



INNOVATIVE PROJECT REPORT (EC438)

A NOVEL FRAMEWORK FOR INTEGRATED SAR-GPS SYSTEM

ANIRUDH NAKRA
2K17/EC/22

ABHIJEET VATS
2K17/EC/03

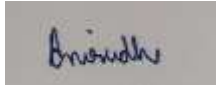
SUBMITTED TO: PROF AVINASH RATRE

A Novel Framework for an Integrated GPS-SAR system

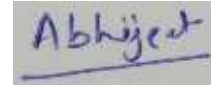
An Innovative Project for EC438 presented

by

Anirudh Nakra 2K17/EC/22



Abhijeet Vats 2K17/EC/03



Supervisor: Prof Avinash Ratre

Submitted to the Undergraduate School of
Delhi Technological University in partial fulfillment
of the requirements for the degree of

BACHELOR OF TECHNOLOGY

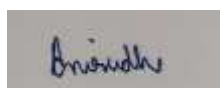
May 2021



Department of Electronics and Communication Engineering,
Delhi Technological University, New Delhi, India

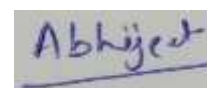
Declaration

We, Anirudh Nakra and Abhijeet Vats, hereby declare that the report attached below has not been plagiarized in anyway. The work has been implemented and reported on our own with no unreferred material. I understand that copying a classmate or peer's work is a form of plagiarism and submitting similar work constitutes plagiarism. I have not and will not allow anybody to copy my work in the fore coming time and pass it off as their own material.



Anirudh Nakra

2K17/EC/22



Abhijeet Vats

2K17/EC/03

Group Contribution:

Anirudh Nakra:

Report making: Synthetic Aperture Radar, GPS, RDA and Back Projection

Programming and Conceptualization: C/A Code, SAR RDA Back Projection

Abhijeet Vats:

Report making: Extended Kalman Filter, System Integration, Introduction

Programming and Conceptualization: GPS Mapping, SAR Interferometry

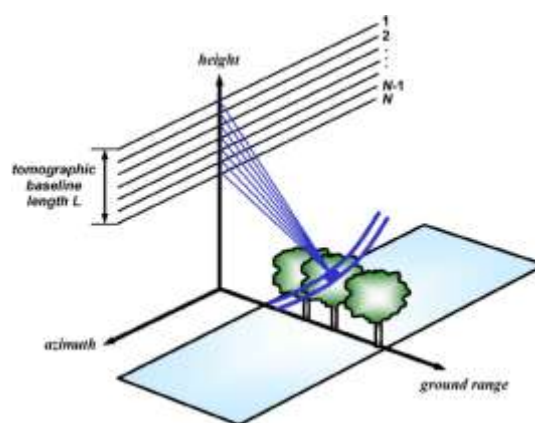
Contents

Introduction.....	3
Theory.....	5
Implementation	13
Results	15
Conclusion.....	24
References	25
Appendix.....	27

Introduction

Synthetic aperture radar is a remote sensing technique. It gathers information by creating power and then recording the power reflected back by the surface. It can detect different physical features of the surface, such as structure, temperature, and wetness. SAR, is a new means of generating images that involves actively illuminating the ground rather than relying on sunlight like optical imaging. The distinctions bring obstacles, but they also provide opportunities to develop new skills. SAR has a number of significant advantages, one of which is as follows: Even the greatest optical camera on an aircraft or a satellite is less useful at night and ineffective when clouds or smoke are present. SAR can see through clouds and smoke and record photographs at night. It is an all-weather, 24-hour technology.

Data is collected using optical sensors in various areas of the electromagnetic spectrum by means of foreshortening, shadowing and layover. These will be discussed in detail in the next chapter. Wavelength, look angle and polarization of the EM wave are also taken into account when we take SAR images. By increasing the frequency, different levels of penetrations can be achieved. For e.g, by using EM waves in the L band, we can determine the thickness of the ice layer upto certain extent. SAR has emerged as a powerful remote sensing method, with several applications in the study of various geographical locations such as the cryosphere, wetlands, and so on. It's become critical to use remote sensing to minimize environmental damage to icebergs, wetlands, and oil spills, all of which can harm eco-sensitive areas.



The Global Positioning System, or GPS, is a remote sensing system that serves as the backbone. This navigation system supplies GPS receivers with spatiotemporal data over a large geographical area. GPS now uses the L band for data transmission and has progressed to the point where it can monitor people to within a few centimeters. A network of four satellites is used by the GPS system.

GPS applications can be used in a variety of fields. It is utilized not just in automobile navigation, but also in astronomy to map newly discovered planets, sports to analyze player statistics, and even biology to tabulate animal behavior and decipher diverse patterns.

Some specific applications can also be utilized by GPS which can range from important situations like in emergency response, disaster management to entertainment activities such as in mobile games like Pokémon go. It finds applications in health and fitness. GPS can track activities of a person, if he or she is walking, running etc. In transportation, we can easily track our items, if they have reached their destinations or not. In supply chain management, it can create awareness and enthusiasm in customers about the delivery. Here, we'll focus on using GPS to improve the SAR imaging system so that it may be used for high-accuracy targeting and navigation in the future.

Our research includes a lot of Kalman filtering. As previously said, GPS technologies are critical for both daily and commercial uses. However, for the GPS to function properly, it must be in touch with at least four satellites at the same time. When the device is in an isolated place or travelling through places with a loss of connectivity, such as a tunnel, the situation becomes more challenging. Kalman filtering can be a good solution here because it delivers an accurate estimate of the device's location based on sensor data. When using a dynamic GPS-SAR integration technique, Kalman filtering should always be taken into account.

After focusing on individual components, we researched on integrating the system using SAR, GPS and Kalman filtering and created a robust framework which is discussed in detail in the implementation part of the report.

Theory

I. Extended Kalman

Extended kalman filter is extensively used in estimation theory. It is an extended version of normal kalman filter as the normal kalman filter is used only for linear models, but when we consider real world scenarios, most of the systems are nonlinear in nature. Therefore, an extended kalman filter becomes important. Extended kalman filter linearizes around the mean and covariance of the system considered. For nonlinear systems, it is considered the best option because of its low computational cost when we compare it to other non linear filters like particle filters. It is extensively used to solve real time problems such as in the navigation system in our case.

Let's consider a model for realization where state transition is occurring

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) + \mathbf{w}_{k-1}$$

$$z_k = \mathbf{h}(\mathbf{x}_k) + \nu_k$$

Here

\mathbf{f} = function of preceding stages

\mathbf{x}_{k-1} = preceding stage

\mathbf{u}_{k-1} = command input

\mathbf{w}_{k-1}, ν_k = gaussian white noise for process model and measurement model

\mathbf{h} = measurement function

z_k = measurement

Here, we consider that covariance is \mathbf{Q} and \mathbf{R} for process model and measurement model.

The algorithm requires us to calculate the jacobian matrix first and then find the partial derivative of the vector.

$$\mathbf{F}_{k-1} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1}}$$

$$\mathbf{H}_k = \left. \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_k^-}$$

After this step, the model is linearized with respect to the present approximate.

Measurement residual	$\hat{\mathbf{y}}_k = z_k - \mathbf{h}(\hat{\mathbf{x}}_k^-)$
Kalman gain	$\mathbf{K}_k = \mathbf{P}_k^- \mathbf{H}_k^T (\mathbf{R} + \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T)^{-1}$
Updated state estimate	$\hat{\mathbf{x}}_k^+ = \hat{\mathbf{x}}_k^- + \mathbf{K}_k \hat{\mathbf{y}}_k$
Updated error covariance	$\mathbf{P}_k^+ = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_k^-$

In the above functions the cap function is used to find the approximate of the variable and + and - sign indicate the posterior and prior values respectively. Due to all these computations which can be performed easily, extended Kalman filters can be used for the purpose of target tracking.

In target tracking, the approximation is done on a 3-dimensional space by calculating the position and velocity of the tracking object using different types of sensors.

Target State--

$$\mathbf{x} = [\mathbf{p}^T, \mathbf{v}^T]^T$$

where,

p= position vector

v= velocity vector

Then, the system is modeled as a near constant velocity model in discrete time space.

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \end{bmatrix} = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}) = \begin{bmatrix} \mathbf{p}_{k-1} + \mathbf{v}_{k-1} \Delta t \\ \mathbf{v}_{k-1} \end{bmatrix} + \mathbf{w}_{k-1}$$

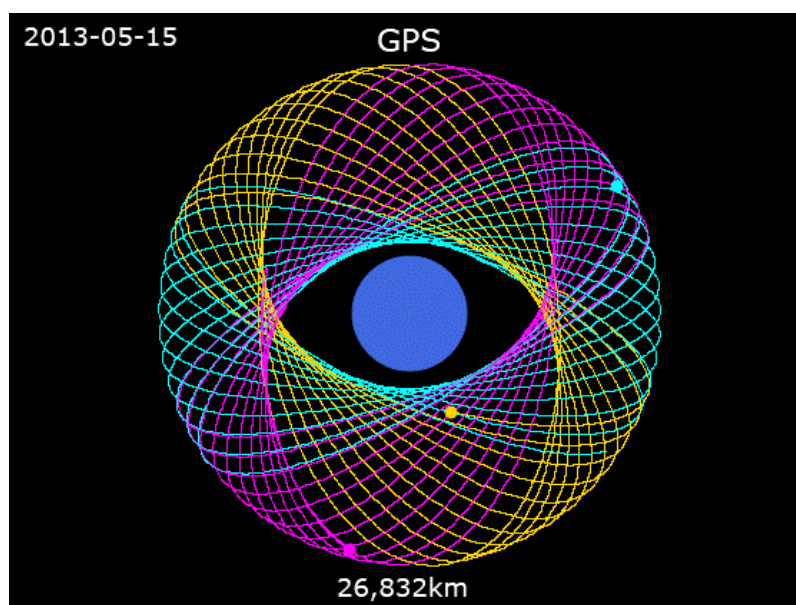
Then, standard deviations are calculated for the noise process in each of the directions namely x,y and z. The relationship is modelled between target state and sensor by setting the target state as variable. And by applying the concepts of extended Kalman filter in each direction, an estimation result is extracted from each axis and velocity of the object on that axis. Then, for each axis prior values are used for plotting the posterior of position and velocity for each axis. Standard error in both the cases are measured by taking into account the root mean squared error. Similar applications like terrain referenced navigation can also be performed using the same procedure as stated above.

The extended Kalman filter also has some disadvantages. If a slight error is there in modelling the process, the system easily diverges and gives the wrong output and error is large. So, we have to be precise in initializing the estimate as well as modelling the process correctly.

II. Global Positioning System (GPS)

At the height of the space race, scientists from the USA and Russia released satellites to get one step closer to manned flights. At the release of the first few Russian satellites, the scientists noticed that after calculating the frequency shifts due to the doppler effects, they could reliably calculate the position of the satellite itself. Since this calculation could locate the satellite, a similar engineered concept could in fact help them identify their own location as well. This led to a brief period of competition with the Russians and the Americans releasing their own proprietary navigation infrastructures. These systems were then widely deployed across military divisions with submarines being one of the main benefactors of this technology. In the late 70's, this technology evolved even further and the foundations of current gen Global Positioning Systems were set.

GPS was initially developed as a USA military specific technology but was later declassified for civilian usage. Current GPS systems employ a network of over 24 satellites that are hovering over Earth in a fixed satellite. These 24 satellites are positioned such that they occupy 6 differing planes, each having 4 satellites in it. The satellites are communicating 4 dimensional spatio temporal data: the three coordinates as well as their time instances. The actual transceiver on the earth that needs to locate itself thus needs information from 4 satellites in order to completely localise it in space and be able to track motion in time domain. These transceivers can actually be perfectly untransmitting i.e passive as well. Measuring the delay of the signal from the satellite to the receiver is done by the receiver itself while relaying this information to the GPS earth stations located in and around the position of the receiver which retransmit and refresh spatiotemporal data and make sure to do it iteratively multiple times every orbit period.



The GPS satellites usually operate in the L band with L1 and L2 bands being used for positioning data and CDMA codes. These frequencies are further subdivided into applicable situations. Civilian users can only make use of certain features within the L band with horizontal and vertical resolutions of around 100m to help with their navigation issues while military personnel can exploit PPS services to track users up to a horizontal and vertical resolution of 20 m, which is upto 5 times better than the civilian issue tech. Strong privacy and security codes are used to ensure that the location data remains private and is not misused or exploited by people having maleficent intentions. AS and SA techniques are used to ensure that jamming and accessing PNR codes is not possible while also not completely destroying the level of service provided to a civilian user. The SA technique ensures that some features are retained by common users.

As explained above, each receiver measures the delay or the TOA of the satellites servicing it. Every Satellite is responsible for generating a signal that would therefore be measured by the passive receiver. A PNR code is communicated between the satellite and the receiver as a priori information. Using this PNR code, delays in the epochs of the code between the satellite and the receiver can be measured to get a final TOA. The satellite also gives context to the epoch information with the specific geographical location included in the message transmitted to the receiver. Other than the obvious satellite and receiver infrastructure, there is an important part of the system that needs to be deployed on Earth. This system is known as the control network or the control segment. The control segment basically ensures that the system is working as intended. Using the control segments, the architecture is controlled via 4 major mechanisms. First of all the control segment checks and controls the relative positions of the satellites and optimises their constellation state and config. It keeps the base time clock to which the TOA, TOT and other major factors responsible for the correct functioning are maintained. It also controls the navigation information sent by the satellite by periodically refreshing and updating it. Other than this, it is also responsible for estimating the behaviour of remote satellite clocks and providing correct versions in case of fault by updating them. Many GPS earth stations are spread over vast continents. Notable stations are located at Hawaii, Australia, Ecuador and Bahrain. These are just 4 of over 11 stations that are currently in service.

The user segment of the GPS system just needs a passive receiver with the ability to tune into L band when required. There is also a need for a stable oscillator, usually a crystal oscillator to keep track of the time to calculate important variables such as TOA and TOT. The device should be able to process the data/ information provided as well and should have a receiver processor on board. In modern times, a display for providing the user critical information has also been shown to be critical for newer applications.

There is a need to discuss the mathematical background behind the seemingly complex technology. First and foremost, there is a need for explaining how the coarse acquisition code is generated for CDMA usage by the satellites. Well the C/A code is nothing but a long sequence of PRN which

uniquely identifies the satellite among 24-32 others. These C/A codes are made from Gold codes due to their excellent correlation and cross correlation properties.

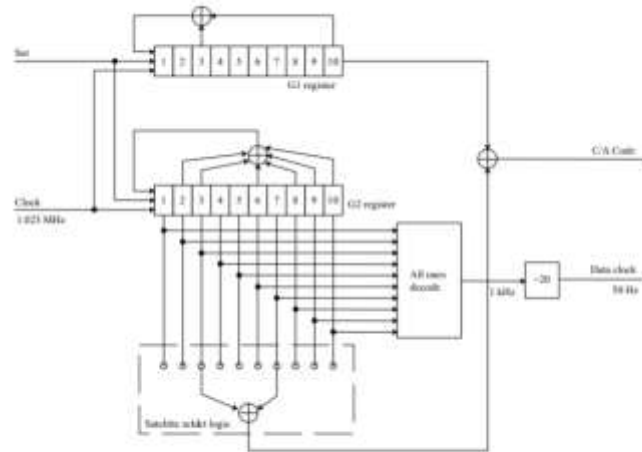


Figure 2.2: C/A code generator.

The PRN numbers can also be generated by Maximal Length Sequences when fed into a Linear Feedback Shift Register that creates such pseudorandom sequences. For a N size LFSR, the length of the MLS is 1 less than two to the power of N. For GPS the value of N is usually taken as 10 especially for the Navstar system which was the first iteration of a GPS architecture. The basic reason why we don't use MLS sequences is based on the advantages of the Gold sequence alone. Although the MLS sequences have good enough autocorrelation properties, their cross correlation is quite bad and thus to enhance their performance multiple MLS's are combined to generate the Gold sequence which brings in the best of both worlds, autocorrelation as well as cross correlation.

For N=10, we use two characteristic polynomials which are defined as follows:

$$P(1) = x^{10} + x^3 + 1$$

$$P(2) = x^{10} + x^9 + x^8 + x^6 + x^3 + x^2 + 1$$

These PRN sequences are irreducible and they decide the feedback of the LFSRs of the C/A code generator. The seed of the LFSRs is 1111111111 which is declared at the X1 epoch, which is defined by the P code. Finally, the different C/A codes are created using phase-shift methodology.

III. Synthetic Aperture Radar:

An optical visualization is the most straightforward way of perceiving satellite-based data. However, there is much more to landscapes and terrain detection than just visual stimuli from the optical rays. There are various different methods that can enable us in understanding the infrastructure and a specific location's identity such as radar information. A new and upcoming kind of this information is based on the bottom-up approach. The bottom approach signifies a revolutionary way in perceiving satellite information: it enables the self-illumination of the surface and thus receives information.

Synthetic aperture radar helps in this alternative methodology of generating high resolution images from a radar that is challenged in that aspect. Synthetic aperture radar or SAR is a novel remote sensing technique that has been exploited in recent times. SAR sensors produce their own stimuli and illuminate the ground while recording the amount of energy reflected and thus making important observations. Since the SAR illumination is heavily affected by the properties of the reflection media, the structure and general properties of terrain are easily representable.

The synthetic aperture radar needs a specific environment to work reliably. First of all, the radar needs to be mounted on a source that is moving in a straight line for effective remote sensing. Usually this is achieved by side mounting the SAR onto an airplane or even a remote satellite. However recently SAR work has been extrapolated to radars situated on aircraft carriers and navy boats with the "Ocean Master" being a notable example. SAR radars are coherent in nature and use the straight lined path to create a fine resolution imagery of the whole landscape. SAR essentially exploits the motion of its source or carrier to create an effective antenna which has a much higher resolution than conventional antennas. By transmitting short pulses and reading its echos, the SAR system can thus simulate a better antenna while not being hardware fitted with one. Since the source is in motion, there is recombination of different echos received at different points in the spatio-temporal space. These factors are the reasoning behind the name of the radar being "synthetic aperture" since it effectively utilises RF fundamentals to "simulate" a high resolution antenna using a conventional antenna present on board and side mounted to vehicular sources.

So why is there a need for SAR besides the obvious resolution improvement? Aren't there other alternatives for the acquisition of terrain data? As it turns out, the finer resolution is just a small part of the many advantages the system provides us with. First of all, the SAR antenna can be modified to be environment invariant. It can essentially negate height and weather issues via dynamic frequency allocation. Thus, SAR bypasses weather-based fluctuations and can be used in differing external environments very effectively due to self-illumination property.

SAR based systems have had numerous applications. From mapping of the Mariana Trench to other planets in our solar system, almost everything is within our scope of implementation. SAR antennas have been demonstrated to be working akin to a parallel set of radars. The system achieves this by

using the concept of time multiplexing. The system, in effect, creates a parallel network of antennas due to the abovementioned platform motion. For a given time of operation, the system stores the unique identifiers, usually amplitudes and phase angles for future analysis. Using this mapped data, we can easily identify and recreate the signal at a later point of time when constructing the landscape. For best detection properties, a sine wave is chosen as the transmitted signal of choice by the SAR system. The impulse has a good amplitude and phase to make the SNR at the transmitted sufficient for great echo reception properties.

The image data is conventionally divided into a matrix of pixels with each pixel acting as a basic element that forms the image. In the case of a SAR antenna, each pixel is characterised with a corresponding signal intensity level or amplitude as well as a phase representation. This intensity is as discussed above correlated to the properties of the reflecting surface i.e the terrain. Thus each pixel contains in its specific reflectance information. These parameters are also affected by atmospheric losses as well as multipath effects. Perhaps the best example for a research on SAR systems is the ongoing NISAR system being developed by NASA and ISRO in collaboration to create a SAR on a deployed satellite. The satellite will be operating on two frequency bands and will be the first remote sensing satellite deploying this technology. It will be studying the natural deformations and evolution of landforms on Earth over the course of its life cycle. The system will be deployed using the L band while also acting as a module in a bigger framework consisting of scientific GPS devices, payload, and recording hardware.

So how do we extract the information from SAR data? We can't understand the raw amplitude and phase values can we? Well, the procedure for the construction of visual information from pixel values as well as other phase information involves a lot of steps. The process of converting this data is quite complicated. The first major step that we take is to define the reflectances of different surfaces. These properties of the surface terrain are elaborated as below:

1. Smooth terrain is pitch dark due to echoes of the stimuli/EM wave not reaching the radar.
2. Rough surfaces in contrast to pt 1 are quite bright due to many EM waves being reflected and fed back to the SAR radar as echoes.
3. Mountainous regions are quite ambiguous. The part of the terrain facing the SAR platform is quite bright whereas the other part is dark due to no echoes reaching the SAR system.
4. Frequency selection is an important procedure because higher frequency will be reflected off smaller interferences such as miniscule rocks and other issues while longer wavelengths and lower frequencies will be more prone to scattering via large obstructions akin to huge rocks.
5. In the special case that the SAR system is near a huge water body where tides are a regular occurrence, the high tides can in fact reflect the EM waves to brighten the image considerably.

6. Radar imaging effects such as Layovers and Foreshortening are also an important part of the change in pixel intensity values. Layover effects arise from the different path propagations of the EM wave incident on the terrain. The layover effects usually arise when a high terrain such as a mountain is considered. This in turn creates the problem of the EM wave getting reflected off the top before it reaches the base of the same landform. This usually occurs in the swath regions of the SAR system. Foreshortening issues are also similar and these two combined issues are a direct result of short incident angles in the vicinity of the SAR line of motion. This leads to the non-illumination of the base features leading to the formation of shadows in the picture obtained.
7. With the recent increase in size and number of skyscrapers, there is another issue in radar image obtained through SAR. The highrise buildings that fall in line with the area just under the SAR system and are at right angles to the line of motion are disproportionately illuminated.
8. As explained above, foreshortening is a major tilt inducing effect in SAR systems and this causes elevated regions to be modeled as a tip instead of a more normal pictorial representation.

Tomography is another concept that is especially relevant to SAR systems. SAR enjoys applications in diverse fields ranging from weather telemetry to oceanography. However, these landscapes are often very complex. To evaluate these terrains, a better technique must be devised. Over the past few years, the field of SAR tomography has taken a huge leap in popularity. SAR interferometry has been improved to a point that additional information can be extracted from the echoes of the SAR illumination. These techniques allow the 3d construction of a terrain rather than a traditional top down image supplied by the SAR system. Multidimensional SAR processing is a tough and innovative technique. It is currently used to map out urban areas, forests and even whole glaciers. By including the temporal domain as well, it is very possible that in the future we will see the concept of evolving 3D models come to life. This 3D SAR technique is able to consolidate height backscattering information as well as target and interference based separation in a single pixel which allows us to analyse the target of interest in much detail. Nowadays, SARs such as InSAR are competitive with state of the art techniques such as LiDAR based systems due to their fine resolution as well as high spatial density. This enables researchers to move towards SAR data when available, especially in remote areas near the poles where drones can be easily deployed with side-mounted 3D SAR equipment.

Implementation

I. System Integration

We have integrated the concepts of SAR, Kalman filtering and GPS acquisition so that targeting with SAR sensors can be achieved with good accuracy. Good accuracy earlier required high resolution SAR, but with this approach, it is possible to increase the accuracy without putting in more cost to the system. By modifying the extended Kalman filter on an airborne SAR system, higher accuracy can be achieved. The GPS acquisition technique which is performed on MATLAB can be used for position and velocity of the aircraft. GPS provides positional accuracy, SAR provides good coverage area and the extended Kalman is used to determine the rate of change of position and velocity, the whole integrated system can achieve better accuracy with respect to actual target position.

The integration of SAR, GPS and the extended Kalman filter can be achieved by broadly two concepts. One is the loose integration and the other is the tight integration.

In loose integration, the raw measurements taken from different sensors are given to the extended kalman filter by grouping the information from two sensors giving them to a single extended kalman filter. This approach is extended in case there are multiple sensors. Then, the extended kalman filter processes this raw information. By this approach, raw measurements are processed multiple times giving more accurate processed measurements. There is a single output from this extended approach which points to a state estimate, having better accuracy.

The second method is tight integration. In this method, all the raw measurements taken from multiple sensors are given to a single extended kalman filter. In this approach, the raw measurements are processed only one time. Both these methods take use of Taylor series expansion to linearize all the nonlinear equations in the navigation system.

Important concepts which we used to integrate the system are sensor fusion, airborne mapping and multi target tracking. Multiple systems with image capturing ability can provide us with ample sensor data. By combining this data from those sensors, we improved the system in which only one sensor was used. For e.g, GPS data can provide errors in case the frequency is high. These errors can be generated from noise as well as various other atmospheric losses.

By combining the data, we can improve upon the data provided by this sensor and then we give the data to the extended Kalman filter for processing. As we know kalman filters are very sensitive to errors, this technique is used to reduce root mean square errors, when we implement it on python.

Airborne mapping includes integrating multiple sensors on the airborne platform, which can be used to process real time data. Multi target tracking is also important when we consider situations like defence where one can be attacked from multiple directions.

The methodology we followed includes GPS signal acquisition, followed by SAR implementation and finally gathering these raw data collected from these and sending it to the extended Kalman filter for processing and extracting information out of that using one of the tight or loose integration. The final step is done by analyzing the covariance matrix and monte Carlo simulation.

II. Range-Doppler Algorithm and Back projection

We implemented the RDA algorithm. Here, a range-dependent azimuth filter is required in the range-Doppler algorithm to minimize aliasing and the impacts of the platform's rapid attitude variation. In the image's pixel values are scaled by r^3 to approximately correct for range roll-off in the images. In addition, the images are presented in decibels with a grayscale range that is randomly selected. When producing the RDA image using the low altitude data, we find that employing a hyperbolic azimuth chirp rather than the standard RDA parabolic azimuth chirp improves the focus. For RDA azimuth compression at range r_a , a typical parabolic azimuth chirp $Az(m)$ can be written as an exponential function. The single look azimuth resolution (half the antenna length) is much lower than the range resolution, spatially square pixels can be created by creating an image with high azimuth resolution and incoherently averaging (multi-looking) pixels in the azimuth direction to reduce azimuth resolution and speckle noise. In the multi-looked image, the speckle level has decreased and the contrast has enhanced. An example of a georectified image created with the multi-looked azimuth compressed image and height information. This is in ground range, or cross-track distance, and shows a map perspective from above. To convert from slant range to ground range in this image, a simple nearest-neighbor range interpolation is utilised. The low altitude highlights the ground range image's range resolution fluctuations.

In our case, time domain back projection is done for reconstruction of the images from the SAR data. It is a simplistic method for creating visuals from raw synthetic aperture radar (SAR) data. SAR backprojection uses an azimuth matched filter for each pixel, accounting for range cell migration as well as aircraft motion. For producing images from SAR data, the time domain backprojection algorithm is an exact approach in which the radar cross-section is computed across a grid of pixels in ground range on the surface by the algorithm. The algorithm for a pixel can be approximated as--

$$A(x_0, y_0) = \sum_n S(n, d[n]) P(d[n]) \exp\{-j4\pi(d[n]/\lambda - d^2[n]K_r/c^2)\}$$

Where

$A(x_0, y_0)$ is complex pixel value of the SAR image.

λ = wavelength

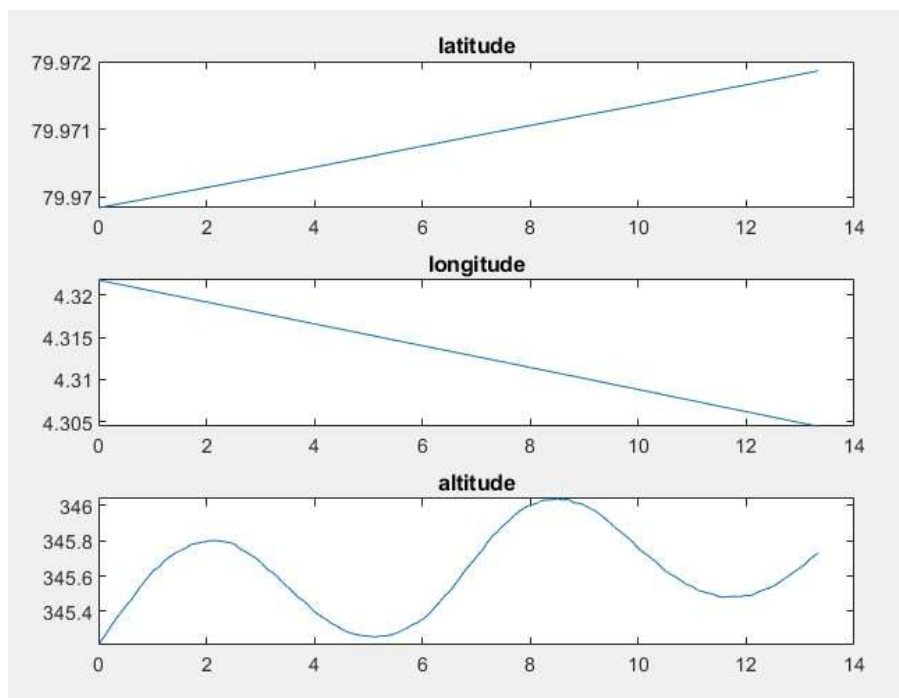
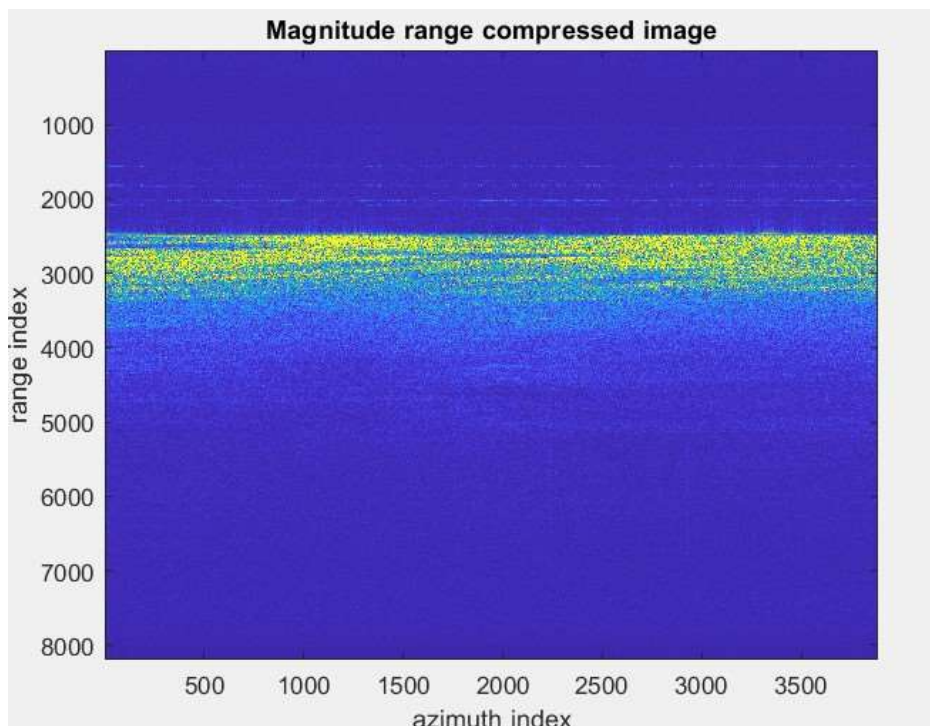
$d[n]$ = distance between the point and the antenna phase centre

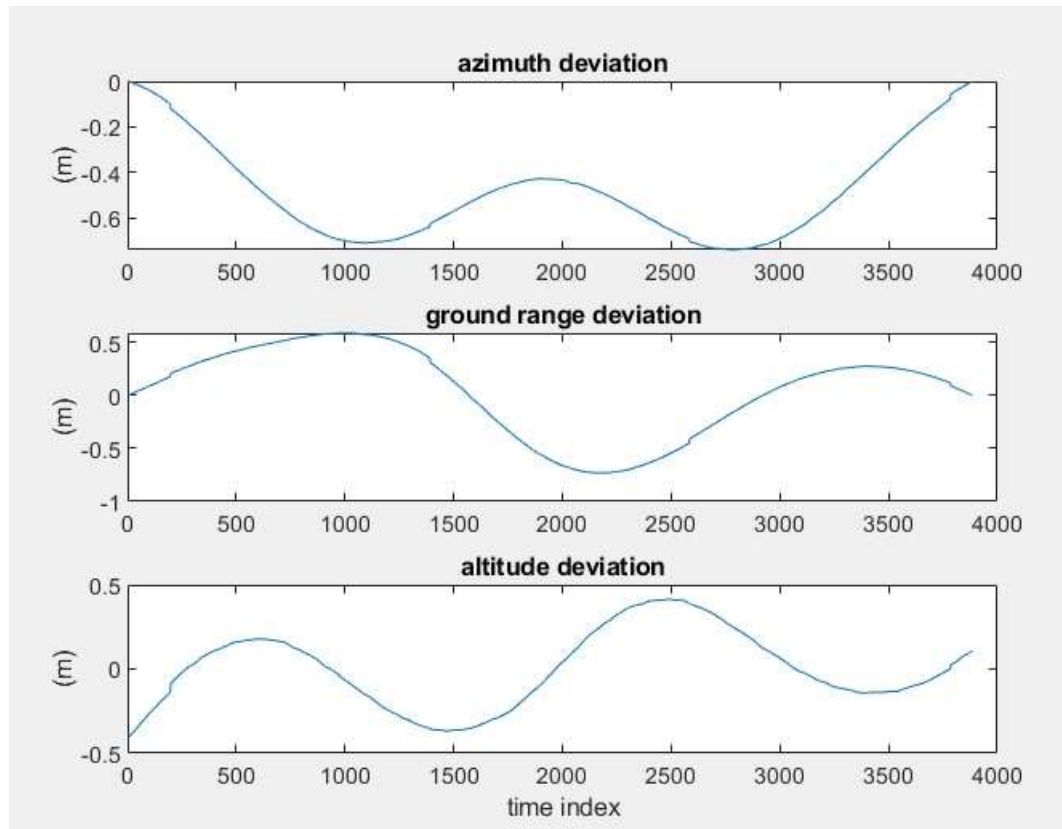
n = pulse number

Results

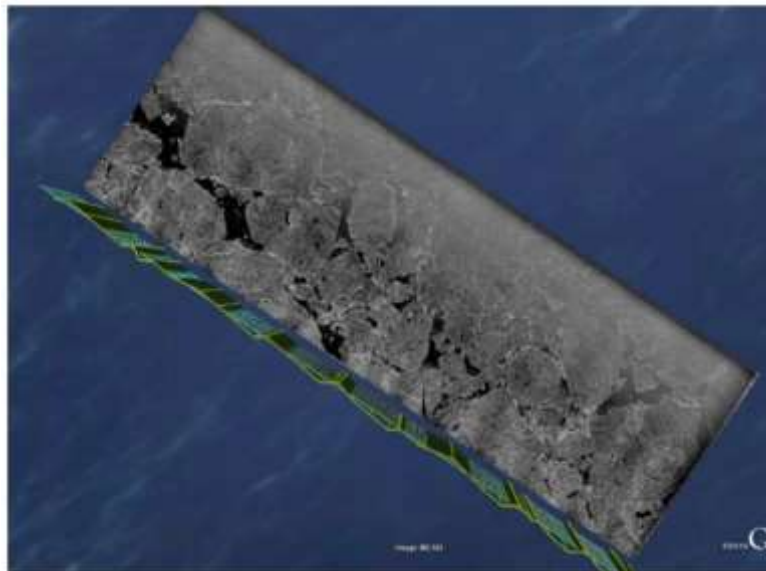


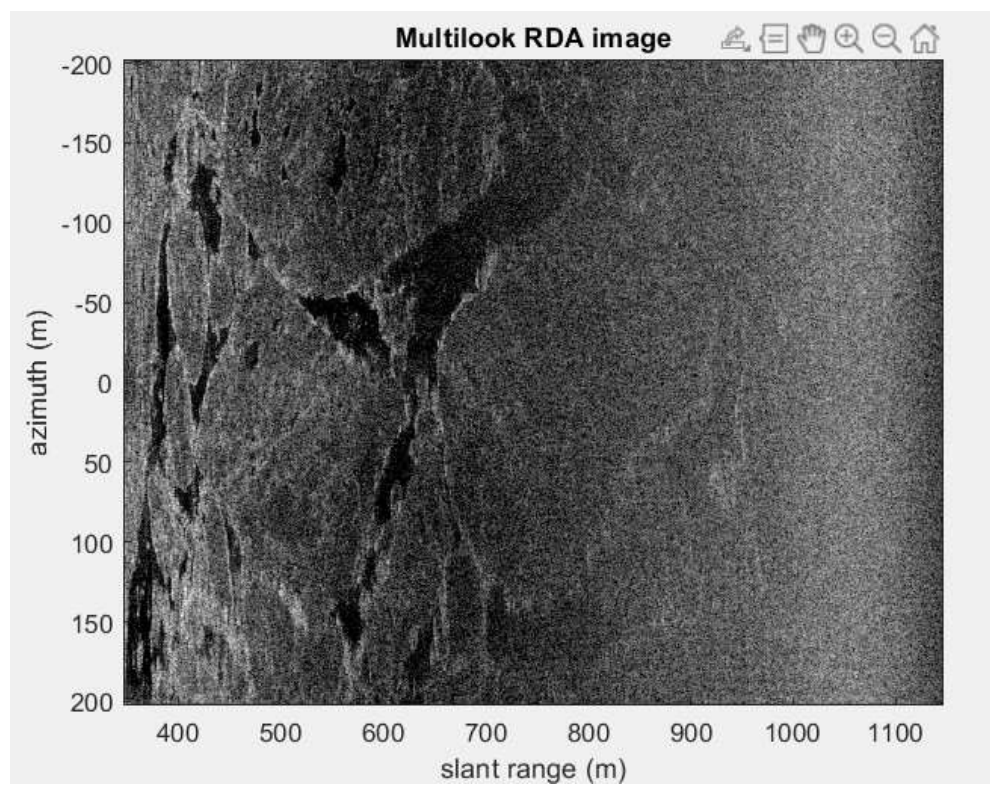
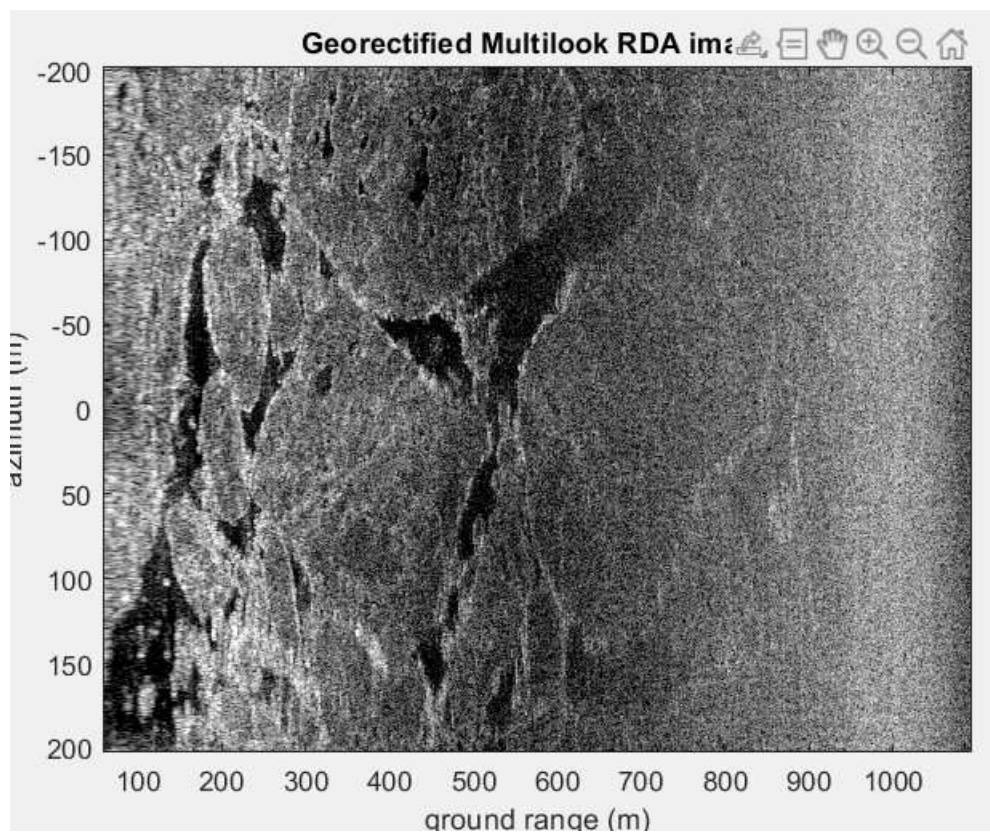
SAR RDA & Backpropagation:

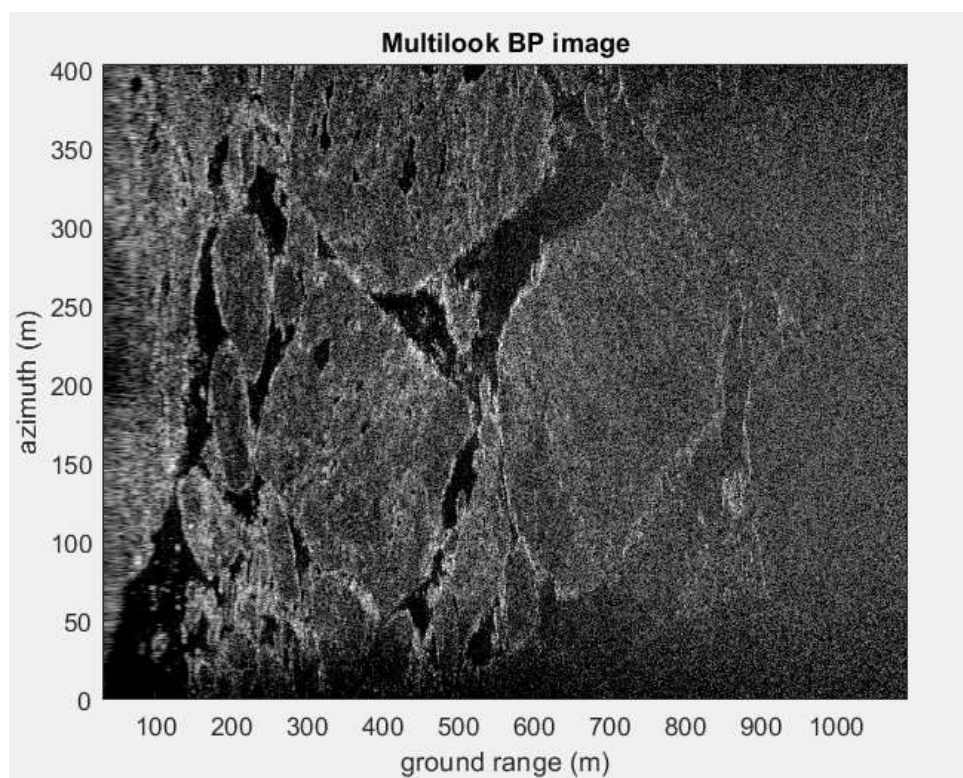
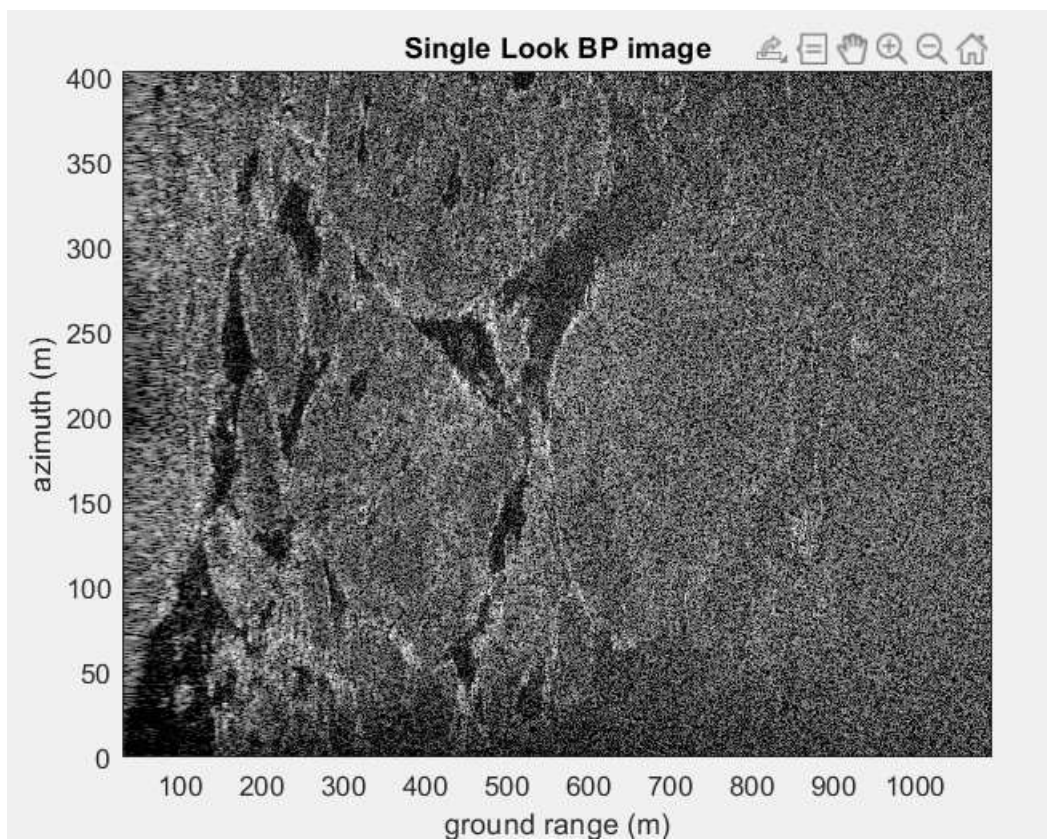


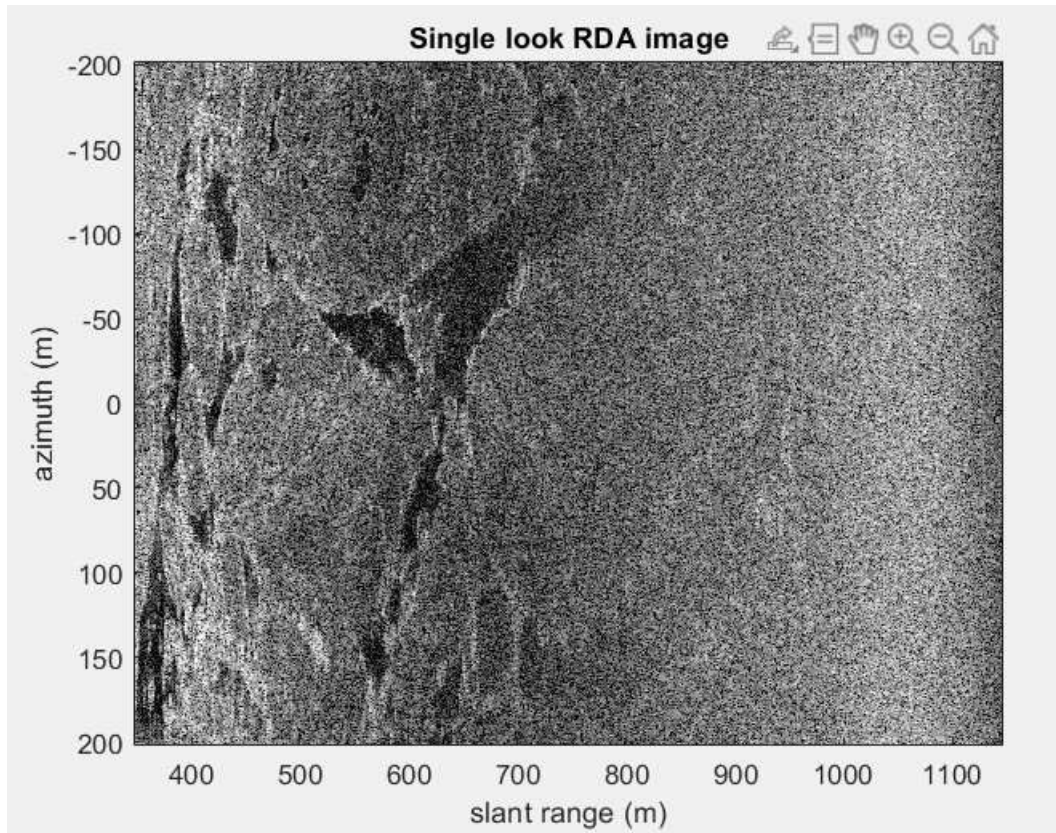


Ground Truth

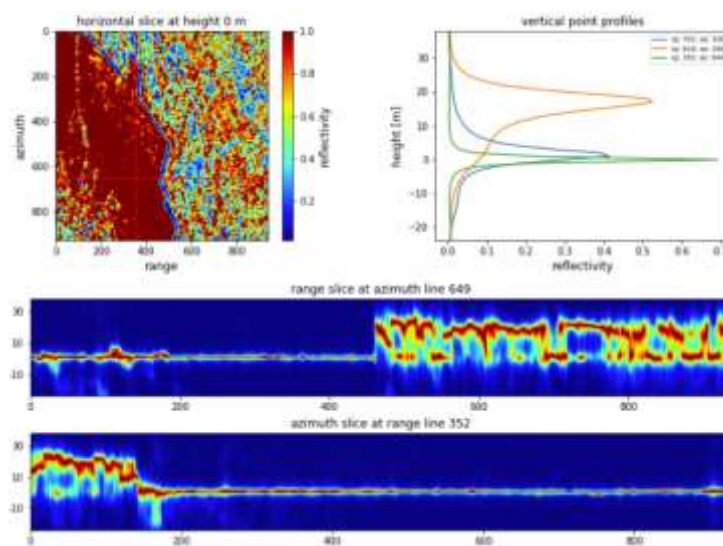


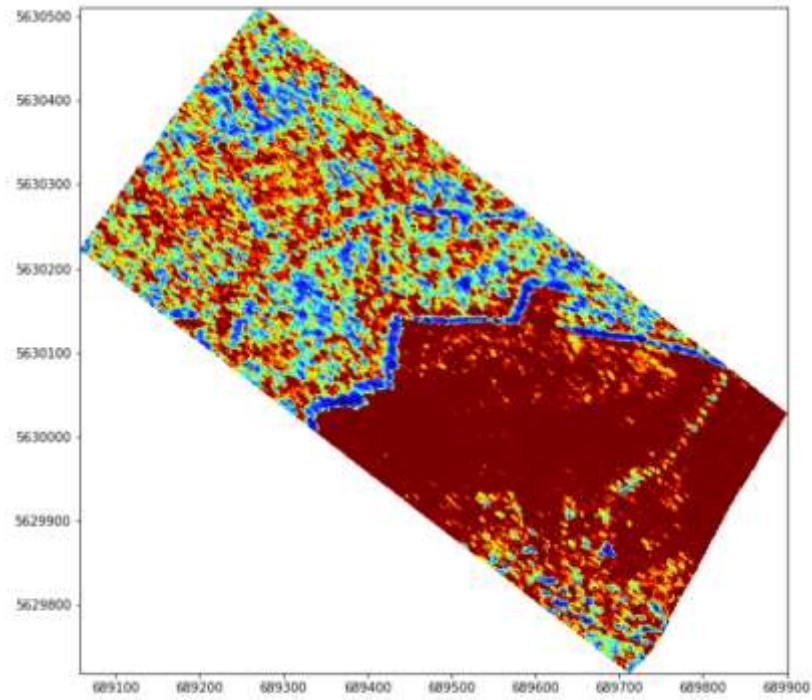






SAR Sliced Interferometry



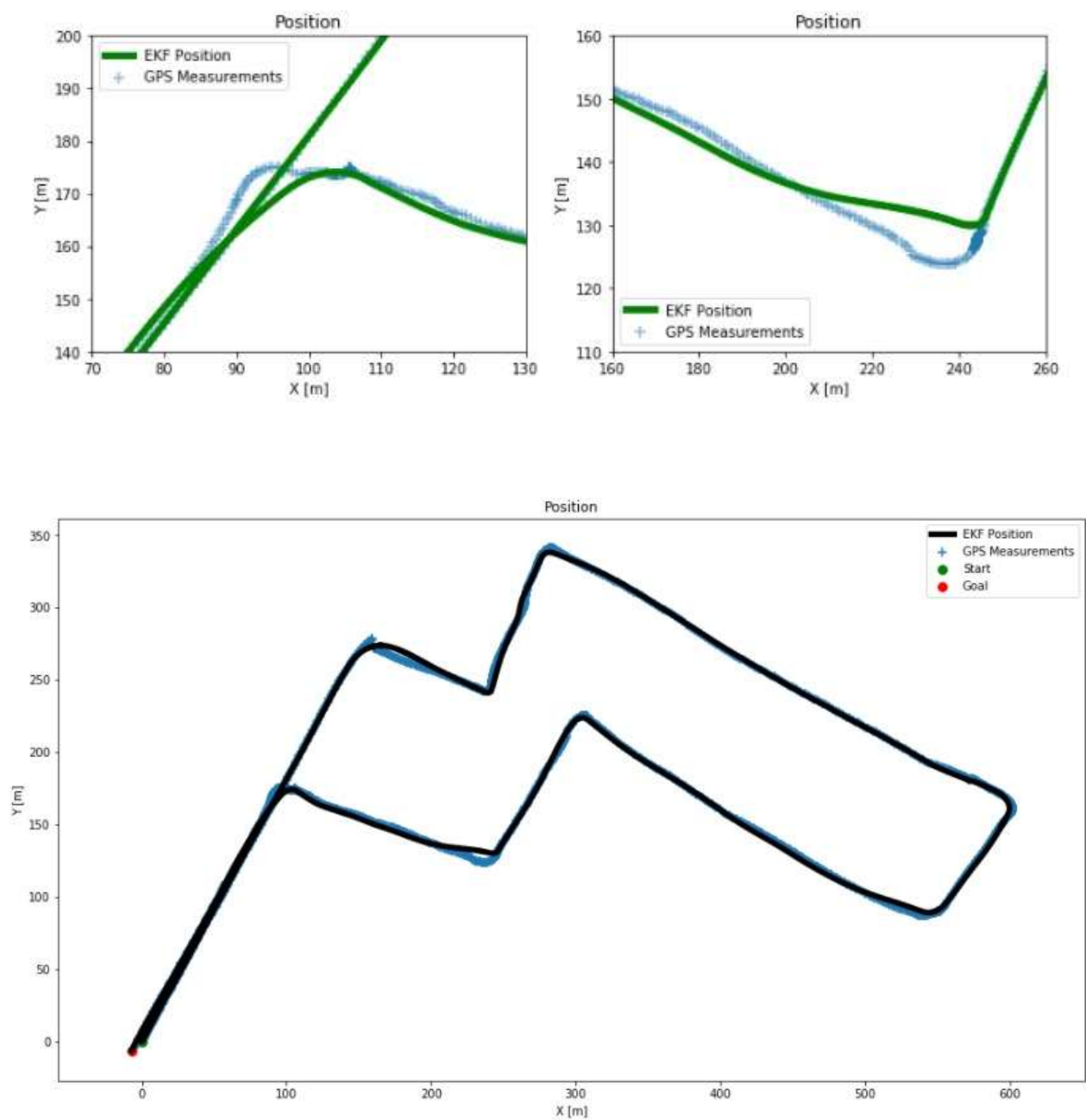


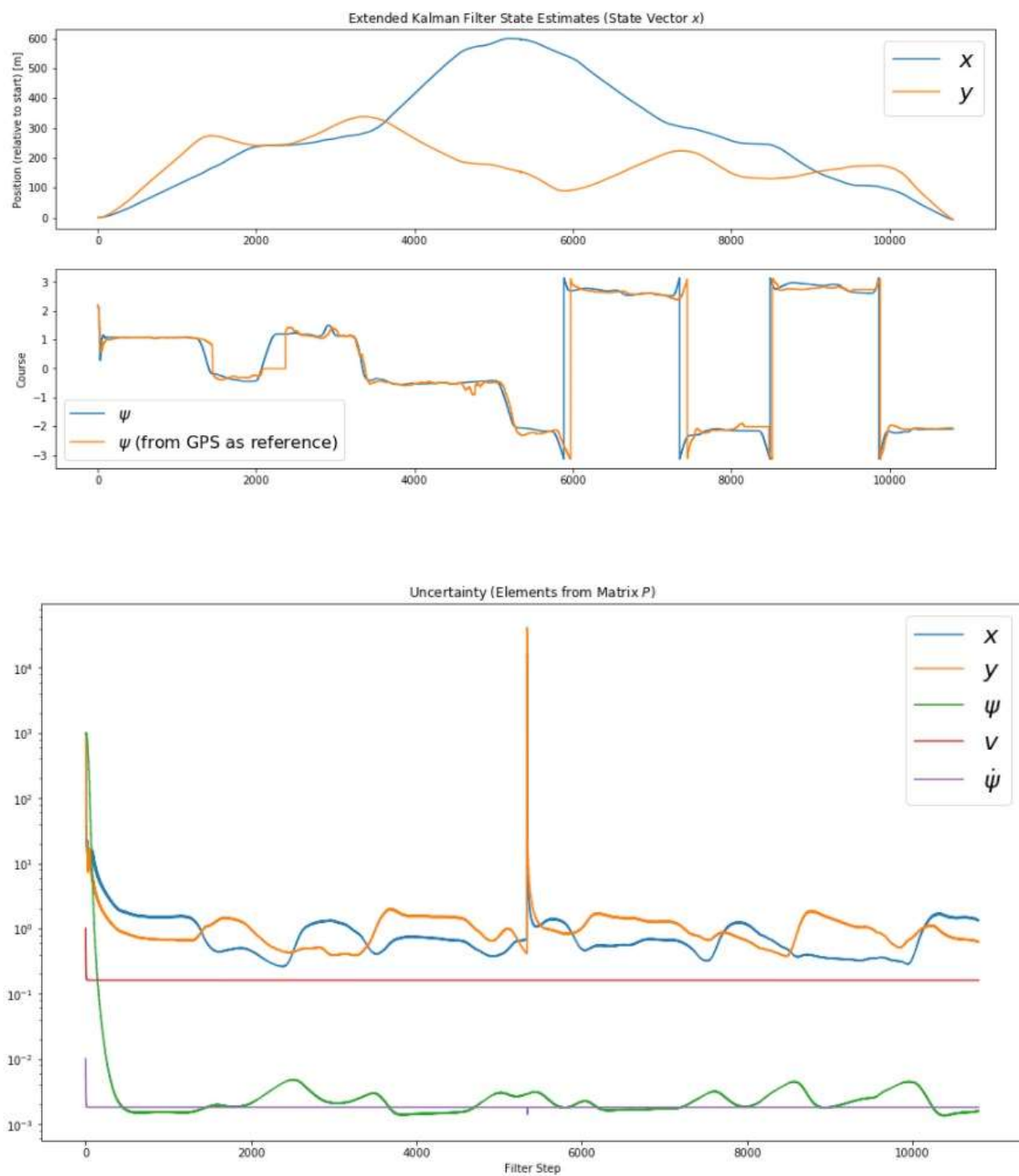
GPS Mapping:





Extended Kalman Filter based tracking:





Conclusion

We have successfully implemented the integration of SAR, GPS and extended Kalman filter in theory and by the results shown and minimized the error in location of target using this integrated approach. The simulations are successfully performed using TensorFlow in python and MATLAB. The integrated system shows better accuracy in terms of GPS targeting. In general, the accuracy of the position of the aircraft leads to accuracy in targeting the SAR imagery. With SAR alone, this level of accuracy could not have been achieved. Throughout the study, the GPS receiver models operated as expected.

With aircraft position accuracy, the SGPS performed slightly better. With location accuracies of 5ft and 3ft, respectively, the DGPS and CPGPS models functioned admirably. The filter has numerous bias states the integrated filter model has problems in this regard. We focused majorly on errors calculation of targets, but in future, the SAR, such models can be made that can further reduce states. While the results provided here were stable when all of the bias states were used, the model can still have demerits. It is worth noting that removing the unobservable states from the filter model should have no effect on overall performance.

References

- [1] Miller, Mikel M. Modified Multiple Model Adaptive Estimation (M3 AE) for Simultaneous Parameter and State Estimation. Ph.D. dissertation, AFIT/GE/ENG/98-02, School of Engineering, Air Force Institute of Technology, Wright-Patterson MB, OH, March 1998.
- [2] White, Nathan A. MMAE Detection of Interference/Jamming and Spoofing in a DGPS-Aided INS. MS thesis, AFIT/GE/ENG/96D-21, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1996.
- [3] Gray, Robert A. An Integrated GPS/INS/BARO and Radar Altimeter System for Aircraft Precision Approach Landings. MS thesis, AFIT/GE/ENG/94D-13, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1994.
- [4] Maybeck, Peter S. Stochastic Models, Estimation and Control, I. New York: Academic Press, Inc., 1982.
- [5] Maybeck, Peter S. Stochastic Models, Estimation and Control, II. New York: Academic Press, Inc., 1982.
- [6] Musick, Stanton H. and Neil Carlson. User's Manual for a Multimode Simulation for Optimal Filter Evaluation (MSOFE). Air Force Avionics Laboratory, Wright-Patterson MB, OH, April 1990. AFWAL-TR-88-1138.
- [7] Neilsen, Robert L. Development of a Performance Evaluation Tool (MMSOFE) for Detection of Failures with Multiple Model Adaptive Estimation (MMAE). MS thesis, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1993.
- [8] Negast, William J. Incorporation of Differential Global Positioning System Measurements Using an Extended Kalman Filter for Improved Reference System Performance. MS thesis, AFIT/GE/ENG/91D-41, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1991.
- [9] Lewantowicz, Zdzislaw H. and Randall N. Paschall. "Deep Integration of GPS, INS, SAR, and Other Sensor Information," Proceedings of ION National Technical Meeting 1998.
- [10] Sheldon, Stuart N. An Optimal Parameter Discretization Strategy for Multiple Model Adaptive Estimation and Control. Ph.D. dissertation, School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB, OH, December 1989.
- [11] Layne, Jeffrey R. Integrated Synthetic Aperture Radar and Navigation Systems for Targeting Applications. WL-TR-97-1185, Air Force Avionics Directorate, Wright-Patterson AFB, OH, September 1997.

- [12] Kaplan, Elliott D. Understanding GPS, Principles and Applications. Boston: Artech House, Inc., 1996.
- [13] Harger, Robert O. Synthetic Aperture Radar Systems, Theory and Design. New York: Academic Press Inc., 1970.
- [14] Omedo, Renato S. and Farnsworth, Kenneth A. "GPS and Radar Aided Inertial Navigation System for Missile System Applications," Proceedings of ION National Technical Meeting 1998, Long Beach, CA, January 21-23, 1998.
- [15] Abbott, Rich "Synthetic Aperture Radar Location Accuracy Assessment," Lockheed Martin Skunkworks Interdepartmental Communication, ASC/RAP, Wright-Patterson AFB, OH, 8 May 1995.
- [16] Grejner-Brzezinska, Dorota A. "Direct Platform Orientation with Tightly Integrated GPS/INS in Airborne Applications," Ohio State University Center for Mapping, Columbus, OH, 1998.

Appendix

➤ Course Acquisition Generation:

```

function g=cacode(sv,fs)
% function G=CACODE(SV,FS)
% Generates 1023 length C/A Codes for GPS PRNs 1-37
%
if nargin<2
    fs=1;
end

if (max(sv)>37) || (min(sv)<1) || (min(size(sv))~=1)
    error('sv must be a row or column vector with integers between 1 and 37\n')
end

if fs<1
    error('fs must be 1 or greater\n')
end

% force integers
testint=round(sv)-sv;
if testint ~= 0
    warning('non-integer value entered for sv, rounding to closest integer\n');
    sv = round(sv);
end

% table of C/A Code Tap Selection (sets delay for G2 generator)
tap=[2 6;
    3 7;
    4 8;
    5 9;
    1 9;
    2 10;
    1 8;
    2 9;
    3 10;
    2 3;
    3 4;
    5 6;
    6 7;
    7 8;
    8 9;
    9 10;
    1 4;
    2 5;
    3 6;
    4 7;
    5 8;
    6 9;
    1 3;

```



```

4 6;
5 7;
6 8;
7 9;
8 10;
1 6;
2 7;
3 8;
4 9
5 10
4 10
1 7
2 8
4 10];

% G1 LFSR:  $x^{10}+x^3+1$ 
s=[0 0 1 0 0 0 0 0 0 1];
n=length(s);
g1=ones(1,n);    %initialization vector for G1
L=2^n-1;

% G2j LFSR:  $x^{10}+x^9+x^8+x^6+x^3+x^2+1$ 
t=[0 1 1 0 0 1 0 1 1 1];
q=ones(1,n);    %initialization vector for G2

% generate C/A Code sequences:
tap_sel=tap(sv,:);
for inc=1:L
    g2(:,inc)=mod(sum(q(tap_sel),2),2);
    g(:,inc)=mod(g1(n)+g2(:,inc),2);
    g1=[mod(sum(g1.*s),2) g1(1:n-1)];
    q=[mod(sum(q.*t),2) q(1:n-1)];
end

%upsample to desired rate
if fs~=1
    %fractional upsampling with zero order hold
    index=0;
    for cnt = 1/fs:1/fs:L
        index=index+1;
        if ceil(cnt) > L    %traps a floating point error in index
            gfs(:,index)=g(:,L);
        else
            gfs(:,index)=g(:,ceil(cnt));
        end
    end
    g=gfs;
end
end

```

➤ GPS Mapping:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image, ImageDraw

class GPSVis(object):

    def __init__(self, data_path, map_path, points):

        self.data_path = data_path
        self.points = points
        self.map_path = map_path

        self.result_image = Image
        self.x_ticks = []
        self.y_ticks = []

    def plot_map(self, output='save', save_as='resultMap.png'):

        self.get_ticks()
        fig, axis1 = plt.subplots(figsize=(10, 10))
        axis1.imshow(self.result_image)
        axis1.set_xlabel('Longitude')
        axis1.set_ylabel('Latitude')
        axis1.set_xticklabels(self.x_ticks)
        axis1.set_yticklabels(self.y_ticks)
        axis1.grid()
        if output == 'save':
            plt.savefig(save_as)
        else:
            plt.show()
```

```

def create_image(self, color, width=2):

    data = pd.read_csv(self.data_path, names=['LATITUDE', 'LONGITUDE'], sep=',')

    self.result_image = Image.open(self.map_path, 'r')
    img_points = []
    gps_data = tuple(zip(data['LATITUDE'].values, data['LONGITUDE'].values))
    for d in gps_data:
        x1, y1 = self.scale_to_img(d, (self.result_image.size[0], self.result_image.size[1]))
        img_points.append((x1, y1))
    draw = ImageDraw.Draw(self.result_image)
    draw.line(img_points, fill=color, width=width)

def scale_to_img(self, lat_lon, h_w):

    old = (self.points[2], self.points[0])
    new = (0, h_w[1])
    y = ((lat_lon[0] - old[0]) * (new[1] - new[0]) / (old[1] - old[0])) + new[0]
    old = (self.points[1], self.points[3])
    new = (0, h_w[0])
    x = ((lat_lon[1] - old[0]) * (new[1] - new[0]) / (old[1] - old[0])) + new[0]

    return int(x), h_w[1] - int(y)

def get_ticks(self):

    self.x_ticks = map(
        lambda x: round(x, 4),
        np.linspace(self.points[1], self.points[3], num=7))
    y_ticks = map(
        lambda x: round(x, 4),
        np.linspace(self.points[2], self.points[0], num=8))
    |
    self.y_ticks = sorted(y_ticks, reverse=True)

from gps_class import GPSVis

vis = GPSVis(data_path='data.csv',
             map_path='map.png',
             points=(45.8357, 15.9645, 45.6806, 16.1557))

vis.create_image(color=(0, 0, 255), width=3)
vis.plot_map(output='save')

print()

```

➤ Sliced Interferometry SAR:

```
% matplotlib inline
from __future__ import print_function
import os
import numpy as np

import matplotlib
import matplotlib.pyplot as plt

from tomography_tutorial import \
    read_data, \
    topo_phase_removal, \
    calculate_covariance_matrix, \
    capon_beam_forming_inversion

from tomography_tutorial.ancillary import listfiles, normalize, geocode

from tomography_tutorial.plotting import DataViewer, Tomographyplot, GeoViewer

inpath = '/path/to/data'
outpath = '/path/to/out'

# define the boxcar smoothing dimension
multi_look = 10

# define the max height for the inversion
height = 70

if not os.path.exists(outpath):
    os.makedirs(outpath)

slc_list = listfiles(inpath, 'SLC_[0-9]_20151013_L_hv$')
phase_list = listfiles(inpath, 'Pha_[0-9]_20151013_L_hv$')
kz_list = listfiles(inpath, 'Kz_[0-9]_20151013_L_hv$')

# read SLC data
outname = os.path.join(outpath, 'SLC_stack')
slc_stack = read_data(slc_list, outname, overwrite=False)

# read files containing flat earth and topographical phase
outname = os.path.join(outpath, 'phase_stack')
phase_stack = read_data(phase_list, outname)

# read wavenumber files
outname = os.path.join(outpath, 'kz_stack')
kz_stack = read_data(kz_list, outname)

%matplotlib notebook
matplotlib.rcParams['figure.figsize'] = (13, 4)

view = DataViewer(slc_list, phase_list, kz_list, slc_stack, phase_stack, kz_stack)

IntSlider(value=1, continuous_update=False, description='image number', layout=Layout(align_it

outname = os.path.join(outpath, 'normalized_stack')
normalized_stack = topo_phase_removal(slc_stack, phase_stack, outname)
```

```

outname = os.path.join(outpath, 'cov_matrix_ml{}'.format(multi_look))
covariance_matrix = calculate_covariance_matrix(normalized_stack, outname, multi_look)

# capon beam forming inversion
outname = os.path.join(outpath, 'capon_bf_ml{}_h{}'.format(multi_look, height))
capon_bf = capon_beam_forming_inversion(covariance_matrix, kz_stack, outname, height)

# compute absolute values of the inversion result
capon_bf_abs = np.absolute(capon_bf)

# normalize the absolute beam forming inversion result per-pixel
caponnorm = np.apply_along_axis(normalize, 2, capon_bf)

%matplotlib notebook
matplotlib.rcParams['figure.figsize'] = (12, 8)
view = Tomographyplot(capon_bf_abs, caponnorm)

SHOW HIDDEN OUTPUT

# first we define the names of the LUT files, which are provided with this tutorial:
lut_range = os.path.join(inpath, 'lut_rg')
lut_azimuth = os.path.join(inpath, 'lut_az')

# then we define a name for the output dataset.
outname = os.path.join(outpath, 'caponnorm_ml{}_h{}_geo'.format(multi_look, height))

# The data in this tutorial is only a subset of a larger dataset which is linked to the LUTs.
# Thus, the function geocode needs to internally convert the pixel coordinates to those of the orig
# Here we define the radar pixel coordinates of our subset within the original dataset:
rg_min = 1610
rg_max = 2547
az_min = 17324
az_max = 18252

# now we can perform the actual geocoding:
geocode(data=caponnorm,
        lut_rg_name=lut_range, lut_az_name=lut_azimuth,
        outname=outname,
        range_min=rg_min, range_max=rg_max,
        azimuth_min=az_min, azimuth_max=az_max)

# Providing a different input dataset of the tutorial, e.g. the SLC data, to the function would worl

%matplotlib notebook
matplotlib.rcParams['figure.figsize'] = (10, 10)
# here you can provide a custom list of band indices so that we can scroll through the
# inversion height range instead of the raw band indices
view = GeoViewer(outname, band_indices=range(-height, height + 1))

SHOW HIDDEN OUTPUT

```


➤ SAR RDA & Back Projection:

```
% close all, clear all;
PLOT=1;          % flag to enable plotting of images
RUNBP = 1;       % flag to enable computation of backprojection
updateTime=15;  % time between backprojection processing status updates
aresScale = 1;  % az resolution scale for BP. values>1 reduce resolution
               % and computation time

%% code can optionally support matlab's use of GPU, set USEGPU to 0
% unless a GPU is available on your machine
USEGPU = 0;      % flag to enable use of GPU on capable platforms, zero
               % value disables use of GPU
matver = sscanf(version, '%d.%d.%d');
if matver(1) <= 7 && matver(2) < 12 && USEGPU == 1
    warning('This version of matlab does not support use of GPU')
    USEGPU = 0;
end

if USEGPU
    cgpuArray = @(x) single(gpuArray(x));
    cgather = @(x) gather(x);
else
    cgpuArray = @(x) x;
    cgather = @(x) x;
end

%% Set up microASAR SAR and processing parameters
fftsize = 4096*4; % size of fft to use for backprojection range compression
N = ceil(fftsize/2);

c0 = 299702547; % speed of light in the atmosphere (m/s)
actual_samples=1702; % number of samples/chirp
PRF = 307.292; % effective pulse repetition frequency (Hz)
azBW=0.1920; % 3dB antenna azimuth beamwidth (rad) (3dB to 3dB)
cabledelay = -1.2; % system and cable delay (m)
f_adc = 24485000; % adc sample rate (Hz)
bw = 170000000; % transmit bandwidth (Hz)
fc = 5.42876e9; % carrier frequency (Hz)
kr = 1.597249139246997e12; % chirp rate
delrad = c0*f_adc/(4*kr*pi); % meters per radian
delsamp = delrad*2*pi/actual_samples; % meters per real sample
delrsamp = delsamp*(actual_samples/fftsize); % meters per interpolated sample
max_range = c0*f_adc/(4*kr); % maximum range without aliasing
lambda=c0/fc; % carrier wave length
ares = lambda/(2*azBW)*aresScale; % azimuth resolution
rres = c0/(2*bw); % slant range resolution

%% load the BYU CASIE-09 sample data
load('flight9_9_sample.mat')
numpulses = size(dat,2);
```

```

if 0 % optional code to reduce data set size for machines with <4GB memory
    Ndec = 8; % factor by which data set is reduced
    numpulses = floor(numpulses/Ndec);
    dat = dat(:,1:numpulses);
    geom = geom(:,1:numpulses);
end

%% range compress the data using a range window and zero padding
% Note that the first 30 samples are set to zero. A Taylor window
% with NBar=4, SSL=-26 minimizes range sidelobes, but other windows
% can be used.
rc = fft(dat.*([zeros(30,1); taylorwin(actual_samples-30,4,-26)]*ones(1,numpulses)), %
fftsize,1);
% discard duplicated half of the spectrum
rc = [rc(1:N,:); zeros(1,size(rc,2))];

% plot the range compressed data
if PLOT
    figure(1), imagesc(abs(rc),[0 5e4]), title('Magnitude range compressed image')
    ylabel('range index'); xlabel('azimuth index');
    drawnow
end

%% plot visuals of the flight geometry
t = (geom(1,:)-geom(1,1))/PRF; % time axis
if PLOT
    figure(2),
    th(1) = subplot(3,1,1);
    subplot(3,1,1), plot(t,geom(2,:))
    title('latitude')
    th(2) = subplot(3,1,2);
    subplot(3,1,2), plot(t,geom(3,:))
    title('longitude')
    th(3) = subplot(3,1,3);
    subplot(3,1,3), plot(t,geom(4,:))
    linkaxes(th,'x')
    title('altitude')
    drawnow
end

%% project lat lon onto a local tangent plane (northing easting surface)
RefLat=mean(geom(2,:)); % select a reference latitude
Conv=1852.23 * 60.0; % latitude conversion factor
Lconv=Conv*cos(RefLat*pi/180); % longitude conversion factor
northing = geom(2,:)*Conv;
easting = geom(3,:)*Lconv;
z = geom(4,:)-cabledelay; % correct height by cabledelay;

```



```

if PLOT
    figure(1),
    hold on, plot(z/delrsamp), hold off
end

%% rotate and move the gps data to a local reference
rotAngle = -atan2((easting(end)-easting(1)), (northing(end)-northing(1)))+pi/2;
cang = cos(rotAngle);
sang = sin(rotAngle);
l_northing = (northing-northing(1));
l_easting = easting-easting(1);
r = cang*l_northing-sang*l_easting;
a = sang*l_northing+cang*l_easting;
%%
if PLOT
    figure(3)
    plot(a(1),r(1),'gs',r(end),a(end),'rs',l_northing(1),l_easting(1),'gx',l_northing(
end),l_easting(end),'rx'),
    figure(4)
    subplot(3,1,1), plot(a-a(end).*(t./t(end))), title('azimuth deviation'), ylabel(
('m'))
    subplot(3,1,2), plot(r), title('ground range deviation'), ylabel('(m)')
    subplot(3,1,3), plot(z-mean(z)), title('altitude deviation'), ylabel('(m)'),
xlabel('time index')
    drawnow
end

%% Backprojection Processing
if RUNBP
    disp('Beginning Backprojection processing')
    % set up grid for backprojection
    ground_range = sqrt(max_range.^2-mean(z).^2);
    y = cgpuArray([ares:ares:a(end)]-ares/2);
    x = cgpuArray([rres+30:rres:ground_range]-rres/2);
    z = cgpuArray(z);
    z2 = z.^2;
    % define image grid
    [X,Y] = meshgrid(x,y);
    % pre-compute constants in exponent
    betapic=4*pi*kr/c0.^2;
    lambda4pi=4*pi/lambda;
    % define output image and phase arrays
    img = cgpuArray(single(complex(zeros(size(X)),0)));
    pphase = cgpuArray(single(complex(zeros(size(X)),0)));

    % perform backprojection

    % use clock tick to measure CPU time

```

```

inittic = tic;
updr = inittic;
if PLOT
    figure(10)
end

%% backprojection loop
for k = 1:npulses
    ty = Y-a(k);
    tx = X+r(k);
    tx2 = tx.^2;
    D2 = (ty.^2+tx2+z2(k));
    D = sqrt(D2); % distance from antenna to each pixel
    % get the azimuth angle estimate
    mz = sqrt(tx2+z(k).^2).*sign(tx); % can approximate by using mean(z) to lower
computation
    az = atan2(ty,mz); % azimuth angle to each pixel
    ids = round(D./delrsamp);
    ids(ids>N) = N+1;
    pphase = exp(-1i*(betapic*D2-D*lambda4pi)); % calculate the expected phase
    % multiply and accumulate phase
    img = img+pphase.*reshape(rc(ids,k),size(ids,1),size(ids,2)).*(abs(az)<
(azBW/2)).*exp(-az.^2*30);

    % plot an updated image
    tdr = toc(updr);
    if tdr > updateTime
        fprintf(1,'%d of %d complete, %fs elapsed, %fs estimated\n',k,npulses,
toc(inittic),toc(inittic)*(npulses-k)/k)
        updr = tic;
        %
        if PLOT
            dimg = db(cgather(img));
            dimg(~isfinite(dimg)) = 0;
            rangescale = mean(dimg(1:find(cgather(y)-a(k)>0,1,'first'),:),1);
            rangescale = max(rangescale)./rangescale;
            rangescale(rangescale > 500) = 500;

            dimg = dimg.*(ones(length(y),1)*rangescale);
            imgMin = prctile1(dimg(dimg>0),10);
            imgMax = prctile1(dimg(dimg>0),99.99);
            set(0,'CurrentFigure',10); % this draws the update, but doesn't make
the figure come to the foreground
            imagesc(x,y,dimg,[imgMin imgMax]), %colorbar
            set(gca,'YDir','normal'), colormap('gray')
            drawnow
            title('Single Look BP image')
        end
    end
end

```

```

end

%%
if PLOT
    figure(10)
    imagesc(x,y,dimg,[imgMin imgMax]), colormap('gray')
    set(gca,'YDir','normal')
    title('Single Look BP image')
    xlabel('ground range (m)'); ylabel('azimuth (m)');
    drawnow
end
%%
img = cgather(img); % bring the gpuArray back to the cpu memory if needed

%% create multi-look backprojection image
NazLook=4;
mimg=zeros([floor(size(img,1)/NazLook) size(img,2)]);
for k=1:NazLook:size(img,1)-NazLook
    kk=(k-1)/NazLook+1;
    mimg(kk,:)=sum(abs(img(k:k+NazLook-1,:)).^2,1)/NazLook;
end

%% plot a multi look image

if PLOT
    dmimg = db(cgather(mimg));
    dmimg(~isfinite(dmimg)) = 0;
    rangescale = mean(dmimg,1);
    rangescale = max(rangescale)./rangescale;
    rangescale(rangescale > 500) = 500;
    dmimg = dmimg.*(ones(size(dmimg,1),1)*rangescale);
    mimgMin = prctile1(dmimg(dmimg>0),10);
    mimgMax = prctile1(dmimg(dmimg>0),99.99);
    figure(11)
    imagesc(x,y,dmimg,[mimgMin mimgMax]), colormap('gray')
    set(gca,'YDir','normal')
    title('Multilook BP image')
    xlabel('ground range (m)'); ylabel('azimuth (m)');
    drawnow
end
disp('Backprojection processing completed')
end

%% =====
%% RDA Processing
disp('Beginning RDA processing')
rdafftsize = 4096;

```

```

%% RDA processing needs evenly spaced pulses data. Since there are gaps
% we first need to fill in the missing pulses. What follows is a simple
% but effective interpolation method for the CASIE-09 SAR data.
pulsen = geom(1,:)-geom(1,1)+1;
dat1 = zeros(actual_samples,pulsen(end));
dat1(:,pulsen) = dat;
dpulsen=diff(pulsen)-1;
skips = pulsen(dpulsen~=0);
dpulsen = dpulsen(dpulsen~=0);
for k = 1:length(skips)
    dat1(:,skips(k)+1:skips(k)+dpulsen(k)) = dat1(:,skips(k)-dpulsen(k)+1:skips(k));
end
Naz = size(dat1,2);

%% interpolate the rest of the data
t1=(1:pulsen(end))./PRF;
a1 = interp1(t,cgather(a),t1,'spline');
r1 = interp1(t,cgather(r),t1,'spline');
z1 = interp1(t,cgather(z),t1,'spline')+cabledelay;

%% range compress the data
% Note that the first 30 samples are set to zero. A Taylor window
% with NBar=4, SSL=-226 minimizes range sidelobes, but other windows
% can be used.
rcl = fft(dat1.*([zeros(30,1); taylorwin(actual_samples-30,4,-26)]*ones(1,Naz)), ⚡
rdafftsize,1);
% discard duplicated half of FFT output
rcl = rcl(1:rdafftsize/2,:);

% compute the slant range per sample
rdadelrsamp = delsamp*(actual_samples/rdafftsize);

%% find the mean linear velocity of the aircraft
vel = a1(end)./t1(end);

ran = (0:rdafftsize/2-1)*rdadelrsamp+cabledelay;
azm = ([1:Naz]-Naz/2-1)*vel/PRF;
% meshgrid az and range vars
[AZ RA] = meshgrid(azm,ran);

disp('calculating azimuth chirp')
% conventional parabolic azimuth chirp
% M = exp(-j*(fc^2/c0*(AZ.^2./(2*RA))-2*fc*vel*AZ./(c0.^2)));
% hyperbolic azimuth chirp
tau_r=2*sqrt(RA.^2+AZ.^2)/c0;
M=exp(-li*(2*pi*fc*tau_r+kr*tau_r.^2));

% azimuth window (rect) based on antenna beamwidth

```

```

M(abs(AZ) > 0.5*RA*azBW) = 0.;
% simple tapered azimuth window
azwin=-600.0;
M=M.*exp(azwin*(AZ./RA).^2);

% fft shift to center frequency at origin
M=fftshift(M,2);

% convert azimuth chirp to frequency domain
M=fft(M,[],2);

% Take range compressed data to the range Doppler domain
A4=fft(rcl,[],2);

% Multiply by fft of azimuth chirp to do matched filtering
A4 = A4.*conj(M);

% Finish the process using ifft
A4 = ifft(A4,[],2);

% Apply an approximate range compensation for image display
rcomp=(ran').^3;
for k=1:size(A4,2)
    A4(:,k)=A4(:,k).*rcomp;
end

%% generate a georectified RDA image
grange = sqrt([(1:rres:sqrt((max_range-2)^2-mean(z1).^2)).^2 + mean(z1).^2]);
grange = grange(find(grange>mean(z1)+5,1,'first'):end);
gA4 = interp1([1:size(A4,1)]*rdadelrsamp,A4,grange);

% generate multilook image
NazLook=8;
C=zeros([size(A4,1) floor(size(A4,2)/NazLook)]);
gC=zeros([size(gA4,1) floor(size(gA4,2)/NazLook)]);
for k=1:NazLook:size(A4,2)-NazLook
    kk=(k-1)/NazLook+1;
    C(:,kk)=sum(abs(A4(:,k:k+NazLook-1)).^2,2)/NazLook;
    gC(:,kk)=sum(abs(gA4(:,k:k+NazLook-1)).^2,2)/NazLook;
end

imrange = ceil((mean(z1)+1)/rdadelrsamp):floor((max_range-2)/rdadelrsamp);

%% plot the RDA images
if PLOT
    % display single look image
    figure(102)

```



```

dA4 = db(rot90(A4(imrange,:)));
dA4(find(dA4==Inf))=NaN;
rllow = prctile1(dA4(:),5);
rlhigh = prctile1(dA4(:),99.9);
azdis = [-vel*t(end)/2 vel*t(end)/2];
imagesc(imrange*rdadelrsamp,azdis,dA4,[rllow rlhigh]);
xlabel('slant range (m)'); ylabel('azimuth (m)');
title(sprintf('Single look RDA image'))
colormap('gray'), %colorbar

% display multi look image
figure(104)
dC = db(rot90(C(imrange,:)));
dC(find(dC==Inf))=NaN;
rmlow = prctile1(dC(:),3);
rmhigh = prctile1(dC(:),99.99);
imagesc(imrange*rdadelrsamp,azdis, dC,[rmlow rmhigh]);
%colorbar
xlabel('slant range (m)'); ylabel('azimuth (m)');
colormap('gray')
title(sprintf('Multilook RDA image'))

%% plot georectified multilook image
figure(105)
dgC = db(rot90(gC));
dgC(find(dgC==Inf))=NaN;
rgmlow = prctile1(dgC(:),3);
rgmhigh = prctile1(dgC(:),99.9);
imagesc(sqrt(grange.^2-mean(zl)^2),azdis,dgC, [rgmlow rgmhigh]);
%colorbar
xlabel('ground range (m)'); ylabel('azimuth (m)');
colormap('gray')
title(sprintf('Georectified Multilook RDA image'))

end

```