Journal of Geographic Information System, 2020, 12, 84-95
https://www.scirp.org/journal/jgis
ISSN Online: 2151-1969
ISSN Print: 2151-1950

Forward Support—Cloud Based Mosaic Imagery for Emergency Operations

Elin Henningsson, Elisha Ibanga, Victor Anguiano

Spatial Sciences Institute, University of Southern California, Los Angeles, United States
Email: ehenning@usc.edu, ibanga@usc.edu, victoran@usc.edu

How to cite this paper: Henningsson, E., Ibanga, E. and Anguiano, V. (2020) Forward Support—Cloud Based Mosaic Imagery for Emergency Operations. *Journal of Geographic Information System*, **12**, 84-95.
https://doi.org/10.4236/jgis.2020.122005

Received: February 18, 2020
Accepted: April 7, 2020
Published: April 10, 2020

Abstract

The Institute for Creative Technologies (ICT) has pursued the creation of One World Terrain (OWT), which aims to provide a set of 3D global terrain capabilities and services that can replicate the coverage and complexities of the operational environment. Research was conducted in support of One World Terrain through development of best practices for the delivery of a raster mosaic via cloud hosting service, created using OptimizeRasters Geoprocoessing Toolbox and the Mosaic Dataset Configuration Script. Though ultimately successful in developing the raster mosaic and hosting it online; JPEG compression lossiness was a key issue with the larger Rose Bowl dataset. Additionally, hosting the imagery via ArcGIS Online was found to increase the compressed file size; making it comparable to the original file size of the data. Future testing should consider usage of an enterprise server to avoid this issue. MRF_LERC compression was identified as the ideal file configuration; and ArcGIS Online was identified as a poor enterprise hosting medium. We have also identified a variety of ways to improve the MDCS script in order to automate the whole process more efficiently.

Keywords

Cloud Storage, Mosaic, Raster, Tiling, Pyramids, OptimizeRasters, MDCS

1. Introduction

Geospatial Intelligence plays a critical supporting role in human security operations. In this space, preparation typically serves as a precursor to success. Emergency managers can limit the impact of disaster through mitigation and contingencies. These actions are enabled from a thorough understanding of the operating environment, which is facilitated by geospatial intelligence. At times there are obstacles to using geospatial intelligence effectively, such as the various methods of collection that can make interpreting the intelligence a slow process. There are also various databases and organizations that maintain data but do not readily share it or adhere to the same standards, limiting its broad application. Additionally, the quantity of aerial imagery collected creates storage issues. In response, the Institute for Creative Technologies has pursued the creation of OWT, which aims to provide a set of 3D global terrain capabilities and services that can replicate the coverage and complexities of the operational environment [1]. This would permit more rapid planning and decision making in support of human security operations, whether that is the management of a natural disaster area or military operations to secure high-value targets. With this ambitious project, come the issues associated with storing and compiling vast quantities of aerial imagery, collected via different sources.

This research supports OWT by narrowing down various delivery methods for raster mosaics via cloud hosting services through implementation of file compression utilizing the OptimizeRasters Geoprocessing Toolbox and the Mosaic Dataset Configuration Script (MDCS). The two applications are used in tandem to prepare data for cloud delivery to the end-user, which would likely be an emergency responder operating in a low-bandwidth environment [2]. Because of the austere environments and high-risk situations that users would likely find themselves in the compression method ultimately chosen for ICT's data needs to be rapidly accessible, facilitating hasty mission planning.

Mosaicking of raster data presents challenges not only in the development of the mosaic, but also in its delivery [3]. Delivery of the mosaic to the end user requires a delivery system compatible with various machines and bandwidths. Users will potentially be in austere environments with limited IT support, necessitating a delivery method that limits requirements of the user. The final application and how it is delivered will be dependent upon mission requirements, the number of persons accessing the data, how it is stored, and how it is to be used.

2. Background

Historically, remotely sensed images were preprocessed with software packages before use in GIS applications. This process involved orthorectification, pan-sharpening, image enhancement, mosaicking, and clipping. Each of these steps results in intermediary results that take up disk space, in addition to taking up time to write the intermediary results to a disk [4]. This traditional workflow is inefficient in disaster scenarios where rapid response is crucial. For this reason, the geospatial intelligence industry has transitioned away from local storage options to cloud environments.

There is a significant body of literature related to the efficiency of producing map tiles for web services under limited bandwidths [5]. Various mosaicking and pyramid geoprocessing toolsets exist, such as the Mosaic Dataset Toolset. However, performance of pyramids and mosaicked images is largely dependent on server capabilities. Therefore cloud storage is generally preferred, since server capabilities are maintained by an outside party. The OptimizeRaster toolset is primarily cloud-based rather than dependent on local storage and a local server, which makes it a better option than other toolsets for ICT's needs [6]. Another option for cloud-based mosaicking is ESRI's ArcGIS Image Server [7]. This tool, however, is older than OptimizeRasters and is designed primarily for ArcGIS Desktop, versions including and before 10.3 [6]. OptimizeRasters can be used in ArcGIS Desktop or ArcGIS Pro, and is available for versions after 10.3.

3. Information and Methods

3.1. Data

For the purposes of this project, two real-world datasets were provided by the Institute for Creative Technologies, in order to construct raster mosaics. These included orthorectified imagery of USC's Catalina Science Center (**Figure 1**), and the Rose Bowl Stadium (**Figure 2**). Although smaller than the requirements of the planned 3D global terrain, these two data sets allow us to test the effectiveness of various compression methods, and have many of the same characteristics, such as large structures in an urban environment (Rose Bowl), or remote locations (Catalina Science Center).

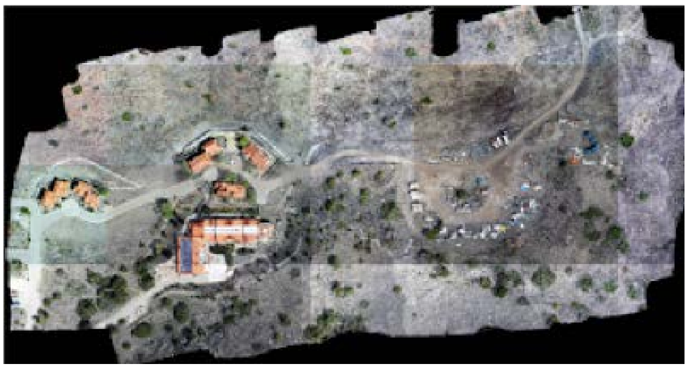


Figure 1. Full extent of the Catalina Science Center.

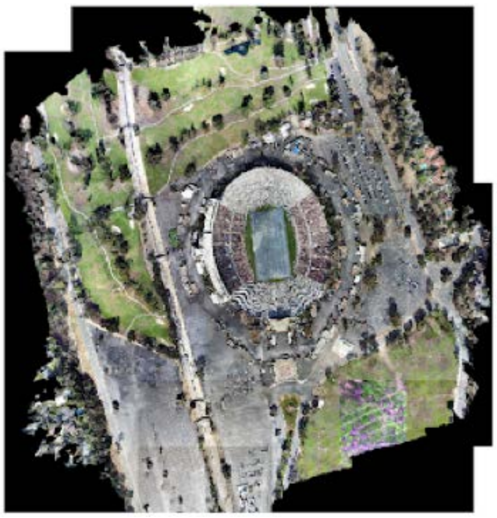


Figure 2. Full extent of the Rose Bowl dataset.

The Catalina Science Center dataset contains orthorectified imagery, meaning it is remotely sensed imagery that has been geometrically adjusted for lens distortion, camera tilt, etc. to increase accuracy. It was collected in 2016 with a DJI Phantom 4. The data was collected in a WGS84 coordinate system in UTM 11N. TIF file sizes range from 19 MB to 49 MB. The overall file size for the Science Center was 1.35 GB. The data was collected in 3 bands (RGB). The pixel type is unsigned integer and the pixel depth is 8 bit.

The Rose Bowl data also contains orthorectified imagery with fifty-nine TIF and TFW files, collected in 2017 with a DJI Phantom 4. It was collected in a WGS84 coordinate system. This data was collected in 3 bands. The pixel cell size is 0.028, 0.028. The pixel depth is 8 bit. The overall file size of the Rose Bowl dataset before compression is 2.71 GB.

3.2. Mosaicking Tools: OptimizeRasters

OptimizeRasters is an open-source geoprocessing toolbox created by ESRI to optimize file conversion and compression. It converts a variety of file formats to more optimized formats such as MRF, tiled TIFF, and Cloud Optimized GeoTIFF. This allows for more efficient and scalable data access. It also transfers to and from an enterprise storage system and/or cloud storage, such as Amazon S3, Microsoft Azure, and Google Cloud Storage. It also creates raster proxies which simplifies data management by storing small files on local systems that reference much larger files stored on a cloud [8]. Since ICT considers itself "data agnostic", the OptimizeRasters tools are useful in converting multiple file types to one uniform and optimized file type [9]. We consider MRF to be the most appropriate file type for ICT's purposes as it is optimized for both enterprise storage and cloud storage, thus supporting its accessibility for those operating outside of standard working conditions [8].

3.3. Meta Raster Format

MRF data is tiled and has pyramids, like Tiled TIFFs and Cloud Optimized GeoTIFFs, and was developed by NASA for the explicit purpose of storing and indexing rasters more efficiently than other file types. This efficiency comes from the ability to individually store the pyramids, index, and metadata as separate files, which means they can be read faster [10].

MRF also supports JPEG and LZW compression in addition to Limited Error Rate Compression (LERC). LERC is an even more efficient compression method than JPEG and LZW because it speeds up data access while also saving additional storage space. Large-bit-depth rasters, such as elevation and digital camera imagery are more prone to lossy compression, which is supported by LERC [11]. Since ICT considers itself "data agnostic," and some of the data may be collected by digital imagery, using the LERC extension during MRF conversion would be a good option for ensuring all data is uniform in the OptimizeRaster process. The MRF file is also relevant because we are moving into a time where our industry is transitioning away from the traditional desktop environment to a cloud-based platform [11]. However, if downloading data is an important component for the final product application's use, then MRF may not be an ideal file type. MRF data takes longer for users to download and takes more space to save [10].

3.4. Mosaicking Tools: Mosaic Dataset Creation Scripts

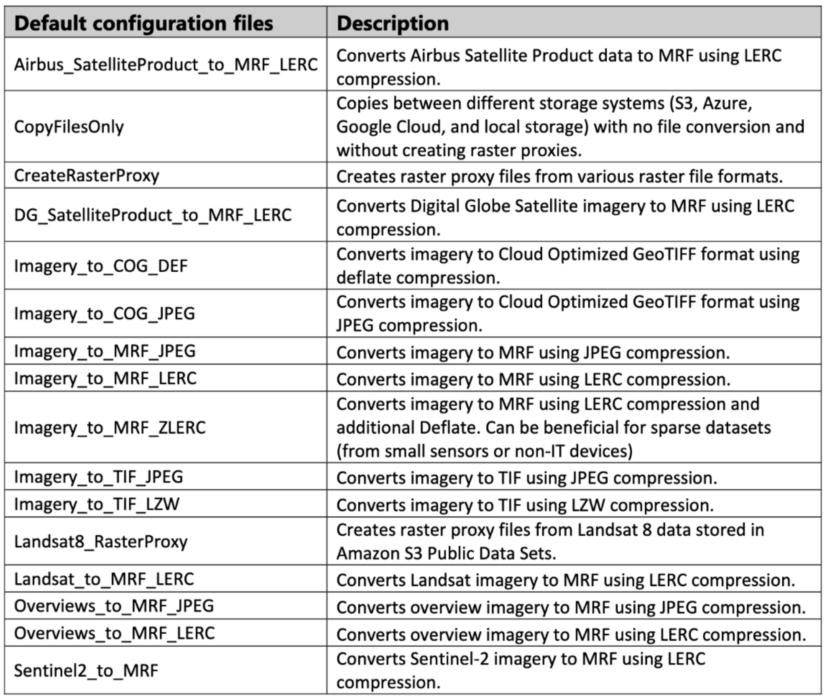
When discussing mosaic dataset creation scripts (MDCS), it is important to consider that a mosaic dataset is a dataset which allows you to store collections of raster/image data, essentially a matrix of cells and pixels organized into a grid, with each "cell" containing a value which represents information, for example temperature. As such, these datasets allow you to manage, query, and view these data collections [12]. MDCS enables the creation of these datasets by automatically adding the raster and dealing with all the desired parameters executed in a single step. This saves the user a lot of time and makes the creation much simpler via an efficient and consistent automation, as well as ensuring best practices are used [13].

Since these workflows are parameterized, MDCS is also very useful for documentation purposes, particularly for quality assurance and control. The main use for MDCS is to avoid having to create each dataset one at a time. MDSC is compatible with ArcMap, a user interface from which MDCS can be called, which is developed to make the UI process much simpler. It does, however, require knowledge of XML files and a base understanding of the creation and usage of mosaic datasets. The reason this automation is so important, providing the user has this foundation of understanding, is because manual creation is prone to error. MDSC provides a convenient solution to make the process of creating mosaic rasters repeatable and sufficiently documented [13].

3.5. Methods

The OptimizeRasters Toolbox was first used to standardize and compress the Rose Bowl and Catalina Science Center datasets. The "Input Path" and "Output Path" parameters are the user's opportunity to add either local or cloud-based sources. See **Figure 3** below for the various "Configuration File" options. In this study, both "Imagery to MRF_JPEG" and "Imagery to MRF_LERC" were used to compare the results of the two options. The MRF_LERC files were then further compressed through the MDCS windows batch file.

ICT was upgrading their infrastructure at the time of this project, so there was no access to ICT's Enterprise Server environment, Microsoft Azure or AWS S3 services. Had there been access, the compressed files would have been uploaded to an enterprise environment and more accessibly downloadable. Instead, the MRF LERC compressed data produced from MDCS was published as a service from ArcCatalog to an ArcGIS Online account (**Figure 4**) [14]. In ArcGIS Online this data is accessible only among USC Spatial Sciences Institute users, but



Default configuration files	Description
Airbus_SatelliteProduct_to_MRF_LERC	Converts Airbus Satellite Product data to MRF using LERC compression.
CopyFilesOnly	Copies between different storage systems (S3, Azure, Google Cloud, and local storage) with no file conversion and without creating raster proxies.
CreateRasterProxy	Creates raster proxy files from various raster file formats.
DG_SatelliteProduct_to_MRF_LERC	Converts Digital Globe Satellite imagery to MRF using LERC compression.
Imagery_to_COG_DEF	Converts imagery to Cloud Optimized GeoTIFF format using deflate compression.
Imagery_to_COG_JPEG	Converts imagery to Cloud Optimized GeoTIFF format using JPEG compression.
Imagery_to_MRF_JPEG	Converts imagery to MRF using JPEG compression.
Imagery_to_MRF_LERC	Converts imagery to MRF using LERC compression.
Imagery_to_MRF_ZLERC	Converts imagery to MRF using LERC compression and additional Deflate. Can be beneficial for sparse datasets (from small sensors or non-IT devices)
Imagery_to_TIF_JPEG	Converts imagery to TIF using JPEG compression.
Imagery_to_TIF_LZW	Converts imagery to TIF using LZW compression.
Landsat8_RasterProxy	Creates raster proxy files from Landsat 8 data stored in Amazon S3 Public Data Sets.
Landsat_to_MRF_LERC	Converts Landsat imagery to MRF using LERC compression.
Overviews_to_MRF_JPEG	Converts overview imagery to MRF using JPEG compression.
Overviews_to_MRF_LERC	Converts overview imagery to MRF using LERC compression.
Sentinel2_to_MRF	Converts Sentinel-2 imagery to MRF using LERC compression.

Figure 3. Configuration templates [9].

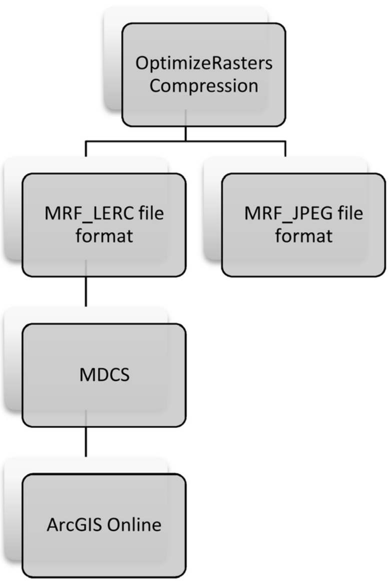


Figure 4. Workflow methodology used for this project.

could be shared with a broader audience if desired through permissions [15]. This change from Azure/AWS S3 to ArcGIS Online provided the opportunity to assess the viability of ArcGIS Online as a web hosting proxy.

4. Results

Through the use of OptimizeRasters and MDCS, we were able to generate raster mosaic files at a much smaller file size than their original, in theory making them ideal for delivery via cloud systems such as AWS S3 and Microsoft Azure [16]. Reduction of file size was key, as although the cloud systems have very impressive speeds; the download of data to physical workstations or mobile devices is limited by the bandwidth of the end user. While MDCS does compress the data well, if the client requires just one specific square raster as opposed to a greater viewing area (raster mosaic), then MDCS may not be ideal. In hosting to ArcGIS Online, we found our hosted file size comparable to its original size prior to being processed in OptimizeRasters and MDCS, making ArcGIS Online an insufficient alternative for hosting data (Table 1).

Due to questions about file sizes produced from JPEG and MRF_LERC compression and their ease of download, we performed both compression methods for both sets of data to discover the output file sizes. We discovered that JPEG compression, which is known to be lossy, removed tiles from the outer edges of the larger Rose Bowl dataset, though the compression size was smaller than MRF_LERC. Due to this, when running MDCS, we chose to use the MRF_LERC compressed files as our input.

4.1. Catalina Science Center

Table 1 provides information on the size of the Catalina Science Center dataset subject to the compression method used, as described previously. However, it is important to note that the file size increased over 700 MBs after the compressed files were hosted to ArcGIS Online (Figures 5-7).

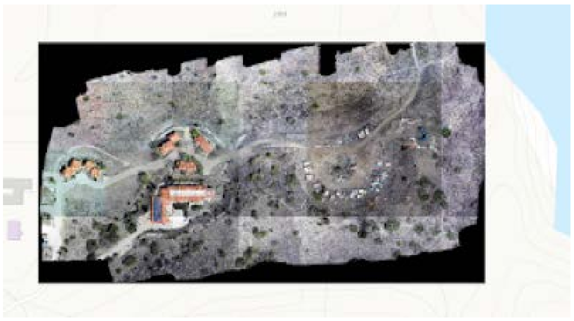


Figure 5. Illustrates the OptimizeRasters tool with MRF_JPEG compression that resulted in a file size of 148 MB.

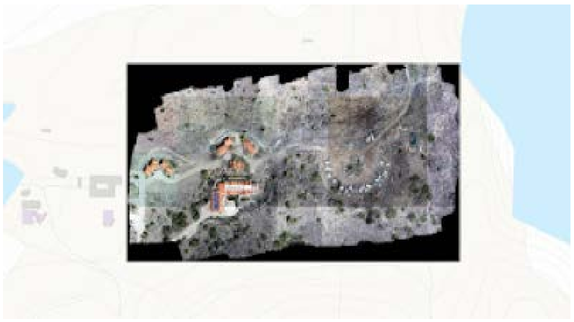


Figure 6. Illustrates the OptimizeRasters tool with MRF_LERC compression that resulted in a file size of 753 MB.

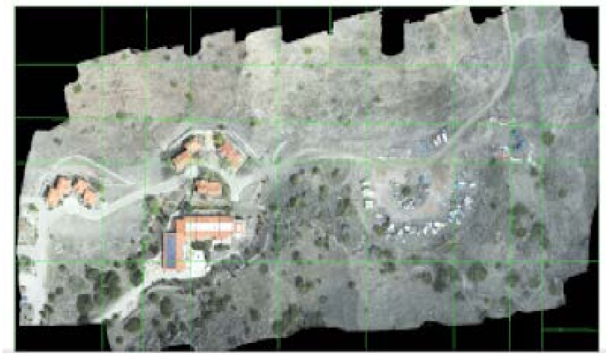


Figure 7. Illustrates the OptimizeRasters output of MRF_LERC compression wth MDCS that resulted in a file size of 1.4 MB.

Table 1. File size changes with different compression methods.

Dataset	File	Size
Catalina Science Center	Original	1.35 GB
	MRF_JPEG	148 MB
	MRF_LERC	753 MB
	MRF_LERC with MDCS	1.4 MB
	Hosted to ArcGIS Online	743 MB
Rose Bowl	Original	2.71 GB
	MRF_JPEG	282 MB
	MRF_LERC	1.4 GB
	MRF_LERC with MDCS	2.07 MB
	Hosted to ArcGIS Online	2.04 GB

4.2. Rose Bowl

The changes in file size of the Rose Bowl dataset subject to the compression method used are shown in Table 1. These results provide two significant take-aways. Firstly, Figure 8 illustrates the known lossiness of the MRF_JPEG method. This outcome resulted in the use of the MRF_LERC method for comparison. Secondly, while the combination of MRF_LERC and MDCS greatly reduced the file size, the result was rendered useless when hosted to ArcGIS Online (Figure 9 and Figure 10).

4.3. Discussion

This project was ultimately successful in developing a raster mosaic and hosting it online, though not without the identification of some key concerns. We found JPEG compression's lossiness to be an issue with the larger Rose Bowl dataset. If ICT's other datasets are of comparable or larger size, it is reasonable to assume that this would be a problem for those datasets as well. For this reason, we used MRF_LERC compression, and would recommend that future research attempts follow this procedure. We also identified imagery hosted in ArcGIS Online to

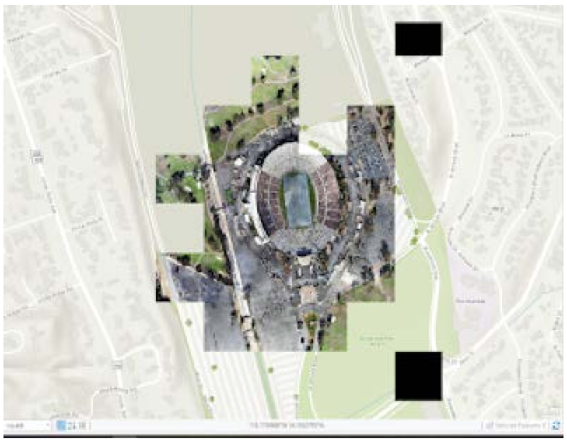


Figure 8. Illustrates the OptimizeRasters tool with MRF_JPEG compression that resulted in a file size of 282 MB.

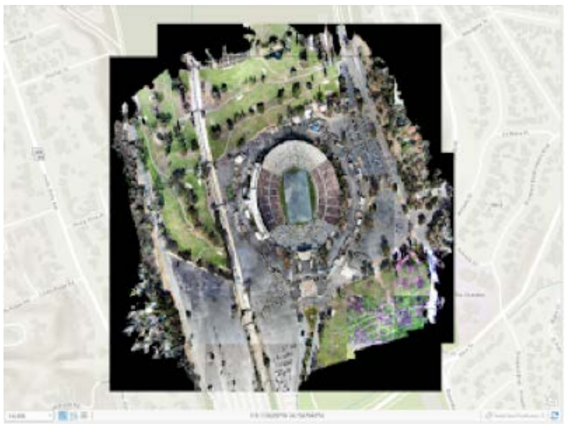


Figure 9. Illustrates the OptimizeRasters tool with MRF_LERC compression that resulted in a file size of 1.4 MB.

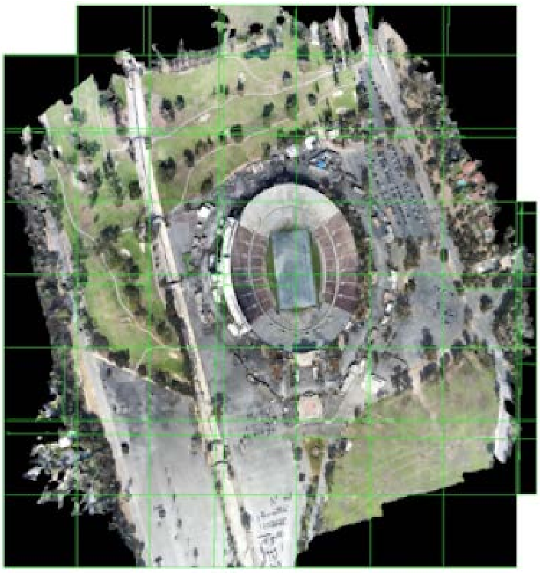


Figure 10. Illustrates the OptimizeRasters output of MRF_LERC compression wth MDCS that resulted in a file size of 2.07 MB.

increase the compressed file size close to the original file size of the data. This project successfully identified MRF_LERC compression as the ideal file configuration, and was successful in eliminating ArcGIS Online as a hosting medium.

5. Recommendations

5.1. Cloud Environment

Due to ICT's infrastructure update at the time of this study, we were not able to integrate their cloud infrastructure into our project via AWS or Azure. For this reason, all inputs and outputs for the OptimizeRasters geoprocessing tool were located in local folders. For projects with large datasets especially, utilizing web hosting would be beneficial. If a study such as this were to be repeated, these authors would recommend utilizing web hosting capabilities rather than ArcGIS Online.

5.2. Mosaic Dataset Creation Scripts

In ModelBuilder, OptimizeRasters can be added to the MDCS script to automate the entire two-step process. The benefit of automating both OptimizeRasters and MDCS together is of course, that it is a valuable way to save time and increase efficiency when working with large datasets. These authors attempted to automate the two processes together but were unsuccessful due to the time constraints of the project. Future research on this topic should also involve adding a script so that all batch files can be run at the same time. This way, the script can be automated to run daily, weekly, etc. Temporal automation would be particularly useful in the case of satellite imagery collection. Both of these refinements to the script are areas where we suggest future work be focused.

6. Conclusions

This project aimed to identify successful file compression and hosting methods for imagery data. By utilizing the OptimizeRasters toolbox and MDCS, development of the raster mosaic and online hosting were ultimately successful. However, JPEG compression lossiness of the Rose Bowl dataset indicated that MRF_LERC compression was the ideal configuration. Additionally, hosting the imagery via ArcGIS Online was found to increase the compressed file size close to the original file size of the data, making the compression obsolete. Future testing should consider usage of an enterprise cloud environment to avoid this issue.

Beyond the scope of ICT's One World Terrain, file compression and hosting have other Human Security applications. Any organization with large amounts of imagery data, require an effective way to store their data and share it within their organization, particularly in times of crisis when data needs to be quickly accessible. Humanitarian, defense, and environmental entities face data emergencies during natural and human imposed disasters. During these emergencies, geospatial personnel need access to imagery data, downloaded to their devices quickly and efficiently so that reputable decisions can be made with this information.

Acknowledgements

This project is the result of research done for a MS degree at the University of Southern California focused on Human Security and Geospatial Intelligence. The authors would like to thank USC for their training in geospatial intelligence, GIS, database management, and remote sensing. We likewise thank our advisors (Col. retired) Dr. Steven Fleming and Dr. Jason Knowles and acknowledge their guidance and review of this project. We would also like to acknowledge the Institute for Creative Technology for providing us with the data and framework necessary for completing this project.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this paper.

References

[1] Institute for Creative Technologies. One World Terrain. http://ict.usc.edu/prototypes/one-world-terrain-owt/

[2] Benkelman, C. ArcGIS Workflows for Optimizing Image Management Services in the Cloud. https://proceedings.esri.com/library/userconf/devsummit17/papers/dev_int_13.pdf

[3] ESRI. Mosaic Dataset. ArcMap. https://doc.arcgis.com/en/imagery/workflows/standard-workflow/overview/overview-mosaic-datasets.htm

[4] Xu, H. and Becker, P. (2012) ArcGIS Data Models for Managing and Processing Imagery. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, **XXXIX-B4**. https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XXXIX-B4/97/2012/isprsarchives-XXXIX-B4-97-2012.pdf https://doi.org/10.5194/isprsarchives-XXXIX-B4-97-2012

[5] Huang, W., Wang, C. and Tang, D. (2018) Design and Applications of Rapid Image Tile Producing Software Based on Mosaic Dataset. *The International Archives of the Photo-Grammetry, Remote Sensing, and Spatial Information Sciences*, **XLII-3**. https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLII-3/2225/2018/isprs-archives-XLII-3-2225-2018.pdf https://doi.org/10.5194/isprs-archives-XLII-3-2225-2018

[6] ESRI. Core Geoprocessing Tools for Raster Data. http://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/core-geoprocessing-tools-for-raster-data.htmsa

[7] ESRI. ArcGIS Server. https://enterprise.arcgis.com/en/server/latest/get-started/windows/what-is-arcgis-image-server-.htm

[8] GitHub. OptimizeRasters. https://github.com/Esri/OptimizeRasters

[9] ESRI (2018) OptimizeRasters: AWS Lambda Implementation. User Documentation, Redlands.

https://github.com/Esri/OptimizeRasters/blob/master/Documentation/OptimizeRasters_UserDoc.pdf

[10] Norton, J. (2019) Cloud Optimized GeoTIFF vs the Meta Raster Format. Element 8423 RSS. https://www.element84.com/blog/cloud-optimized-geotiff-vs-the-meta-raster-format

[11] Becker, P. (2016) MRF as a Cloud Optimized Raster Format and LERC Compression. ESRI White Paper, Redlands. https://pdfs.semanticscholar.org/8267/f0b26a0b9f21c2c7d9ea3fdcc59903ac3157.pdf

[12] ESRI. Raster Pyramids. http://desktop.arcgis.com/en/arcmap/10.3/manage-data/raster-and-images/raster-pyramids.htm

[13] GitHub. Mosaic Dataset Creation Scripts. https://github.com/Esri/mdcs-py

[14] ArcGIS Enterprise. Share Imagery as a Tiled Map Service. https://enterprise.arcgis.com/en/server/latest/get-started/windows/share-imagery-as-an-arcgis-online-tiled-map-service.htm

[15] ESRI (2019) Downloading Data from an Image Service. https://desktop.arcgis.com/en/arcmap/latest/map/web-maps-and-services/downloading-from-an-image-services.htm

[16] Harvey, C. and Patrizio, A. (2019) AWS vs. Azure vs. Google: Cloud Comparison. https://www.datamation.com/cloud-computing/aws-vs-azure-vs-google-cloud-comparison.html