Real-time image processing on handheld devices and UAV

Joakim Andersson
Daniel Persson

Supervisor, Klas Arvidsson (IDA), Jonas Bromö (IT-Bolaget Per & Per AB)
Examiner, Jonas Wallgren (IDA)
Upphovsrätt

Detta dokument hålls tillgängligt på Internet – eller dess framtida ersättare – under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för ickekommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida http://www.ep.liu.se/.

Copyright

The publishers will keep this document online on the Internet – or its possible replacement – for a period of 25 years starting from the date of publication barring exceptional circumstances.

The online availability of the document implies permanent permission for anyone to read, to download, or to print out single copies for his/hers own use and to use it unchanged for non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional upon the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: http://www.ep.liu.se/.

© Joakim Andersson and Daniel Persson
Real-time image processing on handheld devices and UAV

Joakim Andersson  
IDA  
Linköping, Sweden  
Joaan075@student.liu.se

Daniel Persson  
IDA  
Linköping, Sweden  
Danpe503@student.liu.se

ABSTRACT
The forest industry needs an up-to-date overview of certain areas of a forest, to either estimate damages after a storm or assess its overall health. Today, the use of unmanned aerial vehicles (UAV) have exploded. Almost anyone can own one and they are very easy to operate. They are often equipped with accurate sensors and cameras that can be used for several productive applications. This paper investigates if a UAV equipped with positional sensors and a high resolution camera can be used for taking aerial photographs and together with a mobile devices create a coherent orthophoto in real- or near real-time. Three different seam stitching algorithms are tested and evaluated based on speed, accuracy and visual appearance. The results of using a DJH Matrice 100 UAV together with an iPad Air is presented both qualitative and quantitative.

Keywords
Mapping; Orthographic Stitching; UAV; Orthophoto

INTRODUCTION
Today, most information and image gathering for the forest industry is done by airplanes or helicopters. If a company need an updated view over an area it is often associated with high costs and long lead times.

An alternative to large manned aircrafts is the smaller unmanned aerial vehicles (UAV). UAVs can be equipped with high resolution sensors and cameras, and can operate autonomously [1].

Traditionally, images and sensor data has been post-processed on a stand-alone computer to e.g. create a georeferenced map over the photographed area. What if the gathered data could be processed during flight and presented near instantaneously? This paper focuses on testing the possibility of achieving this on a UAV together with the processing on a handheld device.

The company IT-Bolaget Per & Per  
IT-Bolaget Per & Per is located in Mjärdevi Science Park, Linköping, Sweden, and is a small IT firm. They focus on consulting as well as application development.

IT-Bolaget Per & Per is today working primarily on applications providing Geographic Information Systems (GIS) for both private individuals and companies in the forest industry.

The main application is developed for iOS devices. Their app provides the customer with all kinds of GIS data, like height maps, orthophoto, property maps, etc. The data is stored on a central server, but can also be downloaded for offline usage. When customers request new GIS data, it is pushed to the central servers, making it possible to synchronize the locally stored data.

Problem Description
A recently found need to get updated GIS data has been discovered by IT-Bolaget Per & Per. An up-to-date version of geographical images over an area is often desired, especially after e.g. a big storm. Existing methods using rented helicopters or airplanes might be expensive and lead to long waiting times. The data produced also has a need to be post-processed before delivery to customers.

This paper investigates the possibilities of UAVs doing a similar job as manned aerial vehicles photographing an area. The purpose is to let customers update their own maps whenever they desire, at a lower cost and quicker rate. The UAV will be a one-time investment for the customer unlike the recurrent hire of external services.

The goal for this investigation is:

- Implement a prototype that can photograph an area autonomously, process the data, and create a coherent orthophoto and a 3D point cloud in real- or near real time.

The most important problem to investigate is the autonomous processing of image and sensor data during flight. No human interaction is possible during this stage. This makes stitching of the images from the UAV to create an orthographic mosaic difficult. The only human interaction is deciding the area to fly over. The following research question is investigated:

- What image stitching method is best suited for autonomously creating coherent orthophotos in real- or near real-time?

There is a possibility of discovering that this is not possible with current technologies or within our given timeframe of this investigation. Even if this fails to produce a working prototype, the research data could still be useful for future investigations.

Scope and Limitations
This investigation is restricted to only work with a UAV specifically designed for development. A use of an iOS tablet/smartphone is also required for giving the UAV flight directions and for assistant data processing.
Tests will be focused on processing capabilities, data link performance and battery life of the UAV.

The resulting orthophoto is to be presented on the app from IT-Bolaget Per & Per on an iPad Air.

THEORY
What is a UAV?
An Unmanned Aerial Vehicle is an unmanned aircraft system that either uses its onboard computers to fly fully autonomously or is piloted from ground with some kind of remote control [2].

Today there is a growing market for UAVs and they are used for various purposes. The United States military uses UAVs [3] to carry out dangerous missions instead of sending aircrafts with humans as pilots that might be harmed. These unmanned vehicles are controlled by an operator located on ground and are therefore referred to as drones. There is also an increasing use of drones while filming movies and sports [2]. The drones are much cheaper to operate and you do not need to hire big expensive helicopters to do the job.

In this paper we use the autonomous definition of UAV, since we will only give it instructions before flight. During flight the UAV will operate completely on its own.

Choosing the UAV
Since our goal is to implement a map on an iOS device and present it in real or near real time, it introduces a few limitations. There must be a way of communicating between the UAV and the iOS device during flight at large operating ranges.

Another important feature is autonomous flight. To greatly simplify the process of capturing photos over an area, the UAV should be able to support some kind of GPS waypoint navigation system or flight mission control [1]. This could be implemented by us, but an existing implementation would be preferable.

There is a need to find a UAV that is basically ready to fly out of the box. An SDK for controlling and communicating with the UAV is also a requirement. This would save a lot of time and help focus on the main part of this investigation.

Matrice 100
The DJI Matrice 100 [4], as pictured in Figure 1, is a modular development platform aimed for testing and research. Data transmission range using radio (~5.7 – 5.8GHz and ~2.4GHz) is specified to be up to 3.5km and a battery life of up to 19 minutes with the Zenmuse X3 camera, shown in Figure 1. The Matrice 100 has an onboard SDK which allows further development. A mobile SDK that works together with the onboard SDK is also available for the Matrice 100.

An optional camera can be attached to the UAV, capable of taking low distorted 12MP photos. The camera is suspended in a three-axis gimbal, making it possible to compensate for vibrations and angular variations. When a photo is taken, GPS and altitude data are saved in a geotag in the photo file.

The handheld controller can natively communicate with mobile devices such as Android and iOS. This allows for a real-time view of what the UAV is currently observing.

![Figure 1. The Matrice 100 with the Zenmuse X3 camera.](image)

The camera has several “shooting modes” and a playback mode. While the camera is in a shooting mode, it is not possible to read an already taken photo from the memory card. This means that once a photo is taken, the camera mode has to be changed for the iPad to download a photo.

The DJI Mobile SDK features available methods for communicating with the UAV, such as controlling the camera, receiving live video feed and setting navigational waypoints. It is also possible to receive flight system data such as GPS coordinates, altitude and battery status.

Presenting maps
Accurately creating a map from photographs demands a lot of information. The location of the photo must be known to a certain level of accuracy, as well as the orientation. The height the photo was taken at is also important.

To display a map on iOS it must be formatted in a certain way, an example is with MapKit [5]. It is not a matter of just displaying a photograph with high resolution. The photograph must be saved with the correct information i.e. coordinates and altitude. If this information is not included in the photograph it will be difficult to place the photo in the correct location in the existing map application.

The spherical surface of the earth is projected on a two dimensional grid [5], as in Figure 2. Each square or tile when “zoomed in” is divided into four new tiles, and so on. This is often referred to as a quadtree implementation. When the map is max “zoomed out”, that is, when the whole earth is visible in one tile it is called zoom level 0. At zoom level 1, a quarter of the earth is visible in one tile, this pattern can be repeated as far as a desired detail level is achieved. On most map applications, each tile usually is an image with a resolution of 256x256px, or sometimes 512x512px. This means that a photograph must be “tiled”, or cut down into these squares.

If the original photograph is taken with a desired level of detail, it can be tiled into a high zoom level, e.g. level 20. For each zoom level above, it is only a matter of scaling...
down the already tiled images, i.e. each tile on zoom level 19 consists of four scaled down tiles from level 20 and so on.

![Image](image_url)

**Figure 2. How different zoom levels work. Showing zoom level from top to bottom: 0, 1, 2.**

Camera lens distortion is a problem when taking aerial photographs. If lens distortion is big, the photographs can get a “fish-eye” look and needs correction before placing them in the map application.

**Image stitching**

When stitching photographs together there is a requirement of a certain percent of overlap in the photographs taken. This means that a percentage of a photograph taken will also be in the next one. The overlap is to facilitate the process of stitching; if the process can find pieces that are the same in the different photographs, building the mosaic will be easier [6].

**Image crop**

The simplest “stitching” method will be the crop method, where images are just aligned and placed on top of each other. This requires little computation, but the seams will probably be visible. If the missions are planned with waypoints a known distance apart, it is possible to calculate the number of tiles between each waypoint. When a tile at the border between two waypoints collide or overlap, the last tile calculated will simply overwrite the old one on the disk.

**Alpha blending**

The second method to stitch the overlapping images together is to blend them together, without caring for pixel matching. This means that the overlap gradually fades to better hide the seam. This can be implemented with a method called alpha blending. The alpha channel in an image is commonly referred to as the transparency level per pixel. To create a gradual overlap the alpha channel of an image placed on top of another must follow a curve where the alpha is 0 (transparent) to 1 (opaque) [7]. In this paper the curve tested will be linear. Other types of curves can be used, and give other visual appearances, but the performance of image merging should be the same.

**Pixel matching**

The last method to investigate is the pixel matching method. This method includes heavy computation of finding matching pixels in overlapping images. Perfectly matching the overlap results in the seam between the images being invisible as Z. Assaf et al. [7] describes it. A problem of using the last method is that the images cannot be moved, warped or stretched too much, otherwise they lose the correct position in reference to the earth. The basic theory behind pixel matching algorithms is to find “the same” pixels, or groups of pixel in two images, and then transforming the images to align the matching patterns. These groups of pixels are something that significantly stands out from the photo and are called features [8]. Features in photos can be various things, like edges, corners and other shapes. If the same features are found in two photos the features can be placed on top of each other where they match and be stitched together. The features found can differ in size or rotation, the photos might therefore need to be stretched, warped or rotated to perfectly match when stitched together.

**Color correction**

Automatic white balance and exposure control can cause color differences in the photos over time. The algorithms calculating these parameters in the camera might falsely change values when traveling over terrain with different colors.

**Orthorectification**

Orthorectification is a method for aligning images with the “real world”. As G. Zhou [9] describes it, it is one of the most important processing steps that need to be done.

Based on the focal length, camera sensor dimensions, height, GPS coordinates, orientation and resolution of the camera, an image can be rectified to the surface of the earth. If the photo is taken orthogonally, the formula to calculate pixel resolution (P in meters / pixel) is as follows:

$$2 \times h \times \tan (\tan^{-1} \left( \frac{d}{2 + f} \right)) \cdot \frac{r}{\tan^{-1} f} = P$$

Where h is the height, r is the resolution in x or y, d is the length of the sensor in x or y respectively and f is the focal length. The pixel resolution is used to select what height the UAV should fly at, to achieve the desired level of detail in the final map.

**Point cloud**

A 3D point cloud in its simplest form is a set of coordinates in three dimensions. In GIS applications, point clouds are often used for creating digital elevation models (DEM). A DEM can either describe the surface of the earth, or the structures on it, or both.

A common way of creating point clouds is using lasers (LiDAR) [10] from either satellites or airplanes. Since the Matrice 100 is not equipped with a LiDAR, another approach is using stereo photogrammetry. Using two images taken a known distance apart at the same object, triangulation can be
used to calculate the height of the object [11]. OpenCV [12] has algorithms implemented to calculate so called disparity maps from two images. A disparity map in this context is presenting how far different pixels or features have moved between the two images. This distance information together with camera parameters and altitude can then be used to calculate the height of said pixels or features.

RELATED WORK
Autonomous flights using UAVs for mapping areas is no new idea [1, 13, 14]. Mapping has also been done in real or near real time by G. Zhou [9]. The UAV used in G. Zhou’s study was operated by a pilot on the ground and the video stream was captured by another person holding an antenna. The stream was then processed by computer equipment located in a nearby van.

Hugenhoetz et al. did a survey using a winged drone with the purpose to create both an orthographic mosaic as well as a DEM. The information was not processed in real time, but gave promising results [10].

As far as we can tell, the processing has never been done in real-time on a mobile device during flight.

METHOD
Approach
We start by getting familiar with the SDK for both the DJI iOS framework, as well as the onboard SDK for the Matrice.

The first test is with just one photo over an area, and sending it in the correct format to the existing map application. This is to get an overview over where possible optimizations are needed for future development.

The next step is to look in to whether the best approach is to use a photo by photo technique, or a continuous video stream to create a mosaic.

Approach A: Continuous video stream
The Zenmuse X3 camera mounted on the Matrice supports a live video stream to be sent over a radio link to a connected Apple or Android device. The video stream is available with a maximum resolution of 1080p, and a framerate of 60fps. The stream can be processed in real time on the iPad. Another approach is to not stream the video, but save it locally on the UAV. This allows for higher resolution video, up to 4K. The processing can begin after the recording is done. Accessing the data after the recording is not optimal due to the inability of processing the data in real-time, which is our goal. Another approach is to record a video onboard the UAV during short periods of time and then download the recordings. Depending on the resolution and how long the recordings are, the file size could grow quite large which can result in bigger download times of the video files from the UAV. This is a drawback because the camera has to change from the shooting mode to the playback mode for us to be able to download the video file. After the file is downloaded the camera needs to change back to shooting mode for it to be able to record another video. Depending on if the download takes too long or the UAV has too high velocity, some areas might not be recorded.

Approach B: Photo by photo
In this approach photographs are captured at certain points over the defined area. Where the photographs should be taken is calculated at the same time as the route over the area is calculated. When the UAV has taken a photograph it will switch to playback mode which will enable the download of the photograph and then switch back to shooting mode to take the next photo. In this approach there is no need to care about video length because a photo can be captured in much less than a second. One thing that still need to be taken into account is the velocity of the UAV. Depending on the velocity and the camera shutter time the photographs can get “blurry”. This is something that needs to be investigated, whether the photographs can be taken while the UAV is still moving or if it needs to stop to capture each photographs. However, in this study we will only take photos when the UAV is stationary, leaving tests of speed and blurriness for future work.

Choosing an approach
In the video approach there is a lot to consider:

- Is it possible to process all data from a continuous video stream on an iPad?
- If the approach to record short periods of time is chosen, what is the optimal recording time for each video?
- How fast is the UAV able to fly while recording and is there time to download each video file between recordings?

If there is not enough time between the recordings to download each media file, the UAV needs to hover in the air until the file is downloaded. When the download is complete the UAV can continue with the mission and record the next area. If the video approach missions end up taking longer time due to the pauses for download, a need to plan smaller missions is required because of the 15-20 minutes of battery life.

In the photo approach there is only a need to calculate the best spots on the route in the mission to capture photographs. The benefits with the photo approach is that the files will be considerably smaller than the video files. The photographs can therefore be downloaded between each photo taken while the UAV is still moving. This will save time and can result in that the UAV can carry out longer missions. The photo approach has less data to process due to only capturing one photo at each spot on the route. The video approach will capture up to either 25 fps in 4K, or 60fps in 1080p, which can result in large amount of data to process in only a few seconds. Photographs captured by the camera will also include geotag information by default.

The photo approach does not have as much to consider as the video approach. The following research will therefore focus
on the photo by photo approach, due to time constraints in this study, thus leaving the video approach for future works.

**Implementation**

**UAV route planner**

An already existing application for iOS called “MapDemo” used for in-house testing of new features at IT-Bolaget Per & Per is modified. The first step is to implement a mission controller with the DJI SDK, for setting navigation waypoints and mission actions. The actions include rotating the aircraft and camera to correct position, as well as taking photographs.

The existing map in the application is used to create missions. An area is selected by zooming in on the desired area in the map application and then press a button to prepare a mission. The route planner then calculates a path for the UAV to follow, and setting actions for when to take photos. The path is calculated based on desired height, overlap of photos and field of view (FOV) of the camera to completely photograph all parts of the selected area.

The mission is then uploaded to the UAV. When the UAV receives a good GPS signal it can start executing the mission. The UAV will now fly along the predetermined route and take photographs at certain points that was calculated on the iPad.

If the UAV flies out of transmission range during a mission, the default behavior is to return home. This behavior is not changed due to safety concerns. The drawback with losing radio connection is that the iPad will no longer receive photographs taken by the UAV, which will interfere with our real-time processing of the data. This can be solved by storing the photographs on the UAV until radio connection is once again established and the data can be sent. However, this is out of scope for this investigation.

The default setting for the camera is to use automatic white balance and automatic exposure control. The white balance setting is changed to a locked setting for sunlight. The exposure settings is left as automatic to handle possible cloud coverage and light conditions.

**Image processing**

An overview of the complete processing chain for one photo is as follows:

1. Extract location data from the photo.
2. Calculate the bounding box, that is, the coordinates for the corners in the photo based on altitude and the location data.
3. Calculate the most appropriate zoom level to place the tiles from the photo on.
4. Based on the bounding box and zoom level, extract the intersecting tiles of neighboring photos.
5. Calculate the overlapping offset of the photo.
6. Based on the overlap, discard not useable tiles, and place the tiling and stitching operation on a queue.
7. For each tile, cut out the correct tiled image as a new image.
8. Based on selected stitching method, the appropriate methods are called.
9. If the crop method is used, the tile is just saved as is. Otherwise the tile is processed according to the selected stitch method.
10. Once the queue is empty, three zoom levels “above” are created by merging four tiles and scaling them down for each level.

Once a photo is downloaded to the iPad, it can be processed. The first step is to align the photo relative to the surface of the earth, a method called georeferencing. The relevant information needed to achieve basic georeferencing is stored in the Exif header of the images. Once the photo is georeferenced, it is aligned with neighboring photos. The overlap is then calculated and the tiling and stitching process begins.

Since the overlap is often a good portion of the original image, a lot of image data is cut away and discarded. If the overlap used in our algorithms are 60%, only about 30% of the original image is used further in the stitching and tiling chain.

All tiles fitting inside the bounding box of an image are calculated, and since tiles are independent of each other the tile slicing can be done in parallel. Tiles that are not inside the usable area of the image are discarded. All tile slicing operations are then placed in a queue, where a number of worker threads share the workload. A few different numbers of worker threads are tested to see how well the work scales depending on the number of available cores.

Three stitching methods are implemented and tested:

- Cropping
- Alpha blending
- Pixel matching

When using the crop method, no additional computation is needed. As described in the theory chapter, when a new tile happens to overlap an old one, the old tile is overwritten.

The blending method requires a bit more computations. Once two tiles are found to overlap each other between two waypoints they are merged together with a linear gradient alpha mask. First, the old tile is drawn in a CGContext, a type of canvas in iOS, then the second tile is drawn on top with an alpha mask. The mask is basically a greyscale image, where the color tells the drawing method how much of the alpha channel value to use. The mask is generated at runtime, and the direction of the gradient is dependent on the merging direction of the tiles. A special case found is in the corners where several tiles overlap, then a radial gradient is created to better hide the seam. This is not optimal, as the data for the several overlapping tiles in the corner are not stored separately. The first overlap between two tiles in a corner tile
is written to disk, then when the third tile is to be merged, some information is lost and small artefacts are visible.

When the pixel match stitching method is selected, the OpenCV (Open Source Computer Vision) [12] library is used. This library is chosen because it has decent documentation and can be integrated in the existing application without the use of an external GUI. OpenCV has several feature detector/matcher algorithms that can be used for stitching photos together. OpenCV also have a built-in stitcher class that can be modified with different parameters to give desired stitching result. In the implementation of pixel matching the built-in stitcher class is used with slightly modified parameters.

When performing pixel match stitching, three different approaches are used to see which one gives the best result and performs the fastest:

The first approach is to pixel match full scale photos two at the time. Once the iPad have received two photos it can start the pixel matching process. When a new photo arrives, it will be pixel matched together with the last calculated result. This will continue until the iPad have received and stitched the last photo. The resulting image can then be tiled and used in the map application.

The next approach is to perform pixel match stitching using all photos in full scale at the end of a mission. If all photos demands too much resources, bigger chunks of full scale photos are pixel matched.

The last approach is to only perform pixel match stitching on the overlapping tiles. This approach is similar to the blending method, only that overlapping tiles are stitched based on pixel matching instead of alpha blended.

Once the images are tiled and stitched into a mosaic, it can be presented on the map application.

Camera calibration
To calibrate the camera a measuring tape, markers and a compass is used. Three markers are placed on a flat surface with 50m between them in a north-south direction. A photo is then taken at 100m altitude while the UAV is facing north, and another one while the UAV is facing west. Photographs are captured in both north and west direction to be able to calibrate both width and height parameters of the camera.

Three markers are then placed in the map application with the same distance apart used when capturing the photographs. The photo with the markers is then aligned to the map markers, and the offset between the markers in the photo and in the map is measured. The width and height parameters of the camera are then changed accordingly to the measured offset. The camera is correctly calibrated when the markers in the photo align with the markers in the application.

GUI
A graphical user interface (GUI) is implemented to achieve easy control of overlap and altitude while planning missions. This is to avoid the need to bring a computer in the field, and rebuilding the app once a parameter has to be changed. A few text areas displaying GPS status, battery percentage and various debug texts are also implemented.

Evaluation
A set of test photos is taken over a defined area. During this test, the download time for each photo is measured, as well as time intervals between each photo taken. This interval is the allowed time for processing of each photo. The photos are then processed using the methods mentioned above, while measuring time consumption for each step.

To test the orthorectified accuracy of the photos, markers are placed at known coordinates with a known distance between them. The markers will then be photographed and the offset measured. The distance between the markers are also used to calibrate the camera parameters.

Real-time processing is evaluated based on how fast the iPad is able to process the data from the UAV. The optimal and real-time case would be to have a coherent orthophoto when the UAV have finished its mission and have landed. If the iPad is finished processing shortly after the UAV has landed this will be considered as near real-time processing.

To measure time consumption for the stitching algorithms, a test using built in performance measure blocks is implemented. The Entire chain of computing is tested on a set of 25 test images, and is measured 10 times using the measure blocks. The standard deviation is also calculated automatically, giving an indication of how scattered the measurements are.

The stitching methods used in this investigation will be evaluated based on visual appearance, speed and accuracy.

RESULTS
Cropping
Visual result of the crop stitching method is presented in figure 3. The borders between each image is visible due to small misalignments.

Alpha blending
Alpha blending is a small extension of cropping, resulting in tiles not taking much longer to compute. In fact, our measurements showed it to perform almost exactly as fast as cropping. Visual results for blending can be seen in figure 4.

Pixel matching
The implementation of pixel matching only gave a desired result when performing pixel match stitching on chunks of up to ten full scale photos at a time. Pixel matching all photos at once demanded too much resources, the iPad ran out of memory and the application crashed. When pixel matching ten full scale photos the result is seamless, there is no sign of corners or edges. A small piece of the full scaled photo pixel matching result can be seen in figure 5. However, for some images the algorithm failed to find matching features, and the stitching would fail.

Figure 4. Corner with alpha blending.

Figure 5. Pixel matching with full scale photos.

Performing pixel match stitching with overlapping tiles in most cases gave no result at all. When the pixel match failed for the overlapping tiles, the tiles would remain untouched and the result would look like cropping which can be seen in figure 3.

Performance
When the UAV is flying at 100m altitude and has an photo overlap of 60% in all directions, the time between each photo takes approximately 24 seconds. The goal for our algorithms are to complete the whole chain of stitching and tiling within this timeframe to call it real time.

Figure 6. Time components.

Figure 6 describes the time components between each waypoint. The total time is the time it takes for the UAV to complete one waypoint. This includes fly time and download time. The download time of an image is about 8 seconds, resulting in a time frame of 16 seconds for the stitching and tiling chain until another image starts to download.

Figure 7. Computing time versus thread count. Tested on an iPad Air simulator on a Core i7 MacBook Pro (Early 2011).

The number of threads in the tile and stitch work pool is found to heavily impact the performance. Figure 7 shows the average time it takes to tile one image, depending on how many threads are working on it. The stitch method tested was the cropping one. Later tests were performed using 6 threads, since the performance gain using more threads is deemed negligible.

Figure 8 shows the time it takes for one photo to go through the entire chain of computing before it can be presented on the map. Our initial guess that the blending algorithm should
take considerable longer time than the cropping is shown here to be wrong. The main reason for this is due to heavy disk operations, reading and writing tile images. The actual computing time is negligible and within the margin of error.

<table>
<thead>
<tr>
<th>Stitching methods</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Cropping</td>
<td>Alpha blending</td>
<td></td>
</tr>
</tbody>
</table>

![Figure 8. Time in seconds for each stitch method, per photo.](image)

DISCUSSION

Result

We could have optimized the image download chain from the UAV, but due to the available SDK implementations, it was difficult. The UAV now stops after each image is taken instead of moving to the next waypoint while downloading an image simultaneously. This does not affect any visual aspects of the tests, just the overall time it takes to complete a mission. However, if the time between waypoints is decreased, the available time for each image processing step is also smaller.

Maintaining and reporting height of the UAV is a major concern for accuracy of the images. If the measured and calculated height of the image does not match the real value, the image will not be correctly georeferenced, it will either appear too small or too large. We found that the barometer used in the UAV is good at measuring relative altitude differences, but not absolute altitudes. A possible solution for this is to attach better sensors to the UAV, such as laser range finders or LiDARs. Additional sensors draw more current, and adds more weight to an already limited platform, which probably reduces flight time drastically. As a comparison, without any additional weight added to the UAV, a flight time of 17 minutes is achieved. With the DJI Guidance platform attached, a sensor array with five depth cameras and distance sensors, a flight time of 11 minutes is achieved.

To calibrate the altitude measurements reported by the UAV, a small test using a handheld laser range finder was performed. The test was performed by letting the UAV rise to a desired altitude, aim the range finder and measure the range to the UAV. The height the rangefinder was held at was also taken into account. The results showed that the UAV flew at an altitude of 95 meters, while reporting 100 meters.

Heading measurements is also vital for good image matching methods. If a photo is rotated too much, it will be clearly visible. We did not have time to account for heading differences, as this information was not written in the Exif header of the photos taken by the UAV. This could probably be solved by saving additional data at the time an image is taken, using the onboard SDK.

Due to the relative small overlap (around 60%) and altitude (100m) used to test these methods, large trees at the edges of each photo are seen from the side, and it appears as they are leaning out of the image. This is a consideration to either get better coverage by using smaller overlap, or more accurate orthophotos by using larger overlap. If the goal is to get an overview of the forest, it might be acceptable to get some distortion, while covering a larger area.

The Zenmuse X3 camera have a very small lens distortion (0.90%). This means that in most cases there is no need to account for the distortion. However, if an application needs a more accurate representation of the image, or if the distortion is higher, the image could be warped to account for the distortion. This was not done in this paper due to the low distortion.

Camera calibration

The method used for calibrating the camera worked really well. However, the problems with altitude measurements and reporting accurate altitude made the calibration efforts obsolete. Since the UAV flew at different heights during different missions, the camera parameters or the altitude offset needed to be adjusted every time.

Cropping

The crop stitching method is the easiest of the three mentioned methods to implement. However, the visual defects of cropping give an inferior result. The seams between images are very significant, as can be seen in figure 4. The cropping method does not take into consideration if there are any color or small rotational differences in the photos. It is easy to determine where each photo is placed on the map, making the visual result resemble a grid-like map.

Alpha blending

The visual results of alpha blending are more promising than cropping. When two overlapping photos have a small rotational offset, they blend together almost perfectly and the seam is hard to find. If there are larger offsets in the photos, the blending method will not compensate for it and only blend them together. Color differences in photos will also make the seam visible. If there are significant color differences in the photos, the blending method will get a grid-like appearance just as cropping.

Pixel matching

The first tests performed was to pixel match two full scale photos at a time. When the iPad received two photos from the UAV the pixel match stitching process could begin. The result from the two photos would then be stitched together with the next photo that was downloaded from the UAV.
This approach did not work because the built-in stitching class was not able to stitch the result from two photos together with a third photo.

The next approach was to stitch all photos together at once after the UAV took the last photo. This approach had several drawbacks. The first one we found was that the iPad could only process ten full scale photos at a time. More than ten photos resulted in the iPad running out of memory and the application crashed. The next drawback is that OpenCV does not take any georeferencing information into account while pixel matching photos. Not taking the georeferencing into account results in photos being warped, rotated and stretched and loses their original scale. Objects in photos can therefore get a strange shape. A third drawback is that the Exif information the photos contained while being sent to the stitching class is lost when the resulting image comes back. This means that our tiling methods does not have a clue on how to tile this resulting image. The last drawback was that the feature finding algorithms had in some cases difficulties finding and matching features in the photos. We believe this is due to that, in photos only containing trees, it is hard to find the exact same tree in two photos. The feature finding algorithm used tries to find contrast edges in the photos, and it is mostly soft color transitions in a forest. In photos containing an open flat field, it is hard to find any features at all. There needs to be a good portion of both open area and clusters of trees in a photo to get a perfect pixel match result.

The last approach was to stitch only the overlapping tiles. This approach was investigated due to OpenCV not taking the georeferencing information into account. Only stitching the overlapping tiles would partially solve the loss of georeferencing, because tiles are already placed in the right location when they are created. When a tile is created it is given a name representing its location on earth. When two tiles with the same name are found, these two are the ones to do the pixel matching on. However, this approach did not work as expected. The feature finding algorithm used in OpenCV had a hard time finding features in some tiles and also finding the same features in two tiles. This resulted in most of the tile pixel matching operations failing.

We investigated the overlapping tiles further and found that in most cases two overlapping tiles did not entirely align at the same location as they theoretically should. This problem is a result from inaccuracy of coordinates and altitude saved in the photos, as well as camera calibration errors. This explains why OpenCV failed with performing pixel matching on the tiles.

We believe that with a more accurate location measurements and information saved in the photos taken, pixel matching only tiles would be a possible approach.

Point cloud
Some preliminary tests were performed to achieve point clouds using OpenCV. Due to time limitations for this research no real results were found. We found however, that as Hugenholtz et al. [10] describes it, it is difficult to find correct features in aerial photographs with heavy vegetation. This is probably better suited for using LiDARs, since LiDARs return reflections from both ground and vegetation.

Method
The way photos are processed could have been done in a more efficient way. The current algorithm uses about 30% of a photo when flying with an overlap of 60%. It is a waste of time and resources to download data from the UAV that is not going to be used. This could perhaps be optimized by investigating if the onboard processor on the UAV could do some preprocessing of photos before sending them to the iPad. Only downloading the useful part of a photo would both save time and resources. We did not investigate the possibilities of performing any additional computing on the UAV, since it is only available on the DJI Matrice 100 UAV, not on the more commercial focused DJI Phantom 3 and 4. These products cost a lot less, and might be better suited for this type of application.

Another way of saving computing time and resources is to use a camera specifically designed for taking aerial photographs. This type of camera has a narrower FOV, resulting in objects at the edges of a photograph not appearing to lean out from the center as much. A photograph taken with this type of camera might not need to be preprocessed at the same extent, before it is tiled and inserted into the map application. The overlap of photographs could also be decreased since less area of the photo is wasted.

A minor problem occurred when we discovered small rotational differences in our photos. We found that our photos did not include any information about heading. Our solution to this problem was to tell the UAV to fly with the front of the aircraft always pointing at a true north direction. The gimbal would point in the same direction as the UAV to reduce rotational differences. This was not optimal due to e.g. wind that could slightly rotate the aircraft. When a photo got slightly rotated, we did not know how much we should compensate for, to align the photo in the true north direction. A solution to this might have been to save the heading information separately at the same time a photo is taken. This information could then be accessed when the photo is processed. The image processing chain could then rotate the photo according to the information given, before the tiling processes begins.

NASA has a large toolkit for managing aerial photographs, this could perhaps been used in this application. The toolkit includes methods for stitching, and creating DEMs, however, due to the scope of the toolkit we did not have time to try and compile it for the iPad.

An own implementation of pixel matching would perhaps have given better results in our case. With additional time for this investigation that might have been possible. With an own pixel matching implementation the problems using OpenCV could be avoided. The coordinates in photos could be used
as anchor points to limit which areas to perform feature detections on. The uncontrolled warping and stretching of photos while using OpenCV could then be avoided.

**Source criticism**

References were chosen with care, to mainly use well known publishers, such as IEEE and Springer. However, due to not finding any relevant information published, some web links needed to be included [4] [5] [12].

**Wider context**

This application is perhaps not only suited for the forest industry. It could be used whenever a quick aerial overview of a smaller area is needed. The ease of use somewhat sacrifices positional and stitching accuracy, but for applications where that is not the most important aspects, this is a viable approach.

This application could in some sense contribute to a better environment. Whenever people and companies require an up to date map over a small area, this application could be used. Instead of using fuel consuming airplanes or helicopters, a small battery powered UAV might be better suited.

**CONCLUSIONS**

The crop stitching method is not an alternative to use as a stitching method, due to clearly visible seams. The alpha blending method might however be a better alternative, since performance wise, they are very similar. The visual appearance for alpha blending is better than cropping and hides the seam between photos. If there are big rotational differences in the photos, the seam will be visible even with alpha blending. The best visual solution is the pixel match method, although pixel matching is much more difficult to implement and get georeferenced orthophotos from. However, due to the pixel matching algorithm is not always working, no real results could be measured. For real time stitching, this might not be the best solution. Using OpenCV for pixel matching full scale aerial photographs is also not optimal due to memory constraints on the iPad. Pixel matching only overlapping tiles might be a possible solution, with more accurate coordinate information in the photos.

**References**


