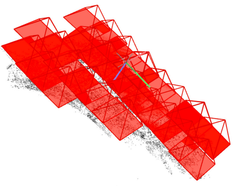
Contributions

The following notebook is a collection of my contributions to the project. I worked with the OpenCV library to automatically crop images, as well as COLMAP, an open source software for reconstruction of image collection. The following image is a COLMAP of images generating from the aerial photography set ch2m-1960-2

Ch2m-1960-2 COLMAP output



In [4]:
```
import os
import cv2
import numpy as np
import requests
from bs4 import BeautifulSoup
from PIL import Image

class ImagePrep():

    # my coding contributions
    def __init__(self, images_path, dest_path, url):
        self.url = url
        self.images_path = images_path
        self.dest_path = dest_path

    def request_images(self):
        """
        Takes in library ap_images flight url. Returns all tifs within flight list.
        dest specifies desired output path.
        """
        path = self.url
        r = requests.get(path)
        soup = BeautifulSoup(r.content, 'html.parser')
        # save files to same image destination
        dest = self.images_path

        # look for 'href' within all <a/>
        imgs = soup.find_all("a")
        tifs = [img.attrs["href"] for img in imgs]

        for tif in tifs:
            if tif.endswith('tif'):
                tif_url = path+tif
                res=requests.get(tif_url, allow_redirects = True)
                img_bytes=requests.get(tif_url).content # download bytes for a image
                with open(os.path.join(dest, tif),"wb") as img_file:
                    img_file.write(img_bytes)

    def photo_count(self):
        path = self.url
        response = requests.get(path)
        soup = BeautifulSoup(response.text, "html.parser")
        list = soup.find_all('a')
        counter = sum(1 for ul in list for li in ul)
        return counter


    def prep_photos(self): # , crop_left=0, crop_top=0, crop_right=0, crop_bottom=0):
        """
        (Altered from original notebook)
        Converts .tif files to .jpg files for ingestion into COLMAP.  May also crop images.
        """

        # inheritance
        images_path = self.images_path
        # setting destination to same folder
        dest_path = self.images_path

        for file in os.listdir(images_path):
            if (len(file.split('.')) > 1) and (file.split('.')[1] == 'tif'):
                file_path = os.path.join(images_path, file)
                image = Image.open(file_path)
                # width, height = image.size
                # image = image.crop((crop_left, crop_top, width-crop_right, height-crop_bottom))
                image.save(os.path.join(dest_path,file.split('.')[0]+".jpg")) # "_cropped.jpg"))
            else:
                pass
        return None

    def crop_photos(self, uniform=False):

        dim = None
        images_path = self.images_path
        dest_path = self.dest_path

        for file in os.listdir(images_path):
            if (len(file.split('.')) > 1) and (file.split('.')[1] == 'tif'):
                file_path = os.path.join(images_path, file)
                img = cv2.imread(file_path)
                gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

                _, thresh = cv2.threshold(gray, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
                cnts, hiers = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

                parent = hiers[0, :, 3]
                hist = np.bincount(np.maximum(parent, 0))
                max_n_childs_idx = np.argmax(hist[1:]) + 1 # hist.argmax()

                c = cnts[max_n_childs_idx]
                x, y, w, h = cv2.boundingRect(c)

                if uniform:
                    center_x, center_y = x + w // 2, y + h // 2
                    crop_w, crop_h = min(w, h), min(w, h)
                    new_x, new_y = max(0, center_x - crop_w // 2), max(0, center_y - crop_h // 2)
                    img = img[new_y:new_y + crop_h, new_x:new_x + crop_w]
                else:
                    img = img[y:y+h, x:x+w, :]

                # skips values with undetectable crop
                if max_n_childs_idx ==0:
                    continue

                dest_path_file = os.path.join(dest_path, file)
                cv2.imwrite(dest_path_file, img)
```

In [6]:
```
IP = ImagePrep('./ch2m-1960-2','./ch2m-1960-2-out', 'https://mil.library.ucsb.edu/ap_images/ch2m-1960-2/')
IP.photo_count()
```

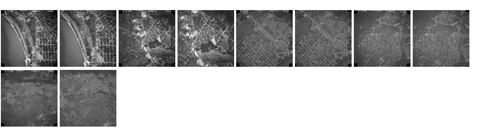
Out[6]: 60

In [10]:
```
%%time
IP.crop_photos(uniform=True)
```

```
CPU times: user 1min 19s, sys: 3.34 s, total: 1min 22s
Wall time: 1min 24s
```

In [7]:
```
def prep(path):
    images_path = path
    dest_path = path
    for file in os.listdir(images_path):
        if (len(file.split('.')) > 1) and (file.split('.')[1] == 'tif'):
            file_path = os.path.join(images_path, file)
            image = Image.open(file_path)
            # width, height = image.size
            # image = image.crop((crop_left, crop_top, width-crop_right, height-crop_bottom))
            image.save(os.path.join(dest_path,file.split('.')[0]+".png")) # "_cropped.jpg"))
        else:
            pass
    return None
prep("storage")
```

Before and after ch2m-1960-2



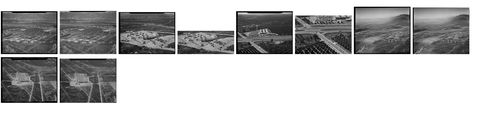
In [5]:
```
IP2 = ImagePrep('./teledyne-4x5','./teledyne-4x5-out', 'https://mil.library.ucsb.edu/ap_images/teledyne-4x5/')
IP2.photo_count()
```

Out[5]: 59

In [6]:
```
%%time
IP2.crop_photos()
```

```
CPU times: user 20 s, sys: 1.06 s, total: 21 s
Wall time: 20.9 s
```

Before and after teledyne-4x5



Geolocating Aerial Photography

The Airphotos repository contains notebooks designed to geolocate aerial photography imagery. The imagery data sets are sourced from UC Santa Barbara's library. I ran, tested, and added functions to this Airphotos repository. The Topography Comparison uses point data, from imagery sets processed by COLMAP, and compares it against USGS topographical data to find the best match.

Notes

The following are notes from running the script from the Airphotos respository.

- bs4 as web scrapes/downloads images (as tiffs) from index. Notebook includes an extra function that returns the number of photos for each specified flight.
- Opencv optionally auto crops and removes vignettes from photos.
- Converted sets of photographs from tiffs to jpegs and run through COLMAP.
- COLMAP creates 3D Constructions of aerial photography sets then exported

point data for ingestion into the code. COLMAP runs the best/fastest on Nvidia GPU.

- Script doesn't like uneven sets. The script outputs a normalized plot of area, so an uneven distribution of camera centerpoints creates a patchy area with less coverage.
- The script also dislikes small sets (n < ~16).
- The script placed ch2m-1960-2 in the Mojave desert area.
- I changed some of the original code in topo_compare() to fix reference error
- Assumptions made for topo_compare parameters:
  - I converted the pixel width/height to inches using a 1/96 coefficient
  - Scale is a parameter, but the units aren't specified in the readme. Might be problematic for future photosets that don't specify scale
- The script only supports searches/photos over California
- Running a full search on new datasets takes 24+ hours
- Ch2m-1960-2 is a set of 60 images. The code is designed for groups of 72.