*cos(v_hdg)-sin(c_scn)*cos(v_pch)*...
    sin(v_hdg))*conj(X)+(-(cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*sin(v_hdg)-sin(c_scn)*...
    cos(v_pch)*cos(v_hdg))*conj(Y)+(-cos(c_scn)*sin(v_rll)-sin(c_scn)*sin(v_pch)*cos(v_rll))*conj(Z)-...
    ((cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*cos(v_hdg)-sin(c_scn)*cos(v_pch)*sin(v_hdg))*...
    vx-(-(cos(c_scn)*cos(v_rll)-sin(c_scn)*sin(v_pch)*sin(v_rll))*sin(v_hdg)-sin(c_scn)*cos(v_pch)*cos(v_hdg))*...
    vy-(-cos(c_scn)*sin(v_rll)-sin(c_scn)*sin(v_pch)*cos(v_rll))*vz)/(((1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+...
    (1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(1/fl*cos(c_elv)*...
    cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg))*conj(X)+(-(1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)...
    +(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(1/fl*cos(c_elv)*...
    cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg))*conj(Y)+(-1/fl*cos(c_elv)*sin(c_scn)*sin(v_rll)...
    +(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll))*conj(Z)-((1/fl*cos(c_elv)*...
    sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)...
    +(1/fl*cos(c_elv)*cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-(1/fl*cos(c_elv)*sin(c_scn)...
    *cos(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+...
    (1/fl*cos(c_elv)*cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(-1/fl*cos(c_elv)*sin(c_scn)*...
    sin(v_rll)+(1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll))*vz+1))^2+(ym-...
    (((-sin(c_elv)*sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*...
    cos(v_hdg)+(-sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*sin(v_hdg))*conj(X)+(-(-sin(c_elv)*...
    sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+...
    (-sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*cos(v_hdg))*conj(Y)+(sin(c_elv)*sin(c_scn)*...
    sin(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*cos(v_rll))*conj(Z)-((-sin(c_elv)*...
    sin(c_scn)*cos(v_rll)+(-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+...
    (-sin(c_elv)*cos(c_scn)*cos(v_pch)-cos(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-(-sin(c_elv)*sin(c_scn)*cos(v_rll)+...
    (-sin(c_elv)*cos(c_scn)*sin(v_pch)+cos(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(-sin(c_elv)*cos(c_scn)*...
    cos(v_pch)-cos(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(sin(c_elv)*sin(c_scn)*sin(v_rll)+(-sin(c_elv)*cos(c_scn)*...
    sin(v_pch)+cos(c_elv)*cos(v_pch))*cos(v_rll))*vz)/(((1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*...
    cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(1/fl*cos(c_elv)*cos(c_scn)*cos(v_pch)-...
    1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg))*conj(X)+(-(1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+(1/fl*cos(c_elv)*...
    cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(1/fl*cos(c_elv)*cos(c_scn)*cos(v_pch)-...
    1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg))*conj(Y)+(-1/fl*cos(c_elv)*sin(c_scn)*sin(v_rll)+(1/fl*cos(c_elv)*...
    cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll))*conj(Z)-((1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+...
    (1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*cos(v_hdg)+(1/fl*cos(c_elv)*...
    cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*sin(v_hdg))*vx-(-(1/fl*cos(c_elv)*sin(c_scn)*cos(v_rll)+...
    (1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*sin(v_rll))*sin(v_hdg)+(1/fl*cos(c_elv)*...
    cos(c_scn)*cos(v_pch)-1/fl*sin(c_elv)*sin(v_pch))*cos(v_hdg))*vy-(-1/fl*cos(c_elv)*sin(c_scn)*sin(v_rll)+...
    (1/fl*cos(c_elv)*cos(c_scn)*sin(v_pch)+1/fl*sin(c_elv)*cos(v_pch))*cos(v_rll))*vz+1))^2;
```

*sensor***Adjustment**

Finite Difference vs. Analytical Derivatives

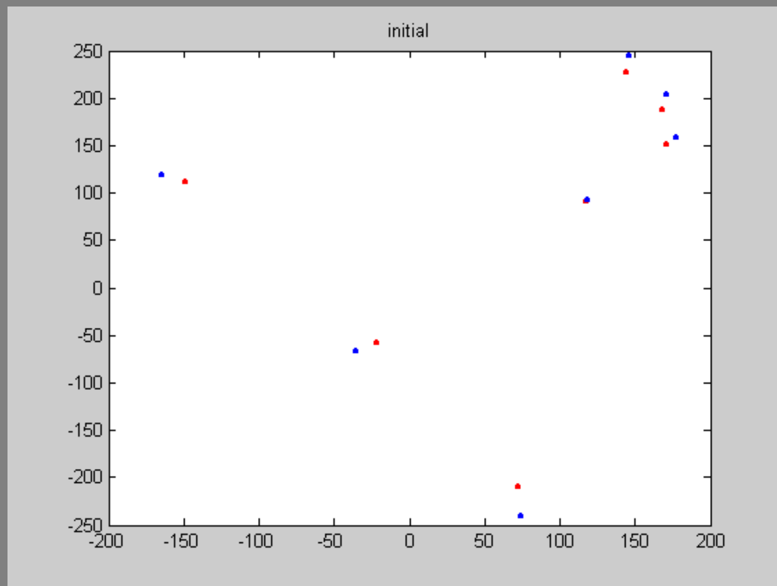
- Derivatives of the surface guide the search
- Finite Difference Derivatives are fast in MATLAB, because analytical terms are too long
- However, computing finite derivatives increases the number of function evaluations

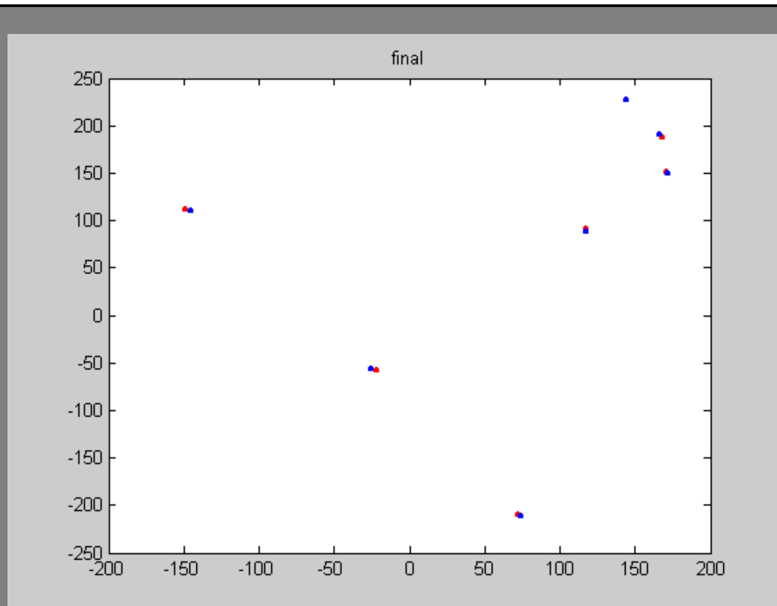
*sensor*Adjustment

Exit Strategy

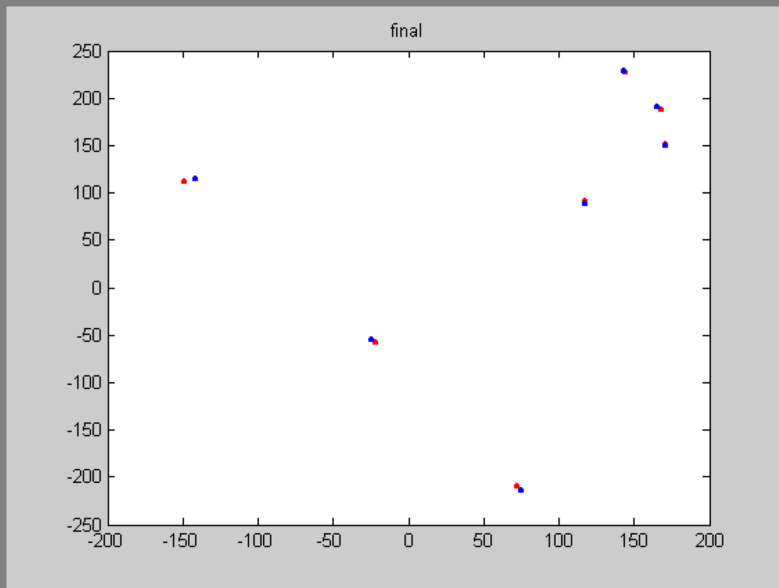
- Multiple stopping criteria:
  - Tolerance of change in sensor model
  - Maximum Iterations, Maximum Function Evaluations

Example





2000 function evaluations



200 function evaluations

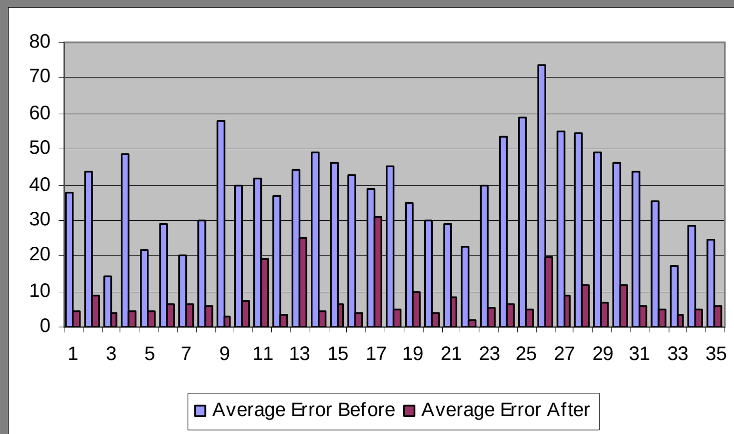
*sensor*Adjustment

Multiplicity of Solutions

- Solution is not unique
  - –Several different sensor models can yield the same error
- Solution found depends on the initial sensor model and the decent strategy
- Essentially we have a 9-D surface with multiple minimum points
- E.g. Focal-length / Aircraft Height are complementary pairs

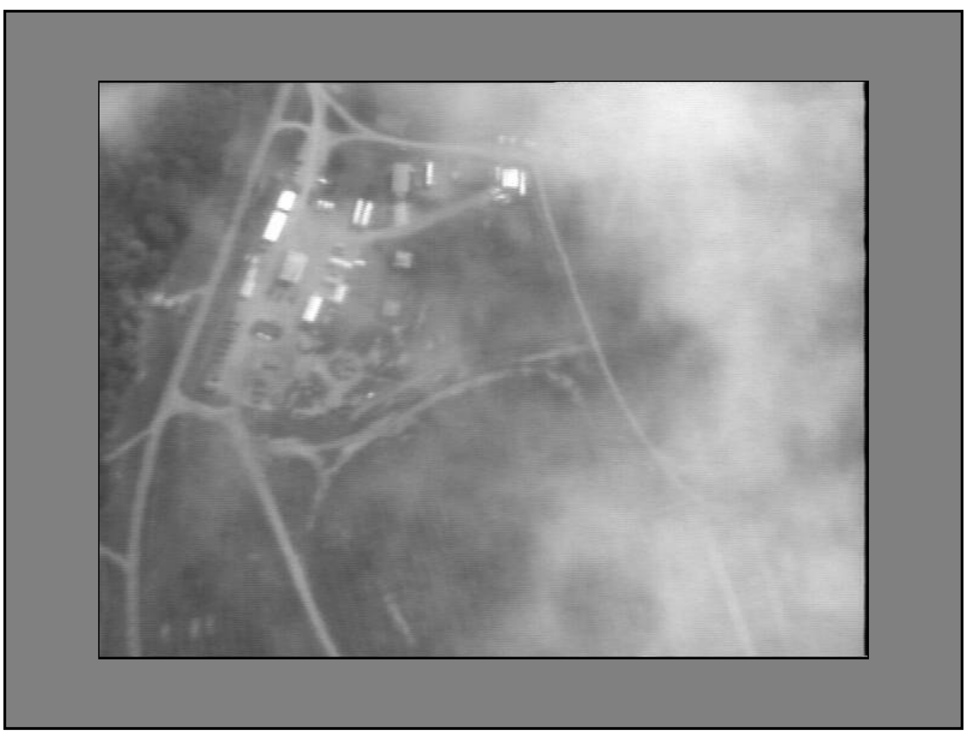
Computer Vision
University of Central Florida

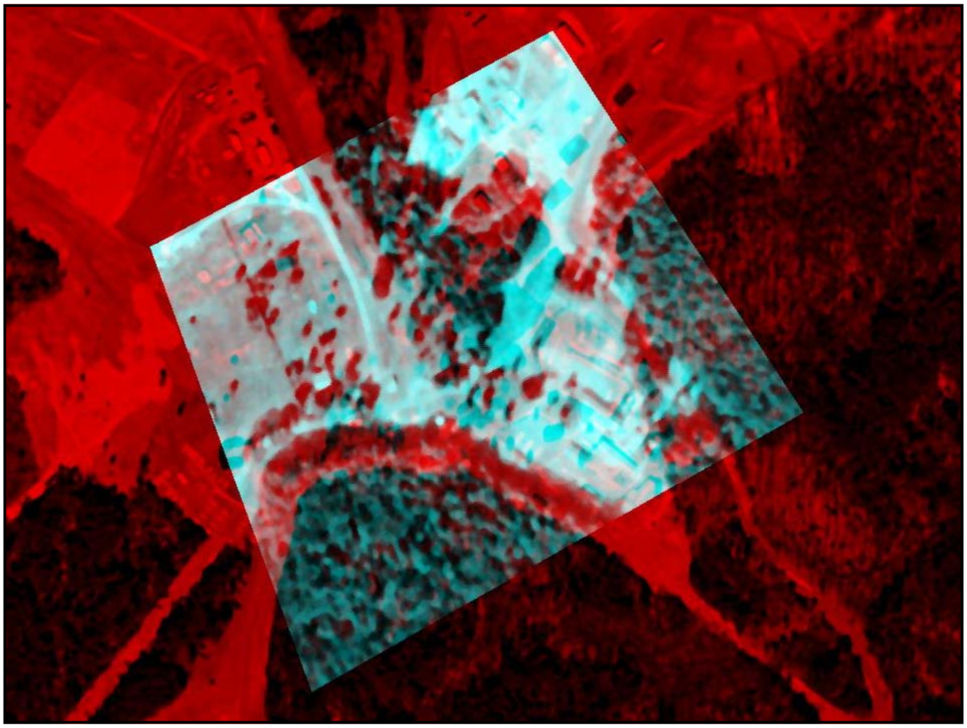
*registration*Results

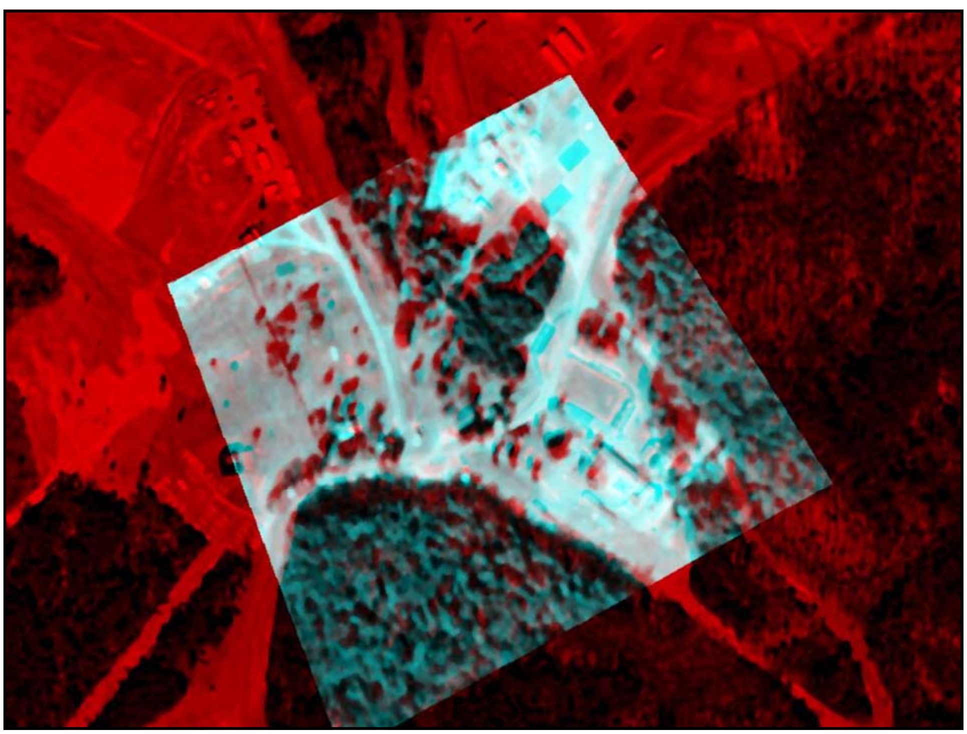


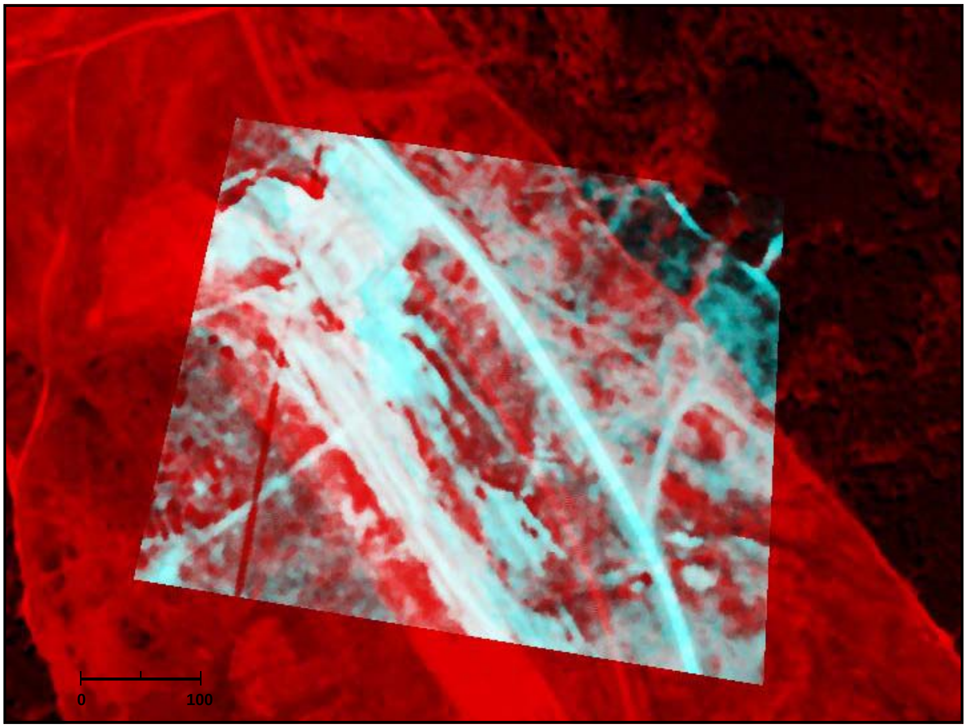
Average Original Error: 32 meters Average Residual Error: 8 meters

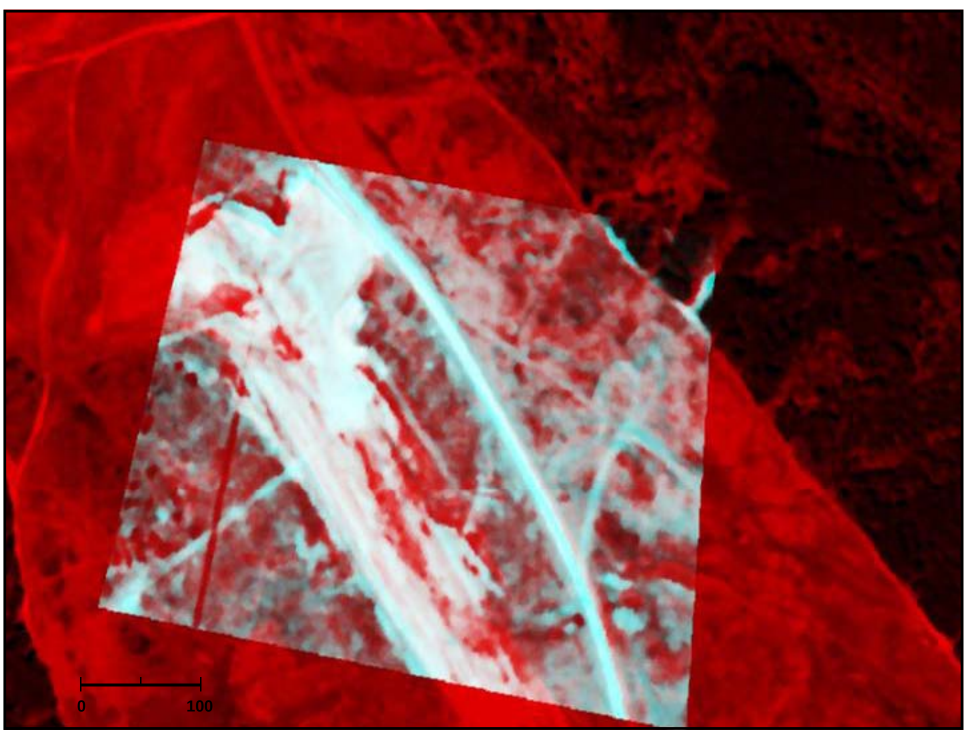
Computer Vision
University of Central Florida

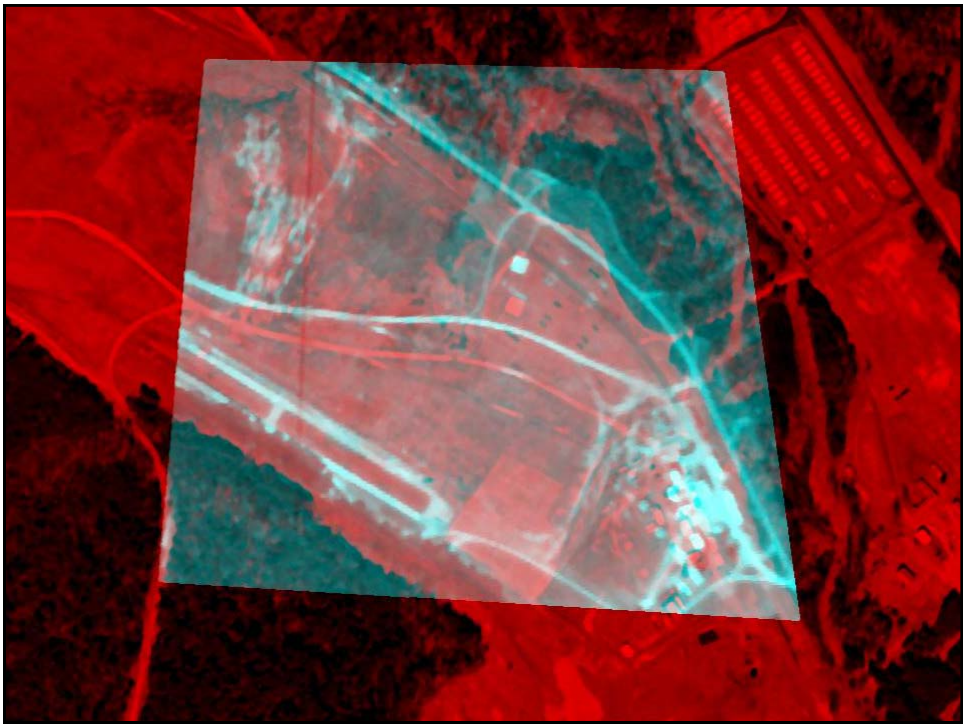


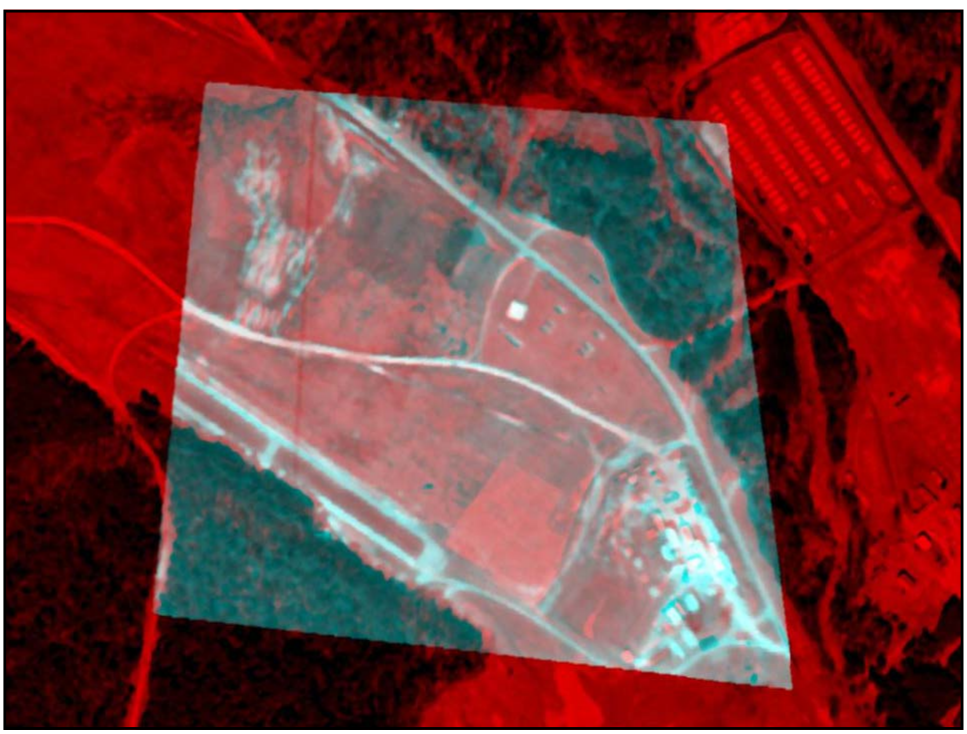


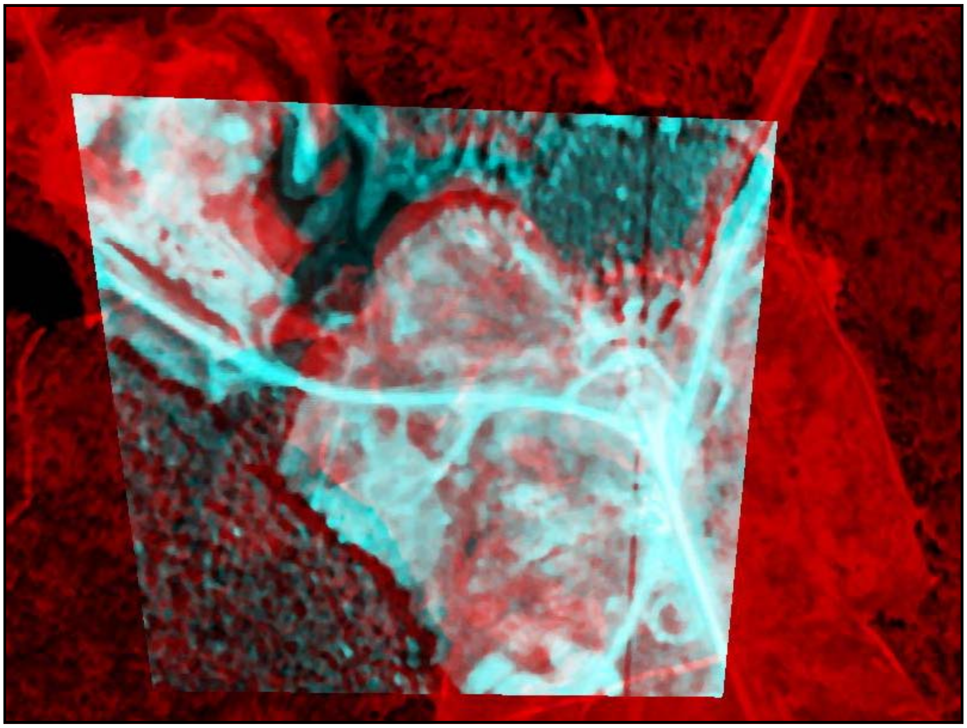


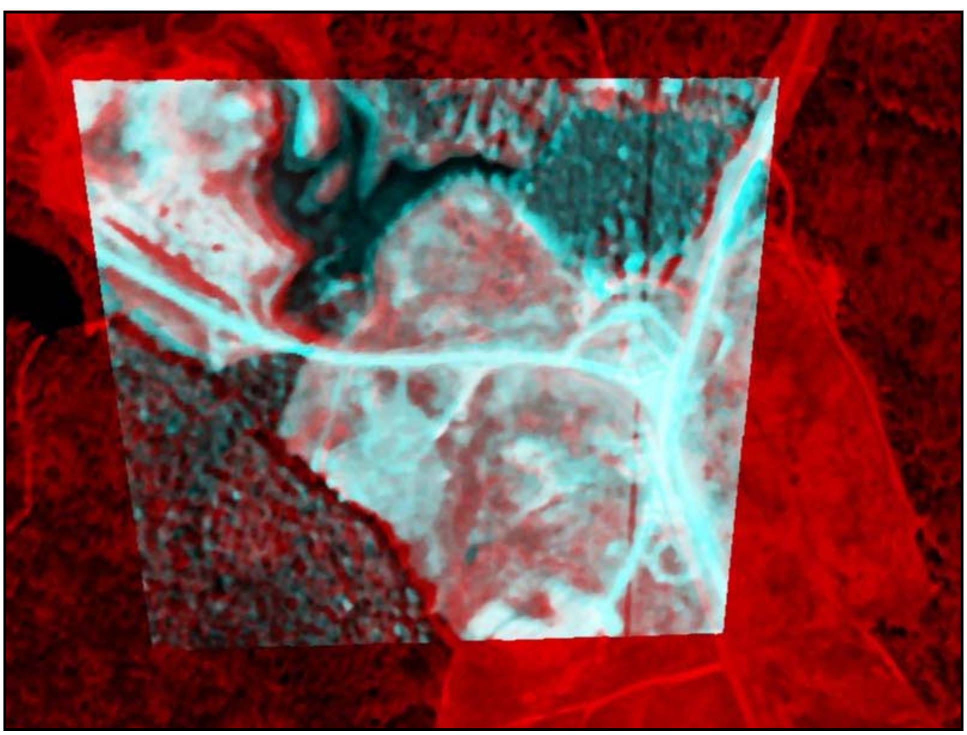

0
100

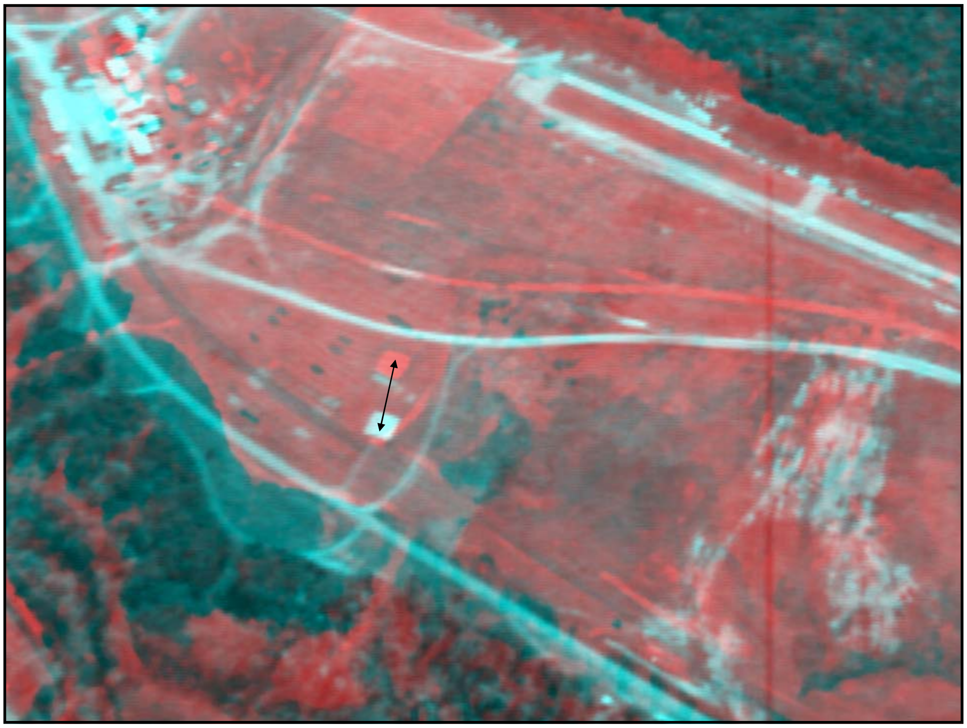

0
100

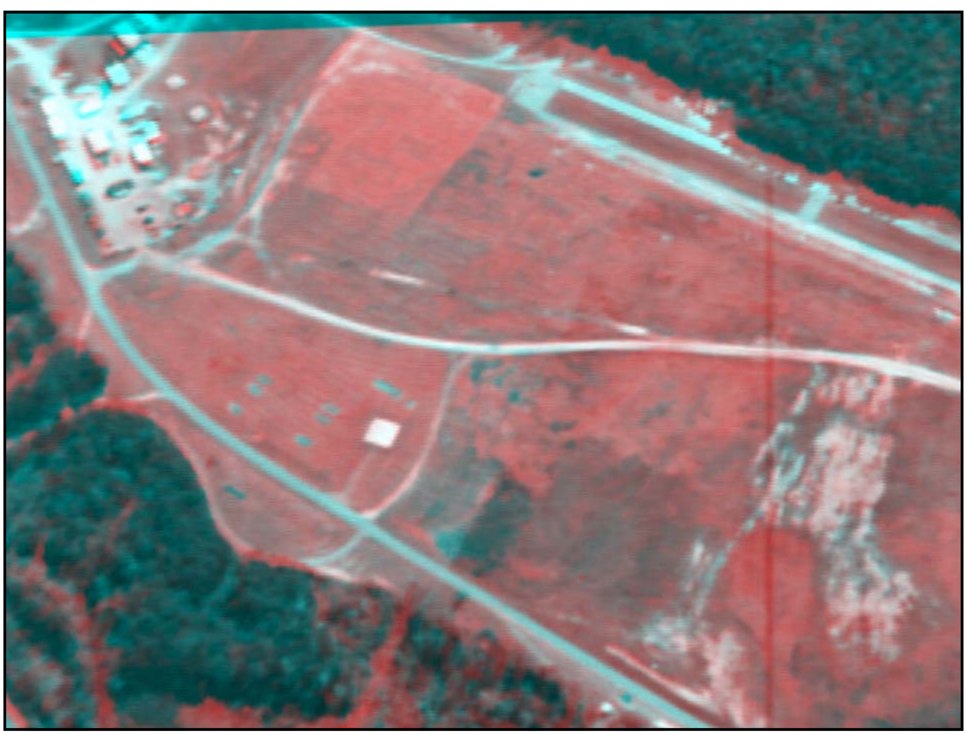


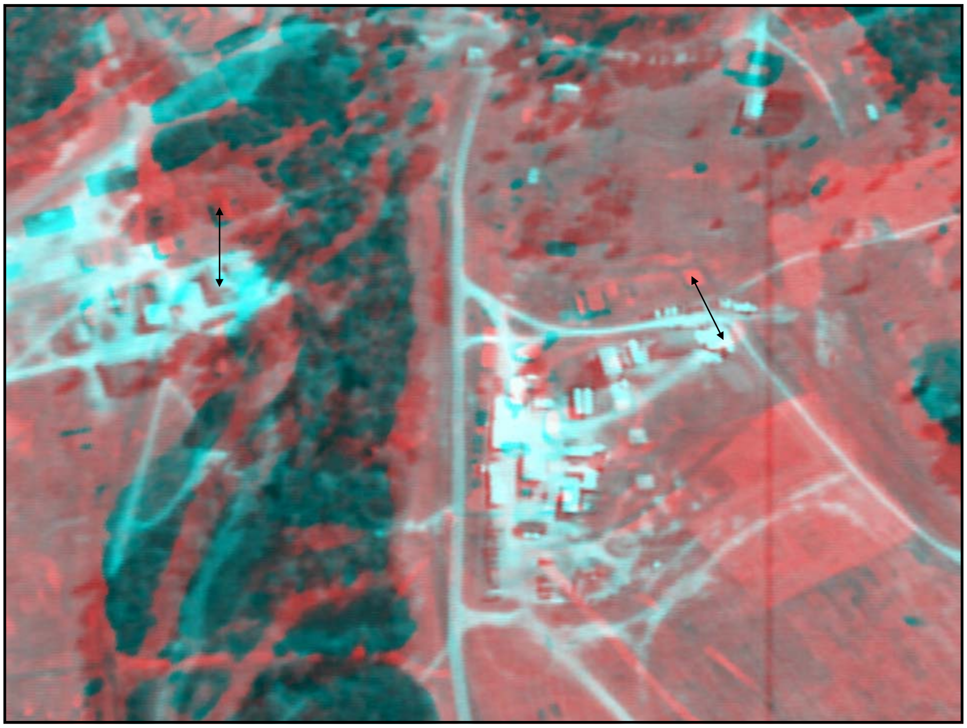


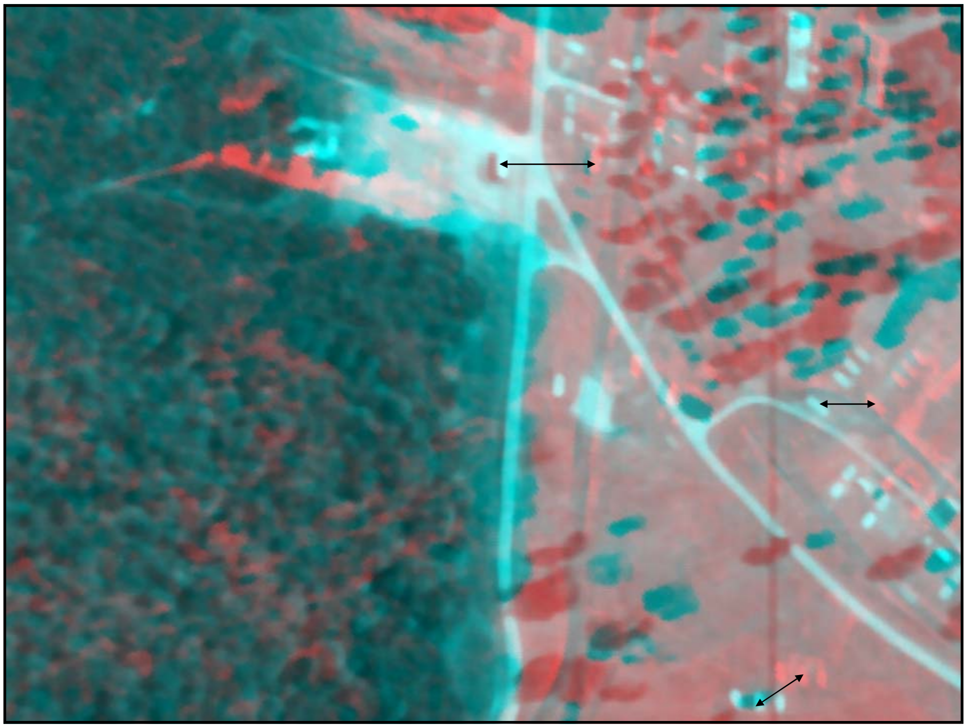


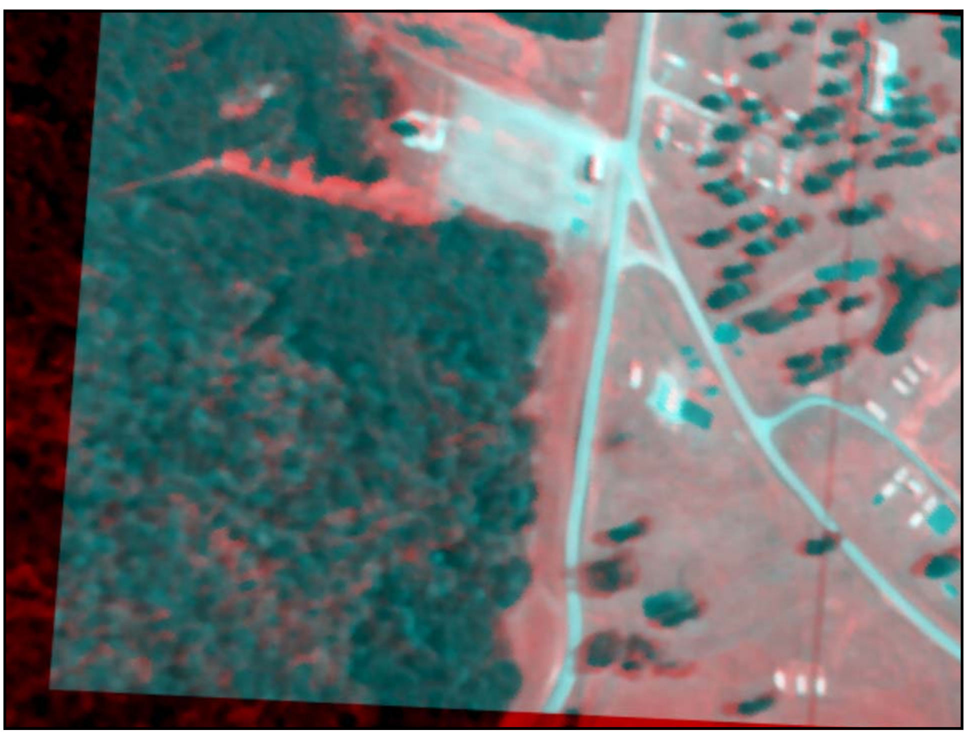


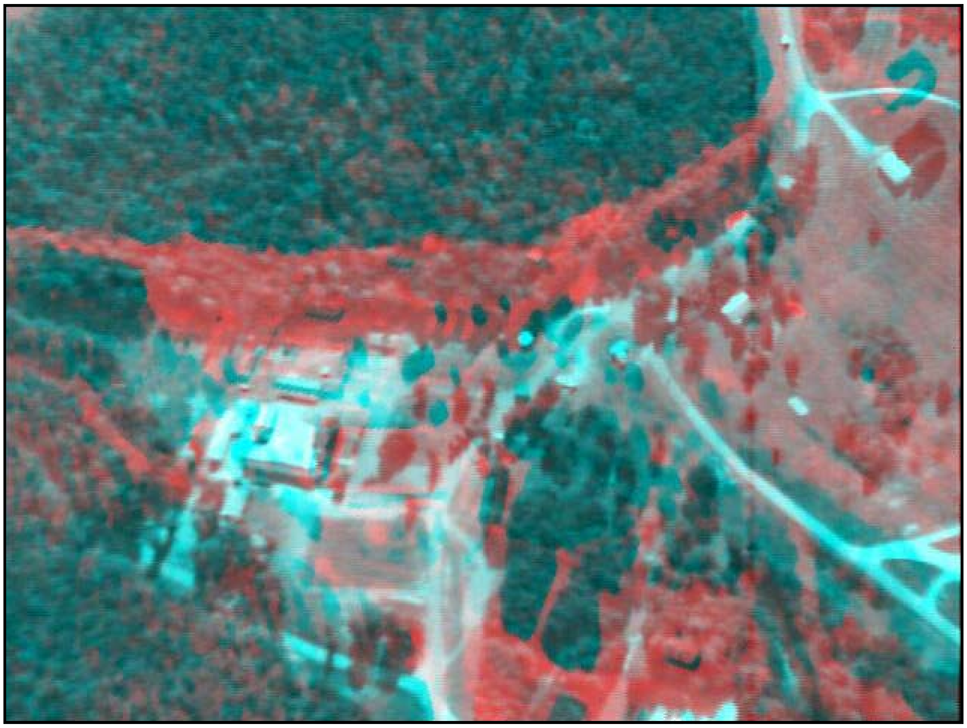


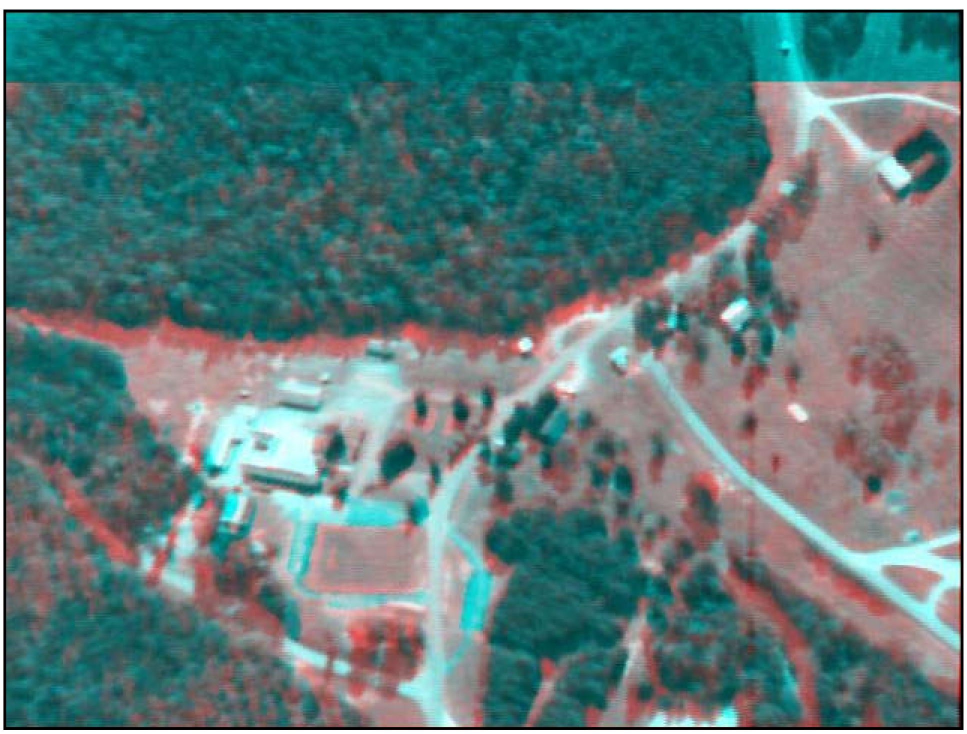


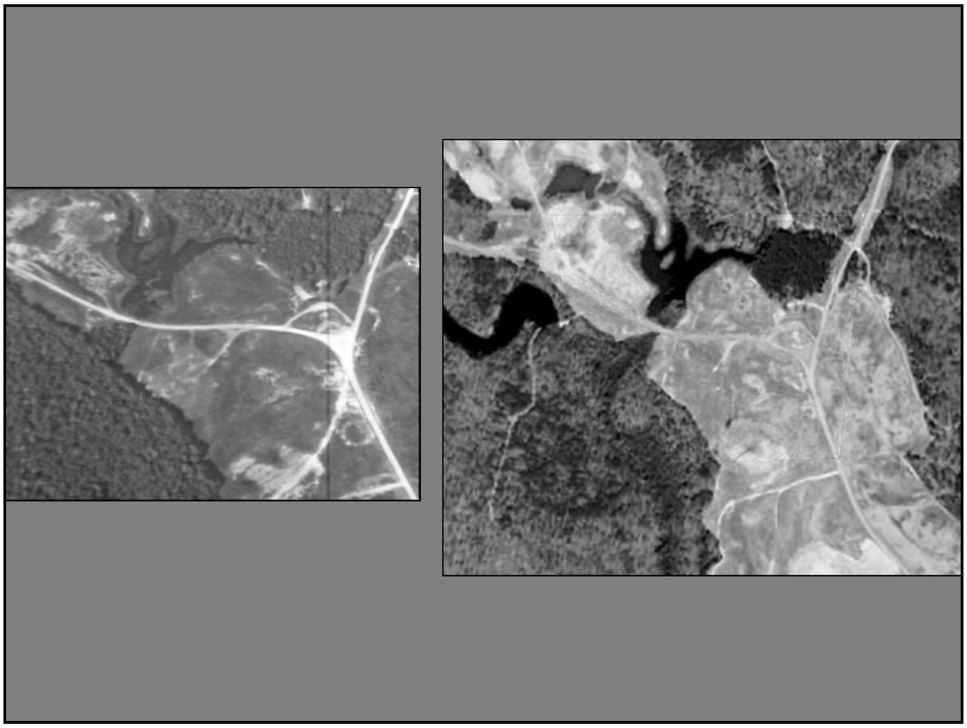


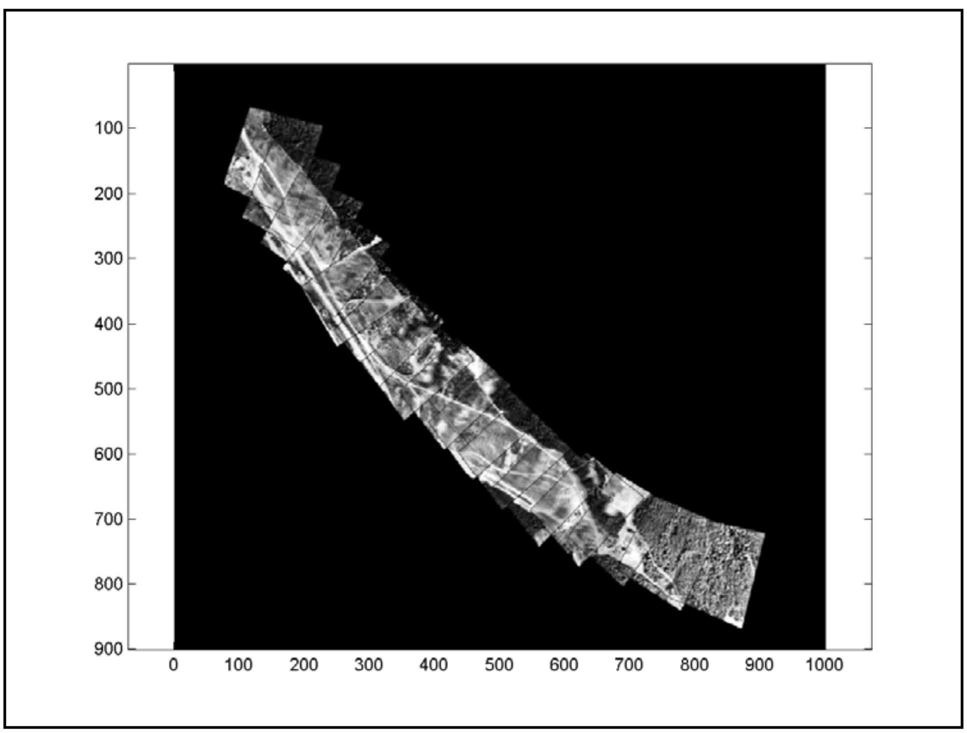


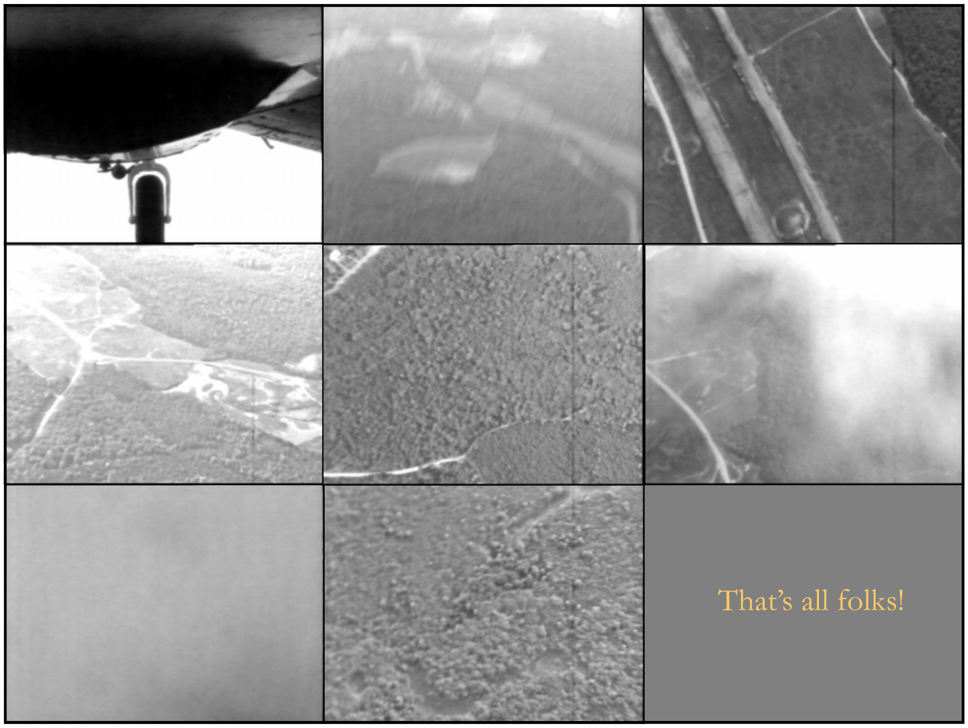









100
200
300
400
500
600
700
800
900
0
100
200
300
400
500
600
700
800
900
1000



- Yaser Sheikh, Sohaib Khan, Mubarak Shah, and R. Cannata, Geodetic Alignment of Aerial Video Frames, In Video Registration, Video Computing Series, Kluwer Academic Publishers, 2003.