

**Improved Geometric Modeling of Spaceborne
Pushbroom Imagery Using Modified Rational
Polynomial Coefficients and the Impact on DSM
Generation**

**Gurpreet Singh
January, 2008**

Improved Geometric Modeling of Spaceborne Pushbroom Imagery using modified Rational Polynomial Coefficients and the Impact on DSM Generation

by

Gurpreet Singh

Thesis submitted to the International Institute for Geo-information Science and Earth Observation in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation, Specialisation: (Geoinformatics)

Thesis Assessment Board

Chairman : Prof.Dr.Ir.A. (Alfred) Stein, ITC
External Examiner : Dr.P.K. Garg, IIT Roorkee.
IIRS Member : Dr. C. Jeganathan
IIRS Member : Dr. Anil Kumar
IIRS Supervisor : Mrs. Shefali Aggarwal

Thesis Supervisor

IIRS : Mrs. Shefali Aggarwal
ITC : Dr. Michel Morgan
Dr. Markus Gerke



iirs

**INTERNATIONAL INSTITUTE FOR GEO-INFORMATION SCIENCE AND EARTH OBSERVATION
ENSCHDEDE, THE NETHERLANDS
&
INDIAN INSTITUTE OF REMOTE SENSING, NATIONAL REMOTE SENSING AGENCY (NRSA),
DEPARTMENT OF SPACE, DEHRADUN, INDIA**

Disclaimer

This document describes work undertaken as part of a programme of study at the International Institute for Geo-information Science and Earth Observation. All views and opinions expressed therein remain the sole responsibility of the author, and do not necessarily represent those of the institute.

Abstract

Two sensor models are widely used for reconstructing imaging geometry in photogrammetry; the physical sensor models and the replacement sensor models. From the past few years, replacement sensor models have found considerable importance since the physical imaging parameters that are useful in producing a rigorous camera model are being withheld by the imagery vendors, deliberately. One of the replacement sensor models that have gained considerable interest in the recent past in photogrammetry and the processing of remote sensing satellite imagery is the Rational Polynomial Coefficient (RPC) sensor model. The RPC sensor model is considered to be a simple and effective way to approximate a rigorous camera model.

However it has been found that the construction of RPCs needs exterior and interior sensor orientation parameters, which may have some residual errors induced in them, which in turn induce biases in the RPCs. And in this research, an attempt has been made to improve the geometric accuracy of the RPC sensor models of IKONOS and CARTOSAT-1 with an effort to reduce the residual errors and biases of the imaging parameters, in particular with the use of ground control points (GCPs).

The research work consisted of deriving a mathematical model which was programmed in JAVA language. The software made has been tested for different sensors such as IKONOS and CARTOSAT-1 with different terrain conditions.

The model was tested for accuracy in a 3D object plain and it was able to attain the anticipated accuracies with lesser number of ground control points. CARTOSAT-1 needed around 4 to 9 GCPs and 3D affine transformation involving the denominators for giving an accuracy of around 4-5 meter vertical and less than a meter horizontal. For IKONOS it was observed that considerable accuracy of less than a pixel can be achieved both in vertical and horizontal just by using 1 to 3 GCPs.

Key words: Replacement sensor models, RPCs, GCPs, Software, JAVA, Object plain.

Acknowledgements

This thesis dissertation marks the end of a long and eventful journey for which there are many people whom I would like to acknowledge for their support along the way. Above all I would like to acknowledge the tremendous sacrifices that my parents made to ensure that I had an excellent education. For this and much more, I am forever in their debt. It is to them that I dedicate this dissertation.

I am deeply indebted to Dr. Michel Morgan my supervisor at ITC, for his direction and guidance on this research topic, and for providing me, with his work which formed the basis of the software building. What ever little has been done is because of his useful criticism and constructive suggestions from time to time.

I am greatly obliged and gratified to my supervisor Mrs. Shefali Aggarwal from IIRS whose help, stimulating suggestions and encouragement helped me all through, during the research and for writing the thesis.

I would like to show my thankfulness to my other supervisor Dr. Markus Gerke from The Dept. of Earth Observation Sciences, ITC for his input and support in the final stages of the study. I am thankful to him for his thoroughness and detail of scrutiny through examining this thesis. His comments improved the thesis and gave significance to the work.

And, it is my duty to show my gratitude to Dr. V.K Dadhwal, Dean IIRS and Mr. PLN. Raju Head Geoinformatics for giving me permission and encouragement to go ahead with the thesis.

Especially I am highly obliged to Mr. Ir.V.(Bas) Retsios (Head software developer, Geo information, ITC) and Dr. C. Jeganathan(Program coordinator, Geoinformatics) for their invaluable assistance on developing the program in Java. Their support made this work very much possible.

I want to thank my colleagues GurdeepSingh, ChandanNayak, DumindaWielekana, RakeshSharma, Dipender Chand , Shumona , Ambika, Rupinder, Samadritha and Sandeep (MSc Geoinformatics and MSc Geohazards) who supported me in my research work with their help, support, interest and valuable hints.

I am thankfull to Ms. Deepti (M.Tech) for her patience to listen to me all the time when I was having problems in research. I would always remember the long discussions on the topic and the way she helped me through the research.

Finally, I would like to give my special thanks to my Colleague and friend Ms. Shashi Dobhal (MSc Geoinformatics) who believed in me throughout the work and encouraged me to work harder.

Table of contents

1. Introduction:	1
1.1. General:	1
1.2. Problem statement and motivation:	3
1.3. Research Objective:	3
1.3.1. Main objective:	3
1.3.1.1. Sub Objectives:	3
1.4. Research Questions:	4
1.5. Research Area and data used:	4
1.6. Software used:	4
1.7. Structure of the thesis:	4
2. Research Area and Data used:	6
2.1. Research area:	6
2.1.1. Chandigarh:	6
2.1.1.1. Location and extent:	6
2.1.1.2. Geology and terrain info:	6
2.1.2. Dehradun:	7
2.1.2.1. Physical characteristics:	7
2.2. Data used:	8
2.2.1. IKONOS:	8
2.2.2. CARTOSAT – 1:	10
3. Background and Literature review:	12
3.1. Photogrammetry:	12
3.1.1. Introduction:	12
3.2. Photogrammetry with Linear array scanners and Frame images:	13
3.2.1. Linear array scanners:	13
3.2.2. Stereo coverage using linear array scanner:	14
3.2.3. Frame image :	14
3.2.4. Coordinate systems with respect to image and scene:	14
3.3. Photogrammetric processing:	15
3.3.1. Interior orientation :	15
3.3.2. Exterior orientation:	16
3.3.3. The collinearity equation:	16
3.3.4. Least square adjustments:	17
3.4. Sensor orientation modelling of pushbroom sensors:	18
3.4.1. Rigorous sensor model:	18
3.4.2. The generalized sensor models:	20
3.4.2.1. The Rational Function Model:	20
3.4.2.2. Direct Linear Transformations (DLT):	21
3.4.2.3. Two-D Affine Model:	22
3.5. Rational polynomial coefficient model:	22
3.5.1. Approaches in Determining RPCs:	24
3.6. Errors in the Exterior orientation parameters of linear array scanners:	25

3.6.1.	Attitude Errors:	25
3.6.2.	Ephemeris Errors:	26
3.6.3.	Drift errors:	26
3.7.	RPC Refinement:	26
3.7.1.	Direct refinement:	26
3.7.2.	Indirect refinement:	27
3.8.	A general brief up of Literature review on RPCs	28
3.9.	Summary	31
4.	Concepts and methodology	32
4.1.	Overall Research Methodology	32
4.1.1.	Conceptual stage	34
4.1.1.1.	Deciding upon the research area and the sensors	34
4.1.1.2.	Deciding the programming language and the software	34
4.1.1.3.	Developing a mathematical model	34
4.1.2.	Field work:	37
4.1.3.	Implementation of Designed mathematical model:	37
4.1.3.1.	Software Building	37
4.1.4.	Accuracy analysis:	50
5.	Results and discussion	52
5.1.	Analysis of Raw RPCs:	52
5.1.1.	Accuracy for CARTOSAT-1 Raw RPC:	52
5.1.2.	Accuracy for IKONOS Raw RPC:	53
5.1.3.	GCP Distribution	54
5.2.	Experiments with models	55
5.2.1.	Modelling with one parameter	55
5.2.1.1.	Using CARTOSAT - 1 with distributed GCPs	55
5.2.1.2.	Using IKONOS with GCPs	58
5.2.2.	Modelling with two parameters:	60
5.2.2.1.	Using CARTOSAT – 1 for the analysis:	60
5.2.2.2.	Using IKONOS for Analysis :	62
5.2.3.	Modeling with three parameters:	63
5.2.3.1.	Using CARTOSAT – 1 for analysis:	63
5.2.3.2.	Using IKONOS for analysis:	65
5.2.4.	Modelling with Four parameters	65
5.2.4.1.	Using CARTOSAT – 1 for the refinement:	66
5.2.4.2.	Using IKONOS for the refinement with 4 parameters:	67
5.2.5.	Using Numerators in the observation equations for Modelling:	68
5.3.	Extraction of DTM from the refined RPCs:	68
5.3.1.	Extraction of DTM from Catosat-1 using raw RPCs:	68
5.3.2.	Extraction of DEM from Catosat-1 using refined RPCs:	70
5.4.	Discussions on the Results and Analysis:	73
6.	Conclusions and Recommendations	75
6.1.	Overall conclusion:	75
6.2.	Sub Conclusions:	75
6.3.	Limitations:	76
6.4.	Recommendations:	76

7. References:	77
8. Appendix:	79
8.1. Annexure 1:	79
8.1.1. Help file for the execution of the software:	79
8.2. Annexure 2:	88
8.2.1. The code for the main class file:	88

Note:

The source code for the built software (for RPC refinement) is given in Annexure 2; (However this source code gives the code for the main class of the program and more classes are required to compile the code). The complete software is provided in a CD, with the thesis. A copy of it is also available with my guides, Dr. Michel Morgan (ITC), Dr.Markus Gerke (ITC) and Mrs. Shefali Aggarwal (IIRS). The help file for running the software and the functions used, is listed in Annexure 1.

List of figures

Figure 2-1 Describes the location of Chandigarh city in India. (Source: Google-earth.com).....	7
Figure 2-2 Describes the location of Dehradun city in India (Source: Google-earth.com).....	8
Figure 2-3 IKONOS stereo pair of Chandigarh	9
Figure 2-4 CARTOSAT-1 Stereo Pair of Dehradun	11
Figure 3-1 linear array scanner (Source: Morgan, 2004)	13
Figure 3-2 Sequence of 1-D images (a) consisting a scene (b) (Source: Morgan, 2004).....	14
Figure 3-3 Shows interior orientation of linear array scanner for SPOT scene (Source: Erdas, 2005)..	15
Figure 3-4 showing the exterior orientation of the camera image (Source: Erdas, 2005).....	16
Figure 3-5 linear regression on X and Y	17
Figure 3-6 RPC Evolution and Generation (Source: Hu et al., 2004).....	24
Figure 3-7 Illustration of VAPs generation (source :Chen et al., 2006).....	25
Figure 4-1 the overall methodology flowchart.....	32
Figure 4-2 Methodology subdivision	33
Figure 4-3 Flow chart showing the implementation of the Design	38
Figure 4-4 Sample for Normalized RPCs.....	39
Figure 4-5 Sample for Denormalized RPCs.....	41
Figure 4-6 L matrix using 2 GCPs	46
Figure 4-7 the structure of matrix for 1 GCP and 1 parameters.....	47
Figure 4-8 the structure of matrix with 2 GCPs and 2 Parameters.....	48
Figure 4-9 the structure of the matrix with 1 GCP, 1 Tiepoint and 1 parameter.....	49
Figure 5-1 Object coordinates of CARTOSAT-1	52
Figure 5-2 Graphs for a) Planimetric errors b) Horizontal errors of Raw RPCs in CARTOSAT-1 ..	53
Figure 5-3 Object coordinates for IKONOS	53
Figure 5-4 Graphs for a) Planimetric errors b) Height errors of IKONOS Raw RPCs.	54
Figure 5-5 Block file showing position of CARTOSAT-1 GCPs.....	54
Figure 5-6 Block file showing Positions of IKONOS GCPs	55
Figure 5-7 Graphs a, b, c, d showing Planimetric and height accuracy with 1 parameter using 17 and 10 GCPs for CARTOSAT-1 image.....	56
Figure 5-8 Graphs a, b, c, d showing Planimetric and height accuracy with 1 parameter using 7 and 4 GCPs for CARTOSAT-1 image.....	57
Figure 5-9 Graphs a, b, c, d showing Planimetric and height accuracy with 1 parameter using 1 and 10 GCPs for IKONOS images.....	59
Figure 5-10 Graphs a, b, c, d showing Planimetric and height errors with 2 parameter using 4 and 17 GCPs for CARTOSAT-1 image.....	61
Figure 5-11 Graphs a, b, showing Planimetric and height errors with 2 parameter using 12 GCPs for IKONOS image.	62
Figure 5-12 Graphs a, b, c, d, e, f shows Planimetric and height errors with 3 parameters using 17, 7 and 6 GCPs for CARTOSAT-1 image.	64
Figure 5-13 Graphs a, b, showing Planimetric and height errors with 3 parameter using 12 GCPs for IKONOS image	65
Figure 5-14 Graphs a, b, c, d shows Planimetric and height errors with 3 parameters using 17 and 9 GCPs for CARTOSAT-1 image.....	66

Figure 5-15 Graphs a, b, shows Planimetric and height error with 4 parameters considering 12 GCPs for IKONOS image.	67
Figure 5-16 CARTOSAT DTM with Original RPCs.....	69
Figure 5-17 DEM surface profile with original RPCs.....	69
Figure 5-18 Quality map of the DTM extracted from the original RPCs.....	70
Figure 5-19 DEM from modified RPCs	71
Figure 5-20 Surface profile of DEM from modified RPCs.....	71
Figure 5-21 Point map of DTM extracted from Refined RPCs.....	72
Figure 5-22 Planimetric accuracy with CARTOSAT- 1.....	73
Figure 5-23 Vertical accuracy with CARTOSAT-1.....	73
Figure 5-24 Planimetric accuracy with IKONOS	73
Figure 5-25 Vertical accuracy with IKONOS	73

List of tables

Table 2-1	Basic geographic Profile of Chandigarh	6
Table 2-2	IKONOS sensor information (Source http://spaceimagine.com).....	9
Table 2-3	IRS – P5 Satellite information (source: http://www.nrса.gov.in/satellites/irs-p5.html)	10
Table 2-4	Major specifications of the sensors (source: http://www.nrса.gov.in/satellites/irs-p5.html)	11
Table 5-1	Over all accuracy with GCPs and Check points of CARTOSAT–1 orientation through 1 parameter model.	56
Table 5-2	Accuracy with only check points of CARTOSAT-1 with one parameter.....	58
Table 5-3	IKONOS orientation accuracy with both GCPs and Checkpoints through 1 parameter model.....	58
Table 5-4	Accuracy with only the check points of IKONOS with one parameter.	60
Table 5-5	CARTOSAT-1 Orientation with 2 parameter model.	60
Table 5-6	IKONOS orientation through 2-parameter model.....	62
Table 5-7	CARTOSAT-1 orientation with 3 parameter model.....	63
Table 5-8	IKONOS orientation through 2 parameter model.....	65
Table 5-9	CARTOSAT-1 orientation with 4 parameter model.....	66
Table 5-10	IKONOS orientation with 4 parameter model	67

1.Introduction:

1.1. General:

With the advancement in technology and increasing general awareness among people, the need for using high resolution satellite images (HRSI) for various applications is realized and hence the ever escalating demand for HRSI. The current and future high resolution earth observations satellites having a spatial resolution of 1 m and less and stereo capabilities will go a long way in the field of mapping in terms of cost, time and accuracy. Accordingly, there is a greater need to fully understand the potential and indeed shortcomings of alternative photogrammetric sensor orientation models for HRSI.

Using High resolution satellites have opened up the opportunities for exploiting their spatial resolution in mapping. Many remote sensing satellites exist that can be effectively used for map production, including SPOT, IRS-1C/D, IKONOS, and recently QUICKBIRD and CARTOSAT-1. One can consider HRSI as an indispensable mapping tool which can be used in the developing world, as they require very little in the way of ground survey data. However, one of the limiting factors is the cost of HRSI which effects its application for large-scale mapping where metric tolerances are most stringent. For example IKONOS imagery has base-level accuracy of 15m available at the cost of about \$US 20/km², whereas the 4m accurate Precision image product costs around \$US 120/km² in Africa and Asia. How ever a very attractive proposition for mapmakers in the developing world, therefore, would be to have meter-level accuracy /Precision at the price of base-level Geo imagery.(Satirapod et al., 2004) These days most of the high resolution satellites use linear array pushbroom sensors. “Based on the pushbroom scanning geometry, a number of investigations have been reported regarding the geometric accuracy of linear array images (Westin, 1990; Chen and Lee, 1993; Li,1998; Tao et al., 2000; Toutin,2003; Grodecki and Dial,2003).” as cited in(Chen et al., 2006).

The generation of such image and its derived products with high accuracies from HRSI requires exceptional efforts through photogrammetric techniques.To get the full geometric potential of the high and very high resolution, space images require a correct mathematical model or a three dimensional interpolation function which is based on sensor geometry and orientation. Different orientation models affect the accuracy of the image-derived products such as the Digital Terrain Models (DSM) and the orthoimages and to comprehend this, we need to understand the role of various sensor models. A sensor model can be used for describing the geometric relationship between the 3D ground coordinates and the 2D image space coordinates. (Hu et al., 2004)There are two broadly used imaging geometric models. These are the rigorous sensor model and the replacement sensor model. The rigorous sensor model is used for the reconstruction of the physical imaging settings; these settings include physical parameters about camera, such as focal length, principal point location, pixel size and lens distortions and also orientation parameters of the image such as position and altitude of the image. One of the most efficient and popular way to implement transformations based on rigorous sensor model is through Co-linearity conditions. The rigorous sensor model has been conventionally used in the photogrammetric processing because of clear separation between various physical parameters. In

contrast, a replacement sensor model does not include sensor position and orientation information. (Di et al., 2003)

There are three main replacement sensor models, namely, the grid interpolation model, The RPC model and the universal real-time sensor model (USM). These models can be considered generic in nature and their parameters are mathematical and do not have any physical significance in the imaging process. (OGC, 1999; Hu et al., 2004) Rational functions have been accepted as a standard with the mapping community, because of its popularity with the U.S military community in the past. Rational functions are often applied in photogrammetric and remote sensing processing to represent the transformation between the image space and the object space whenever the rigorous model is made unavailable to the users intentionally or unintentionally. The physical parameters for rigorous model are most commonly not supplied with the high resolution imagery like IKONOS and the recent CARTOSAT-1. Now –a- days, physical and abstract mathematical models are used with pushbroom satellite imagery, the imagery vendors do not provide ephemeris data (or ancillary data) for the end users, so one has to rely on mathematical sensor models for extracting 3D geospatial information. Imagery vendors of push broom sensors such as IKONOS and CARTOSAT-1 provide users with RPC files which are rational polynomial coefficients. The RPC can be considered as a re-parameterization of the rigorous sensor orientation model. An RPC model is the ratio of two polynomials which can be derived from the rigorous sensor model and the corresponding terrain information, which does not reveal the sensor parameters. However it has been seen that using RPC with out the aid of Ground control points give rise to geo positioning errors. This can be contributed to the inherent interior and exterior orientation errors in the rigorous sensor model, which subsequently give rise to the errors in the RPC. These errors in geo positioning can be compensated by additional parameters in the image space that effect a translation of image coordinates, with spatial intersection. The additional parameters which are introduced help in the removing the biases in the interior orientation parameters such as principal point, detector position, lens distortions, focal length and those in exterior orientation which constitute of position and attitude errors who are attributed to object space (Grodecki and Gene, 2003).

To understand and set up these parameters correctly, one needs to fully understand and reconstruct the rigorous model, which if known, does not require any RPCs. Our objective, however, is to reconstruct the RPC without the knowledge of the rigorous model by selecting and analyzing the importance and significance of these parameters, to reach high accuracy in the object space.

As now we know that nowadays RPCs are provided instead of physical parameters with HRSI, if one would want to generate DSM from the recently launched high resolution imagery he would have to use the RPCs. However with the supplied RPC's one can generate DEM with limited Precision because of the bias in the polynomial coefficients. Therefore one needs to refine the supplied RPC to achieve higher accuracy levels. (Cho et al., 2003) .It has been observed that with the recent innovation of bias-compensated RPC adjustment, it's demonstrated that sensor orientation of sub-pixel level can be achieved with minimal ground control. (Fraser et al., 2006)

With the modified RPCs, it is possible for the end users to directly obtain spatial information from the imagery with higher precision levels than supplied. . Thus, modifying/correcting RPCs will open the gates for generating accurate photogrammetric products.

1.2. Problem statement and motivation:

RPC is becoming popular in today's world and is being used in a lot of photogrammetric applications such as generation of DEM, Orthoimages etc. Even the commercial-off-the-shelf digital photogrammetric workstations have also incorporated the RPC and related techniques. A number of photogrammetric systems such as LH systems SOCET Set, ZI imaging and ERDAS have incorporated RPCs. However the RPC's provided by the vendors may not always represent the real imaging process well. This could be so due to various reasons such as the biases in the rigorous sensor model for generating RPCs, systematic errors or could also be marketing strategy from the image vendors, as they want to sell high precision products at relatively higher prices.

It has been learnt that RPCs can be modified or refined either in image space or object space, if additional control information is available. The RPCs may be modified or refined using two general approaches, the direct approach or the indirect approach. The direct refining approach is used for updating the original RPCs themselves, but it uses more control points while the indirect refining approach introduces complementary or concealed transformations in the image space or the object space and they do not change the original RPCs.(Hu et al., 2004) .Therefore a model needs to be developed with which one can refine the existing RPC values using few control points and mathematical modelling, which will enable the end users to get photogrammetry products with a higher geo-positional accuracy easily even without knowing the nuances of photogrammetric processing in detail.

This research deals with the aspect of modifying the supplied RPC coefficients values (80 per image) for the two high resolution satellite sensors IKONOS and CARTOSAT-1.

1.3. Research Objective:

1.3.1. Main objective:

The main objective of the research is to develop a model and method, which can be used for modifying the RPC supplied by the imagery vendor, so as to achieve better accuracy.

1.3.1.1. Sub Objectives:

- Different models or combinations of parameters will be analysed and the best combination will be found for both the images in order to reduce the errors.
- The minimum control point requirement will be found for modifying the RPC values for both the images in order to attain significant accuracy.
- Comparison will be made between the DSM generated using vendors supplied RPCs and the refined RPCs.

1.4. Research Questions:

- The first research question talks in general of how to improve the RPC with the model?
- What should be done in order to minimize the ground control point requirement?
- How can one assess improvement of RPCs?
- What is the impact of refined RPCs on the derived Products such as DTM?

1.5. Research Area and data used:

The data sets used corresponds to two sensors IKONOS and CARTOSAT-1 corresponding to different terrain conditions, flat terrain and moderately hilly terrain. This is done so as to check the accuracy of the model in variable terrain conditions.

The two sites correspond to Chandigarh city area and Dehradun and its surrounding area, in India, where Chandigarh has a flat terrain and Dehradun has moderately hilly terrain.

1.6. Software used:

The lists of software that have been used during the research are as follows:

- Java development kit 1.6.2: Java programming language has been used for building the software to execute different models.
- Matlab and Excel: For statistical analysis.
- Leica photogrammetric suit : For extracting DTMs and DEMs

1.7. Structure of the thesis:

This thesis contains five chapters.

- The first chapter (Introduction) gives a brief introduction about the topic and also makes us aware about the problem statement, research objectives, research questions, study area, the data used and the software's that have been used.
- The second chapter gives the required briefing about the research area and sensor data that have been used during the research.
- The Third chapter (Back ground and literature review) gives detailed background about rational polynomial coefficients and the significant literature has been reviewed for the understanding. It also discusses about the previous work that has been done in the related field.
- The fourth chapter talks in detail about the methodology that has been adopted for the improvement of the rational polynomial coefficients,
- The fifth chapter (Results and discussions) discusses the results of comparison of different combinations using both the sensor data.

- The sixth chapter (Conclusion) answers the research questions and concludes the thesis, giving future recommendations.

2. Research Area and Data used:

This chapter puts light on the Characteristics of data used and research area selected for the research. It starts by giving the overview about the research areas and proceeds with the sensor characteristics.

2.1. Research area:

A brief description of the research area selected for the research is given as under.

2.1.1. Chandigarh:

2.1.1.1. Location and extent:

The study area Chandigarh, lies in the northwest of India having boundary coordinates $78^{\circ}00'36.06''\text{E}$ – $78^{\circ}02'37.26''\text{E}$ and $30^{\circ}24'14.91''\text{N}$ - $30^{\circ}21'46.98''\text{N}$ and shares its borders with the state of Haryana in the south and Punjab in the north .The exact cartographic co-ordinates of Chandigarh are 30.74°N 76.79°E . It has an average elevation of 321 meters (1053 feet).

Basic geographic Profile of Chandigarh is as under	
Area	114 sq Kms.
Longitude	$76^{\circ}47'14\text{ E}$
Latitude	$30^{\circ}44'14\text{ N}$
Altitude	304-365 meters above MSL
Monsoon	July-September
Temperature	Winter(Nov- Jan, 2005) 3°C - 14°C Summer(April- july,2004) 31°C - 44°C

Table 2-1 Basic geographic Profile of Chandigarh

2.1.1.2. Geology and terrain info:

The union territory of Chandigarh is located in the foothills of shivalik hill ranges in the north, which form a part of fragile Himalayan ecosystem. It is occupied by Kandi (Bhabhar) in the northeast and

Sirowel (Tarai) and alluvial plains in the remaining part. The subsurface formation comprises of beds of boulders, pebbles, grave, sand, silt, and clay. The area is drained by two seasonal rivulets viz. sukhna choe in the east and patiala-Ki-Rao choe in the west.



Figure 2-1 Describes the location of Chandigarh city in India. (Source: Google-earth.com)

2.1.2. Dehradun:

2.1.2.1. Physical characteristics:

The city of Dehradun is situated in the south central part of Dehradun district. Dehradun city lies at $30^{\circ} 19' N$ and $78^{\circ} 20' E$. Two intermittent streams viz. Rispana river and Bindal river, on the east and west respectively marks the physical limits of Dehradun municipality. The city is located at an altitude of 640 m above MSL. The lowest altitude is around 600 m in the southern part and the highest altitude is around 1000 m in the northern part.



Figure 2-2 Describes the location of Dehradun city in India (Source: Google-earth.com)

2.2. Data used:

Two satellite data sets of IKONOS and CARTOSAT – 1 were used. A brief description of the satellite data is given as under.

2.2.1. IKONOS:

IKONOS was the first commercial high-resolution satellite to be placed into orbit. IKONOS is owned by space imaging, a USA based earth Observation Company. The OSA sensor onboard IKONOS is based on the pushbroom principal and can simultaneously take panchromatic and multispectral images. In addition to a high spatial resolution of 1m panchromatic and 4 m multispectral, it has also the radiometric resolution using 11- bit quantization.

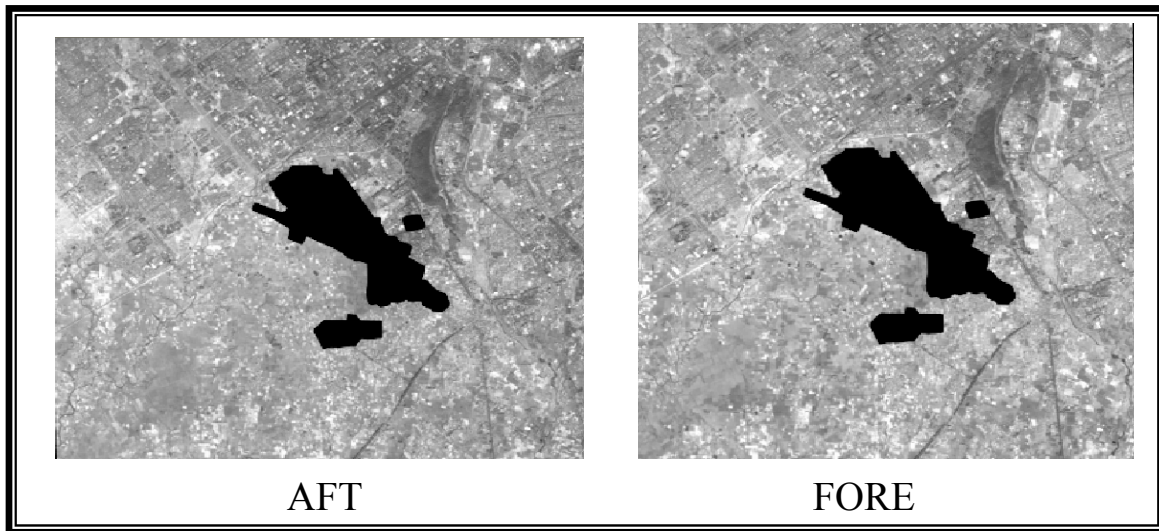


Figure 2-3 IKONOS stereo pair of Chandigarh

The figure 2 -3 above shows the stereo pair of IKONOS as on date 14th- December- 2003 used in the research, and the black patch in the middle of the scene is the area masked out for security reasons. The details of the IKONOS sensor are given as under in table 2-2.

System	IKONOS
Orbit	681 km, 98.2° inclination Sun- synchronous, 10:30 AM crossing, 14 days repeat cycle.
Sensor	Optical sensor assembly(OSA)
Swath width	11 km
Off – nadir viewing	50° Omni directional
Revisit time	1 – 3 days
Spectral bands(μm)	0.45 – 0.52 (1), 0.52 – 0.60 (2), 0.63 – 0.69(3), 0.76 – 0.90 (4), 0.45 – 0.90(PAN)
Spatial resolution	1 m panchromatic, 4 m (bands 1 – 4) www.spaceimaging.com
Data archive at	

Table2-2 IKONOS sensor information (Source <http://spaceimagine.com>)

2.2.2. CARTOSAT – 1:

CARTOSAT-1 is a global mission. The nominal life of the mission is planned to be five years. The satellite was launched by the indigenously built Polar Satellite Launch Vehicle on May 05, 2005. The satellite information is given by the table 2-3 as shown below.

Orbit	Polar, sun-synchronous
Orbital Altitude	618 km
Semi Major Axis	6996.14 km
Eccentricity	0.001
Inclination	97.87 degrees
Local time	10:30 A.M
Revisit	5 days
Repetevity	126 days
Orbits/day	14
Period	97 minutes

Table 2-3 IRS – P5 Satellite information (source: <http://www.nrsa.gov.in/satellites/irs-p5.html>)

The payload system of IRS-P5 consists of two panchromatic solid state cameras - Fore and Aft, mounted at +26 degrees and -5 degrees with respect to nadir to generate stereoscopic image of the area along the track. Both the cameras work on the 'pushbroom scanning' concept using linear arrays of Charge Coupled Devices (CCDs) as sensors. The spacecraft body is steerable to compensate the earth rotation effect and to force both fore and aft cameras to look at the same ground strip when operated in stereo mode.

The CARTOSAT-1 fore and aft scenes of date 5th-February- 2005 were used for the research purpose. The fore and aft scene of the Dehradun city can be observed by the following figure 2-4. The images also show that the terrain is rugged and has lot of undulation with lesser Himalayas covering the city.

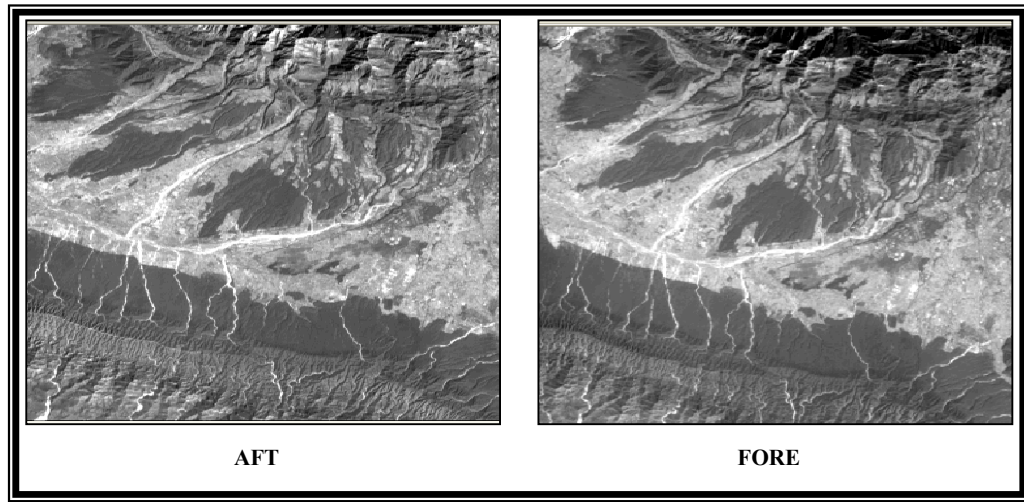


Figure 2-4 CARTOSAT-1 Stereo Pair of Dehradun

The major specifications of the CARTOSAT-1 sensor have been given as below.

Parameter	Specification
Swath Fore	29.42 km
Aft	26.24 km
IGFOV Fore	2.452 m (Across track)
Aft	2.187 m (Across track)
Ground sample distance	2.54 m (Along Track)
Spectral band	0.5 – 0.85 microns
Quantization	10 bits (1024)
Number of detectors	12 K
Pixel size	7 x 7 micron
Integration time	0.336 ms
Focal length	1945 mm
Data rate per Camera	336 Mbps
Data compression Ratio	3.2:1
Type of compression	JPEG like
Data rate transmitted to ground	105 Mbps

Table 2-4 Major specifications of the sensors (source: <http://www.nrsa.gov.in/satellites/irs-p5.html>)

3. Background and Literature review:

This chapter initially provides the reader with a brief background of photogrammetry and its concepts. An introduction to photogrammetry is given in the section (3.1). In the section (3.3), the concepts related to photogrammetric processing are discussed with respect to the frame images as well as the linear array scanners (push broom sensors). And a brief introduction to the line scanners and frame camera images, is given in the section (3.2). Sensor orientation models are discussed in detail in section (3.4) and the rational polynomial coefficient model is explained in section (3.5). A brief overview of Errors in exterior orientation is given in (3.6). Overview of RPC refinement techniques used in the past is provided in section (3.7). Section (3.8) summarises all important literature reviewed and gives the major findings with each paper.

3.1. Photogrammetry:

In this section introduction to photogrammetry is given. It also familiarises us with concepts and techniques used in photogrammetry.

3.1.1. Introduction:

Photogrammetry is the “art, science and technology of obtaining reliable information about physical objects and the environment through the process of recording, measuring and interpreting photographic images and patterns of electromagnetic radiant imagery and other phenomenon”(Erdas, 2005) . Photogrammetry has been largely used for extracting topographic information from aerial maps and satellite images. Photogrammetry has evolved from analog photogrammetry to Analytical photogrammetry and now we are witnessing the age of digital photogrammetry. In analog photogrammetry optical or mechanical instruments were used for reconstructing 3D geometry from two photographs that overlap. The main product of the analog photogrammetry was topographical maps. The shift from analog to analytic photogrammetry was realised with the introduction of computers, in this phase the devices used were analog/digital hybrids. Outputs of analytical photogrammetry are not only topographic maps, but also digital maps and DEMs. We are living in the phase of digital photogrammetry in which the digital images are stored and processed on the computers. Digital photographs can be scanned or can be directly captured by the digital cameras. Digital photogrammetry has highly automated many photogrammetric tasks such as automatic DEM extraction and digital orthophoto generation. Digital photogrammetry uses highly sophisticated software's to automate the tasks used in conventional photogrammetry. Leica photogrammetric suit (LPS) is one such software. (Erdas, 2005)The geographic information collected using photogrammetric approaches save time and money, and also maintain highest of accuracies.

Nowadays photogrammetric techniques are also used for geometrically correcting the images for both frame images and push broom sensors, instead of using convention techniques of polynomial transformations based on general functions, which were used in the past. These conventional methods had a drawback that they could only process a single image at a time and used greater number of

GCPs. However photogrammetric techniques can rectify more than one image and with lesser number of GCPs using the technique of least square bundle block adjustments.

The concepts related to photogrammetric techniques used in both frame images and linear array scanners are talked about in brief in the upcoming paragraphs starting with the definition of frame cameras and linear array scanners as under.

3.2. Photogrammetry with Linear array scanners and Frame images:

In this section a brief introduction will be provided about linear array scanners and frame cameras

3.2.1. Linear array scanners:

The principle behind pushbroom sensor is the use of array of charged coupled devices (CCDs) for measuring the electromagnetic energy. A CCD- array is a line of photo- sensitive, solid state detectors. A single detector can be as small as $5 \mu\text{m}$. (Bakker et al., 2004)

Linear array scanners or line cameras (push broom sensors) capture the image in 1- Dimension or 1- D i.e. it captures a very narrow strip from the ground per snapshot. These sensors maintain ground coverage and the resolution by having more pixels in 1-D array. Successive snapshots of the ground are taken as the as the camera moves along the flight direction. The scene of the area of interest is taken by stitching the 1-D images, captured through each snapshot. Each set of 1-D image has a single exposure station and there fore each set has different exterior orientation parameters. In case of linear array image there are many 1-D images and each has its own exposure station, there fore a scene taken from linear array scanner will have different exposure stations at different times (Morgan, May 2004) The following figure 3-1 describes the linear array scanner.

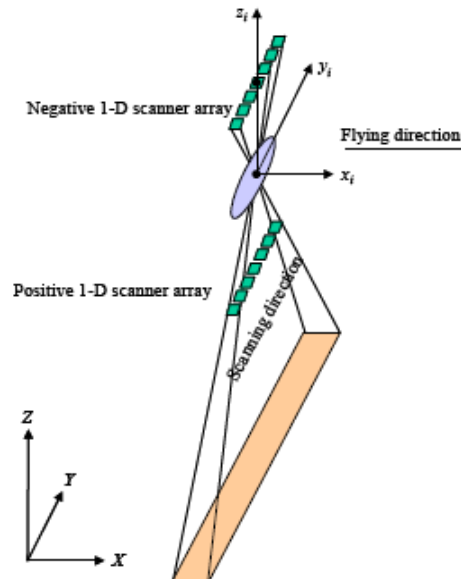


Figure 3-1 linear array scanner (Source: Morgan, 2004)

3.2.2. Stereo coverage using linear array scanner:

Push broom sensors have shown the ability of off-nadir viewing, i.e. the scanner can be pointed towards the areas left or right of the orbit track. This helps these sensors to produce stereo image pairs, *aft* and *fore* at almost the same time under the same conditions, such as season, weather, and plant life.(Bakker et al., 2004) The stereo products can be produced along or across the flight direction by turning the camera left and right or front and back. Along track stereo are considered to have more advantages than across track as the time delay between stereo acquisitions is smaller with along track ones.(Tonolo and Poli, 2003) IKONOS produces stereo along the track and SPOT uses across track stereo generation technique.(Morgan, 2004)

3.2.3. Frame image :

Frame images are usually taken from the conventional photographic cameras mounted on some aircraft or low orbital satellites and on NASA space shuttle missions. Films were used in analog frame cameras in the past. However, CCDs instead of film are used in digital cameras nowadays.(Bakker et al., 2004) An image can be defined as the recorded sensory data associated with one exposure station. The whole scene in this case is taken in one snapshot and consequently it is one complete image.(Morgan, 2004)

3.2.4. Coordinate systems with respect to image and scene:

The relationship between the sensor that is used to capture imagery, the imagery itself, and the ground, is used in photogrammetry, this relationship can be understood by using three variables (x , y , z) associated with coordinate system. (Erdas, 2005)However it is important to differentiate between image coordinates and scene coordinates as the mathematical model that relates a point in the object space with the corresponding point in the image space is contained in collinearity equations which again uses the exterior orientation parameters of the image in which point appears. The difference can be seen in figure 3-2 where i and y are the scene coordinates while x_i and y_i are the image coordinates for image number i .

Only x_i and y_i can be used in the collinearity equations, while i indicate the image number or the time of exposure.(Morgan, 2004)

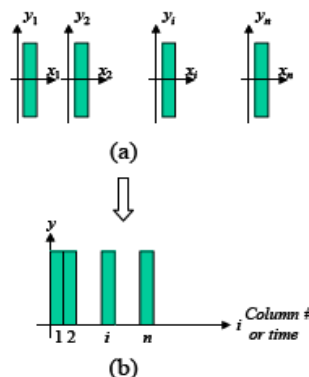


Figure 3-2 Sequence of 1-D images (a) consisting a scene (b) (Source: Morgan, 2004)

3.3. Photogrammetric processing:

Photogrammetric processing implies solution of two main problems one of them is “space intersection”(Erdas, 2005) and its solution helps in generating digital elevation models. It solves the basic problem of determining the object-space coordinates (x , y , and z) of a point in the image space (l , s). The other problem is called as “space resection”(Erdas, 2005) and its solution helps in generation of orthoimages and for 3D vectorization. It calculates point pixel coordinates or image coordinates (l , s) from object space coordinates. Some of the concepts such as “interior orientation”(Erdas, 2005), “exterior orientation”(Erdas, 2005), “collinearity equations”(Erdas, 2005), and “least squares”(Herve, 2003) are used throughout in this research and extensively in applications such as space intersection and resection, sensor orientation models etc. For getting clearer understanding of the terms used ahead, these concepts are explained in brief in the sections (3.3.1) to (3.3.4). For detailed information please refer to the references mentioned between the texts.

3.3.1. Interior orientation :

Interior orientation reconstructs the position of the perspective centre of the sensor with respect to the image, with the help of the principal point and the principal distance. It usually defines the internal geometry of the camera or sensor as it exists at the time of data capture. It includes parameters for detector positions, principal point focal length, perspective centre, optical distortions etc.(Grodecki and Gene, 2003) In case of satellite interior orientation the transformation between file coordinate system and image coordinate system is kept constant. And for each scan line separate bundle of light ray is defined.

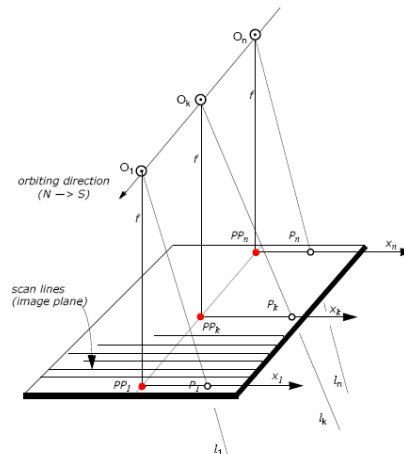


Figure 3-3 Shows interior orientation of linear array scanner for SPOT scene (Source: Erdas, 2005)

In figure 3-3 the interior orientation of push broom scanner is explained where

PP_k = image point

x_k = x value of image coordinates for scan line k

f = focal length of the camera

O_k = perspective center for scan line k , aligned along the orbit

PP_k = principal point for scan line k

l_k = light rays for scan line, bundled at perspective center O_k (Erdas, 2005)

3.3.2. Exterior orientation:

Exterior orientation reconstructs the position, inclination and rotation of the sensor with respect to the terrain coordinate system. It defines the position and angular orientation associated with an image. The elements of exterior orientation define the characteristics associated with an image at the time of exposure or capture. The positional elements of exterior orientation are X_o , Y_o and Z_o . Three rotation angles are commonly used to define angular orientation. They are omega (ω), phi (Φ), and kappa (κ). Using these three rotation angles, the relationship between the image space coordinate system (x , y , and z) and ground space coordinate system (X , Y , and Z) can be determined as shown in Figure (3.4). A 3×3 matrix referred to as the orientation or rotation matrix M , is used for defining the relationship. The rotation matrix can be defined as follows:

$$M = \begin{pmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{pmatrix} \quad \text{Equation 3-1}$$

The symbology of equation 3-1 is taken from Erdas field guide.

This matrix is derived by using the sequential rotation of omega about the x-axis, phi about the y-axis, and kappa about the z-axis. In case of exterior orientation with respect to linear array scanners there is a different exposure station for each scan line therefore each scan has its own set of exterior orientation parameters. (Erdas, 2005)

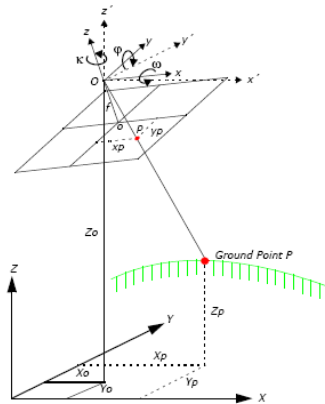


Figure 3-4 showing the exterior orientation of the camera image (Source: Erdas, 2005)

The exterior orientation parameters include position and attitude. The GPS receivers that are on the satellite are used for determining satellite ephemeris, i.e. camera position with respect to time. Star trackers and gyros on board measure camera attitude as a function of time. (Grodecki and Gene, 2003)

3.3.3. The collinearity equation:

Collinearity equation is one of the most important and useful formulation in photogrammetric applications. Collinearity equation defines the relationship between the camera/sensor the image and the ground. This equation forms the basis of the collinearity condition that is used in most Photogrammetric operations. "The collinearity condition specifies that the exposure station, ground point, and its corresponding image point location must all lie along a straight line, thereby being

collinear”.(Erdas, 2005) For every measured GCP, there are two corresponding image coordinates (x and y). Therefore, two collinearity equations can be formulated to represent the relationship between the ground point and the corresponding image measurements. In the context of bundle block adjustment (Erdas, 2005), these equations are known as observation equations.

$$x_p - x_o = -f \left[\frac{m_{11}(X_p - X_{o_i}) + m_{12}(Y_p - Y_{o_i}) + m_{13}(Z_p - Z_{o_i})}{m_{31}(X_p - X_{o_i}) + m_{32}(Y_p - Y_{o_i}) + m_{33}(Z_p - Z_{o_i})} \right] \quad \text{Equation 3-2}$$

$$y_p - y_o = -f \left[\frac{m_{21}(X_p - X_{o_i}) + m_{22}(Y_p - Y_{o_i}) + m_{23}(Z_p - Z_{o_i})}{m_{31}(X_p - X_{o_i}) + m_{32}(Y_p - Y_{o_i}) + m_{33}(Z_p - Z_{o_i})} \right] \quad \text{Equation 3-3}$$

Where the terms used in the equation 3-2 and 3-3 can be referred to figure 3-4 and Equation 3-1.

The symbology of equation 3-2 and 3-3 is taken from Erdas field guide.

The collinearity conditions are extensively used with the concept of least squares in photogrammetry described as under.

3.3.4. Least square adjustments:

The least square method is a very popular technique that has been used over ages to produce solution for parameter estimation and also for fitting of the data. It is one of the oldest techniques of modern statistics as it was first published in 1805 by the French mathematician Legendre in a now classic memoir. However it was discovered that this method was being used in as early as 1795.(Herve, 2003) It can be defined as “A mathematical procedure for finding the best-fitting curve to a given set of points by minimizing the sum of the squares of the offsets ("the residuals") of the points from the curve.” (Weisstein, 2007)The least squares parameter estimation method is a variation of the probability plotting methodology in which one mathematically fits a straight line to a set of points in an attempt to estimate the parameters. The method of least squares requires that a straight line to be fitted to a set of data points such that the sum of the squares of the vertical offset from the points to the line is minimized, if the regression is on Y, or the line be fitted to a set of data points such that the sum of the squares of the Horizontal offset from the points to the line is minimized, if the regression is on X. The regression on Y is not necessarily the same as the regression on X. The only time when the two regressions are the same (*i.e.* will yield the same equation for a line) is when the data lie perfectly on a line.

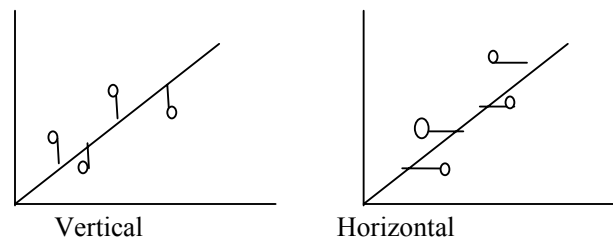


Figure 3-5 linear regression on X and Y

3.4. Sensor orientation modelling of pushbroom sensors:

Push broom sensor modelling can be broadly divided into two kinds of models, the parametric or Rigorous sensor models and the non – parametric or generalized sensor models.(Tonolo and Poli, 2003) Both the models will be explained in the section below.

3.4.1. Rigorous sensor model:

Rigorous or exact modelling of the pushbroom sensors tells us about the actual formation of scene at the time of photography. This model is capable of fully delineating the imaging geometry between the Object space and the image space.(Chen et al., 2006) Rigorous sensor models have been considered as the most precise models. These sensor models are based on collinearity equations (Equation 3-2 and Equation 3-3) where m_{ij} are elements of rotational matrix with three angles of $(\omega_i, \varphi_i, \kappa_i)$. X_o and Y_o are interior orientation parameters and X_{o1} , Y_{o1} , Z_{o1} and $\omega_i, \varphi_i, \kappa_i$ includes exterior orientation parameters.(Di et al., 2003) For the linear array sensor this model requires the knowledge of the exterior orientation parameters of each image in the scene, The EO parameters are stable or do not change their values during consecutive images in a scene usually the space based scenes. Therefore most of the rigorous models adopt collinearity equations as shown in equation 3.2 and equation 3.3 which use polynomial method for representing the EOP.(Morgan, 2004)

$$X_{0i} = X_0 + \Delta X_i + \dots + \Delta X_{nX} i^{nX} \quad \text{Equation 3-4}$$

$$Y_{0i} = Y_0 + \Delta Y_i + \dots + \Delta Y_{nY} i^{nY} \quad \text{Equation 3-5}$$

$$Z_{0i} = Z_0 + \Delta Z_i + \dots + \Delta Z_{nZ} i^{nZ} \quad \text{Equation 3-6}$$

$$\omega_i = \omega + \Delta \omega_i + \dots + \Delta \omega_{n\omega} i^{n\omega} \quad \text{Equation 3-7}$$

$$\phi_i = \phi + \Delta \phi_i + \dots + \Delta \phi_{n\phi} i^{n\phi} \quad \text{Equation 3-8}$$

$$\kappa_i = \kappa + \Delta \kappa_i + \dots + \Delta \kappa_{n\kappa} i^{n\kappa} \quad \text{Equation 3-9}$$

Where:

i is the image number (which is directly related to the time of exposure);

(X_{0i}, Y_{0i}, Z_{0i}) are the spatial location of the exposure station of image i ;

$(\omega_i, \varphi_i, \kappa_i)$ are the rotation angles of image i ;

(nX, nY, nZ) are the degrees of the polynomials of X_{0i} , Y_{0i} and Z_{0i} , respectively;

$(n\omega, n\varphi, n\kappa)$ are the degrees of the polynomials of ω_i , φ_i and κ_i , respectively;

(X_0, Y_0, Z_0) are the spatial location of the exposure station of the first image in the scene (image 0);

$(\Delta X, \Delta Y, \Delta Z)$ are the linear changes of scanner location (components of the scanner velocity vector – the first order components of the scanner location);

$(\Delta X_{nX}^{th}, \Delta Y_{nY}^{th}, \Delta Z_{nZ}^{th})$ are the nX^{th} , nY^{th} and nZ^{th} order components, respectively, of the scanner location;

$(\omega, \varphi, \kappa)$ are the rotation angles of the first image in the scene;

$(\Delta\omega, \Delta\varphi, \Delta\kappa)$ are the linear changes of the rotation angles (the first order components of the scanner rotation angles); and

$(\Delta\omega_{n\omega}^{th}, \Delta\varphi_{n\varphi}^{th}, \Delta\kappa_{n\kappa}^{th})$ are the $n\omega^{th}$, $n\varphi^{th}$ and $n\kappa^{th}$ order components, respectively, of the scanner rotation angles.

The symbology of the equations is taken from (Morgan, 2004)

The orbital relations or the polynomial order can also be different for each polynomial. One can also say that values of nX , nY , nZ , $n\omega$, $n\varphi$ and $n\kappa$ may differ, depending on the scanner movement pattern. Their typical values are therefore scanner-dependent. The parameters that have been used in Equations are either given (directly) from the navigation units such as GPS and INS sensors mounted on the platform, or indirectly estimated using ground control in bundle adjustment. (Morgan, 2004)

RF model is generally generated based on the rigorous sensor model, where after the rigorous sensor bundle adjustment is performed, we can generate multiple evenly distributed object/ image grid points these can be used as control points. Then the virtual RFCs are generated using these virtual control points. However, if one has sufficient GCPs, RFCs can be solved without the help of rigorous sensor model. (Di et al., 2003)

Rigorous approach can be considered as physical modeling of the image acquisition process, including geometrical sensor model (internal orientation elements for the image line), sensor position model (coordinates of the sensor, they are linear elements of external orientation) and sensor attitude (angular elements of external orientation). One can provide the sensor's internal geometry like a set of line-of-sight vectors for detectors of specified numbers in the sensor detector line, or some approximating model like 2D-central projection sensor. The sensor position can be modeled by orbital relations or polynomials for each of three sensor coordinates; the sensor attitude is commonly presented as the sum of on-board measured attitude angles and polynomial refinements determined by the adjustment procedure. It has been observed that high resolution satellites have multiple sensor lines that are artificially combined into a single synthetic array described by the present camera models. Lens and other distortions are modelled by polynomials rather than rigorously, at any given time; it is also not possible to provide orientation for every scan line. Therefore it will not be wrong to say that high resolution camera models can be regarded as empirical approximations and not completely rigorous models. (Fraser et al., 2006)

3.4.2. The generalized sensor models:

The non parametric approach uses the transformation between the image and ground coordinates and is represented by some general functions which do not use the physical imaging process.(Tonolo and Poli, 2003) The replacement models can be considered as an alternative when rigorous models are not available. The generalized sensor model is a kind of approximate sensor model which gives the approximate transformations between the scene and the object coordinates. These generalized sensor models can be considered as generic models for all types of sensors.

This class of models include rational function models, direct linear transform models, 2-D and 3-D affine transformation models etc. These kinds of models are further explained as under in section (3.4.2.1) to (3.4.2.3).

3.4.2.1. The Rational Function Model:

The Rational function model is one of the commonly used Generalised sensor orientation model in today's world. This model became popular as US intelligence community initially started using this model.(OGC, 1999) has already decided to adopt it as a standard image transfer format. With the up come of High resolution satellite images some of the imagery vendors have stopped giving the sensors exterior and interior orientation parameters due to security reasons, instead they provide with generalised sensor models in the form of Rational function models. RFM have smoothed the requirement to obtain physical sensor model parameters.(Fraser and Hanley, 2005)

“The Rational Function Model tries to relates object point coordinates (X, Y, Z) to image pixel coordinates (l, s) or vice visa, as any physical sensor models would do, but in the form of rational functions that are ratios of polynomials”. (Hu et al., 2004) or “RFs perform transformations between the image and object spaces through the ratio of two polynomials”.(Di et al., 2003)The RFM is a generalized sensor model which can be considered as an approximate solution for Rigorous sensor model. For high-resolution satellite images having a small field of view and a high orbit precision, this model provides a good approximation in the geometrical processing. The model uses pair of ratios from the two polynomials as shown in equations 3-10 and 3-11.(Chen et al., 2006; Di et al., 2003; Grodecki and Dial, 2003; Hu et al., 2004)These equations represent the forward transformation from object coordinates to scene coordinates. The RFM model can also do backward transformation from scene coordinates to object coordinates as can be verified from equation 3- 12.(Di et al., 2003; Morgan, 2004) .In general equation 3-12 is called as downward Rational functions and equation 3-10 and equation 3-11 are called as upward rational functions.

$$x = \frac{p_a(X,Y,Z)}{p_b(X,Y,Z)} = \frac{\sum_{i=0}^{m1} \sum_{j=0}^{m2} \sum_{k=0}^{m3} a_{ijk} X^i Y^j Z^k}{\sum_{i=0}^{n1} \sum_{j=0}^{n2} \sum_{k=0}^{n3} b_{ijk} X^i Y^j Z^k}$$

Equation 3-10

$$y = \frac{p_c(X,Y,Z)}{p_d(X,Y,Z)} = \frac{\sum_{i=0}^{m1} \sum_{j=0}^{m2} \sum_{k=0}^{m3} c_{ijk} X^i Y^j Z^k}{\sum_{i=0}^{n1} \sum_{j=0}^{n2} \sum_{k=0}^{n3} d_{ijk} X^i Y^j Z^k}$$

Equation 3-11

$$X = \frac{B_1^r(x, y, Z)}{B_2^r(x, y, Z)}$$

$$Y = \frac{B_3^r(x, y, Z)}{B_4^r(x, y, Z)}$$

Equation 3-12

Where in equations 3- 10 and 3-11 (x, y) are the normalized line (row) and sample (column) index of pixels in image space; (X, Y, Z) are normalized coordinate values of object points in ground space; normalized values are to the range of -1.0 to 1.0 according to their image and geometric extend.(Di et al., 2003; NIMA, 2000) polynomial coefficients a_{ijk} , b_{ijk} , c_{ijk} , d_{ijk} are called rational function coefficients (RFCs). In equation 3-10 $(X \text{ and } Y)$ represent normalised Planimetric object coordinates and (x, y, Z) are image row and column and Z in elevation on ground. The total power of all the ground coordinates in equation 3-10 and 3-11 is usually limited to three. In such a case, each numerator or denominator is of twenty-term cubic form. The polynomial coefficients are also called RPCs, namely, rapid positioning capability, rational polynomial coefficients or rational polynomial camera data. (Hu et al., 2004)

3.4.2.2. Direct Linear Transformations (DLT):

Direct linear transformations can be considered as rigorous sensor model for frame cameras and replacement sensor model for linear array scanners as EOP is different for different images in the scene.(Fraser et al., 2002) In this case there exists the direct relationship between the rigorous sensor model and the Rational function. The collinearity equation (3-2 and 3-3) can be written again, so that the image coordinates are rational functions of the object coordinates or inversely, the interior and exterior orientation parameters can be computed from DLT.(Di et al., 2003)

$$x = \frac{A_1X + A_2Y + A_3Z + A_4}{1 + A_9X + A_{10}Y + A_{11}Z}$$

$$y = \frac{A_5X + A_6Y + A_7Z + A_8}{1 + A_9X + A_{10}Y + A_{11}Z}$$

Equation 3-13

Where A_1, \dots, A_{11} are the DLT parameters. DLT consists of first-degree forward rational functions with common denominators. Therefore, these parameters can be obtained directly (using IOP and EOP) or indirectly (using GCP).

SDLT is a self calibrating DLT that was introduced by (Wang, 2000) as cited in.(Morgan, 2004) Introduces a new parameter A_{12} in the equation representing additional correction to the image coordinates as shown in equation 3-14.(Morgan, 2004)

$$x = \frac{A_1X + A_2Y + A_3Z + A_4}{1 + A_9X + A_{10}Y + A_{11}Z}$$

$$y - A_{12}xy = \frac{A_5X + A_6Y + A_7Z + A_8}{1 + A_9X + A_{10}Y + A_{11}Z}$$

Equation 3-14

The symbolologies for equations are taken from (Morgan, 2004).

3.4.2.3. Two-D Affine Model:

Two-D affine transformation is one of the most studied and used approximation model as it uses less parameters (equation 3-15) It can be used with scanners having narrow AFOV and moving with constant attitude.

$$x = A_1X + A_2Y + A_3Z + A_4$$

$$y = A_5X + A_6Y + A_7Z + A_8$$

Equation 3-15

It would be worth mentioning here that before using the model in Equation, a perspective-to-affine transformation must be applied to transform the scenes from their original state, as a perspective projection, to a 2-D Affine model as the raw scenes obtained from linear array scanners comply with perspective geometry. (Okamoto et al., 1992; Okamoto et al., 1996; Okamoto and Fraser, 1998; Ono et al., 1999; Hattori et al., 2000; Ono et al., 2000).as cited in (Morgan, 2004) .The affine model has been further extended to relief corrected affine models in (Fraser et al., 2002) where the author tries to extend the affine model to reduce height errors of the terrain.It was found in tests involving various Geo images that in affine transformation between the two spaces, with GCPs in different projection systems, the two scale terms get deviated from unity in the fourth or fifth decimal place only. The projection of the GCPs onto a reference plane leads to corrections of the Planimetric X and Y coordinates. “The relief-corrected affine transformation can be used not only for object-to-image transformation and orthoimages generation (Kersten et al., 2000; Baltsavias et al., 2001) as cited in (Fraser et al., 2002), but also for 3D spatial intersection and automatic DSM generation using two or more images, including application to constrained image matching and epipolar resampling.”(Fraser et al., 2002)

3.5. Rational polynomial coefficient model:

“The term - RPC model - often refers to a specific case of the RFM that is in forward form, has third-order polynomials, and is usually solved by the terrain-independent scenario”.(Hu et al., 2004) RPCs are usually provided by imagery vendors of IKONOS, Quick bird, CARTOSAT – 1 etc, so RPCs will be discussed in more detail here. The RPC model utilized for this high resolution satellite provides a transformation from image to object space coordinates in a geographic reference system. Primarily due to numerical conditioning of the estimation process involved in generating RPCs, the actual expressions comprising quotients of two third-order polynomials usually relate normalised (scaled and offset) line and sample coordinates (l_n, s_n) to normalised latitude, longitude and ellipsoidal height (x, y, z):

$$l = l_n L_s + L_o \quad \text{Equation 3-16}$$

$$s = S_n S_s + S_o \quad \text{Equation 3-17}$$

$$x = (\phi - \phi_o) / \phi_s \quad \text{Equation 3-18}$$

$$y = (\lambda - \lambda_o) / \lambda_s \quad \text{Equation 3-19}$$

$$z = (h - h_o) / h_s \quad \text{Equation 3-20}$$

$$l_n = \text{Num}_l(x, y, z) / \text{Den}_l(x, y, z) \quad \text{Equation 3-21}$$

$$s_n = \text{Num}_s(x, y, z) / \text{Den}_s(x, y, z) \quad \text{Equation 3-22}$$

Where

$$\begin{aligned} \text{Num}_l(y, x, z) = & a0 + a1x + a2y + a3z + a4xy + a5xz + a6yz + a7x^2 + a8y^2 + a9z^2 + a10xyz + a11x^3 + a12xy^2 + \\ & a13xz^2 + a14yx^2 + a15y^3 + a16yz^2 + a17x^2z + a18y^2z + a19z^3 \end{aligned} \quad \text{Equation 3-23}$$

$$\begin{aligned} \text{Den}_l(y, x, z) = & b0 + b1x + b2y + b3z + b4xy + b5xz + b6yz + b7x^2 + b8y^2 + b9z^2 + b10xyz + b11x^3 + b12xy^2 + \\ & b13xz^2 + b14yx^2 + b15y^3 + b16yz^2 + b17x^2z + b18y^2z + b19z^3 \end{aligned} \quad \text{Equation 3-24}$$

$$\begin{aligned} \text{Num}_s(y, x, z) = & c0 + c1x + c2y + c3z + c4xy + c5xz + c6yz + c7x^2 + c8y^2 + c9z^2 + c10xyz + c11x^3 + c12xy^2 + \\ & c13xz^2 + c14yx^2 + c15y^3 + c16yz^2 + c17x^2z + c18y^2z + c19z^3 \end{aligned} \quad \text{Equation 3-25}$$

$$\begin{aligned} \text{Den}_s(y, x, z) = & d0 + d1x + d2y + d3z + d4xy + d5xz + d6yz + d7x^2 + d8y^2 + d9z^2 + d10xyz + d11x^3 + d12xy^2 + \\ & d13xz^2 + d14yx^2 + d15y^3 + d16yz^2 + d17x^2z + d18y^2z + d19z^3 \end{aligned} \quad \text{Equation 3-26}$$

In equations 3- 16 to equation 3- 26, l and s are the image line and sample coordinates; l_s and l_o are the line scale and offset terms, and S_s and S_o are the corresponding sample terms; ϕ , λ , h represent

latitude, longitude and height; and φ_s , λ_s and h_s , and φ_o , λ_o and h_o are the corresponding scale and offset terms. The expressions for Num_l , Num_s , Den_l and Den_s are similarly constructed, giving rise to 80 RPC coefficients represented by $a[20]$, $b[20]$, $c[20]$ and $d[20]$.

The symbology used in the equations 3-16 to 3-22 are taken from (Fraser et al., 2006)

The RPC camera model has an advantage that it allows satellite vendors to withhold certain confidential sensor information without denying the public use of the satellite imagery. However, this feature also means that the physical imaging parameters may not be directly adjusted to perform adjustments to the RPC camera model. (Fraser et al., 2006)

3.5.1. Approaches in Determining RPCs:

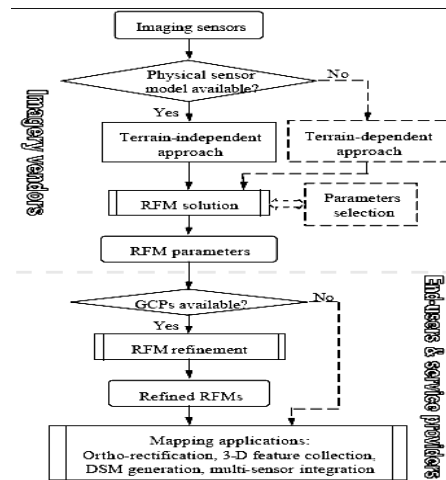


Figure 3-6 RPC Evolution and Generation (Source: Hu et al., 2004)

There are basically two approaches used to determine the RPCs. One approach uses numerous ground control points and does not use the physical parameters. This approach uses plentiful of polynomial terms without establishing the grids, and the solution is highly dependent on the actual terrain relief, the distribution and the number of GCPs. RPC behaves as a rubber-sheeting model, and the over-parameterization may cause the design matrix of the normal equations to become almost rank deficient because of the complex correlations among RPCs. Therefore it may not be used instead of the replacement sensor model if high accuracy is required. This approach is also called as **Terrain dependent approach**. The second approach “utilizes the satellite on-board orientation, which includes orbital parameters and attitude data in generating enough transformation anchor points. This method achieves a high precision under the circumstances that the on-board orbital parameters and attitude data are accurate. As most high resolution satellites are equipped with instruments such as GPS, INS, and star trackers, they are capable of providing satisfactory orientation measurements.” (Chen et al., 2006) The horizontal coordinates (X, Y) of a point of the 3-D object grid are calculated from a point (l, s) of the image grid using the physical sensor model with specified elevation Z. RPCs are estimated using a direct least-squares solution with an input of the object grid points and the image grid points. This approach is also called as **Terrain independent approach**. This terrain-independent

computational scenario makes the RPCs a perfect and safe replacement to the physical sensor models, and has been widely used to determine the RPCs.(Hu et al., 2004).The concept used in Terrain dependent approach using the satellite on-board data to generate transformation anchor points for the determination of RPCs has been demonstrated very nicely in (Grodecki and Dial, 2003)

The concept uses the anchor point generation to determine the image coordinates for a set of multilayer virtual ground points by using satellite orientation parameters. At first the satellite on-board data is utilized to develop the physical sensor model. Afterwards, numerous three dimensional grid points in the object space are generated where they are distributed with different elevations. This helps in generating the corresponding image coordinates for these three-dimensional grid points by using the physical sensor model. The generated points are called virtual anchor points (VAPs). The figure 3-7 shows the modified representation of Grodecki and Dial (2003) for the geometric concept of VAP generation.

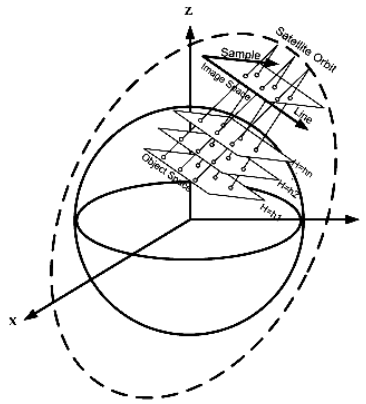


Figure 3-7 Illustration of VAPs generation (source :Chen et al., 2006)

Although the terrain dependent approach is very accurate for generation of RPCs but there may exist errors in the interior and exterior orientation parameters such as position and attitude of the camera. These errors need to be understood so that the errors or biases in the RPCs can be rectified.

3.6. Errors in the Exterior orientation parameters of linear array scanners:

It has been verified that for high resolution satellite systems that the in-track and cross-track position errors are almost completely correlated with pitch and roll attitude errors so that they can not be separately estimated, moreover yaw and radial errors are negligible thus it is merely necessary to estimate roll and pitch.(Grodecki and Dial, 2003) These errors that are embedded in the linear array scanners are briefed up in the following section.

3.6.1. Attitude Errors:

Attitude errors can be considered as the error in the attitude angle of the camera. Attitude angles are roll (rotation about the in-track direction), pitch (rotation about the cross-track direction), and yaw (rotation about the line-of-sight). (Grodecki and Dial, 2003)

3.6.2. Ephemeris Errors:

Ephemeris errors can be conventionally decomposed into in-track and cross-track, radial components, where radial errors are negligible. For narrow FOV cameras small horizontal displacement are equal to small angular rotations. As a result roll errors are completely correlated with cross-track errors. The same is the case for a pitch and in track errors. (Grodecki and Dial, 2003)

3.6.3. Drift errors:

The attitude and ephemeris errors are largely biases, there exists the possibility that these errors would drift as a function of time. For example gyro errors without sufficient compensation from the star trackers could introduce an error in attitude rate, for IKONOS these errors have been found to be small, less than few pixels per 10 km.(Grodecki and Dial, 2003)

3.7. RPC Refinement:

From the above discussion it has been found that there could be errors in the exterior orientation parameters of the linear array scanners. These errors are inherited subsequently in the RPCs generated. So one could say that the generated RPCs could have bias in them which affects the accuracy of the derived image or object coordinates. To remove these biases one needs to refine RPCs.

It has been found that the RPCs can be refined in the domains of image space or object space when additional ground control is available. RPCs can be modified using two ways, one being the direct way and the other being an indirect way. The direct refining methods update the original rational polynomial coefficients and hence, the updated RPCs can be transferred without the need for changing the existing image transfer format. Whereas, the indirect refining introduces complementary or concatenated transformations in the image or object space, and they do not change the original RPCs directly.(Hu et al., 2004) The methods above are explained briefly in the following sections

3.7.1. Direct refinement:

In the direct refinement, the RPCs are computed when both original and a set of additional GCPs collected from the ground are available. The original GCPs refer to those used for collecting the original RPCs and the additional GCPs are those that are used not for collecting the initial set of RPCs. The refinement is carried out incorporating both the original set of GCPs and the additional GCPs into the solution. The values from existing RPCs are used to help to converge the solution. If only the new GCPs are available existing RPCs can be updated using an incremental method based on kalman (Hu and Tao, 2002; Bang et al,2003) as cited in (Hu et al., 2004) filtering or the sequential least square solution. In this technique the RPCs are corrected by adding weighted residuals from new measurements.

3.7.2. Indirect refinement:

Indirect refinement methods have been tested and used with various models, (Bianconi et al., 2007; Chen et al., 2006; Cho et al., 2003; Fraser et al., 2006; Fraser and Hanley, 2003; Grodecki and Dial, 2003; Tonolo and Poli, 2003) all have used indirect refinement method in modelling with terrain independent RPCs. To refine a simple forward RFM it needs to append a simple complementary transformation in the image space at the right side of the equation 3-21 and 3-22 to eliminate the error source. (Grodecki and Dial, 2003) has proposed the comprehensive block adjustment math model in which two complementary polynomials that are adjustable to model the effects originated from the uncertainty of spacecraft telemetry and geometric properties of the IKONOS sensor. The first-order polynomials Δl and Δs are defined by

$$\Delta l = l' - l = a_0 + a_l \cdot l + a_s \cdot s \quad \text{Equation 3-27}$$

$$\Delta s = s' - s = b_0 + b_l \cdot l + b_s \cdot s \quad \text{Equation 3-28}$$

The symbology for the equations have been taken from (Hu et al., 2004)

Where $(\Delta l, \Delta s)$ express the discrepancies between the measured line and sample coordinates (l', s') and the RFM projected coordinates (l, s) of a GCP or tie point; the coefficients $a_0, a_l, a_s, b_0, b_l, b_s$ are the adjustment parameters for each image. He indicated that each of the polynomial coefficients has physical significance for IKONOS products, and thus the model which he named it as RPC-BA does not present the numerical instability problem. In detail, the constant a_0 (b_0) absorbs all in track (cross-track) errors causing offsets in the line (sample) direction, including in-track (along-track) ephemeris errors satellite pitch (roll) attitude error, and the line (sample) component of principal point and detector position error.

The correction vector to the approximate values of both the model parameters and the object point coordinates is given by Equation

$$\Delta x = ((A^T C w^{-1} A)^{-1} A^T C w^{-1} w) \quad \text{Equation 3-29}$$

Where \mathbf{A} is the design matrix of the block adjustment equations; w is the vector of misclosures for model parameters; Cw is the covariance matrix. It also introduces an extra parameter (eg a polynomial) in either an image space or the object space and improves the accuracy by fitting the RPC measured coordinates which have the coordinates calculated by the GCPs. The adjustable functions Δx and Δy are typically, polynomials of the image coordinates, x and y . Such methods are known to be particularly applicable to a system with a narrow field of view, imaging a relatively flat area.

(Fraser and Hanley, 2003) develops a simple method for removal of bias from the IKONOS imagery where from equation 3-21 and 3-22 effectively, all original terms in the numerator of each expression are modified. In the case of the x - equation, each coefficient a_k is replaced by $(a_k - b_k \Delta x_i)$. Where Δx_i and Δy_i are the bias corrections added into original RPCs. This is shown with the help of equation 3-30

$$\begin{aligned} \text{Num} &= (a_1 - b_1 \Delta x_i) + (a_2 - b_2 \Delta x_i) \cdot Y \\ &+ (a_3 - b_3 \Delta x_i) \cdot X + \dots + (a_{20} - b_{20} \Delta x_i) \cdot Z^3 \\ \text{Den} &= (c_1 - d_1 \Delta y_i) + (c_2 - d_2 \Delta y_i) \cdot Y \\ &+ (c_3 - d_3 \Delta y_i) \cdot X + \dots + (c_{20} - d_{20} \Delta y_i) \cdot Z^3 \end{aligned} \quad \text{Equation 3-30}$$

(Fraser et al., 2006) also refines the RPCs using indirect technique where a practical bias compensation approach is used by adding the adjustment parameter as shown below

$$l + A_0 + A_1l + A_2s = \frac{\text{Num}_L(U, V, W)}{\text{Den}_L(U, V, W)} L_s + L_0$$

$$s + B_0 + B_1l + B_2s = \frac{\text{Num}_S(U, V, W)}{\text{Den}_S(U, V, W)} S_s + S_0$$

Equation 3-31

Here, the parameters A_i , B_i describe an affine distortion of the image. There are essentially three choices for ‘additional parameter’ sets for bias compensation:

- i) A_0, A_1, \dots, B_2 , which describe an affine transformation,
- ii) A_0, A_1, B_0, B_1 , which model shift and drift, and
- iii) A_0, B_0 which represent image coordinate translations.

In another approach, a method of adjusting the object space coordinates that involves the addition of an adjustment term to each of the three object space coordinates. Each adjustment term is approximated by a cubic polynomial function of the object space coordinates. The adjustment method requires determination, and optimization, of some 60 coefficients of the cubic polynomial functions (Grodecki and Dial, 2003).

3.8. A general brief up of Literature review on RPCs

General literature reviewed is been briefed up in this section. Here is the description of some of the related work that has been referred to, for the current project:

- (Fraser et al., 2002) did a lot of work and tried to refine RPCs using DLT and affine models, he also extended the affine transformation model to relief corrected affine transformation model. The results achieved can be summarized as follows: For control configurations of four, six and eight GCPs and 20–25 checkpoints, the affine model produced RMS 3D positioning accuracies of 0.3–0.5 m in XY and 0.5–0.7 m in Z, for both stereo and three-image geometries. It was however observed that Accuracy deteriorated only slightly with decreasing numbers of control points, while use of three instead of two images improved the results, especially in height. For the same number of control points, the affine model yielded slightly better results than the RPC-based spatial intersection. For the same GCP sets, the DLT yielded slightly lower 3D positioning accuracy of 0.3–0.6 m in XY and 0.5–0.9 m in Z, and it was found to exhibit stability problems for certain GCP configurations.
- (Ian and Tao, 2002) presents an article which reviewed the developments in the use of RPC models since 2000. It concludes stating that a prime driving force to use RPC is the decision

of space imaging not to release the IKONOS sensor model, but to provide the information necessary to restitution the data in the form of RPC. It also believes that RF model can be considered as an effective representation of the rigorous sensor model. The major findings of the paper are based on the review which can be described as follows:

- a) The paper acknowledges the work of Toa and his co-workers, where he says that they have carried out a lot of research in testing RPCs with different formulations, mainly with aerial photography and spot and have concluded in the paper Tao and HU (2001a, 2001b, 2001c) cited in(Ian and Tao, 2002) that RPCs can give a very high accuracy in the case of aerial photographs and spot data in terrain independent case. RPCs with unequal denominators often give better results, and in addition, it depends on the accuracy and placement of the GCPs. Tao (2001a) cited in (Ian and Tao, 2002) proposes computation methods for improvement of the numerical instability of the solution.
 - b) The RPCs for the GEO product and Precision products for IKONOS have been analysed and the paper concludes that GEO products which are expected to produce a RMS positioning accuracy of about 25m, are derived solely from satellite ephemeris and attitude data, whereas those for Precision products are computed with the additional aid of ground control. The work done by Hanley and Fraser (2001) as cited in(Ian and Tao, 2002) tested IKONOS Geo product by first projecting the control points onto ‘planes of control’, to minimize the effect of terrain, and then transform the image to these points using Similarity, affine and projective transformations. The results show that 0.3-0.5m geopositioning accuracy is achievable from the Geo product without using the rational function solution.”
 - c) The paper talks about the work of (Grodecki and Dial, (2001)) as cited in (Ian and Tao, 2002) where he reports with IKONOS that tests with 140 ground control points gave horizontal accuracy of the order of 1m, while vertical accuracy was of the order of 2m’ from a controlled stereo pair over San Diego.
 - d) Fraser et al (2002a) as cited in(Ian and Tao, 2002) These tests conclude that IKONOS gives z vertical accuracy of 0.83- 0.90 meters with 4 GCPs (less than a meter). And 0.89-0.90 meters with one GCP.
- (Di et al., 2003) enlightens in the conclusion of his paper that the rigorous sensor model has the explicit physical sensor parameters that can be used efficiently for calibration, debugging algorithms, improving computational efficiency by separating correlated parameters. He states that The RF coefficients are scene specific because IO and EO parameters are merged together and they use the control points. He conducted the experiments on three data sets one is aerial image and corresponding DEM, taken along Ohio lake Erie shore, in 1977 Second data constituted two stereo pairs of one meter resolution of IKONOS geo stereo images, Third data is asset of HRSC (High resolution stereo camera image).He conducted three experiments in which the first tried to generate RF coefficients using DLT model, In the second experiment he tried to refine the geopositioning accuracy of IKONOS geo stereo image using two methods, it was found that first method which used parameters to refine RPCs used more GCPs for refinement then the ones which refine RPCs through object space . Accuracy of 1.9 to 0.9 meters was achieved using 6 to 10 GCPs. He also gives the related work done on the RF models in the past years.
 - (Grodecki and Gene, 2003) This paper describes how to block adjust high-resolution satellite imagery described by Rational Polynomial Coefficient (RPC) camera models and illustrates the method with an example [3.7.2]. By incorporating priori constraints into the adjustment model,

multiple independent images can be adjusted with or without ground control. In the experiment it was found that average error without GCPs were -5.0, 6.2 and 1.6 meters in longitude, latitude, and height. The addition of one GCP reduces the average error to -2.4, 0.5, 1.1 meters. While additional ground controls further reduce the average error and the standard deviation remains virtually unchanged at 1 meter in longitudes and latitudes and 2 meters in height. This experiment was done in IKONOS images with six stereo strips and a large number of well distributed GCPs.

- (Fraser and Hanley, 2003) A method for the removal of exterior orientation biases in rational function coefficients (RPCs) for IKONOS imagery is developed and presented in this paper. []. The Two factors account for the biases being essentially equal in the stereo image. First, recorded exterior orientation is unlikely to undergo significant change within the 100 or so seconds between the recordings of the two along-track images and, second, stereo images are routinely subjected to initial bundle adjustment. From the experiments it was found that only bias corrected RPCs of IKONOS is capable of generating sub-pixel accuracies on the 3D plain and with just one good ground control point.
- (Cho et al., 2003) proposes two methods for generation of DEM with IKONOS using modified RPCs. He proposes two of the methods, First is using Pseudo GCPs in Normalized Cubic and the second is DEM generation using Epipolar Image. In the conclusion he makes a statement that addition of 5 GCPs reduces the error by 50% in case of IKONOS.
- (Toutin, 2003) did a exhaustive study on error tracking of IKONOS geo images with 3D parametric model and he found that When ground control points (GCPs) have an accuracy poorer than 3 m, 20 GCPs over the entire image are a good compromise to obtain a 3- to 4-m accuracy in the bundle adjustment. When GCP accuracy is better than 1 m, ten GCPs are enough to decrease the bundle adjustment error of either panchromatic or multiband images to 2 to 3 m.
- (Hu et al., 2004) gives the whole overview of RFM and its concepts concludes by saying that the “Photogrammetrists have overcome restrictions placed on the use of the data by vendors using RFM refining methods, which ensure that the exploitation results are as accurate as what can be achieved using physical sensor models, and are also economical using low price products.” He further states that the RFM provides an open standard for photogrammetric interoperability, is not dependent on particular sensors, and is extensible for block adjustment. He concludes saying that in future, RFM will be considered as an important key tool for obtaining the relationship between ground coordinates and image coordinates.
- (Fraser and Hanley, 2005) This paper talks about the impressive geopositioning accuracy attained with the RPC bundle adjustment with bias compensation supports the view that this sensor orientation model has the same metric potential as rigorous model formulations for both IKONOS and Quickbird imagery, Implicit in this conclusion is that the RPCs produced by space Imaging , Inc., and Digital globe, are equivalent to the rigorous model and thus there should be no concern regarding their applicability in stereo imagery covering any type of terrain.
- (Fraser et al., 2006) describes in conclusion that aim of the paper has been essentially two folded, firstly it gives an overview of the RPC model for HRSI, and secondly it tries to highlight both the

practicability and accuracy potential of RPC block adjustment. It has been demonstrated that bias compensated RPC block adjustment can yield sub pixel accuracy, which for the case of both IKONOS and Quick bird imagery, translates to a sub-meter geopositioning capability. Moreover, the attainment of such accuracy via bias-corrected RPCs in subsequent orthoimages and DSM generation, and in feature extraction, is possible with lower-cost image products.

- (Kumar, 2006) first analysed the DEM generated from the CARTOSAT-1 stereo image over Dehradun and says that while using only RPC information for CARTOSAT –1 stereo data, the error in height was in the range 100 to 200m .While using eight GCP's for CARTOSAT –1 stereo data, the error was reduced to 2 to 13m. It was found that accuracy of contours generated from CARTOSAT –1 stereo data was very accurate and close to ground height.
- (Nandakumar and srivastava, 2006) A joint summary of the The ISPRS-ISRO CARTOSAT -1 Scientific Assessment Programme (C-SAP) include 10 test sites, consisting of one or two stereo pairs from CARTOSAT-1 along with the reference data sets, out of which the summary of 5 test sites was provided in the paper. Most of the summary contained the accuracies achieved with the DEM using CARTOSAT-1..
- (Chen et al., 2006) In this investigation, he provides a least squares collocation procedure for the RSM and RFM. The study compares the RSM and RFM error of the RFM. Experiment was conducted on FORMOSAT-2 satellite images and results indicate that the RSM performs just slightly better than the RFM in this study. For a standard scene, the model error between the physical sensor model and the rational function model is 0.01 pixels. For a strip with 15 scenes, the model error increases to 0.15 pixels.

3.9. Summary

This chapter gave us an insight into photogrammetric processing with linear array scanners. The theory behind replacement sensors and rigorous sensors was also discussed in details. It can be said that various models for refining the RPCs have been proposed for scanners usually with narrow FOV. However most of them have mostly been analysed for IKONOS sensors .There have been a lacking to test a model which works on both CARTOSAT-1 stereo images and IKONOS stereo. Moreover there are no models which work directly on the RPCs with out using priory constraints like weighted covariance matrix. A simple model that can be considered valid for systems with a large field of view and suitable to moderately hilly terrain conditions with respect to CARTOSAT-1 are yet to be standardized, and this project aims to develop a solution of this kind.

4. Concepts and methodology

4.1. Overall Research Methodology

The methodology in the research is designed in order to answer the research questions, and has been adapted based on the formulation of the research questions, literature review, and the concept building. The overall methodology can be understood by the following flow chart or diagram given below

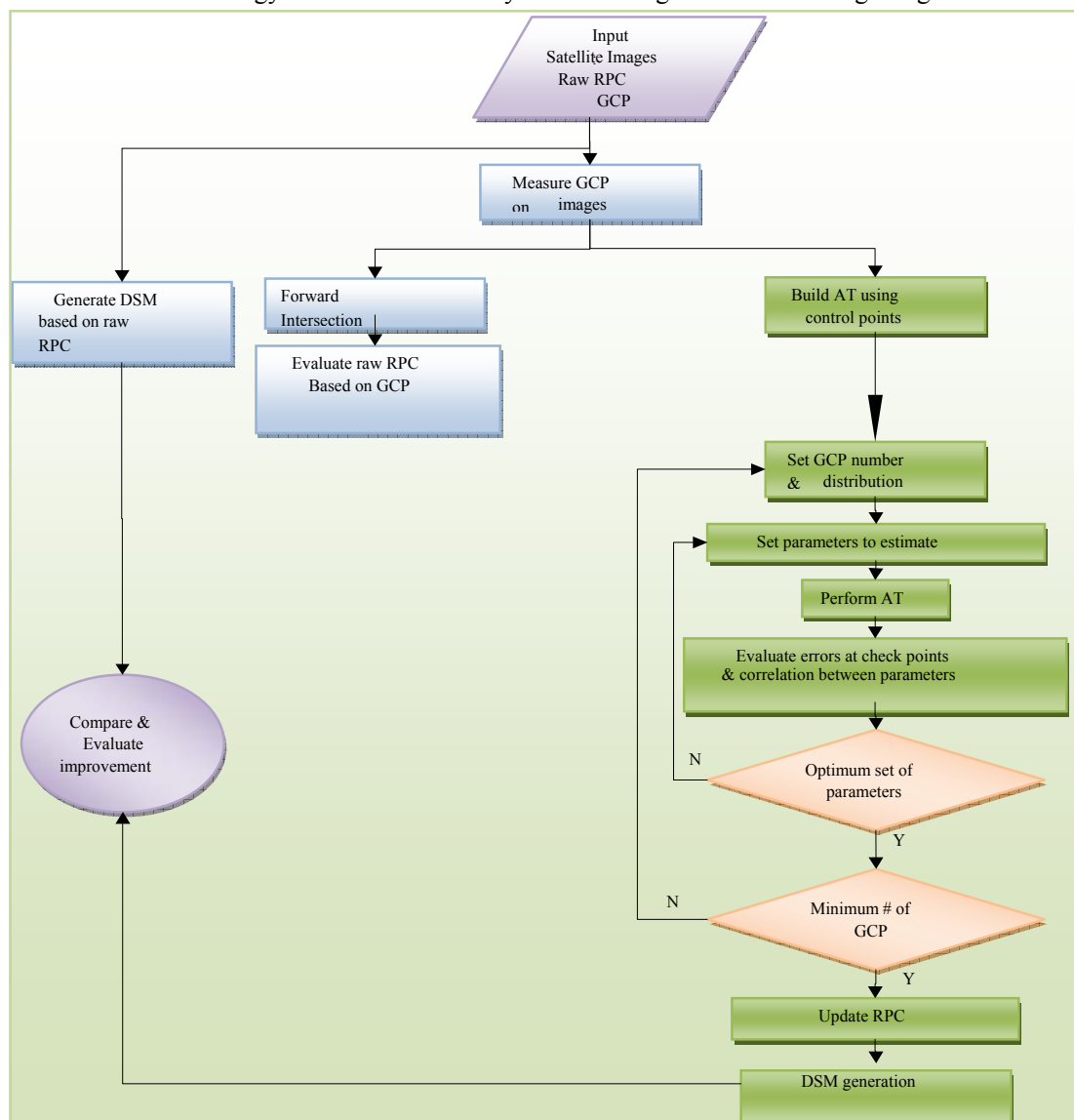


Figure 4-1 the overall methodology flowchart

The methodology used focuses on achieving the main objective of the research, i.e. to find the best set of parameters with least number of GCPs for refining the RPCs provided by the vendors. This method will focus on adding the approximate parameters directly into the RPCs that will be fitted with image coordinates in the domain of known object coordinates. This model developed will not be using any weighted priority constraints to achieve numerical stability. The model will be tested with both CARTOSAT-1 data and IKONOS data. The over all approach to achieve this model is described by overall flow chart in the Figure-1. This flow chart can be briefed up as under.

- Step1: Input the Satellite data with the given RPCs file, this has to be accompanied by ancillary data of ground control points.(GCPs)
- Step2: GCPs need to be measured on the satellite images to get image coordinates of the selected ground control points.
- Step3: Evaluating raw RPC based on measured GCP and forward intersection.
- Step 4: Building Aerial Triangulation using control points.
- Step 5: Set different parameters in the model to change the original RPC.
- Step 4: Use least squares adjustment to estimate parameters.
- Step 5: Analyze the improvement based on checkpoint analysis.
- Step 6: Find the best set of parameters to minimize the errors at the check points.
- Step 7: Change number of GCP to obtain the min requirement of GCP to estimate the parameters.
- Step 8: Generate DTM from Raw RPC and Corrected RPC.
- Step 9: Compare DTM generated using raw RPC & corrected RPC.
- Step10: Draw conclusions using statistical approach.

The over approach is further divided into separate sections so as to help in making the task simpler. These subdivisions can be shown the following flow chart:

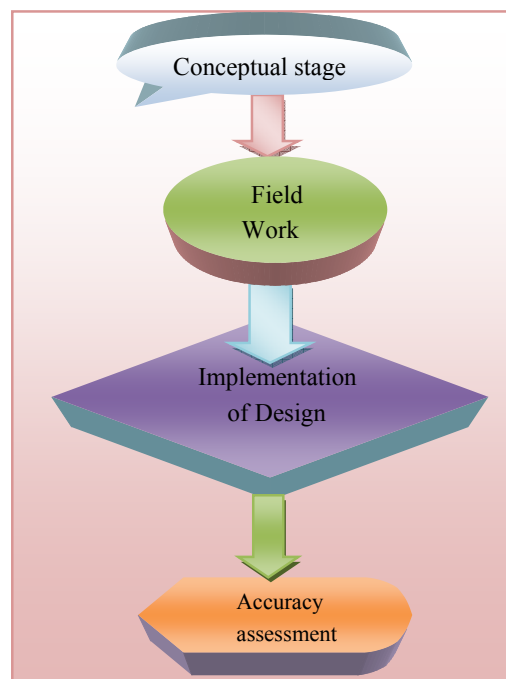


Figure 4-2 Methodology subdivision

The figure 4-2 explains how the research has been carried out starting from the conceptual stage where concept building was done through discussions and literature review, Field work was done for collecting the GCPs, implementation of the mathematical design is done by making a tool using java programming language and lastly accuracy assessment is done using statistical analysis. They are discussed in more details in the following section

4.1.1. Conceptual stage

Conceptual stage can be further subdivided into components as described below.

4.1.1.1. Deciding upon the research area and the sensors

The study is carried out in areas with varied topography so as to observe the requirement of GCPs for RPC refinement in varied terrain types, and also to study the positional accuracies achieved. Two test sites are considered, Chandigarh with a plain terrain and Dehradun which has moderate topography. Along with it data from two sensors IKONOS and CARTOSAT -1 have been chosen for the study in order to test the robustness of the program.

It is observed from literature review, that the refining methods that have been attempted so far do not show valid results in the vertical accuracy in hilly regions, and are mostly used with narrow FOV and flat area. Although there has been considerable amount of work in refinement of RPCs in the past, very less has been done in improving the accuracy of the CARTOSAT – 1 RPCs in the prescribed methods. Therefore both the sensors are being used to test the models developed for refining the RPCs for improvements in their respective positional accuracies.

4.1.1.2. Deciding the programming language and the software

JAVA language has been chosen for the programming because of its user friendly syntax and the platform independence. This option was considered as most of the mathematical calculations and operations that had to be performed during the research required a language which was user friendly and has to have an ability to be used with any mathematical and statistical background such as matlab etc.

Matlab has been chosen for statistical analysis. Leica photogrammetric suite helped in extraction of DTMs.

During the process of implementation an attempt was also made to consider matlab for programming the design. It was found that it had inbuilt functions for fitting and partial derivation, however it could not process all the rational polynomials in one solution, as matlab needed more time to understand and learn, java was considered one of the better options.

4.1.1.3. Developing a mathematical model

The literature is reviewed and discussion helped in building the frame work for the model, the literature reviewed can be summarized as follows which lead to developing a model at last.

The physical imaging parameters that accompany the remote sensing imagery may include orbital data, sensor (camera) attitude data, focal length, data timing etc. These physical imaging parameters are unique to each satellite and each sensor and are useful in producing a rigorous camera model. In

Other words, the camera model is used to relate object-space (ground) coordinates to image-space coordinates. One of the physical camera models is the one that relates a point in the object space coordinate system to the image space coordinate system using equations 3-22 and 3-23. In these equations *Num* and *Den* are the functions that, relates both the ground coordinates and the image coordinates and this function depends on the physical imaging parameters. Nowadays, in satellite based sensors most of the imaging parameters are well measured however there may exist a bias for one or more physical imaging parameters due to the small residual error. This may lead to corresponding errors in the image-space coordinates and, as a consequence, errors are produced in the image using the image-space coordinates. These images which have some error in the exterior and interior orientation parameters produce errors in the overall rigorous models, thus spreading the same to the RPC models. The geometric accuracy of the camera sensor can be improved by attempting to reduce the residual errors and biases in the model, in particular, with the use of ground control points (GCPs). The use of GCPs involves knowledge of object-space coordinates as well as the image-space coordinates. The known object-space coordinates may be processed by a given sensor model to produce a pair of Calculated image-space coordinates. A difference between the calculated pair of image-space coordinates and the known pair of image-space coordinates may then be used to adjust the imaging parameters. This can be attempted by using known mathematical algorithm known as the least squares.

The new model developed takes the original set of coefficients from the RPC model and adds new adjustable functions ΔNum and ΔDen for both line and sample to the normal equations based on the known ground control points. These new adjustable functions are iteratively solved in a single solution for both the images and fitted with the set of new image coordinates which are actually the differences between determined and actual image coordinates.

The new adjustable functions can be defined as follows:

$$\Delta Num_l = A0 + A1x + A2y + A3z + A4xy + A5xz + A6yz + A7x^2 + A8y^2 + A9z^2 + A10xyz + A11x^3 + A12xy^2 + A13xz^2 + A14yx^2 + A15y^3 + A16yz^2 + A17x^2z + A18y^2z + A19z^3$$

Equation 4-1

$$\Delta Den_l = B0 + B1x + B2y + B3z + B4xy + B5xz + B6yz + B7x^2 + B8y^2 + B9z^2 + B10xyz + B11x^3 + B12xy^2 + B13xz^2 + B14yx^2 + B15y^3 + B16yz^2 + B17x^2z + B18y^2z + B19z^3$$

Equation 4-2

$$\Delta Num_s = C0 + C1x + C2y + C3z + C4xy + C5xz + C6yz + C7x^2 + C8y^2 + C9z^2 + C10xyz + C11x^3 + C12xy^2 + C13xz^2 + C14yx^2 + C15y^3 + C16yz^2 + C17x^2z + C18y^2z + C19z^3$$

Equation 4-3

$$\Delta Den_s = D0 + D1x + D2y + D3z + D4xy + D5xz + D6yz + D7x^2 + D8y^2 + D9z^2 + D10xyz + D11x^3 + D12xy^2 + D13xz^2 + D14yx^2 + D15y^3 + D16yz^2 + D17x^2z + D18y^2z + D19z^3$$

Equation 4-4

In the equation 4-1 to 4-4 the terms from $A0 - A19$ are adjustment coefficients for numerator line and $B0-B19$ are adjustment coefficients for denominator line, $C0-C19$ are adjustment coefficients for numerator sample and $D0-D19$ are adjustment coefficients for denominator sample. x, y, z are the normalized latitude, longitude and height of the GCP points

To refine the RPCs the adjustment function of defined polynomial order ($1^{st}, 2^{nd}, 3^{rd}$) are added on to rational functions to capture the difference between the original and calculated object coordinates.

In the model developed the selection of adjustment coefficient is known as parameter selection.

So if only 1 parameter is selected for refinement the adjustment function is of 1^{st} order polynomial and only $A0, C0$ adjustment coefficients are as $B0$ and $D0$ are always constant 1, similarly if 2 parameters are selected for refinement then adjustment function contains $A0+A1x, B0+B1x, C0+C1x$, and $D0 + D1x$.

This can be summarized as follows

$$1 \text{ Parameter} = \Delta Num_l = A0 \text{ and } \Delta Num_s = C0. \quad \text{Equation 4-5}$$

$$2 \text{ Parameters} = \Delta Num_l = A0+A1x, \Delta Den_l = B0+B1x \text{ and } \Delta Num_s = C0+C1x, \Delta Den_s = D0 + D1x. \quad \text{Equation 4-6}$$

$$3 \text{ Parameters} = \Delta Num_l = A0+A1x+A2y, \Delta Den_l = B0+B1x+B2y \text{ and } \Delta Num_s = C0+C1x+C2y, \Delta Den_s = D0 + D1x+D2y. \quad \text{Equation 4-7}$$

$$4 \text{ Parameters} = \Delta Num_l = A0+A1x+A2y+A3z, \Delta Den_l = B0+B1x+B2y+B3z \text{ and } \Delta Num_s = C0+C1x+C2y+C3z, \Delta Den_s = D0 + D1x+D2y+D3z. \quad \text{Equation 4-8}$$

The addition of adjustment function to the rational polynomial coefficient can be shown in equation 4-9

$$l_n = Num_l(x, y, z) + \Delta Num_l / Den_l(x, y, z) + \Delta Den_l \quad \text{Equation 4-9}$$

$$s_n = Num_s(x, y, z) + \Delta Num_s / Den_s(x, y, z) + \Delta Den_s \quad \text{Equation 4-10}$$

The equations are then fitted using least squares technique and the solutions for the adjustment coefficients are added to the original coefficients to get new refined RPCs. The fitting of with respect to least square solution involves generation of jacobian matrices which is further explained in detail in the implementation section of this Chapter.

4.1.2. Field work:

GCP or ground control point can be considered as the main component for establishing an accurate relationship between the images in a project, the camera/sensor, and the ground. GCPs are taken at identifiable features located on the Earth's surface that have known ground coordinates in X, Y, and Z. A full GCP has X,Y, and Z (elevation of the point) coordinates associated with it. Horizontal control only specifies the X,Y, while vertical control only specifies the Z. The following criteria are followed in acquiring GCPs:

- 1) GCP is identified as a point, which should be clearly identified and positioned on the image data as well as corresponding map.
- 2) GCP collected should be permanent in nature.
- 3) The surrounding regions of GCPs should have good contrast which helps in precisely positioning the point on the image. Some of the landmarks on the Earth's surface are commonly used as GCPs:
 - Intersection of roads
 - Utility infrastructure (e.g., fire hydrants and manhole covers)
 - Intersection of agricultural plots of land(Erdas, 2005)

Differential GPS (DGPS) was used for collecting the GCPs and approximately 12 GCPs were collected for each scene from Chandigarh and 20 for Dehradun area

4.1.3. Implementation of Designed mathematical model:

This section of the methodology is most important for meeting the objectives, as this is a practical approach to get the results from the mathematical framework that has been built during the conceptual stage. This part can act as a single entity which can solve all the problems and can help in answering all the research questions. The following section gives a detailed description of Implementation of the programming the model.

4.1.3.1. Software Building

A program that can read Raw RPC files and object and image coordinate files from the user and give modified RPC file and object coordinates file from the modified RPC file was developed. This software could be realized with the help of other programming modules that were provided to me by my guide Dr. Michel Morgan, ITC, The Netherlands. These programming modules were integrated into my software. The modules that were provided are as follows:

- A program which can read image space coordinates and return output coordinates for the object space, using the RPC file.(Source: Dr. Michel Morgan, ITC, The Netherlands)
- A program which can read object space coordinates and return output image space coordinates using the RPC file. .(Source: Dr. Michel Morgan, ITC, The Netherlands)
- A program that can Denormalize and normalize the RPCs.(Source: Dr. Michel Morgan, ITC, The Netherlands)

The process of developing this program for modification of RPC can be shown by the following figure 4-3 flow chart which is divided into three sections of input, process and the output shown by

different colours. The program has been designed with the help of java jar file known as Jama matrix. This interface helped me to manipulate the matrices and the inverse of the given matrix could be found rather easily. The main class used in the software built has been given in the Annexure 2. The three main subdivisions of the flow chart are explained in detail in the following subsections.

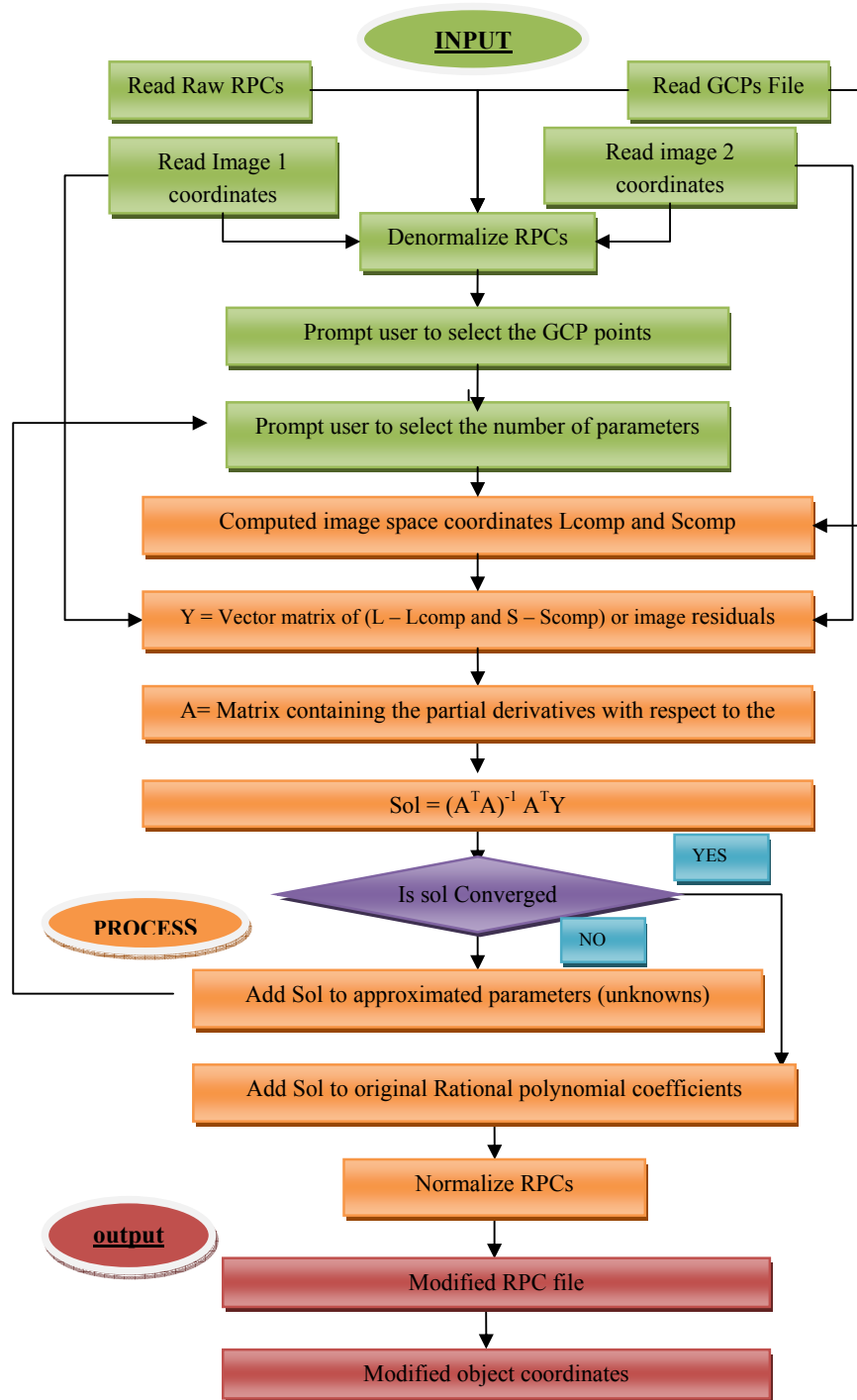


Figure 4-3 Flow chart showing the implementation of the Design

- **Input**

Rational polynomial coefficient files that are given with imagery are in an encrypted format, and are not readable as normal text files. To read the rational polynomial coefficients, the encrypted files need to be decrypted and this can be done by any decryption module found in the current image processing software's. ENVI software has been used for decryption in this research. The structure of the decrypted RPC file can be shown as under.

```

LINE_OFF: +006007.83 pixels
SAMP_OFF: +005951.38 pixels
LAT_OFF: +30.35749018 degrees
LONG_OFF: +077.91678371 degrees
HEIGHT_OFF: +1250.000 meters
LINE_SCALE: +009529.21 pixels
SAMP_SCALE: +009446.12 pixels
LAT_SCALE: +00.18000000 degrees
LONG_SCALE: +000.20750000 degrees
HEIGHT_SCALE: +1050.000 meters

-----
LINE_NUM_COEFF_1: +5.014335863333104E-04
LINE_NUM_COEFF_2: -1.692991268843592E-01
- ----- so on till 20
- -----
LINE_NUM_COEFF_19: -8.060223904726890E-01
LINE_NUM_COEFF_20: +2.390334023116225E-02

-----
LINE_DEN_COEFF_1: +1.000000000000000E+00
LINE_DEN_COEFF_2: +5.937890572162863E-02
----- so on till 20
- -----
LINE_DEN_COEFF_19: +1.079434112980258E+00
LINE_DEN_COEFF_20: -2.833003174792646E-02

-----
SAMP_NUM_COEFF_1: +7.603640811098092E-06
SAMP_NUM_COEFF_2: +8.168390114322921E-01
----- so on till 20
- -----
SAMP_NUM_COEFF_19: -1.797850895099866E-01
SAMP_NUM_COEFF_20: +8.402393972301402E-05

-----
SAMP_DEN_COEFF_1: +1.000000000000000E+00
SAMP_DEN_COEFF_2: +5.167958132858572E-02
----- so on till 20
- -----
SAMP_DEN_COEFF_19: +3.396394424394623E-02
SAMP_DEN_COEFF_20: -9.236263935756101E-03

```

Figure 4-4 Sample for Normalized RPCs

There are the linear scaling factors for latitude, longitude and height which are represented by *LAT_SCALE*, *LONG_SCALE* and *HEIGHT_SCALE*, respectively, and the linear translation factors for latitude, longitude and height which are represented by *LAT_OFF*, *LONG_OFF* and *HEIGHT_OFF*, respectively. Both these factors are used for converting the actual object space coordinates to normalized object space coordinates which the RPCs work on. They are given at the starting of the RPC file.

However the *LINE offset*, *SAMP offset* and *LINE scale*, *SAMP scale* together are used for converting the normalized image space coordinates produced by RPC to original image space coordinates. The *LINE_NUM_COEFFs*, *LINE_DEN_COEFFs*, *SAMP_NUM_COEFFs* and *SAMP_DEN_COEFFs* are the coefficient values which are given with scale and translation factors.

The raw RPC file shown here is in the normalised form or in other terms the values of the coefficients lie between -1 and +1, and the normalised form of RPCs is due to the linear scaling and translating; the *line_offset* and *Sample_offset*; *line_scale*, and *Sample_scale* values which all are given in the first 10 rows of the file.

To work with the given RPC file which is in normalised form one would require to either convert the given Ground coordinate point values to the normalised form or to denormalise the RPCs themselves. The normalization of the given control points can be done by using the equation mentioned below

$$\begin{aligned} Y_{normalized} &= Y_{actual} - LAT_OFF / LAT_SCALE \\ X_{normalized} &= L_{actual} - LONG_OFF / LONG_SCALE \text{ and} \\ Z_{normalized} &= Z_{actual} - HEIGHT_OFF / HEIGHT_SCALE. \text{(Grodecki and Dial, 2003)} \end{aligned}$$

Now the changed normalised image coordinates can be converted to original denormalised coordinates for further use by using the equation

$$\begin{aligned} L_{actual} &= (L_{normalized} * SAMP_SCALE) + SAMP_OFF. \\ S_{actual} &= (S_{normalized} * LINE_SCALE) + LINE_OFF. \text{(Grodecki and Dial, 2003)} \end{aligned}$$

Denormalization of the RPC can be done by converting all the scaling factors to 1 and translating factors to 0. Now the changed Denormalised coefficients do not lie between 0 and 1 but tend to restore their original values.

The structure of the file with changed values looks as follows:

```

LINE_OFF: 0
SAMP_OFF: 0
LAT_OFF: 0
LONG_OFF: 0
HEIGHT_OFF: +0
LINE_SCALE: 1
SAMP_SCALE: 1
LAT_SCALE: 1
LONG_SCALE: 1
HEIGHT_SCALE: 1

-----
l_NUM_COEFF_1: 1497503.8142787197
l_NUM_COEFF_2: -15248.219505371002
- ----- so on till 20
- -----
l_NUM_COEFF_19: -3.3823001933765842e-005
l_NUM_COEFF_20: 1.477641483422664e-018

-----
l_DEN_COEFF_1: 1
l_DEN_COEFF_2: -0.011962136750771638
----- so on till 20
- -----
l_DEN_COEFF_19: -4.0150110890966339e-013
l_DEN_COEFF_20: -2.4732389881224347e-023

-----
-

s_NUM_COEFF_1: -5097474.0226296801
s_NUM_COEFF_2: 198714.54370315871
----- so on till 20
- -----
s_NUM_COEFF_19: -4.6047867484073298e-006
s_NUM_COEFF_20: -3.7138149815600061e-016

-----
_s_DEN_COEFF_1: 1
s_DEN_COEFF_2: -0.023360597031369151
----- so on till 20
- -----
s_DEN_COEFF_19: 1.9516513814650265e-011
s_DEN_COEFF_20: -6.3304304058876586e-020

```

Figure 4-5 Sample for Denormalized RPCs

Both Normalized RPCs and denormalised RPCs were used initially and were found to increase complexity and redundancy in program designing and increased time for processing so only denormalised RPCs were chosen for the program.

There are 80 coefficients which are used in the given RPC files which have been broken down into 4 arrays of size 20 each for use in the program designing. They are described as follows:

Image 1:

$$\begin{aligned} \text{LINE_NUM_COEFF_}[20] &= a[20] \\ \text{LINE_DEN_COEFF_}[20] &= b[20] \\ \text{SAMP_NUM_COEFF_}[20] &= c[20] \\ \text{SAMP_DEN_COEFF_}[20] &= d[20] \end{aligned} \quad \text{Equation 4-11}$$

As the stereo pair is used for the process of refinement the other image RPC coefficient values are also assigned in the same manner to 4 more arrays of size 20 each

Image 2:

$$\begin{aligned} \text{LINE_NUM_COEFF_}[20] &= a1[20] \\ \text{LINE_DEN_COEFF_}[20] &= b1[20] \\ \text{SAMP_NUM_COEFF_}[20] &= c1[20] \\ \text{SAMP_DEN_COEFF_}[20] &= d1[20] \end{aligned} \quad \text{Equation 4-12}$$

The other input files are the GCP file and image coordinate files. The GCP file contains the true X , Y , Z coordinate values taken from the ground in degree, decimal and their corresponding IDs on the image. The image coordinate file contains the L and S coordinate values in ASCII which depicts the corresponding row and column values in the image with respect to the GCP IDs. The program reads the user given file names with *.txt extensions for image coordinates and GCP files and * .RPC extension for RPC files. Then it denormalise the Raw RPC files and reads the corresponding polynomial coefficients.

Having read all the files, the program prompts the user to enter the IDs of the GCPs and the number of parameters to estimate. The parameter defines the model type as explained in equation 4-5 to 4-8. and hence the immediate input for the program would be to set the number of parameters. If the user enters the number of parameters as 1 then the program assigns 2 different array variables, for each image. The array size depends upon the number of parameters selected by the user as shown below

$Aim1[1]$, $Aim2[1]$, $Cim1[1]$, $Cim2[1]$.

Here $im1$ in $Aim1$ and $Cim1$ represents image1 and $im2$ in $Aim1$ and $Cim1$ represents image 2. The $A[]$, $C[]$ are the adjustable coefficients.

Similarly if the user enters number of parameters as 2 then the program assigns 4 different array variables for each image, and the array size depends upon the number of parameters. And, these four

new parameters are added to the already existing parameters and the total count of the total number of approximations becomes 12.

$$Aim1[1], Aim2[1], Cim1[1], Cim2[1] + Aim1[2], Bim1[2], Cim1[2], Dim1[2], Aim2[2], Bim2[2], Cim2[2], Dim2[2].$$

Equation 4-13

The program then, prompts the user with a choice of self assigning the initial approximations for all the adjustable coefficients which gives the user liberty to assign the initial approximation to parameters manually or accept the approximation value as 0.0001 assigned automatically by the program.

- **Process**

In the process of RPC refinement, least squares technique is used during triangulation to estimate or adjust the input parameters so as to adjust interior and exterior orientation.

The residuals associated with the input data have to be minimized for an accurate solution and this requires an iterative processing through the least squares approach. Determining the corrections to the unknown parameters based on the criteria of minimizing input measurement residuals is possible through the least squares adjustment. It is possible to derive the residuals then, from the difference between the measured (i.e., user input) and the computed value for any particular measurement in a project.

In the process of block triangulation, a functional model can be formed based upon the collinearity equations. The functional model refers to the specification of an equation that can be used to relate measurements to parameters. The residuals, which are minimized, include the image coordinates of the GCPs along with the known ground coordinates of the GCPs, also the residuals of the parameters added to the collinearity equation. A simplified version of the least squares condition can be broken down into a formulation as follows:(Erdas, 2005)

$$V = AX - L, \text{ including a weight matrix } P.$$

Equation 4-14

Where:

V = the matrix containing the image coordinate residuals

A = the matrix containing the partial derivatives with respect to the unknown parameters, including exterior orientation, interior orientation, XYZ tie point, and GCP coordinates

X = the matrix containing the corrections to the unknown parameters

L = the matrix containing the input observations (i.e., image coordinates and GCP coordinates)

The symbology used in equation 4-14 and 4-15 are taken from(Erdas, 2005)

The A matrix is formed by differentiating the functional model, which is based on collinearity equations, with respect to the unknown parameters which can be added to the observation equations for removing the exterior orientation errors and to find out x , y and z coordinates of tie points , etc. The L matrix is formed by subtracting the true values with results obtained from the functional model

with newly added approximations. The solution matrix or X matrix contains the corrections to the unknown parameters or approximations. The X matrix is calculated in the following manner:

$$X = (A^T P A)^{-1} = A^T P L \quad \text{Equation 4-15}$$

Where:

X = the matrix containing the corrections to the unknown parameters t

A = the matrix containing the partial derivatives with respect to the unknown parameters

t = the matrix transposed

P = the matrix containing the weights of the observations

L = the matrix containing the observations

After completion of the least squares iteration, the corrections to the unknown parameters are added to the initial estimates. This iterative process of least square continues until a the residual values are converged to a specific value which can be a value which is very close to zero so that the values in the solution matrix do not change much. The difference between the initial measurements and the new estimates is obtained to provide the residuals. Residuals provide preliminary indications of the accuracy of a solution. The residual values indicate the degree to which a particular observation (input) fits with the functional model..

To be more precise with the getting the solution matrix X can be taken out as follows:

The Normal Equations are obtained as follows:

$$A^T A P X = A^T P L \quad [AX = L + V] \quad \text{Equation 4-16}$$

Where

$A^T A = N$ is normal equation coefficients.

P is the weight matrix.

And V is the residual.

Multiply both sides by $(A^T A)^{-1}$

$$(A^T A)^{-1} (A^T A) X = (A^T A)^{-1} A^T L \quad \text{Equation 4-17}$$

$$IX = (A^T A)^{-1} A^T L \quad \text{Equation 4-18}$$

$$X = N^{-1} A^T L \quad \text{Equation 4-19}$$

➤ **Structure of L matrix**

The process of getting the L matrix involves getting the computed image coordinates from the rational polynomial equations having the added adjustment functions. These adjustment functions are added at the end of the polynomial equations. The computed image coordinates can be derived from the following observation equations for each of the ground control points.

For image 1:

$$l_{computed(img1)} = \frac{Num_{imag1}(x, y, z) + \Delta Num_{imag1}}{Den_{imag1}(x, y, z) + \Delta Den_{imag1}}$$

Equation 4-20

OR

$$a0+a1x+a2y+a3z+a4xy+a5xz+a6yz+a7x^2+a8y^2+a9z^2+a10xyz+a11x^3+a12xy^2+a13x+a14yx^2+a15y^3+a16yz^2+a17x^2z+a18y^2z+a19z^3+Aim1[0]+Aim1[1]x+.....Aim1[no. of Parameters]$$

Equation 4-21

$$b0+b1x+b2y+b3z+b4xy+b5xz+b6yz+b7x^2+b8y^2+b9z^2+b10xyz+b11x^3+b12xy^2+b13x+b14yx^2+b15y^3+b16yz^2+b17x^2z+b18y^2z+b19z^3 + Bim1[0]+ Bim1[1]x+.....Bim1[no. of parameters]$$

Equation 4-22

$$l_{computed(img1)} = \text{Equation 4-21/Equation 4-22}$$

$$S_{computed(img1)} = \frac{Num_{simagel}(x, y, z) + \Delta Num_{simagel}}{Den_{simagel}(x, y, z) + \Delta Den_{simagel}}$$

Equation 4-23

OR

$$c0+c1x+c2y+c3z+c4xy+c5xz+c6yz+c7x^2+c8y^2+c9z^2+c10xyz+c11x^3+c12xy^2+c13xz^2+c14yx^2+c15y^3+c16yz^2+c17x^2z+c18y^2z+c19z^3 + Cim1[0]+ Cim1[1]x+.....Cim1[no. Of parameters]$$

Equation 4-24

$$d0+d1x+d2y+d3z+d4xy+d5xz+d6yz+d7x^2+d8y^2+d9z^2+d10xyz+d11x^3+d12xy^2+d13x+d14yx^2+d15y^3+d16yz^2+d17x^2z+d18y^2z+d19z^3 + Dim1 [0] +Dim1 [1] x+.....Dim1 [no. Of parameters]$$

Equation 4-25

$$S_{computed(img1)} = \text{Equation 4-24/Equation 4-25}$$

For image 2:

$$l_{computed(img2)} = \frac{Num_{imag2}(x, y, z) + \Delta Num_{imag2}}{Den_{imag2}(x, y, z) + \Delta Den_{imag2}}$$

Equation 4-26

OR

$$a10+a11x+a12y+a13z+a14xy+a15xz+a16yz+a17x^2+a18y^2+a19z^2+a110xyz+a111x^3+a112xy^2+a113xz^2+a114yx^2+a115y^3+a116yz^2+a117x^2z+a118y^2z+a119z^3+Aim2[0]+Aim2[1]x+Aim2[2]y+..... Aim2[no. of Parameters]$$

Equation 4-27

$$b10+b11x+b12y+b13z+b14xy+b15xz+b16yz+b17x2+b18y2+b19z2+b110xyz+b111x3+b112xy2+b113xz2+b114yx2+b115y3+b116yz2+b117x2z+b118y2z+b119z3+Bim2[0]+Bim2[1]x+Bim2[2]y..... Bim2[no. of Parameters]$$

Equation 4-28

$$l_{computed}(img2) = Equation\ 4-27 / Equation\ 4-28$$

$$S_{computed}(img2) = \frac{Num_{simagel}(x,y,z) + \Delta Num_{simage2}}{Den_{simage2}(x,y,z) + \Delta Den_{simage2}}$$

Equation 4-29

OR

$$c10+c11x+c12y+c13z+c14xy+c15xz+c16yz+c17x2+c18y2+c19z2+c110xyz+c111x3+c112xy2+c113xz2+c114yx2+c115y3+c116yz2+c117x2z+c118y2z+c119z3 + Cim2[0] + Cim2[1]..... Cim2[no. Of parameters]$$

Equation 4-30

$$d10+d11x+d12y+d13z+d14xy+d15xz+d16yz+d17x2+d18y2+d19z2+d110xyz+d111x3+d112xy2+d113xz2+d114yx2+d115y3+d116yz2+d117x2z+d118y2z+d119z3 + Dim2[0] + Dim2[1]..... Dim2[no. Of parameters]$$

Equation 4-31

$$S_{computed}(img2) = Equation\ 4-30 / Equation\ 4-31$$

The terms used in the above equations, hold their original meaning as they are described in the previous .equations (4-5 to 4-10) Once the image coordinates are computed from the Polynomial equations, they are used for making a vector matrix L of length equal to (4 * number of GCPs) input by the user. L matrix size changes when we consider Tie points also for the refinement, as it adds four rows more for each tie point that we take into L matrix.

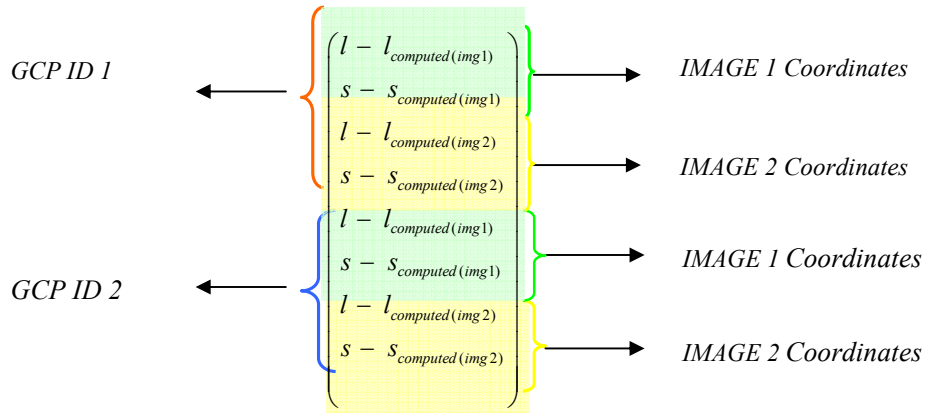


Figure 4-6 L matrix using 2 GCPs

Where

L is the matrix of 2 GCPs

l = Row of image

$l_{computed}$ = taken from the equation 4-20 and 4-22 (Computed Row)

$S_{computed}$ =calculated from the equation 4-21and 4-23(computed column)

➤ **Structure of A matrix (jacobian matrix)**

A matrix is obtained using the partial derivatives of the computed image space coordinates with respect to the unknown parameters input by the user. The size of A matrix depends on the number of GCPs used and the number of parameters entered by the user. The size of the A matrix using GCPs is equal to

$$A = [Row, column] = [(gpsIDs.size() * 4, (((nrParams - 1) * 8) + 4))]$$

Equation 4-32

The size of the matrix A changes when considering Tie points also for the adjustment and becomes to

$$A = [Row, Column] = [(gpsIDs.size() * 4 + TieIDs * 4, (((nrParams - 1) * 8) + 4) + TieIDs * 3)]$$

Equation 4-33

Where

$nrParams$: Stands for number of parameters input by the user or in other terms model definition.

$gpsIDs.size$: Tells us about the number of GCPs selected by the user.

$TieIDs$. Size: Tells us about the number of tie points selected by the user.

The structure of (A Jacobian matrix) is one of the most important aspects of the research as a lot depends on how the variables containing values are placed in the matrix and the order in which they are placed. The convergence of the solution matrix and finding the accurate values depend on the placement of the values in the matrix.

The design of the A matrix can be shown by considering some simple examples as follows

$$A = \begin{pmatrix} \begin{array}{cc|cc} \text{Image} & & \text{Image 2} & \\ \downarrow & & \downarrow & \\ \frac{\partial l_{computed}(img1)}{\partial Aim1[1]} & 0 & 0 & 0 \\ 0 & \frac{\partial S_{computed}(img1)}{\partial Cim1[1]} & 0 & 0 \\ 0 & 0 & \frac{\partial l_{computed}(img2)}{\partial Aim2[1]} & 0 \\ 0 & 0 & 0 & \frac{\partial S_{computed}(img2)}{\partial Cim2[1]} \end{array} \end{pmatrix}$$

Figure 4-7 the structure of matrix for 1 GCP and 1 parameters.

Structure of the matrix changes as we move from 1 parameter to higher parameters as it adds 8 columns to the matrix with each parameter.

$$A = \begin{matrix} \begin{matrix} \text{Image 1} \\ \downarrow \end{matrix} & \begin{matrix} \text{Image 2} \\ \downarrow \end{matrix} \\ \begin{pmatrix} \frac{\partial l_{com(img1)}}{\partial Aim1[1]} & 0 & \frac{\partial l_{com(img1)}}{\partial Aim1[2]} & \frac{\partial l_{com(img1)}}{\partial Bim1[2]} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial s_{com(img1)}}{\partial Cim1[1]} & 0 & 0 & \frac{\partial s_{com(img1)}}{\partial Cim1[2]} & \frac{\partial s_{com(img1)}}{\partial Dim1[2]} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial l_{com(img2)}}{\partial Aim2[1]} & 0 & \frac{\partial l_{com(img2)}}{\partial Aim2[2]} & \frac{\partial l_{com(img2)}}{\partial Bim2[2]} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial s_{com(img2)}}{\partial Cim2[1]} & 0 & 0 & \frac{\partial s_{com(img2)}}{\partial Cim2[2]} & \frac{\partial s_{com(img2)}}{\partial Dim2[2]} \\ \frac{\partial l_{com(img1)}}{\partial Aim1[1]} & 0 & \frac{\partial l_{com(img1)}}{\partial Aim1[2]} & \frac{\partial l_{com(img1)}}{\partial Bim1[2]} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{\partial s_{com(img1)}}{\partial Cim1[1]} & 0 & 0 & \frac{\partial s_{com(img1)}}{\partial Cim1[2]} & \frac{\partial s_{com(img1)}}{\partial Dim1[2]} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial l_{com(img2)}}{\partial Aim2[1]} & 0 & \frac{\partial l_{com(img2)}}{\partial Aim2[2]} & \frac{\partial l_{com(img2)}}{\partial Bim2[2]} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{\partial s_{com(img2)}}{\partial Cim2[1]} & 0 & 0 & \frac{\partial s_{com(img2)}}{\partial Cim2[2]} & \frac{\partial s_{com(img2)}}{\partial Dim2[2]} \end{pmatrix} \end{matrix}$$

Figure 4-8 the structure of matrix with 2 GCPs and 2 Parameters

Similarly if we want to add tie points to the equation we have to add (3 * number of Tiepoints taken) columns to the matrix and (4 * number of tie points) as a row in the given matrix.

Where

$$\frac{\partial l_{com(img1)}}{\partial Aim1[1]} = \text{Corresponds to partial differentiation with respect to first adjustable coefficient}$$

Variable of image 1 = Aim1 [1]

Similarly the other terms in the matrix correspond to the partial derivative of the equation in the numerator with respect to denominator.

This structure of A matrix changes as we try to use tie points for their refinement as each tie point adds 4 rows and 3 columns to the matrix

$$\begin{array}{c}
 \text{Image 1} \quad \quad \quad \text{Image 2} \quad \quad \quad \text{Tie point} \\
 \downarrow \quad \quad \quad \downarrow \quad \quad \quad \downarrow \\
 A = \begin{pmatrix}
 \frac{\partial l_{\text{computed}(\text{img1})}}{\partial \text{Aim}[1]} & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & \frac{\partial s_{\text{computed}(\text{img1})}}{\partial \text{Cim}[1]} & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & \frac{\partial l_{\text{computed}(\text{img2})}}{\partial \text{Aim2}[1]} & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & \frac{\partial s_{\text{computed}(\text{img2})}}{\partial \text{Cim2}[1]} & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & \frac{\partial l_{\text{computed}(\text{img1})}}{\partial X} & \frac{\partial l_{\text{computed}(\text{img1})}}{\partial Y} & \frac{\partial l_{\text{computed}(\text{img1})}}{\partial Z} \\
 0 & 0 & 0 & 0 & \frac{\partial s_{\text{computed}(\text{img1})}}{\partial X} & \frac{\partial s_{\text{computed}(\text{img1})}}{\partial Y} & \frac{\partial s_{\text{computed}(\text{img1})}}{\partial Z} \\
 0 & 0 & 0 & 0 & \frac{\partial l_{\text{computed}(\text{img2})}}{\partial X} & \frac{\partial l_{\text{computed}(\text{img2})}}{\partial Y} & \frac{\partial l_{\text{computed}(\text{img2})}}{\partial Z} \\
 0 & 0 & 0 & 0 & \frac{\partial s_{\text{computed}(\text{img2})}}{\partial X} & \frac{\partial s_{\text{computed}(\text{img2})}}{\partial Y} & \frac{\partial s_{\text{computed}(\text{img2})}}{\partial Z}
 \end{pmatrix}
 \end{array}$$

Figure 4-9 the structure of the matrix with 1 GCP, 1 Tiepoint and 1 parameter.

Various methods have been tried out in the program to get the partial derivatives like approximate or numeric derivation which uses the first law of differentiation and symbolic differentiation which uses the regular differentiation. It has however been found that there is convergence problem in the solution matrix, if we are using the first law of differentiation whereas; it tends to converge faster using symbolic differentiation.

The final solution matrix or **X** matrix is obtained using the formula in equation 4-19:

$$X = N^{-1}A^T L$$

Where

$$N = A^T A.$$

After obtaining the final solution matrix which is a linear vector matrix, the values in the solution matrix are used for replacing the initial approximation for the unknown parameters input by the user. The new estimates for the unknown parameters are replaced to recompute the input observations such as the image coordinate values, iteratively until we get to a solution matrix which is either stable, and do not change its value more than a certain threshold. After the convergence occurs and we have a stable solution matrix, the values of the solution matrix is added to original rational polynomial coefficient values given by 4 different arrays such as a [], b [], c [], and d [] given for both images. The final replaced RPC values are in denormalised form and have to be converted into normalised form and this also is done by the normalization program.

The refined RPC are used with the image coordinates files to get to the object coordinates in the 3D plain, x, y, z this is done by image to object .exe(source DR. Michel Morgan, ITC), this gives us the true value of the selected image points on the ground.

4.1.4. Accuracy analysis:

Accuracy analysis is done using matlab-7 software; this helped in plotting graphs for errors, and to extract spatial and Planimetric accuracies produced by the models. The result of the analysis is a Root mean square error (RMSE) that defines the degree of correspondence between the computed values and the original values. Lower RMSE values indicate better results.

The Planimetric and vertical accuracies are derived by finding the difference between the original object coordinates and the object coordinates computed from the modified RPCs. For finding the spatial and Planimetric accuracy following formulas given by (Tiegum, 2005) are used:

$$\text{RMSE in X} = mx = \sqrt{\frac{1}{n} \sum_{i=1}^n (x_{orig} - x_{comp})^2} \quad \text{Equation 4-34}$$

$$\text{RMSE in Y} = my = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_{orig} - y_{comp})^2} \quad \text{Equation 4-35}$$

$$\text{RMSE in Z} = mz = \sqrt{\frac{1}{n} \sum_{i=1}^n (z_{orig} - z_{comp})^2} \quad \text{Equation 4-36}$$

$$\text{Planimetric or Horizontal RMSE} = \sqrt{mx^2 + my^2} \quad \text{Equation 4-37}$$

$$\text{Vertical accuracy} = mz \quad \text{Equation 4-38}$$

Spatial RMSE is also extracted here with the help of following formula

$$\text{Spatial accuracy} = \sqrt{mx^2 + my^2 + mz^2} \quad \text{Equation 4-39}$$

Where

$(x_{orig} - x_{comp})$ = Error with respect to latitude.

$(y_{orig} - y_{comp})$ = Error with respect to longitude.

$(z_{orig} - z_{comp})$ = Error with respect to Height.

x_{orig} = Original X with respect to ground.

y_{orig} = Original Y with respect to ground.

z_{orig} = Original H with respect to ground.

x_{comp} = Computed X with RPCs.

y_{comp} = Computed Y with RPCs.

z_{comp} = Computed Z with RPCs.

n = Number of GCPs and checkpoints.

Accuracy at the check points are also extracted by considering only check points and leaving out the GCPs.

DEMs are extracted from both original and modified RPCs using Leica photogrametric suite. Accuracy of DEMs is shown with the help of point maps and surface profiles.

5. Results and discussion

Having reviewed the model design and the program to execute the techniques, implementation of the designed tool is now, discussed in this chapter. The accuracy of the RPCs provided by the imagery vendors prior to their modification of any sort is extracted initially. Various models for improving the accuracies in the 3D object space are then analyzed through a set of experiments. Henceforth, results are discussed with a mention of the conclusions drawn.

5.1. Analysis of Raw RPCs:

The horizontal and vertical accuracy for RPCs provided with the image can be well shown by the accuracy of object coordinates achieved from the RPCs and the real object coordinates. The horizontal and vertical accuracy is given by root mean square error derived from equation 4-37 and equation 4-38. This is shown for both IKONOS and CARTOSAT-1.

5.1.1. Accuracy for CARTOSAT-1 Raw RPC:

Planimetric RMSE in meters = 6.5943 meters.

Vertical RMSE in meters = 43.7632 meters

The object coordinates files for CARTOSAT-1 image is shown as below

gps3dn - Notepad				objecttest - Notepad			
File	Edit	Format	View	File	Edit	Format	View
1	78.023801	30.387136	636.70619	1	78.023769270464015	30.387507577362307	592.79054264681429
2	78.011012	30.332567	617.6461	2	78.010989789049489	30.332947364114757	572.16235990901237
3	77.956639	30.343857	534.1943	3	77.956615557418544	30.344245655223382	490.04481434447968
4	77.940106	30.358743	552.916	4	77.940077724940096	30.359132434958539	509.31579011531454
5	77.887624	30.35265	506.0046	5	77.88760109385052	30.353043083093734	464.04755224124011
6	77.92127	30.332088	503.260894	6	77.921246975443211	30.332476756161252	458.77795430432008
7	77.960692	30.323632	553.995648	7	77.960671217415239	30.324020610295539	508.8036010348892
8	77.854437	30.366415	485.954387	8	77.854410362252182	30.366799854294165	443.13005426799839
9	77.871299	30.374272	508.5352	9	77.871267503061617	30.374661460040333	465.71561229953284
10	77.870238	30.394477	541.511857	10	77.870204917368284	30.394869776568488	500.12781088476743
11	77.807424	30.477496	445.8504	11	77.80737961041828	30.477872287217501	407.37695569772228
12	77.934634	30.405835	702.180421	12	77.934598742060814	30.40621359558833	660.78836181421116
15	77.792324	30.42163	433.8881	15	77.792223855380371	30.422024793800627	391.93165272149025
21	77.937924	30.405956	715.1322	21	77.937910341156368	30.406365510634682	668.46215532813346
22	77.929537	30.439885	891.4877	22	77.929497618781127	30.440311464152003	844.82086370833611
23	77.778655	30.375042	437.1802	23	77.778603548763925	30.375474412017049	399.13426679310589
25	77.892973	30.320927	493.2815	25	77.892921003353308	30.321330184944497	447.9688426560092

a) GPS object coordinates

b) Object coordinates by RPCs

Figure 5-1 Object coordinates of CARTOSAT-1

The columns in the above given figure 5-1 represent the Ground point IDs, latitude, longitude and height respectively. The latitude and the longitude are given in degree minutes and the height is represented in meters. The degree, minutes values are converted into meters and are represented with the graphs below, in the graphs the Planimetric error is scaled to 1000.

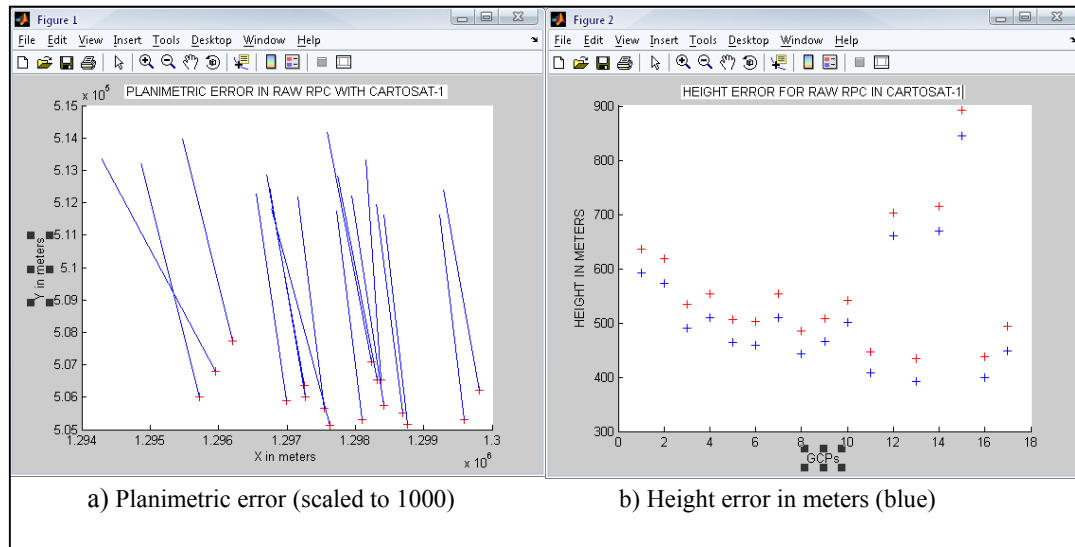


Figure 5-2 Graphs for a) Planimetric errors b) Horizontal errors of Raw RPCs in CARTOSAT-1

From figure 5-2 it can be analysed that CARTOSAT-1 scene of moderate terrain, has a height error, represented in the graph- b by the shift of blue cross from red cross. A constant shift of 40 – 45 meters is observed for every point which can be considered as very high for a high resolution sensor. The errors are systematic in this case as most of the GCPs which are mostly collected from the valley and only one GCP ID 21 was collected from the elevated area. The tests with other scenes of CARTOSAT-1 have shown that this error can go as high as 100-150 m and they were not systematic. The RPCs provided by CARTOSAT-1 uses WGS84 ellipsoid and datum and the collected GCPs use the same projection system.

5.1.2. Accuracy for IKONOS Raw RPC:

Planimetric RMSE in meters = 1.0166 meters

Vertical RMSE in meters = 5.7012 meters

The object coordinates files for CARTOSAT-1 image is shown as below

ch_gps - Notepad			
File	Edit	Format	View Help
1	76.730341	30.687432	266.180300
2	76.817337	30.642578	254.713800
3	76.837075	30.708984	291.105400
4	76.822530	30.701427	282.981800
5	76.752392	30.712114	276.265500
6	76.762007	30.691261	269.462300
7	76.725502	30.665538	258.574300
8	76.802911	30.650991	258.425500
9	76.828356	30.685108	273.623500
10	76.802795	30.682701	272.748600
11	76.842054	30.690738	281.560000
12	76.794549	30.720408	288.399800

a) GPS object coordinates

objecttest - Notepad			
File	Edit	Format	View Help
1	76.730326323217099	30.687372262559137	271.549
2	76.817324409115699	30.642518638161981	260.095
3	76.837061181205058	30.708924898042991	296.902
4	76.822516316462867	30.701365733437818	287.652
5	76.75237758812483	30.712057810707059	282.246
6	76.761993584851709	30.691201540767544	274.836
7	76.725487120984951	30.665479539865981	263.456
8	76.802895637533155	30.650929138925807	263.376
9	76.828339124945018	30.685047255980141	278.730
10	76.802782412870073	30.682643518874762	278.432
11	76.84204282639152	30.690676544970092	286.947
12	76.794539793343844	30.720349778359125	294.497

b) Object coordinates by RPCs

Figure 5-3 Object coordinates for IKONOS

Figure 5-3 shows the real object coordinates from the ground and measured object coordinates.

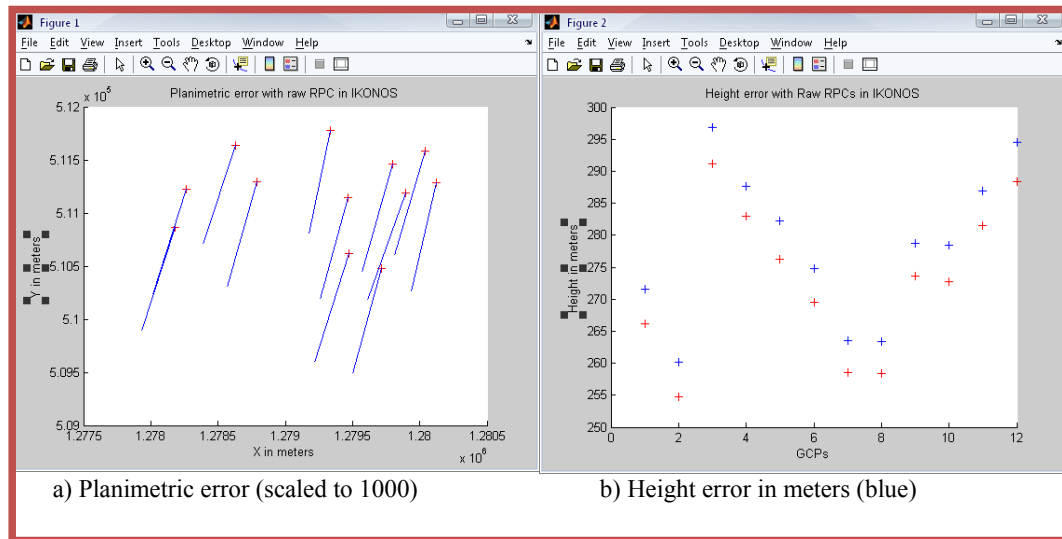


Figure 5-4 Graphs for a) Planimetric errors b) Height errors of IKONOS Raw RPCs.

In case of IKONOS data as from figure 5-4 the RPCs have better quality and raw RPCs generate less error compared to CARTOSAT-1. IKONOS also has an advantage of plain area as compared to moderate with CARTOSAT-1.

5.1.3. GCP Distribution

The position of GCP IDs on the image can be shown over the block file with LPS software; the following are figures of the block files for both CARTOSAT and IKONOS.

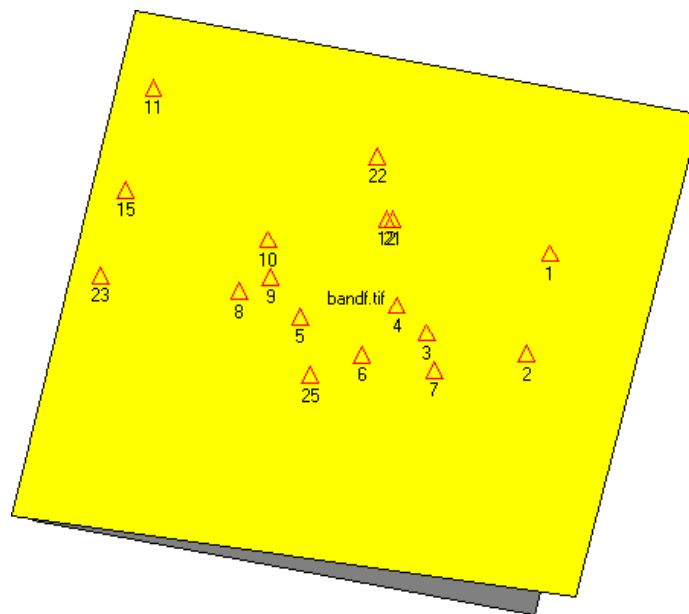


Figure 5-5 Block file showing position of CARTOSAT-1 GCPs

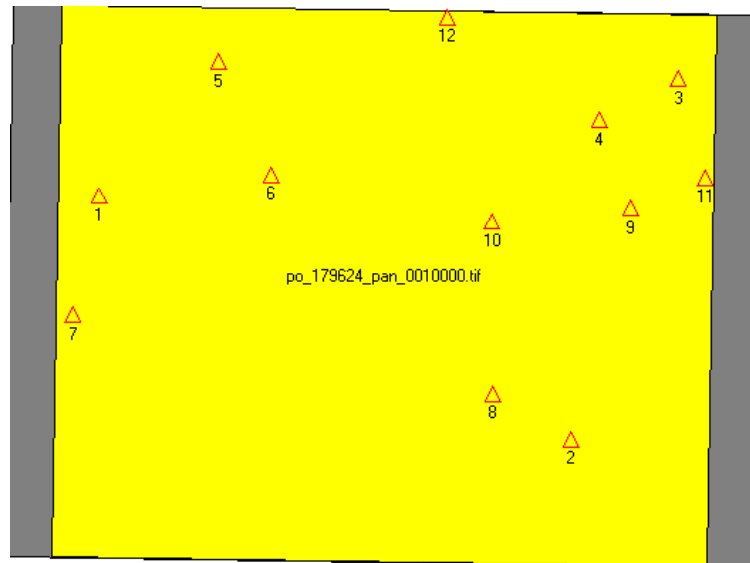


Figure 5-6 Block file showing Positions of IKONOS GCPs

It can be well seen from figure 5-5 and 5-6 that the GCPs are taken in well distributed manner however most of the GCPs taken in the Dehradun area are from the central part as the areas around are covered by high mountains.

5.2. Experiments with models

Different combinations of the parameters were tried to find the optimum least numbers of GCPs required for achieving greater accuracy in the 3D object plain. Experiments were conducted in both sets of data and in different terrain conditions. The experiments conducted and there results are shown in the following section of this chapter.

5.2.1. Modelling with one parameter

The assessment for orientation was done using one parameter and different number of GCPs on both the images. If 1 parameter is considered for improvement it reflects only the removal of bias from the RPCs. Only $a[1]$, $c[1]$ or shift parameters are modified for both the images as $b[1]$ and $d[1]$ are taken as constant value 1 which is not changed.

5.2.1.1. Using CARTOSAT - 1 with distributed GCPs

CARTOSAT- 1 was oriented with 1 parameter using different number of GCPs which were distributed across the scene; their positions can be shown by their IDs in the block file given. Finding the error between the original object coordinates and the derived object coordinate checked Planimetric and spatial accuracies, there are 20 GCPs collected in the area out of which 17 were found to be good .The table 5-1 below shows the results achieved with the same.

No. of GCP	No of Chk Points	Planimetric RMSE in meters	Spatial RMSE in meters	RMSE X in meters	RMSE Y in meters	RMSE Z in meters
17	0	1.3966	29.7763	0.5062	1.3016	29.7436
15	2	1.3922	30.4489	0.4902	1.3031	30.4170
13	4	1.4239	29.3638	0.4936	1.3357	29.3292
10	7	1.5431	32.7993	0.5761	1.4315	32.7630
7	10	1.6636	34.8013	0.6126	1.5467	34.7615
4	13	1.4683	31.5577	0.5852	1.4321	31.5235
1	16	1.7420	34.3194	0.6804	1.6036	34.2752

Table 5-1 Over all accuracy with GCPs and Check points of CARTOSAT-1 orientation through one parameter model.

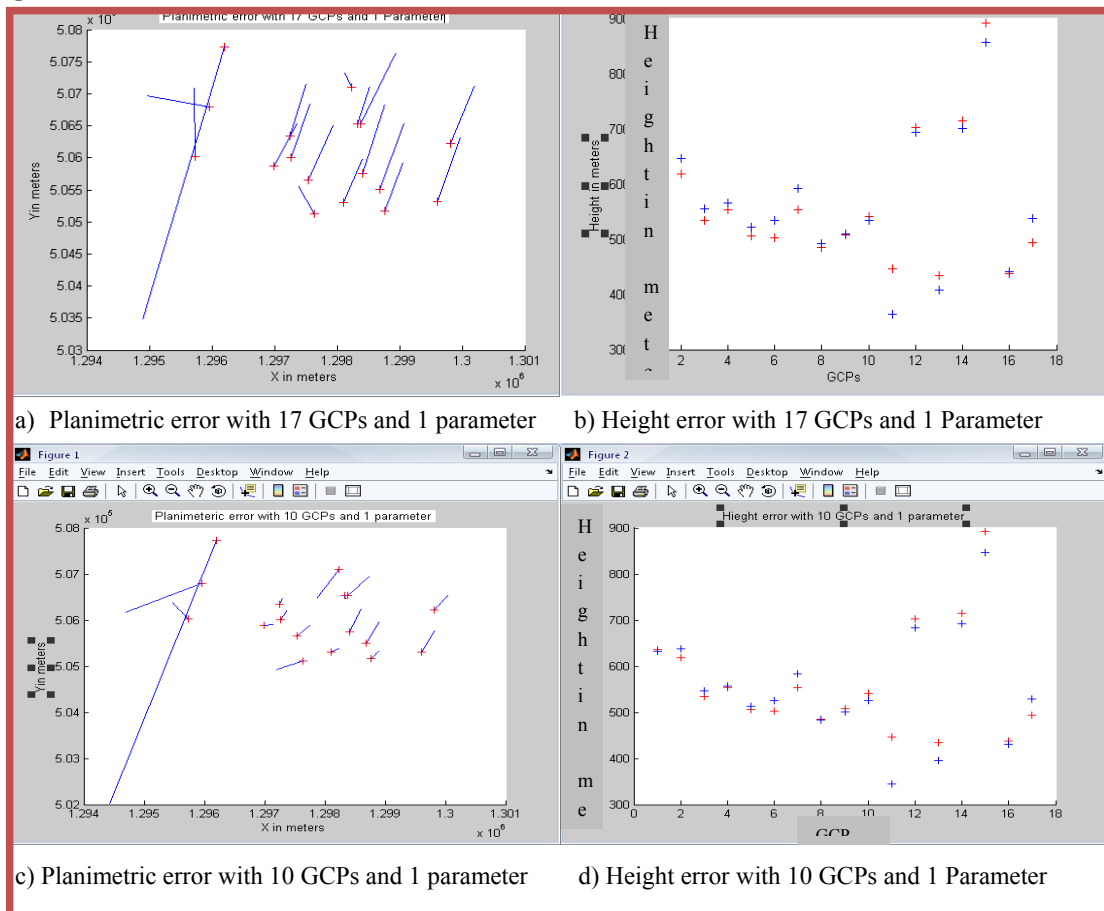


Figure 5-7 Graphs a, b, c, d showing Planimetric and height accuracy with 1 parameter using 17 and 10 GCPs for CARTOSAT-1 image

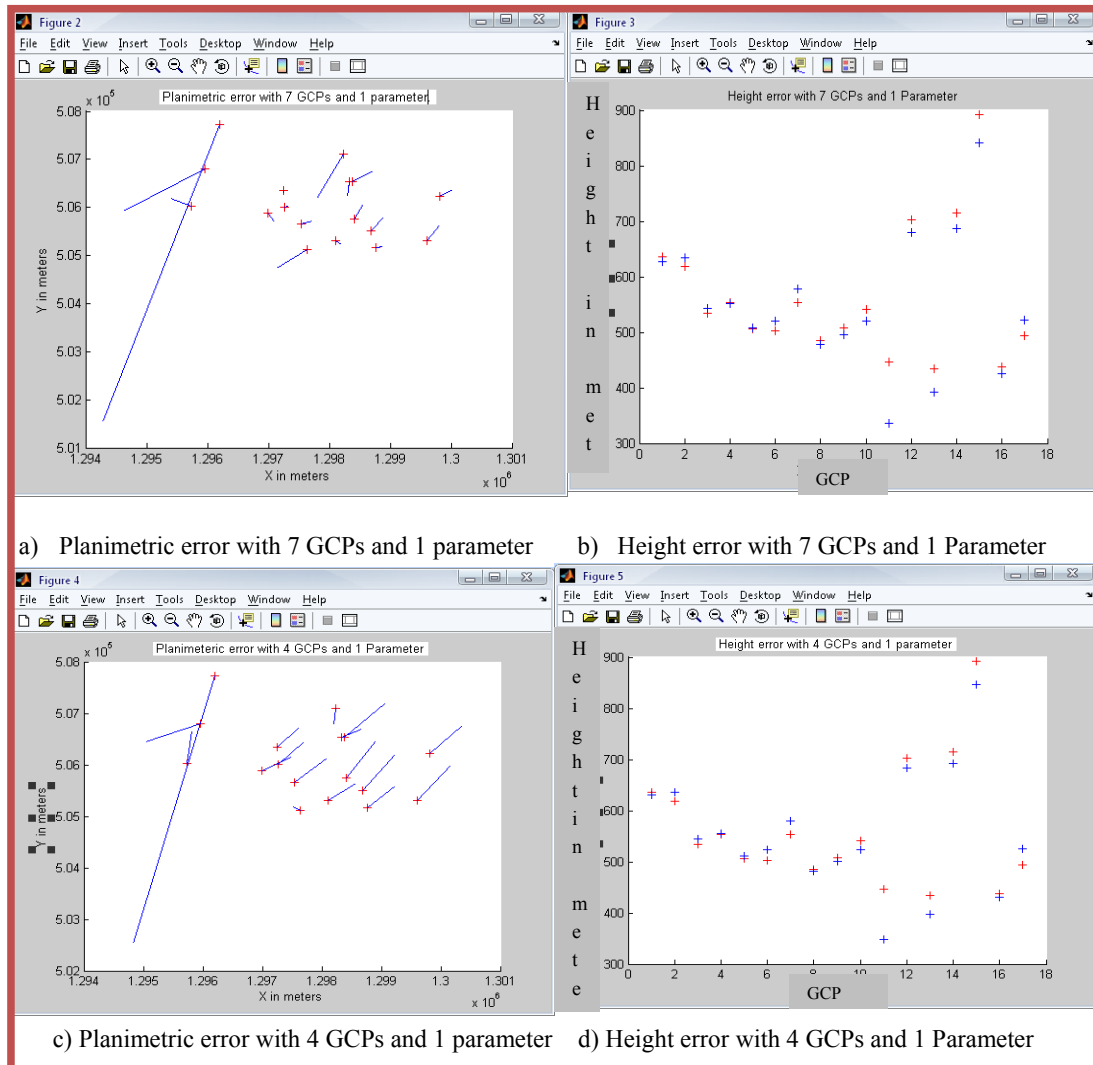


Figure 5-8 Graphs a, b, c, d showing Planimetric and height accuracy with 1 parameter using 7 and 4 GCPs for CARTOSAT-1 image

In figure 5-7 and 5-8 the graphs compare Planimetric and spatial accuracy achieved with different GCPs, The errors in Planimetric graph shown by blue line stretched over the red crosses have been scaled to 1000 to show the results clearly, Height error Graph is true representation of height values on the ground in meters. Where red Cross shows the true value of the coordinate taken from the round and blue cross shows the value measured by the RPCs. It can be observed from the graphs that one point shows much larger deviation from the true value both in height and plane. This error is checked and verified for point ID 22 which is collected from the elevated region of the study area, and is on the mountain side. Some more GCPs where also collected from regions near to that point (they are not included as their precision was questionable) were also tested in the model with one parameter and all show similar deviation from true values.

Now to evaluate the same results only at the check points following table is drawn:

No. of GCP	No of Chk Points	Planimetric RMSE for in meters	Spatial RMSE for chk in meters	RMSE X chk pts	RMSE Y chk pts	RMSE Z chk pts
17	0	NA	NA	NA	NA	NA
15	2	0.9102	35.4219	0.2080	0.8861	35.4102
13	4	1.0937	21.2486	0.4936	0.9334	21.2205
10	7	0.4335	25.9214	0.2831	0.3283	25.9178
7	10	2.1346	41.0223	0.7725	1.9910	40.9668
4	13	1.6582	37.8888	0.6657	1.5367	37.8525
1	16	1.7945	35.3715	0.7012	1.6518	35.3259

Table 5-2 Accuracy achieved with only check points of CARTOSAT-1 with one parameter.

Table 5-2 considers only check points for accuracy assessment both in plain(x, y) and height (z). Considering the results given in Table 5-1, Table 5-2 and Figure 5-7 and 5-8 we can conclude that CARTOSAT – 1 needs more that 1 parameter to achieve better accuracy, and we can also say that with one parameter optimum number of GCPs used should be 4 (spread at four corners of the scene). It is worth noticing that only 2 meter height difference can be seen if one uses all GCPs or 4 GCPs.

5.2.1.2. Using IKONOS with GCPs

IKONOS data set was tested with 1parameter by different number of GCPs, Planimetric and spatial errors were checked similarly as in the case of CARTOSAT – 1 data set. For the IKONOS 12 GCPs were collected for Chandigarh area. Following table 5-3 shows the results for the same.

No. of GCP	No of Chk Points	Planimetric RMSE in meters	Spatial RMSE in meters	RMSE X in meters	RMSE Y in meters	RMSE Z in meters
12	0	0.0426	0.4662	0.0330	0.0269	0.4642
10	2	0.0442	0.4725	0.0349	0.0270	0.4705
7	5	0.0438	0.4669	0.0339	0.0277	0.4648
4	8	0.0431	0.4764	0.0330	0.0278	0.4744
1	11	0.0501	0.4689	0.0422	0.0270	0.4663

Table 5-3 IKONOS orientation accuracy with both GCPs and Checkpoints through 1 parameter

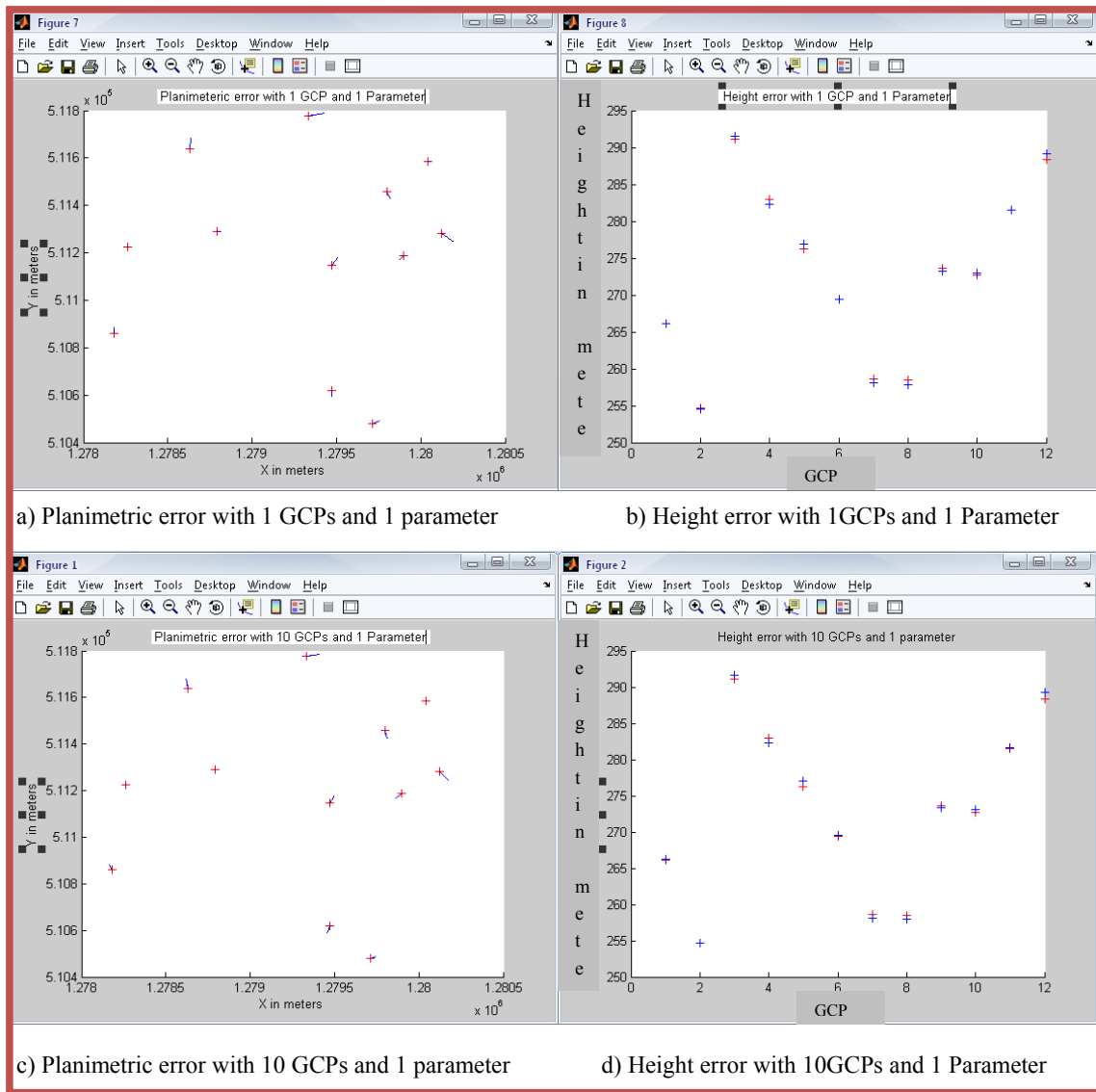


Figure 5-9 Graphs a, b, c, d showing Planimetric and height accuracy with 1 parameter using 1 and 10 GCPs for IKONOS images

In figure 5-9 graphs are shown to plot the both Planimetric and height error considering 1 and 10 GCPs from table 5-3. The accuracy of IKONOS is also evaluated by only considering check points shown from the following table 5-4. From all the analysis done with one parameter over IKONOS it can be concluded that IKONOS gives sufficiently good accuracy by considering 1 parameter and 1 GCP, but for consistent accuracy, a good distribution of points 2 to 3 GCPs are recommended.

No. of GCP	No of Chk Points	Planimetric RMSE in meters	Spatial RMSE in meters	RMSE X in meters	RMSE Y in meters	RMSE Z in meters
12	0	NA	NA	NA	NA	NA
10	2	0.05242	0.5512	0.0289	0.0370	0.5705
7	5	0.0338	0.6669	0.0439	0.0267	0.6648
4	8	0.0531	0.5764	0.0370	0.0178	0.5758
1	11	0.0601	0.6689	0.0522	0.0260	0.6653

Table 5-4 Accuracy achieved with only the check points of IKONOS with one parameter.

5.2.2. Modelling with two parameters:

When Modelling with 2 parameters, 6 coefficient values change or are modified for 1 image, these are $a[1]$, $a[2]$, $c[1]$, $c[2]$, $b[2]$, $d[2]$. Here also $b[1]$ and $d[1]$ are kept constant 1. This comes under first order Polynomial transformation.

5.2.2.1. Using CARTOSAT – 1 for the analysis:

CARTOSAT- 1 was oriented with two parameters using different number of GCPs which were chosen from across the scene.

No. of GCP	No of Chk Points	Planimetric RMSE in meters	Spatial RMSE in meters	RMSE X in meters	RMSE Y in meters	RMSE Z in meters
17	0	0.5767	3.4991	0.2495	0.5199	3.4513
15	2	0.5893	3.3326	0.2596	0.5291	3.2800
13	4	0.8637	5.9184	0.2812	0.8167	5.8550
10	7	0.8988	5.9754	0.2764	0.8552	5.9074
7	10	0.8596	6.2563	0.2684	0.8166	6.1969
4	13	1.0150	7.3730	0.4119	0.9276	7.3028

Table 5-5 CARTOSAT-1 Orientation with two parameter model.

Graphs have been plotted for the results given in table 5-5 as shown in figure 5-10 below

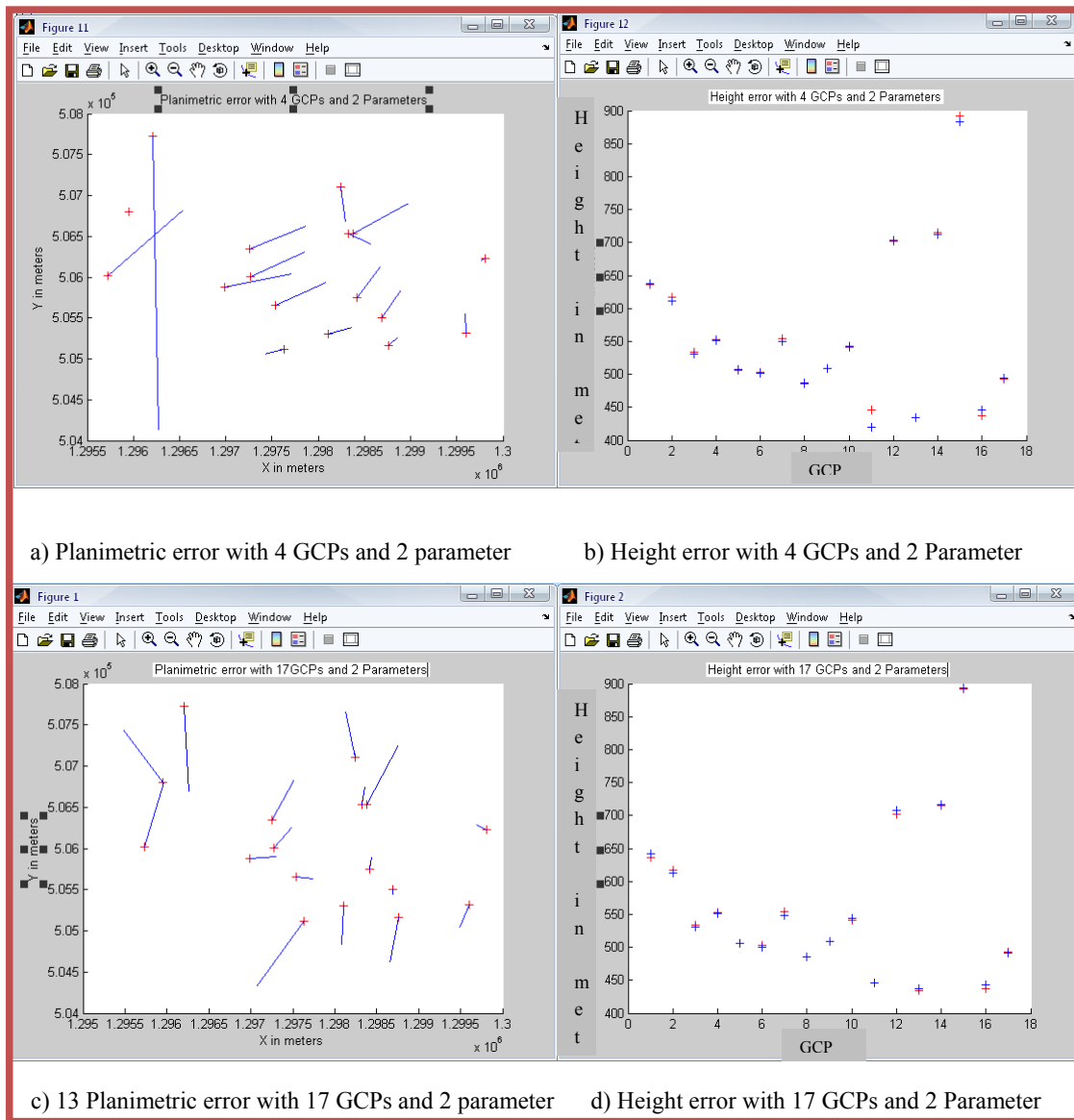


Figure 5-10 Graphs a, b, c, d showing Planimetric and height errors with 2 parameter using 4 and 17 GCPs for CARTOSAT-1 image

Figure 5-10 gives us a very clear picture if compared to figure 5-8 where it is observed that moving from one parameter to two parameters reduces the height error drastically, bringing it down by almost 25 meters over the same points. However it can also be observed that for ID 22 the error can only be reduced if considered all the points in the solution.

5.2.2.2. Using IKONOS for Analysis :

IKONOS is again tested using 2 parameters and different GCPs. The results are shown in the table below.

No. of GCP	No of Chk Points	Planimetric RMSE in meters	Spatial RMSE in meters	RMSE X in meters	RMSE Y in meters	RMSE Z in meters
12	0	0.0367	0.3567	0.0295	0.0219	0.3548
10	2	0.0404	0.3649	0.0338	0.0221	0.3626
7	5	0.0451	0.4517	0.0371	0.0256	0.4494
4	8	0.0500	0.4051	0.0437	0.0243	0.4020

Table 5-6 IKONOS orientation through 2-parameter model.

Graphs have been plotted for 12 GCPs shown in figure

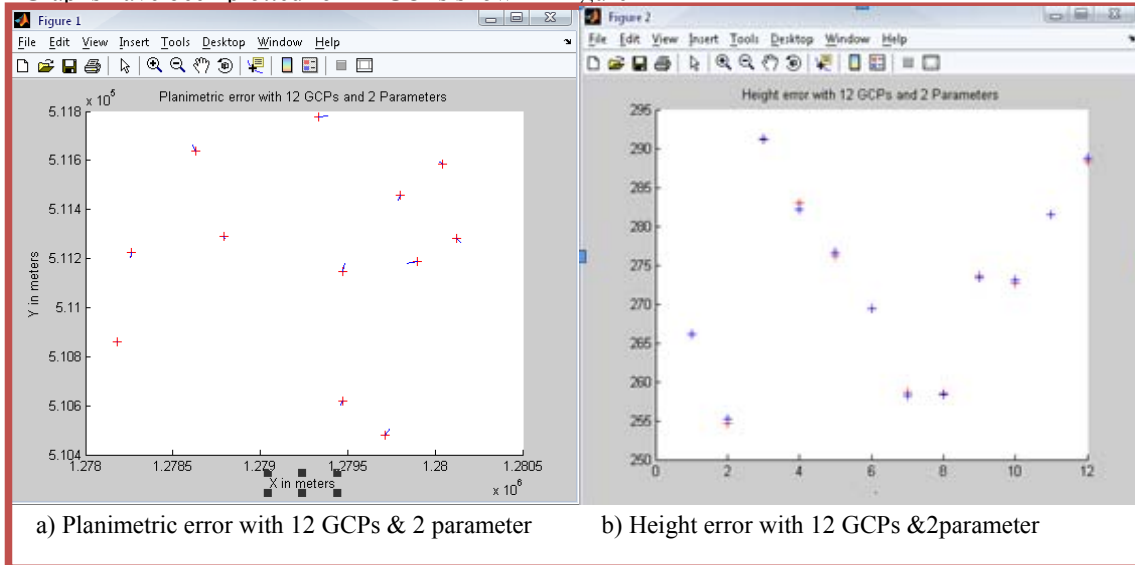


Figure 5-11 Graphs a, b, showing Planimetric and height errors with 2 parameter using 12 GCPs for IKONOS image.

From the Figure 5-11 and table 5-6 it can be seen that not much difference in improvement of vertical and Planimetric accuracy moving from one two parameters in case of IKONOS.

5.2.3. Modeling with three parameters:

When Modelling with 3 parameters , 10 coefficient values change or are modified for 1 image , these are $a[1]$, $a[2]$, $a[3]$, $c[1]$, $c[2]$, $c[3]$, $b[2]$, $b[3]$, $d[2]$, $d[3]$. Here also $b[1]$ and $d[1]$ are kept constant 1. This comes under first order Affine Polynomial transformation

5.2.3.1. Using CARTOSAT – 1 for analysis:

The results for CARTOSAT – 1 with 3 parameters have been given in the table below

No. of GCP	No of Chk Points	Planimetric RMSE in meters	Spatial RMSE in meters	RMSE X in meters	RMSE Y in meters	RMSE Z in meters
17	0	0.2828	1.9984	0.2318	0.1620	1.9783
15	2	0.3048	2.2896	0.2527	0.1704	2.2692
12	5	0.4780	2.4904	0.4410	0.1843	2.4441
7	10	1.8597	6.6585	0.5709	1.7669	6.3935
6	11	6.4451	36.6127	1.7450	6.2044	36.0410

Table 5-7 CARTOSAT-1 orientation with 3 parameter model

From the figure 5-12 and table 5-7 it was observed that CARTOSAT – 1 showed the vertical accuracy of around 2.49 meters i.e. one pixel using 12 GCPs and even showed lesser using all the GCPs, however it was also observed that solution gave absurd values if used with less than 7 GCPs, although the solution converged after 30 iterations, so it can be concluded that 7 GCPs can be regarded as minimum for using 3 parameter solution. More over from the figure it is also seen that GCP ID 22 still is giving more error with taking 7 GCPs and 3 parameters.

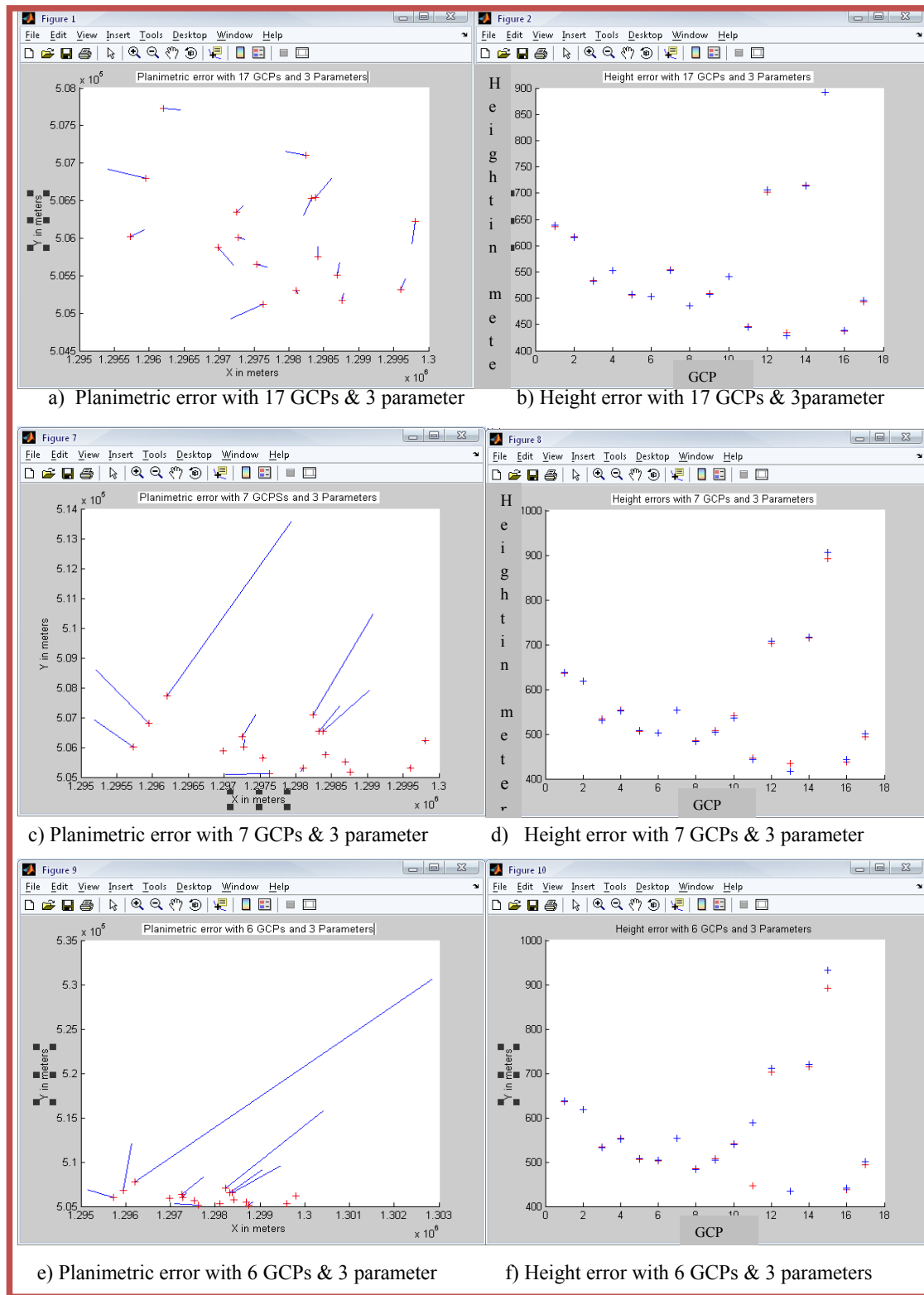


Figure 5-12 Graphs a, b, c, d, e, f shows Planimetric and height errors with 3 parameters using 17, 7 and 6 GCPs for CARTOSAT-1 image.

5.2.3.2. Using IKONOS for analysis:

Analysis of IKONOS has been shown in the table given below:

No. of GCP	No of Chk Points	Planimetric RMSE in meters	Spatial RMSE in meters	RMSE X in meters	RMSE Y in meters	RMSE Z in meters
12	0	0.0333	0.2751	0.0266	0.0200	0.2731
9	3	0.0390	0.2875	0.0301	0.0248	0.2849
6	6	0.0509	0.3790	0.0442	0.0254	0.3755

Table 5-8 IKONOS orientation through 2 parameter model

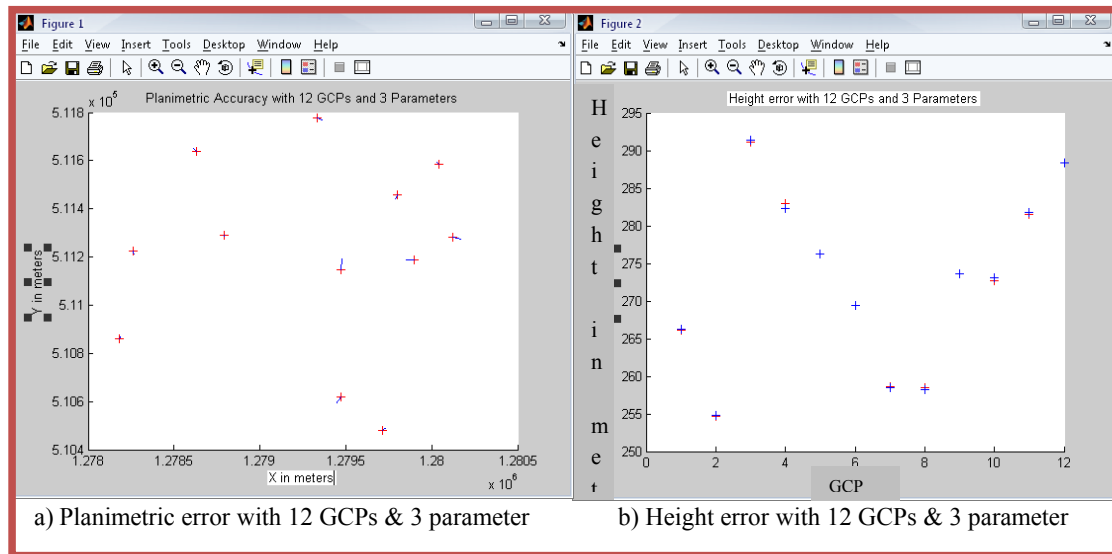


Figure 5-13 Graphs a, b, showing Planimetric and height errors with 3 parameter using 12 GCPs for IKONOS image

From the figure 5-13 and table 5-8 it can be seen that the accuracy has improved.

5.2.4. Modelling with Four parameters

Four parameters constitute $a[1]$, $b[2]$, $c[1]$, $d[2]$, $a[2]$, $b[3]$, $c[2]$, $d[3]$, $a[3]$, $b[4]$, $c[3]$, $d[4]$, $a[4]$, $c[4]$ coefficients to change for 1 image, these coefficients require greater no. of GCPs to change as the

unknowns become more. Refining RPCs with these parameters are also known as 3D affine transformations.

5.2.4.1. Using CARTOSAT – 1 for the refinement:

CARTOSAT – 1 data has been used for refinement with four parameters, the results of the experiments have been shown below.

No. of GCP	No of Chk Points	Planimetric RMSE in meters	Spatial RMSE in meters	RMSE X in meters	RMSE Y in meters	RMSE Z in meters
17	0	0.2609	1.7483	0.2148	0.1480	1.7287
13	4	0.5785	4.1766	0.3609	0.4522	4.1364
9	8	0.7664	5.3826	0.6515	0.4061	5.3278

Table 5-9 CARTOSAT-1 orientation with 4 parameter model

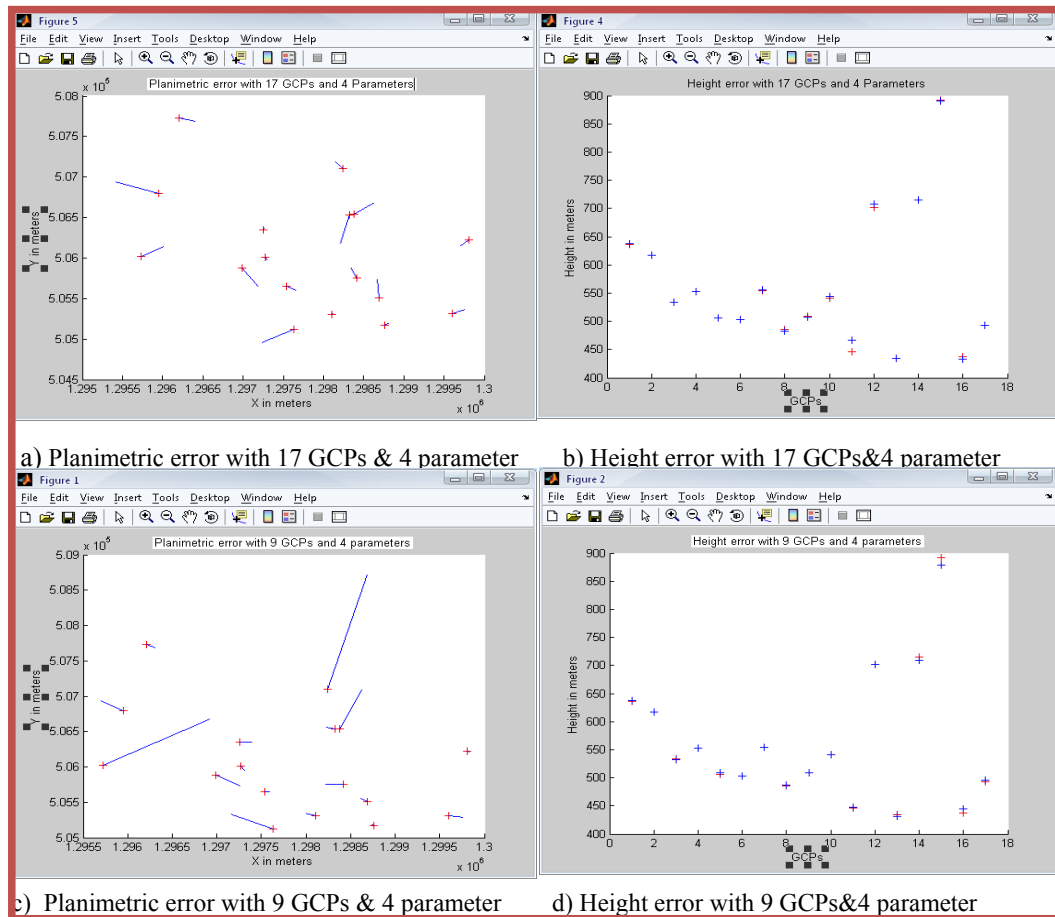


Figure 5-14 Graphs a, b, c, d shows Planimetric and height errors with 3 parameters using 17 and 9 GCPs for CARTOSAT-1 image.

From figure 5-14 and table 5-9 it can be observed that around 5 meter accuracy can be attained using 9 GCPs and 4 parameters, more over the error of ID 22 is also reduced using 4 parameters and 9 GCPs

5.2.4.2. Using IKONOS for the refinement with 4 parameters:

Analysis of IKONOS with 4 parameters has been shown in the table below:

No. of GCP	No of Chk Points	Planimetric RMSE in meters	Spatial RMSE in meters	RMSE X in meters	RMSE Y in meters	RMSE Z in meters
12	0	0.0274	0.2382	0.0196	0.0192	0.2367
9	3	0.0515	0.3608	0.0269	0.0439	0.357

Table 5-10 IKONOS orientation with 4 parameter model

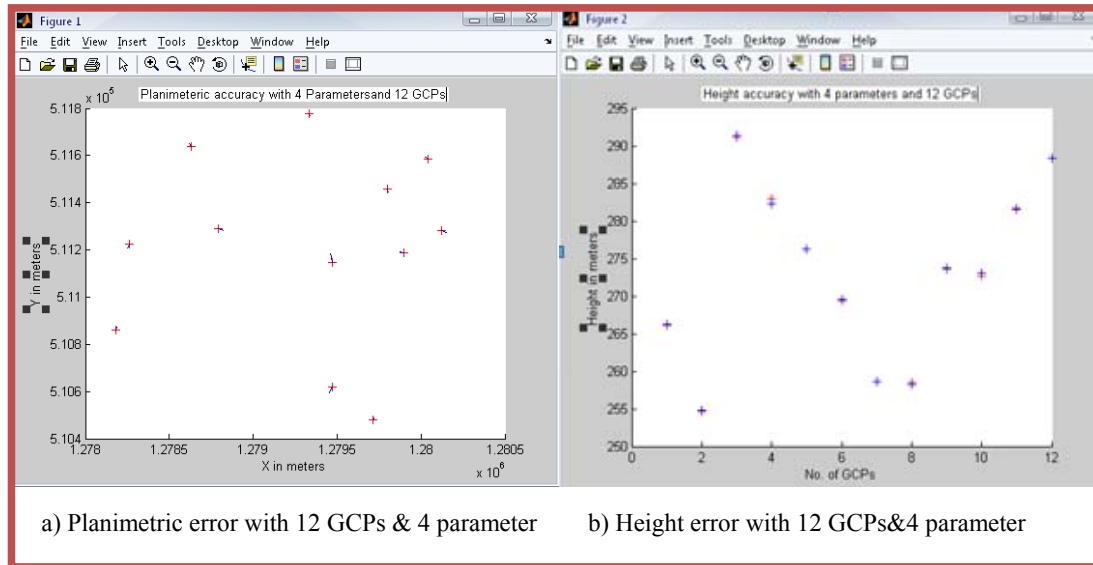


Figure 5-15 Graphs a, b, shows Planimetric and height error with 4 parameters considering 12 GCPs for IKONOS image.

From the figure 5-15 and table 5-10 it can be observed that IKONOS gives almost the same accuracy as compared to previous results with 1, 2, and 3 parameters which also gave sub meter accuracy.

5.2.5. Using Numerators in the observation equations for Modelling:

Models were built and attempted with different parameter combinations and it was found that refining only numerators in the rational polynomial equation could give us reliable solution accuracy with the ease. Numerators remove the chances of zero denominator value error from the equations and helps in getting the solutions with more effectiveness. Only thing is that with the taking numerator terms is that the equations have to be carried to the order of second or third polynomial for the refinement. At first the models were tested by numerators only i.e. refining only a [], c [] terms of the equation. It was found that numerators behaved in similar ways as when taking both numerator and denominator, when refined with GCPs, The difference that was observed was that the equation terms turns up to second order polynomial equation, or even third order to achieve the same results as was achieved by only refining rational equations to 3d affine This sometimes leads to more requirement of GCPs to converse the equation, In case of IKONOS it was found that $a[1]$, $c[1]$ parameters give good accuracy with 1 GCPs which is same if taken the case of considering numerator and denominator both. However in case of the CARTOSAT-1 scene the following are the observations that were made, that is considered as the best from all.

- Over all accuracy with 6 GCPs and 2 parameters $a[1]$, $c[1]$, $a[2]$, $c[2]$
Planimetric RMSE in meters = 1.4896 meters
Vertical RMSE in meters = 6.3762 meters

By doing this experiment we can conclude that the numerator coefficients such as $a1[]$ and $c1[]$ play the key term for determining object coordinates and refining only these terms effect the improvement.

5.3. Extraction of DTM from the refined RPCs:

Digital Terrain Model(DTM) was extracted with the help of RPCs, using LPS or Leica photogrammetric suite, There are several different formats of DTM that can be extracted using LPS such as DEM, Terra model, TIN, 3d shape, socket set TIN etc. During the experiment the DTM was extracted twice, once by using the original RPCs and second by using the refined RPCs.

5.3.1. Extraction of DTM from Catosat-1 using raw RPCs:

CARTOSAT –1 DTM was extracted by using the RPCs provided with the imageries and the output can be shown as follows:

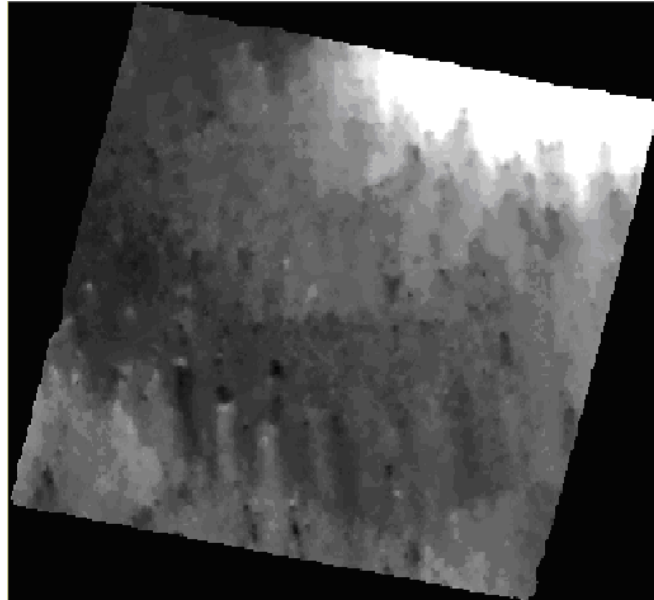


Figure 5-16 CARTOSAT DTM with Original RPCs

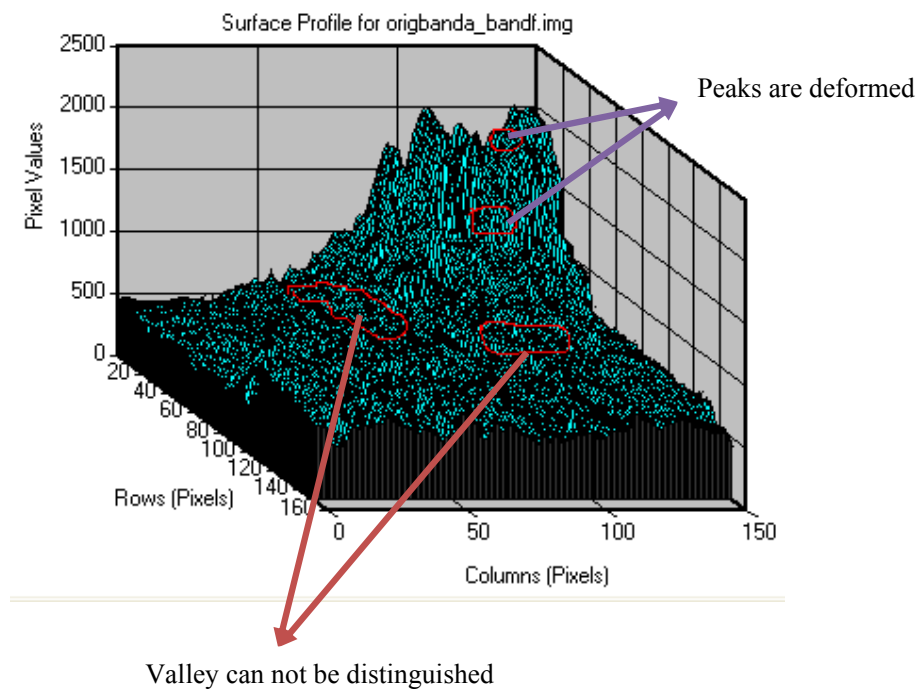


Figure 5-17 DEM surface profile with original RPCs

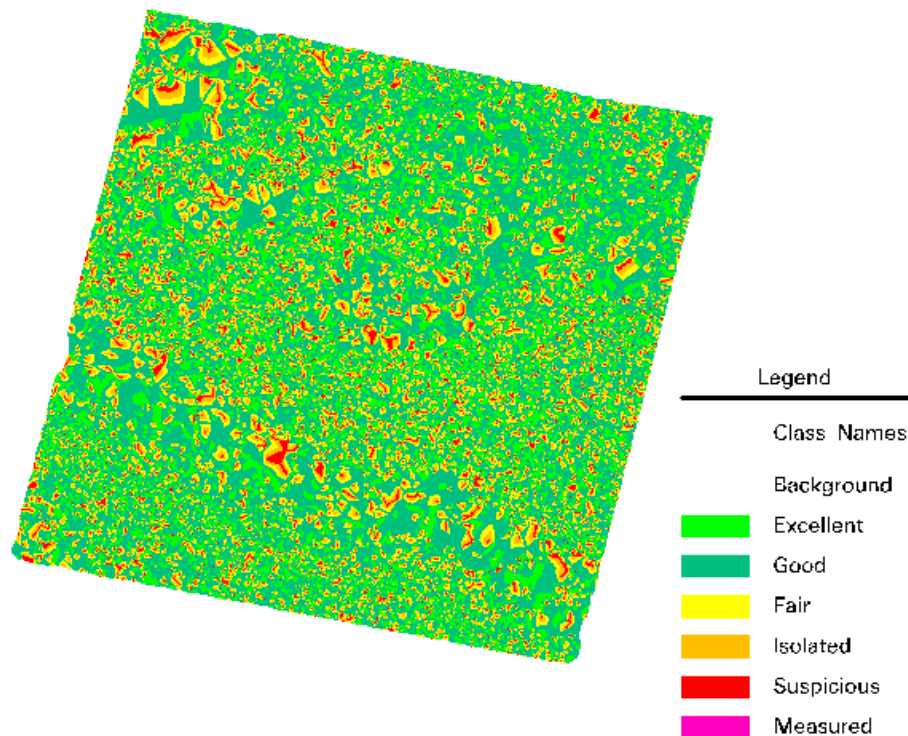


Figure 5-18 Quality map of the DTM extracted from the original RPCs

The DTM was generated from the raw RPCs as shown in figure 5-16. The quality of the DTM was analysed by generating the surface profile and quality map or point map. Observation from the DEM infers that the height values are overlapping and one cannot really distinguish between the elevation values.

Figure 5-17 shows the surface profile of the scene extracted with the help of the DTM and it could be observed that the valley shows undulation which is not the case on the ground; also the peaks are not very sharp. The point map which shows the quality of DTM is also generated. Point map shows the Quality of correlated DTMs Postings using colours as shown in the figure 5-18. Points having Excellent have green colour and correlation value between 1 and 0.85 , Good has correlation value between 0.7 and 0.85 represented by dark green colour, fair lie between 0.75 and 0.5 and points which do not have immediate neighbours are isolated points. It can be observed that there are lot of patches in the quality map extracted which fall under the suspicious and isolated category.

5.3.2. Extraction of DEM from Catosat-1 using refined RPCs:

DTM was extracted using refined RPC by 9 GCPs and 4 Parameters and the output can be shown as follows:

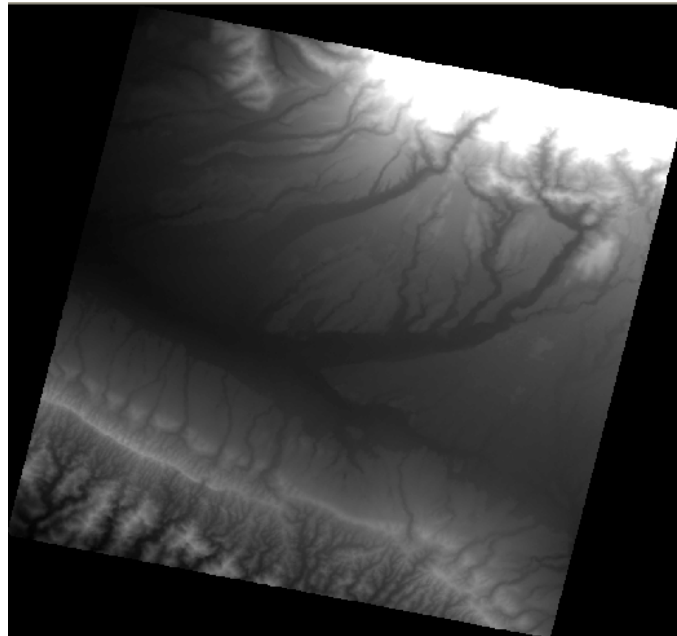


Figure 5-19 DEM from modified RPCs

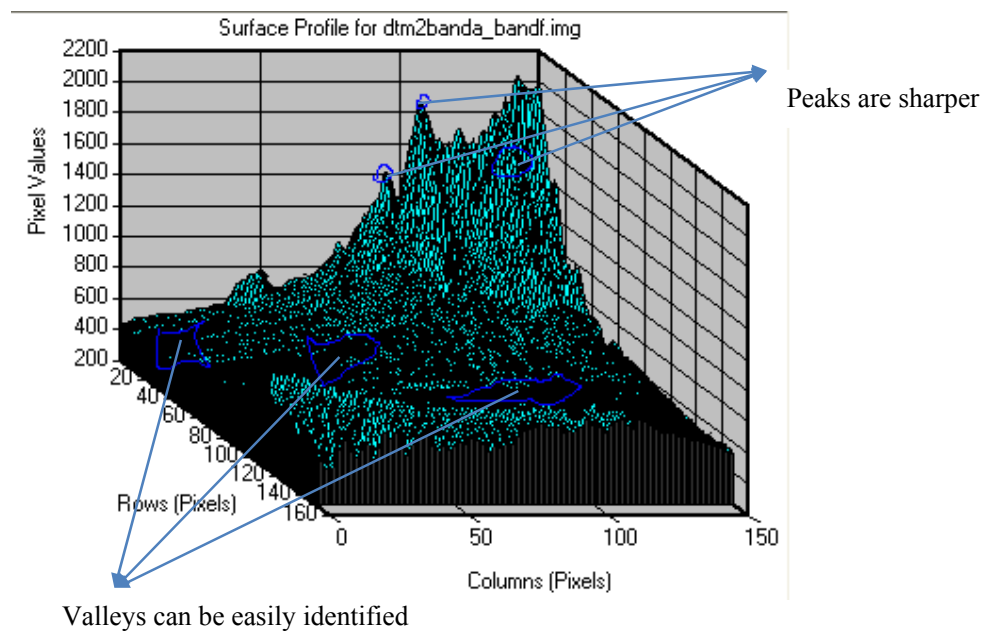


Figure 5-20 Surface profile of DEM from modified RPCs

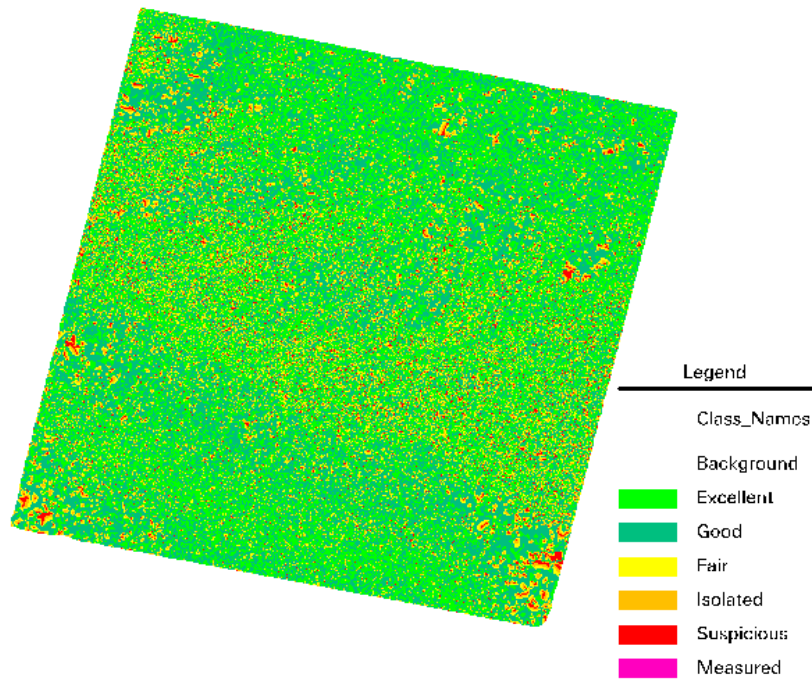


Figure 5-21 Point map of DTM extracted from Refined RPCs

The DTM generated using the refined RPCs is shown in figure 5-19. It can be observed by looking at the DTM that now the one can distinguish between various elevation values as they do not overlap now, the surface profile in figure 5-20 also shows valleys in the area which is true to ground, the peaks also have sharper edges. The point map for DTM of the refined RPCs shown in figure 5-21 show very less red and yellow, and most of the area is filled by green, which represents excellent.

This DTM has been generated in Leica photogrammetry suite with out using any GCPs and only using the refined RPCs generated by the software. It has also been observed that LPS also generates similar kind of DTM but with 12 or more GCPs and with 3rd order polynomial rectification. So we can conclude that the RPCs refined using the build software can be used for extracting accurate photogrammetric products.

5.4. Discussions on the Results and Analysis:

Assessment of RPC refinement requires a combination of parameters to be tested in order to find the optimum number of GCPs needed to achieve higher accuracy in the 3D object plain for high-resolution images IKONOS and CARTOSAT respectively. IKONOS data was tested for the plain terrain of Chandigarh whereas CARTOSAT has been used for the city of Dehradun, which is relatively undulating.

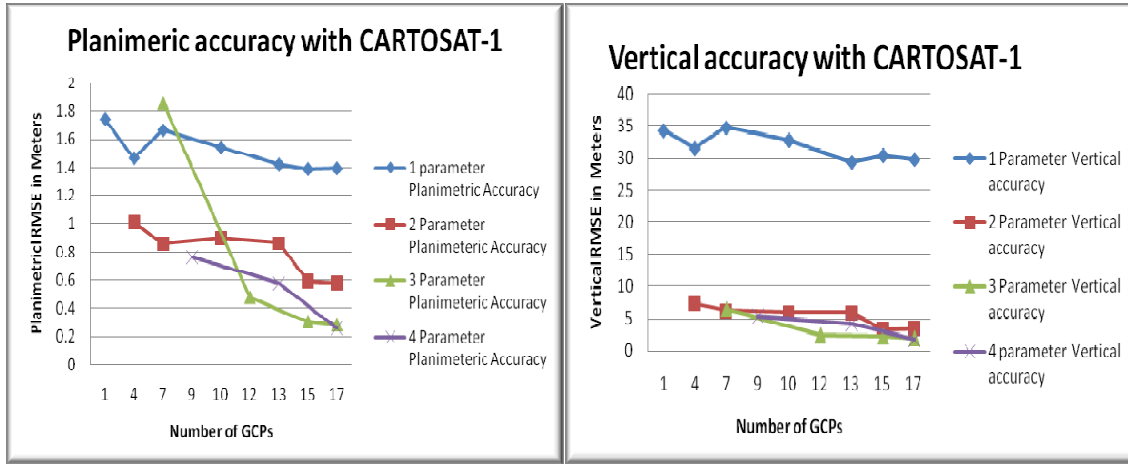


Figure 5-22 Planimetric accuracy with CARTOSAT-1 Figure 5-23 Vertical accuracy with CARTOSAT-1

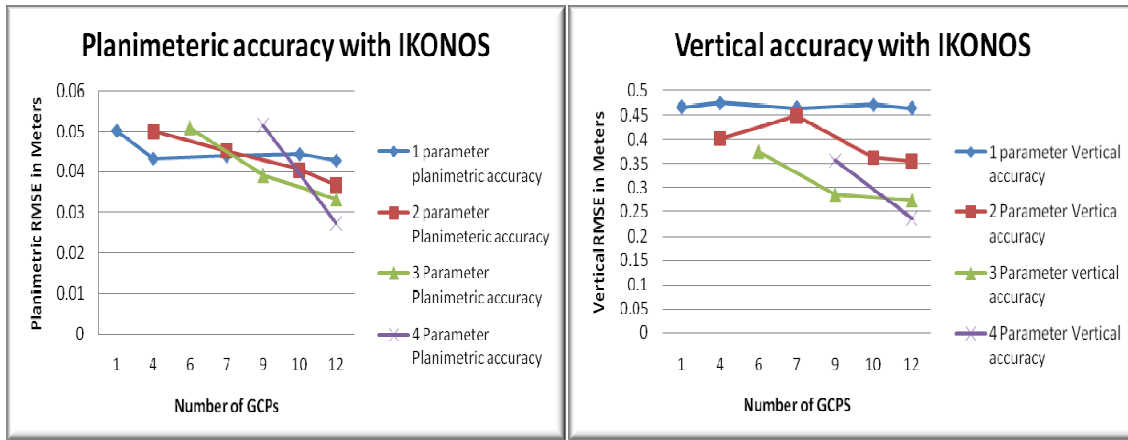


Figure 5-24 Planimetric accuracy with IKONOS Figure 5-25 Vertical accuracy with IKONOS

Four experiments were conducted to assess the orientation by modifying a series of parameters from one to four with varied number of GCPs for both the images, CARTOSAT and Dehradun.

In the first Experiment the model is executed with 1 parameter, it accounts only for the removal of bias from the RPCs. And in this case, the line numerator coefficient $a[1]$, sample numerator coefficient $c[1]$ or shift parameters are modified for both the images as the line denominator coefficient $b[1]$ and sample denominator coefficient $d[1]$ are taken as a constant value 1, which are unaltered.

The observed vertical accuracy with one parameter and one GCP was 34 meter with CARTOSAT data and it reduced to 29 meter when 17 GCPs were used, in comparison to 43 meter of height with the raw RPCs without any GCPs. Therefore, it can be said that the refinement does not deliver a significant change in the accuracy with a single parameter for CARTOSAT. In case of IKONOS, the accuracy observed was as high as 0.4 meter with even a single GCP wherein the accuracy with raw RPCs was 5.6 meter. And it remains the same with 12 GCPs when modelled with only one parameter (0.4 meter).

The second experiment was modification considering 2 parameters. In this case, the 6 coefficient values $a[1]$, $a[2]$, $c[1]$, $c[2]$, $b[2]$, $d[2]$ are modified for each of the RPCs, Aft and Fore and $b[1]$, $d[1]$ are kept constant which equals to 1. This procedure accounts for the first order Polynomial transformation. 7.3 meter was the observed accuracy when using 4 GCPs with CARTOSAT and 3.4 meter was with 17 GCPs. A remarkable reduction in the error has been shown for CARTOSAT when two parameters are considered. On the other hand, IKONOS yields 0.4 meter with 4 GCPs and 0.35 meter with as many as 12 GCPs, which is not quite significant when compared with the results of the initial model, which modified one parameter giving almost the same accuracy.

The third experiment categorically is a first order Affine Polynomial transformation with denominators which modifies 10 coefficient values of the RPCs of each of the images Aft and Fore, and these are $a[1]$, $a[2]$, $a[3]$, $c[1]$, $c[2]$, $c[3]$, $b[2]$, $b[3]$, $d[2]$, $d[3]$ respectively wherein $b[1]$ and $d[1]$ still remain constant (1). For CARTOSAT, the accuracy achieved was 6.3 meter with 7 GCPs, which reduced to 1.97-meter with 17 GCPs. However when the same was applied on IKONOS, the accuracy increased by 0.1 to 0.2 meters as can be observed from the resulting values, which were around 0.2 meter.

The final experiment, was a 3D affine transformation which constituted of modifying four parameters which includes changing of the following coefficients $a[1]$, $b[2]$, $c[1]$, $d[2]$, $a[2]$, $b[3]$, $c[2]$, $d[3]$, $a[3]$, $b[4]$, $c[3]$, $d[4]$, $a[4]$, $c[4]$ for each of the images. And this transformation requires greater number of GCPs, as the number of unknowns is more. An accuracy of 5.32 meter is attainable with 9 GCPs and an accuracy of 1.72 meter is obtained with 17 GCPs for CARTOSAT-1. In IKONOS not much change in accuracy was observed from the last experiment with 3 parameters and it was still around 0.2

From the above analysis, one can observe that the RPCs of CARTOSAT need more number of GCPs and parameters to achieve the desired accuracy of less than a pixel. The optimum solution from RPC modification can be either with 3 parameters and 7 GCPs, which yields accuracy of 6 meter or with 4 parameters and 9 GCPs for an accuracy of 5 meter, depending on the number of GCPs accessible.

The best transformation for IKONOS is with one parameter, which yields an accuracy of 0.4 meter even with a single GCP. Testing with two parameters also may be sought, if necessary and it would yield good accuracies of around 0.3 meter, but with 12 GCPs.

However, it is note worthy to observe that the accuracy of the DEM with RPC refinement is scene dependent (terrain conditions) and plain areas need less number of GCPs relative to the hilly areas.

\

6. Conclusions and Recommendations

6.1. Overall conclusion:

The main objective of the current research work was to develop a model and a method, which could be used for refining the RPCs provided by the image vendor, so as to improve the positional as well as the accuracy in height. The model was built and analysed for CARTOSAT-1 and IKONOS data sets during the analysis. Results were drawn and it was found that the model gave almost similar results in comparison with (Fraser et al., 2002; Fraser and Hanley, 2003; Grodecki and Dial, 2003; Toutin, 2003) in case of IKONOS and (Kumar, 2006) in case of CARTOSAT-1. The model further needs to be analysed with more data sets corresponding to hilly terrains in case of IKONOS and Plain terrains in case of CARTOSAT-1.

6.2. Sub Conclusions:

In order to address the objectives, five-research questions were posed during the study, which helped in building an overall model, which could be used for the refinement. Literature review, Model building in java and Analysis of the results helped in answering the research questions proposed.

The first question takes a broader view and in general examines how the RPC can be modified. It was observed from the literature study that RPCs can be improved in a myriad ways. However, most of the techniques use GCPs with least square adjustment models for the refinement. The current research has adopted a direct approach for refinement. In this approach RPC coefficient are updated/modified so as to achieve better accuracy. The original supplied coefficient values are updated using addition of approximate coefficients method, where by the approximate coefficients are added to the polynomial equations, which are fitted using least square adjustment models, using difference of real image coordinates and calculated image coordinates. The approach to refine the RPC values till date was using polynomial adjustment during the triangulation process, where the biases are added to the image coordinates. To conclude the presented methodology can be used for updating the original RPC values rather than the addition to the image coordinates.

The second research question relates to the usage of minimum ground control points. It was observed from the study that higher accuracy is achieved in the 3D object plain if there is increase in the number of parameters and the GCPs used. However if one would increase the number of parameters and keep the same number of GCPs, considerable improvement of accuracy can be observed. A higher level of accuracy was achieved when the GCPs were considered at the corners of the scene with optimum number of parameters, which is terrain dependent as well sensor dependent. The best set of parameters can be found by finding the correlation between the parameters; And a correlation matrix is made which can tell how much the parameters are related. The parameters having higher correlation are kept and one having lower correlation is taken out. However due to lack of time in my case, the approximate coefficients were considered from the beginning i.e. a_0 in the increasing order and the approximation functions were built with the same. These approximation function or parameters were tested with various GCPs, and the best set of parameters was selected which gave the best accuracy on

3D plain with least number of GCPs. In case of flat areas or flat areas the best accuracy achieved using IKONOS is 0.23 meters using 9-12 GCPs but one could obtain sub pixel accuracy by using 1 GCP and 1 Parameter. Therefore 1 to 3 GCPs with IKONOS on the plain areas can be considered as better option. Where as in case of Moderately hilly terrain it took us 9 GCP and 4 parameters to fit the point ID 22 which was at the higher elevation and it gave us a vertical RMSE of around 5 meters; However lower RMSE can also be achieved using 12 or more GCPs with 3 and 4 parameters.

To conclude, selecting the optimum parameters and Corner GCPs with good accuracy can minimize the requirement of Ground control points. Optimum parameters can be found out by finding out the correlations between the parameters, and highly correlated parameters should be considered in the adjustment.

The third question is on how we can asses the improvement in the RPCs. To conclude, it can be assessed statistically by finding the achieved accuracy in the object and the image space using GCPS and Check points.

The last question relates to study the impact of improved RPCs on the derived products. It has been observed that derived products such as DTM/orthoimages show considerable improvement in the vertical and Planimetric accuracy. Hence we can conclude that if one uses the refined RPCs, no further photogrametric processing is required and one can generate DTMs/orthoimages with considerable accuracy.

6.3. Limitations:

Time period of the research played a major limiting factor which affected the analysis and the design of the model leaving a lot of issues to deal with. One of the issues was to make a model in which the Tie points could also be included in the adjustment. A lot of attempts were made to do the same but the matrix of $A^t A$ shows the problem of numerical instability and singularity in nature and hence could not find the inverse of the matrix. It needed more time to find the solution for the same. Second issue was that the approximate coefficients in the approximate function should be selected automatically, prior to including them in the adjustment. This could be done using a correlation matrix, which finds out the coefficients having higher values. These approximate function coefficients then could be considered for the adjustment. Implementing the same demands more time, as the complexity in implementation increases. Finally, the model made needs to be tested with the same data on different scenes with varying terrain conditions. This was the major limitation as I could not get the same.

6.4. Recommendations:

The developed RPC refinement program can be further extended for designing solutions for the limitations of this research work. There's a lot of research still left in this field where one can still explore new methods of getting higher accuracies in 3D object plane using lesser GCPs.

7. References:

- IKONOS sensor model support tour guide, Prepared by Space Imagine.2004
- Bakker , H.W. et al., 2004. Principles of Remote sensing ITC Educational Textbook Series, 2004. ITC, Enschede.
- Bianconi , M., Crespi, M., Fratarcangeli, F. and Giannone, F., 2007. New Software for RPC Use and Generation, pp.6 pdf accessed 3rd december 2007,. <http://www.ceg.ncl.ac.uk/rspsoc2007/papers/204.pdf>
- Chen, L.-C., Teo, A.-T. and Liu, C.-L., 2006. The Geometrical Comparisons of RSM and RFM for FORMOSAT-2 Satellite Images. Photogrammetric Engineering and Remote Sensing, 72(5): 573-579.,pdf accessed on 2nd december 2007, http://dpl.csr.sr.ncu.edu.tw/journalpapers/J2006_PEnRS_TheGeometricalComparison of RSMAndRFMforFORMOSAT2.pdf
- Cho, W., Bang, I.K., Jeong, S. and Kim, K.-O., 2003. Automatic DEM Generation Using IKONOS Stereo Imagery. IGRASS 7(21): 4289-4291.
- OGC, 1999. The Open GIS Abstract Specification - Topic 7: Earth Imagery.,pdf accessed on 9th december 2007, <http://www.opengis.org/docs/99-107.pdf>
- Di, K., Ma, R. and Li, X.R., 2003. Rational Functions and Potential for Rigorous sensor Model Recovery. Photogrammetric Engineering and Remote Sensing, 69(1): 33-41.
- Erdas field Guide Lieca Geosystems, GIS and Mapping 2005
- Fraser, C., Baltsavias, E. and Gruen, A., 2002. Processing of IKONOS imagery for sub metre 3D positioning and building extraction. ISPRS Journal of Photogrammetry and remote sensing, 56: 177-194.
- Fraser, C.S., Dial, G. and Grodecki, J., 2006. Sensor orientation Via RPCs. ISPRS Journal of Photogrammetry and remote sensing, 60: 182-194.
- Fraser, C.S. and Hanley, H.B., 2003. Bias Compensation in Rational Functions for IKONOS satellite Imagery. Photogrammetric Engineering and Remote Sensing, 69(1): 53-57.
- Fraser, C.S. and Hanley, H.B., 2005. Bias-compensated RPCs for sensor Orientation of High-resolution Satellite Imagery. Photogrammetric Engineering and Remote Sensing, 71(8): 909-915.
- Grodecki, J. and Dial, G., 2003. Block adjustment of high -resolution satellite images described by rational functions. Photogrammetric Engineering and Remote Sensing, 69(1): 59-68.
- Grodecki, j. and Gene, D., 2003. Block Adjustment of High-Resolution Satellite Images Described by Rational Polynomials. Photogrammetric Engineering and Remote Sensing, 69(1): 59-68.
- Herve, A. (Editor), 2003. Least Squares. Encyclopedia of Social Sciences Research Methods. Sage, Thousand Oaks (CA).

-
- Hu, Y., Tao, V. and Croitoru, A., 2004. Understanding the Rational Functional Model: Methods and Applications. GeoICT Lab, York University, Toronto, pp. 6.pdf accessed on 6th december 2007 ,<http://www.geoict.net/Resources/Publications/IAPRS2004/RFM2394.pdf>
 - Ian, D. and Tao, V., 2002. An update on the use of Rational Functions for Photogrammetric Restitution. ISPRS Journal of Photogrammetry and remote sensing, 7(3): 22-29.
 - Kumar, A., 2006. CARTOSAT –1 (IRS –P5) Stereo Data Processing – A Case Study of Dehradun Area, web page accessed on 3rd december 2007 <http://www.nrsa.gov.in/satellites/irs-p5.html>
 - Nandakumar, R., Srivastava, P.K., 2006, ISPRS-ISRO Cartosat-1 Scientific Assessment Programme, The ISRS - ISPRS-TC IV Goa Symposium held during Sep. 27-30, 2006 Space Applications Centre, ISRO, Ahmedabad – 380 015 INDIA, pdf accessed on 10th -december 2007 <http://www.nrsa.gov.in/satellites/Nan.pdf>
 - Morgan, M., 2004. Epipolar Resampling of Linear Array Scanner Scenes, University of Calgary, CALGARY, ALBERTA, 187 pp.
 - NIMA, 2000. The Compendium of Controlled Extensions for the National Imagery Transmission Format (NITF).
 - Satirapod, C., Phalakarn, B., Jongruguenun, T., Trisirisatayawong, I. and Fraser, C., 2004. Enhancing the Prospects for Mapping from high-resolution satellite imagery in the developing world. Geo-Image Technology Research Unit, Dept. of Survey Engineering, Chulalongkorn University, Dept. of Geomatics, University of Melbourne, Bangkok, Thailand, pdf accessed on 12th december-2007, <http://www.image-info.com/isprs2004/comm6/papers/684.pdf>
 - Tiegun, W., 2005. The capability of IKONOS image for 1:10000 mapping in China, International institute for Geo-information science and earth observation, Enschede.
 - Tonolo, G., F and Poli, D., 2003. Georeferencing of Eros - A1 high resolution images with rigorous and rational function model, Hannover., PDF accessed on 8th december 2007, http://www.photogrammetry.ethz.ch/general/persons/daniela/pub/hannover_eros_03.pdf
 - Toutin, T., 2003. Error Tracking in IKONOS Geometric Processing Using a 3D Parametric Model. Photogrammetric Engineering and Remote Sensing, 69(1): 43-51.
 - Weisstein, 2007. Least squares., web page accessed on 12th December-2007 , <http://mathworld.wolfram.com/LeastSquaresFitting.html>

8. Appendix:

8.1. Annexure 1:

8.1.1. Help file for the execution of the software:

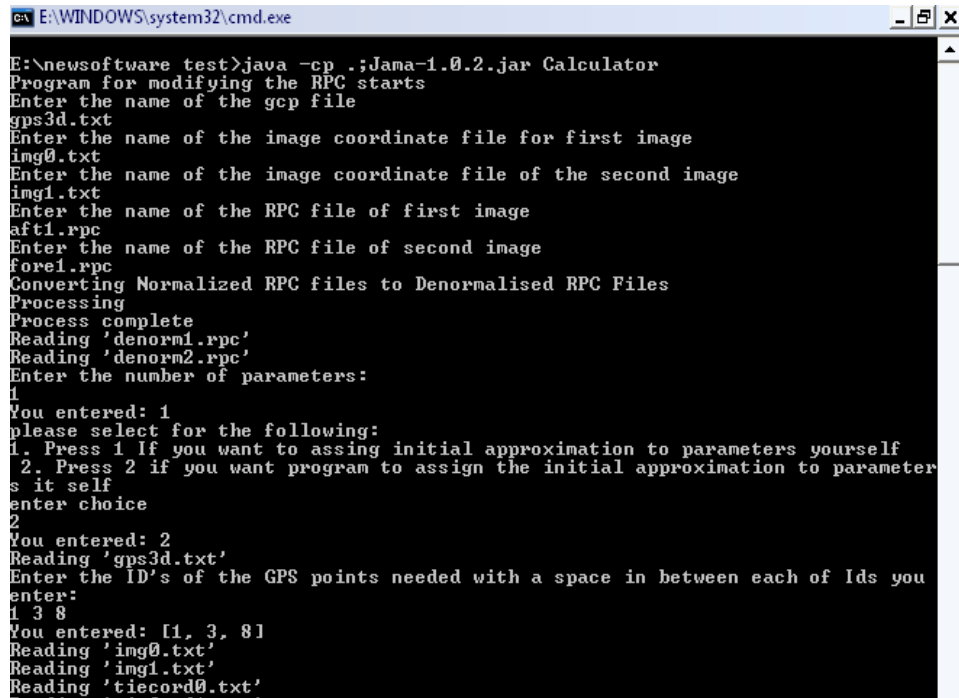
This is an executable document will explain the some coding in steps. This document will also explain how the some of functions work and provides the real time functions which are used.

1. Introduction to my program and how it is executed from the folder.

1. The whole java program is contained in the folder, this folder contains various files most of these files are the input files which are read by the program.
2. There are modules like denormalization, Normalization, Image to object Transformations that have been called into the main class i.e. Calculator .java.
3. There are the class files, which are used for execution of the program.
4. There are java language source files which have the same name as class files and are actually having the code which can be manipulated.
5. There are also a batch files which actually compile the whole code and also runs it.
6. There is one jar file named as jama.jar file, which is actually a class package used for matrix manipulations.
7. The input files are asked from the user where the user has to give the names of GPS coordinate file and 2 image coordinate files with *.txt extension. The RPC files have also to be given the extension of *.RPCs.
8. The class files or java language source files are point3d.java, point2d.java, arrayreader.java, and calculator. java etc
9. Arrayreader.java code is used to read the files and point2d is used for reading (line and sample) from image cord files and poin3d.java are used for reading(lat,long and height) from Gpsid files.
10. The main program is written in calculator. Java and cal. java source code files, which actually gives the whole logic. Calculator. Java gives the main file for de-normalized RPC and cal. java is used for normalized RPC.
11. To execute the program we need to first compile it using compile batch files and then run it using run batch files. When ever we make changes to our source code we need to compile it again and then run it.
12. To execute the program all these files have to be in the same folder and on drive preferably.
13. For executing this we need to install java run time environment on our machine with JDK i.e. Java development KIT which can be easily downloaded from sun Microsystems in very less time.
14. The program returns the output as notepad files which have to be named with valid extensions for both the modified RPCs and also gives us the out put notepad file which has the object coordinates extracted from the modified RPCs.

The program works on the consol where it asks for the input and returns the output in the same folder. The out put console can be seen with the help of these screen shots

A program that could read Raw RPC files, image coordinate files and GCP file from the user. And gives a refined RPC files has been developed. The other product of the program is an 3d object coordinate file which gives x, y, z values which are produced from the modified RPCs and the image coordinates. Here are some of the screen shots of the program console showing various phases of the program.



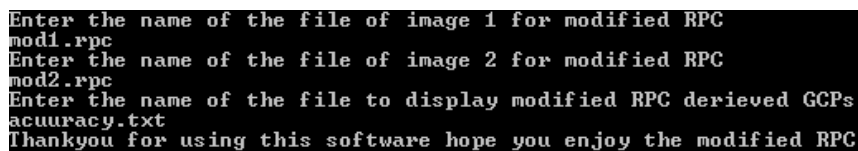
```

C:\E:\WINDOWS\system32\cmd.exe
E:\newsoftware test>java -cp .;Jama-1.0.2.jar Calculator
Program for modifying the RPC starts
Enter the name of the gcp file
gps3d.txt
Enter the name of the image coordinate file for first image
img0.txt
Enter the name of the image coordinate file of the second image
img1.txt
Enter the name of the RPC file of first image
aft1.rpc
Enter the name of the RPC file of second image
fore1.rpc
Converting Normalized RPC files to Denormalised RPC Files
Processing
Process complete
Reading 'denorm1.rpc'
Reading 'denorm2.rpc'
Enter the number of parameters:
1
You entered: 1
Please select for the following:
1. Press 1 if you want to assign initial approximation to parameters yourself
2. Press 2 if you want program to assign the initial approximation to parameters itself
Enter choice
2
You entered: 2
Reading 'gps3d.txt'
Enter the ID's of the GPS points needed with a space in between each of IDs you enter:
1 3 8
You entered: [1, 3, 8]
Reading 'img0.txt'
Reading 'img1.txt'
Reading 'tiecord0.txt'

```

Figure showing screen shot for input in the software console

In the above screen shot taken from the original console screen of the software build we can see that how it asks a user to input the names of the GCP and image coordinate files with *.txt format and the RPC files with *.RPC format. It also lets the user to make a choice for selecting the number of parameters and if he himself wants to assign the initial approximations or wants the program to do it for him. User also provides GCP IDs with a space between each ID.



```

Enter the name of the file of image 1 for modified RPC
mod1.rpc
Enter the name of the file of image 2 for modified RPC
mod2.rpc
Enter the name of the file to display modified RPC derived GCPs
accuracy.txt
Thankyou for using this software hope you enjoy the modified RPC

```

Figure showing screen shots of output file names

In this screen shot the software asks the user to enter the names of the modified RPC files and also the file which generates 3d object coordinates from the modified RPCs.

```

C:\E:\WINDOWS\system32\cmd.exe
The matrix of partial differntiation is as follows
-75508.9286      0.0000      0.0000      0.0000
  0.0000    -10096.6827      0.0000      0.0000
  0.0000      0.0000    -112105.1946      0.0000
  0.0000      0.0000      0.0000    -8386.3051
-78021.4305      0.0000      0.0000      0.0000
  0.0000    -9587.3838      0.0000      0.0000
  0.0000      0.0000    -147892.5515      0.0000
  0.0000      0.0000      0.0000    -8458.4553
-83510.9677      0.0000      0.0000      0.0000
  0.0000    -9606.0057      0.0000      0.0000
  0.0000      0.0000    -119595.6975      0.0000
  0.0000      0.0000      0.0000    -8608.3492

The Y matrix is as follows
-13.9
-16.9
-25.2
 7.7
-14.7
-15.9
-26.1
 7.5
-14.6
-15.1
-25.5
 6.9

A transpose matrix is
0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000
0.0000 0.0000 0.0000 0.0000

Identity check
Inverse matrix into linear matix Y_ is:
The final solution matrix is
 0.0
 0.0
 0.0
-0.0

```

Figure screen shot of the Matrices

The above screen shot shows the matrixes which are used in the process of the modification here.

The solution matrix is converged by iterating the above so that its value comes close to 0.

➤ Some functions used in the code are explained below:

- Reads the RPC files.

Reads image0.Rpc
Reads image1.Rpc

When reading the RPCs files we assign

$a[0] \text{---} a[19]$, $b[0] \text{---} b[19]$, $c[0] \text{---} c[19]$, $d[0] \text{---} d[19]$, values for image0 Rpc files.

$a1[0] \text{---} a1[19]$, $b1[0] \text{---} b1[19]$, $c1[0] \text{---} c1[19]$, $d1[0] \text{---} d1[19]$, values For image1 Rpc file

- Asks the user to input the no of parameters. If the user inputs the number the program asks user to put approximations for the parameters.
- If user put 1 for no of parameters he will be asked to input the approximations manually or the computer can automatically input the same are represented by

$Aim2[1] = ?$ $Aim1[1] = ?$ $Aim2[2] = ?$ $Aim1[2] = ?$
 $Bim2[1] = ?$ $Bim1[1] = ?$ $Bim2[2] = ?$ $Bim1[2] = ?$
 $Cim2[1] = ?$ $Cim1[1] = ?$ $Cim2[2] = ?$ $Cim1[2] = ?$
 $Dim2[1] = ?$ $Dim1[1] = ?$ $Dim2[2] = ?$ $Dim1[2] = ?$ -----etc.

Here im1 represents image0 and im2 represents image 1.

- We have created functions, which represent the polynomial equations of 3rd order

Our polynomial equations are as following:

$$a0+a1x+a2y+a3z+a4xy+a5xz+a6yz+a7x^2+a8y^2+a9z^2+a10xyz+a11x^3+a12xy^2+a13xz^2+a14yx^2+a15y^3+a16yz^2+a17x^2z+a18y^2z+a19z^3 \quad eq-1$$

$$b0+b1x+b2y+b3z+b4xy+b5xz+b6yz+b7x^2+b8y^2+b9z^2+b10xyz+b11x^3+b12xy^2+b13xz^2+b14yx^2+b15y^3+b16yz^2+b17x^2z+b18y^2z+b19z^3 \quad eq-2$$

$$c0+c1x+c2y+c3z+c4xy+c5xz+c6yz+c7x^2+c8y^2+c9z^2+c10xyz+c11x^3+c12xy^2+c13xz^2+c14yx^2+c15y^3+c16yz^2+c17x^2z+c18y^2z+c19z^3 \quad eq-3$$

$$d0+d1x+d2y+d3z+d4xy+d5xz+d6yz+d7x^2+d8y^2+d9z^2+d10xyz+d11x^3+d12xy^2+d13xz^2+d14yx^2+d15y^3+d16yz^2+d17x^2z+d18y^2z+d19z^3 \quad eq-4$$

All the polynomial coefficients a0—a19, b0-b19, c0-c19, d0-d19 are extracted from the image0 Rpc files.

$$a10+a11x+a12y+a13z+a14xy+a15xz+a16yz+a17x^2+a18y^2+a19z^2+a110xyz+a111x^3+a112xy^2+a113xz^2+a114yx^2+a115y^3+a116yz^2+a117x^2z+a118y^2z+a119z^3 \quad eq- 5$$

$$b10+b11x+b12y+b13z+b14xy+b15xz+b16yz+b17x^2+b18y^2+b19z^2+b110xyz+b111x^3+b112xy^2+b113xz^2+b114yx^2+b115y^3+b116yz^2+b117x^2z+b118y^2z+b119z^3 \quad eq-6$$

$$c10+c11x+c12y+c13z+c14xy+c15xz+c16yz+c17x^2+c18y^2+c19z^2+c110xyz+c111x^3+c112xy^2+c113xz^2+c114yx^2+c115y^3+c116yz^2+c117x^2z+c118y^2z+c119z^3 \quad eq-7$$

$$d10+d11x+d12y+d13z+d14xy+d15xz+d16yz+d17x^2+d18y^2+d19z^2+d110xyz+d111x^3+d112xy^2+d113xz^2+d114yx^2+d115y^3+d116yz^2+d117x^2z+d118y^2z+d119z^3 \quad eq-8$$

All the polynomial coefficients a10—a119, b10-b119, c10-c119, d10-d119 are extracted from the image1 Rpc files.

Similarly we do the same thing for image1 Rpc files.

Now the functions that I have made to generate such polynomials are given as follows

First function :

```
public static double[] factors(double x, double y, double z) {
    double [] factors = {1, x, y, z, x * y, x * z, y * z, x * x, y * y,
        z * z, x * y * z, x * x * x, x * y * y, x * z * z, x * x * y,
        y * y * y, y * z * z, x * x * z, y * y * z, z * z * z };
    return factors;
}
```

This function double factors[] which has data type as double takes x y z as its parameters and puts them in an array double factors[].

This function will factors will give us the array of double which will return the factors[0]=1;

Factor[1] = x;

Factors[2] = y;

Factors[3] = z;

Factors[4] = x*y ;

And so on----

Second function: this function calls the factors function in its body.

```
public static double P(double x, double y, double z, double[] my_array) {
    if (my_array.length < 20) {
        System.out.println("a should have 20 elements");
        return 0;
    }

    double[] f = factors(x, y, z);

    double result = 0;
    for (int i = 0; i < f.length; ++i)
        result += my_array[i] * f[i];

    return result;
}
```

This function P actually takes x y z as parameters and also myarray[] as the parameter and runs from i=0 to i = factors length .

Its gives user the result as the product of my_array[] and factor[]

Here my_array[] has the value of RPC coefficients which are read from Rpc files.

So we can actually put array of a [] and also array of b [] in place of my_array [] when calling this function.

So $p(x, y, z, a)$ will give us **eq-1** and the function $p(x, y, z, b)$ will give us **eq-2**
And so on.

So we attain the polynomial equation from these functions.

- Reads the GpsId file
- Asks the user to enter the ids of the GCPs

The user can enter the ids of the GCPs by giving the space between the 2 ids

For example if the user enters 1 3 4 5 then the program will process the GCPs id 1 first then it will repeat the same process for id 3 and id 4.

So when the user takes up the GCPs id 1 our program will start its calculations

For id 1 my program will take the x, y, z values from the GPS id file.

The code to do the following is given as under

Reading the class point 3d and assigning the file values the variables x, y, z

```
for (Integer id : gpsIDs) {  
    Point3d point = ht.get(id);  
    double x = point.x;  
    double y = point.y;  
    double z = point.z;  
    System.out.println("Processing id="+id+"x="+x+"y="+y+" z="+z);  
}
```

The variables are used to hold the values.. These variables can be printed at any place of the code using system.out.println command.

- Now the program will read imagecord0 file and then imagecord1 file from which it will get the values for line and sample.

Reading imagecord0 file

Reading imagecord1 file

Now the variable l with datatype as double gets the value of line from imagecord0 file for the same id as that of the gps id in this case it is 1 as we are working on id 1.

Also variable s with datatype as double gets the value of sample for imagecord0 file for id 1 same as that of gps id file.

```
double l = imc0.get(id).x;  
double s = imc0.get(id).y;
```

Now the variable l1 with datatype as double gets the value of line from imagecord1 file for the same id as that of the gps id in this case it is 1 as we are working on id 1.

Also variable s with datatype as double gets the value of sample for imagecord1 file for id 1 same as that of gaps id file

```
double l1 = imc1.get(id).x;  
double s1 = imc1.get(id).y;
```

- Now for calculating the Lcomputed and Scomputed values which as a whole depends on the number of parameters.

So we need to add the value of the approximations of the parameters that we have selected to the **eq-1 to eq-8** and it needs to follow the same order as the polynomial coefficient itself does

So actually if we select 1 parameter for approximation then we need to approximate values for?
Let's approximate the values as

$$\text{Aim1 [1]} = 1 \quad \text{Bim1 [1]} = 0 \quad \text{Cim1 [1]} = 1 \quad \text{Dim1 [1]} = 0$$

Which are approximations for RPC of image0.RPC files and

$$\text{Aim2 [1]} = 1, \text{Bim2 [1]} = 0, \text{Cim2 [1]} = 1 \quad \text{Dim2 [1]} = 0$$

Which are approximations for RPC values of image1.RPc file?

Now if we are putting the array of Am1[i] in the place of a[i] array in the function P that we have already created it gives us values which can be added into the polynomial equations .

So the variable Lcomp of datatype as double will hold the following values

$$\text{eq-1} + \text{P(x, y, z, Aim1)} / \text{eq-2} + \text{P(x, y, z, Bim1)}$$

This can be expanded as follows

$$\begin{aligned} & a_0 + a_1x + a_2y + a_3z + a_4xy + a_5xz + a_6yz + a_7x^2 + a_8y^2 + a_9z^2 + a_{10}xyz + a_{11}x^3 + a_{12}xy^2 + a_{13}xz^2 + a_{14}yx^2 + a_{15}y^3 + \\ & a_{16}yz^2 + a_{17}x^2z + a_{18}y^2z + a_{19}z^3 + 1 \quad / \text{ (divided by)} \end{aligned}$$

$$\begin{aligned} & b_0 + b_1x + b_2y + b_3z + b_4xy + b_5xz + b_6yz + b_7x^2 + b_8y^2 + b_9z^2 + b_{10}xyz + b_{11}x^3 + b_{12}xy^2 + b_{13}xz^2 + b_{14}yx^2 + b_{15}y^3 + \\ & b_{16}yz^2 + b_{17}x^2z + b_{18}y^2z + b_{19}z^3 + 0 \end{aligned}$$

Similarly Scomputed will hold the value of **eq-3+P(x, y, z, Aim1)/eq-4+P(x, y, z, Bim1)** which can be expanded as follows

$$\begin{aligned} & c_0 + c_1x + c_2y + c_3z + c_4xy + c_5xz + c_6yz + c_7x^2 + c_8y^2 + c_9z^2 + c_{10}xyz + c_{11}x^3 + c_{12}xy^2 + c_{13}xz^2 + c_{14}yx^2 + c_{15}y^3 + \\ & c_{16}yz^2 + c_{17}x^2z + c_{18}y^2z + c_{19}z^3 + 1 \quad / \text{ (divided by)} \end{aligned}$$

$$\begin{aligned} & d_0 + d_1x + d_2y + d_3z + d_4xy + d_5xz + d_6yz + d_7x^2 + d_8y^2 + d_9z^2 + d_{10}xyz + d_{11}x^3 + d_{12}xy^2 + d_{13}xz^2 + d_{14}yx^2 + d_{15}y^3 + \\ & d_{16}yz^2 + d_{17}x^2z + d_{18}y^2z + d_{19}z^3 + 0. \end{aligned}$$

The code that has been used for the same is as follows:

The same applies for values of image1.RPC files

Where the variables l1comp and s1 computed stops the values.

$$\begin{aligned} \text{double Lcomp} &= (\text{P(x, y, z, a)} + \text{P(x, y, z, Aim1)}) / (\text{P(x, y, z, b)} + \text{P(x, y, z, Bim1)}); \\ \text{double scomp} &= (\text{P(x, y, z, c)} + \text{P(x, y, z, Cim1)}) / (\text{P(x, y, z, d)} + \text{P(x, y, z, Dim1)}); \\ \text{double l1comp} &= (\text{P(x, y, z, a1)} + \text{P(x, y, z, Aim2)}) \end{aligned}$$


```

/ (P(x, y, z, b1) + P(x, y, z, Bim2) );
double s1comp = (P(x, y, z, c1) + P(x, y, z, Cim2))
/ (P(x, y, z, d1) + P(x, y, z, Dim2) );

```

- This step will be telling us about the differentiation functions that I have made .

For making A matrix we need to use the differentiation functions. I have used symbolic differentiation method and numeric differentiation functions given at the end. Here I have tried to make functions which can differentiate with respect to the approximations we have made.

These functions have been given at the last of the code. For example this can be one of the examples
public static double dLdAA(double x, double y, double z, double [] A, double [] B, double [] a, double [] b, int index, int n) {

```

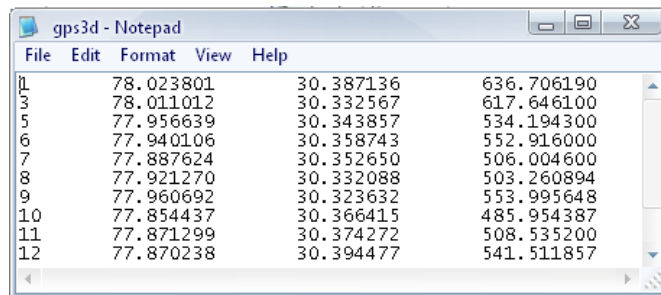
double [] c1 = A;
for(int i=0;i<parasize;i++)
System.out.println("A "+" "+A[])
for(int i=0;i<parasize;i++)
System.out.println("c1 "+" "+c1[i]);
double [] d1 = new double[parasize];
d1[index] = c1[index] + eps;

double xy = P(x, y, z, a);
double xyy = P(x, y, z, b);
double xla = pd(x, y, z, A,index);
double xl = pd(x, y, z, d1,index);
double xll = ppp(x, y, z, A, n);
double xb = ppp(x, y, z, B,n);
double xlaa = (ppp(x, y, z, A, n) - pd(x, y, z, A,index));
double sub = d1[index] - A[index];
double R = (xy + xlaa + xl) / (xyy + xb);
double r = (xy + xll) / (xyy + xb);
double result = (R-r) / sub;

return result;
}

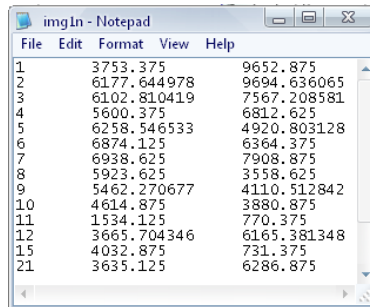
```

- The program can read lat long values in degrees and height in meters the example for the object and the image coordinates can be shown as under.



Line	Latitude	Longitude	Height
1	78.023801	30.387136	636.706190
3	78.011012	30.332567	617.646100
5	77.956639	30.343857	534.194300
6	77.940106	30.358743	552.916000
7	77.887624	30.352650	506.004600
8	77.921270	30.332088	503.260894
9	77.960692	30.323632	553.995648
10	77.854437	30.366415	485.954387
11	77.871299	30.374272	508.535200
12	77.870238	30.394477	541.511857

Figure: showing the layout of GCP file that program accepts



The image shows a Notepad window titled 'imgIn - Notepad'. The window contains a table with three columns: a line number, a first coordinate value, and a second coordinate value. The data is as follows:

1	3753.375	9652.875
2	6177.644978	9694.636065
3	6102.810419	7567.208581
4	5600.375	6812.625
5	6258.546533	4920.803128
6	6874.125	6364.375
7	6938.625	7908.875
8	5923.625	3558.625
9	5462.270677	4110.512842
10	4614.875	3880.875
11	1534.125	770.375
12	3665.704346	6165.381348
15	4032.875	731.375
21	3635.125	6286.875

Figure: showing the layout of image coordinate file that program accepts

8.2. Annexure 2:

Here the coding of the main class file is given as under.

8.2.1. The code for the main class file:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Hashtable;
import java.util.StringTokenizer;
import Jama.Matrix;
import java.util.*;
import java.io.*;

public class Calculator {
    public static int parasize=0;
    public static double eps = 0.0000000001;
    /**
     * @param args
     */
    public static void main(String[] args)throws IOException {
        System.out.println("Program for modifying the RPC starts");
        Thread thred = Thread.currentThread();

        int ch;
        int chr = 0;
        int chh = 0;
        String im0 = "";
        String im1 = "";
        String imcd0 = "";
        String imcd1 = "";
        String gps = "";
        String img_de0 = "denorm1.rpc";
        String img_de1 = "denorm2.rpc";
        try
        {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter the name of the gcp file");
            gps = br.readLine();
        }
        catch(Exception e)
        {
            System.out.println("Error" +e);
        }
        try
        {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter the name of the image coordinate file for first image ");
            imcd0 = br.readLine();
        }
        catch(Exception e)
        {
            System.out.println("Error" +e);
        }
        try
        {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter the name of the image coordinate file of the second image");
            imcd1 = br.readLine();
        }
        catch(Exception e)
        {
            System.out.println("Error" +e);
        }
    }
}
```

```

        try
        {
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            System.out.println("Enter the name of the RPC file of first image ");
            im0 = br.readLine();
        }
        catch(Exception e)
        {
            System.out.println("Error" +e);
        }
    try
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the name of the RPC file of second image ");
        im1 = br.readLine();
    }
    catch(Exception e)
    {
        System.out.println("Error" +e);
    }

    try
    {
        func t = new func();

        t.Denorm(im0,img_de0);

        t.Denorm(im1,img_de1);

        System.out.println("Converting Normalized RPC files to Denormalised RPC Files");
        System.out.println("Processing");
        Thread.sleep(7000);
        System.out.println("Process complete");
    }
    catch(InterruptedException e)
    {
        System.out.print("Main thread interrupted");
    }

    ArrayReader ar = new ArrayReader(img_de0);
    ar.skipLines(10);
    double[] t = ar.readArray(80);
    double[] a = new double[20];
    double[] b = new double[20];
    double[] c = new double[20];
    double[] d = new double[20];
    int j;
    for( j=0;j<20;j++)
    {
        a[j]=t[j];
        //System.out.println("a[j]" +a[j]);
    }

    for( int k=0;k<20;k++,j++)
    {
        b[k]=t[j];
        //System.out.println("b[k]" +b[k]);
    }

    for( int l=0;l<20;l++,j++)
    {
        c[l]=t[j];
        //System.out.println("c[l]" +c[l]);
    }

    for( int m=0;m<20;m++,j++)
    {
        d[m]=t[j];
        //System.out.println("d[m]" +d[m]);
    }

    ArrayReader ar1 = new ArrayReader(img_de1);

```

```

        ar1.skipLines(10);
        double[] t1 = ar1.readArray(80);
        double[] a1 = new double[20];
        double[] b1 = new double[20];
        double[] c1 = new double[20];
        double[] d1 = new double[20];

        for(j=0;j<20;j++)
        {
            a1[j]=t1[j];
            //System.out.println("a1[j]"+a1[j]);
        }

        for(int k=0;k<20;k++,j++)
        {
            b1[k]=t1[j];
            //System.out.println("b1[k]"+b1[k]);
        }

        for(int l=0;l<20;l++,j++)
        {
            c1[l]=t1[j];
        }

        for(int m=0;m<20;m++,j++)
        {
            d1[m]=t1[j];
        }

        /*double [] Aim1 = new double [];
        double [] Bim1 = new double [20];
        double [] Cim1 = new double [20];
        double [] Dim1 = new double [20];
        double [] Aim2 = new double [20];
        double [] Bim2 = new double [20];
        double [] Cim2 = new double [20];
        double [] Dim2 = new double [20];

        for (int i = 0; i < 20; i++)
        {
            Aim1[i] = 0;
            Bim1[i] = 0;
            Cim1[i] = 0;
            Dim1[i] = 0;
            Aim2[i] = 0;
            Bim2[i] = 0;
            Cim2[i] = 0;
            Dim2[i] = 0;
        }
        */
        //System.out.println("please read carefully before starting up with the program.This program will ask for tie points please enter 36 , this
        does not effect the program as tie points come later which is still under construction,you can enter gps ids by giving the space between the ids
        ");
        //System.out.println("This program gives you every step in between you can check them by holding the scrol bar so that you can
        check each matrices formed then release it slowly and slowly so that it is possible for you to get the whole program");

        BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

        int nrParams = readInt("Enter the number of parameters:", in);
        parasize=nrParams;

        double [] Aim1 = new double [nrParams];
        double [] Bim1 = new double [nrParams];
        double [] Cim1 = new double [nrParams];
        double [] Dim1 = new double [nrParams];
        double [] Aim2 = new double [nrParams];
        double [] Bim2 = new double [nrParams];
        double [] Cim2 = new double [nrParams];
        double [] Dim2 = new double [nrParams];
        for (int i = 0; i < nrParams; i++)
        {

```

```

        Aim1[i] = 0;
        Cim1[i] = 0;

        Aim2[i] = 0;
        Cim2[i] = 0;}
    for (int i = 0; i < nrParams; i++)
{
    Bim1[i] = 0;

    Dim1[i] = 0;
    Bim2[i] = 0;
    Dim2[i] = 0;}

System.out.println("please select for the following: \n1. Press 1 If you want to assing initial approximation to parameters yourself\n 2. Press 2
if you want program to assign the initial approximation to parameters it self");

ch=readInt("enter choice ",in);
switch(ch)
{
    case 1:
    {
        for (int i = 0; i<nrParams; i++) {
            Aim1[i] = readDouble("Enter parameter Aim1["+i+"]: ", in);
            Bim1[i] = readDouble("Enter parameter Bim1["+i+"]: ", in);
            Cim1[i] = readDouble("Enter parameter Cim1["+i+"]: ", in);
            Dim1[i] = readDouble("Enter parameter Dim1["+i+"]: ", in);

            Aim2[i] = readDouble("Enter parameter Aim2["+i+"]: ", in);
            Bim2[i] = readDouble("Enter parameter Bim2["+i+"]: ", in);
            Cim2[i] = readDouble("Enter parameter Cim2["+i+"]: ", in);
            Dim2[i] = readDouble("Enter parameter Dim2["+i+"]: ", in);
        }
        break;
    }

    case 2:

    {

        for (int i = 0; i <nrParams; i++)
        {
            Aim1[i] = 0.000000001;
            Cim1[i] = 0.000000001;

            Aim2[i] = 0.000000001;
            Cim2[i] = 0.000000001;

        }

        for (int i = 1; i < nrParams; i++)
        {
            Bim1[i] = 0.000000001;

            Dim1[i] = 0.000000001;
            Bim2[i] = 0.000000001;
            Dim2[i] = 0.000000001;

        }
        break;
    }
}

    if(nrParams == 1)
    {
        ArrayList y1List = new ArrayList<Double>();
        ArrayList y2List = new ArrayList<Double>();
        ArrayList yt1List = new ArrayList<Double>();
        ArrayList yt2List = new ArrayList<Double>();
        PointReader pr = new PointReader();

```

```

        Hashtable <Integer, Point3d> ht = pr.readPoints3d(gps);
        ArrayList <Integer> gpsIDs = readArray("Enter the ID's of the GPS points needed with a space in between each of
Ids you enter:", in);
        Hashtable <Integer, Point2d> imc0 = pr.readPoints2d(imcd0);
        Hashtable <Integer, Point2d> imc1 = pr.readPoints2d(imcd1);
        Hashtable <Integer, Point2d> tiec0 = pr.readPoints2d("tiecord0.txt");
        Hashtable <Integer, Point2d> tiec1 = pr.readPoints2d("tieCord1.txt");
        Matrix AA = new Matrix(gpsIDs.size() * 4 , nrParams * 4);
        int matrixRow = 0;

    if (gpsIDs != null) {
        for (Integer id : gpsIDs) {
            Point3d point = ht.get(id);
            double x = point.x;
            double y = point.y;
            double z = point.z;
            System.out.println("Processing id="+id+" x="+x+" y="+y+" z="+z);

            double lme = P(x, y, z, a) / P(x, y, z, b);
            double sme = P(x, y, z, c) / P(x, y, z, d);

            double l = imc0.get(id).x;
            double s = imc0.get(id).y;
            double lmea = P(x, y, z, a);
            double lmeb = P(x, y, z, b);

            double lmec = P(x, y, z, c);
            double lmed = P(x, y, z, d);
            for(int dd = 0; dd < 20; dd++)
                System.out.println(" a1[d] " + " dd"+" " +a1[dd]);
            double lmea1 = P(x, y, z, a1);
            double lmeb1 = P(x, y, z, b1);

            double lmec1 = P(x, y, z, c1);
            double lmed1 = P(x, y, z, d1);
            //double l1 = P(x, y, z, a1) / P(x, y, z, b1);
            //double s1 = P(x, y, z, c1) / P(x, y, z, d1);
            double l1 = imc1.get(id).x;
            double s1 = imc1.get(id).y;
            double lme1 = P(x, y, z, a1) / P(x, y, z, b1);
            double sme1 = P(x, y, z, c1) / P(x, y, z, d1);

            //System.out.println("L = " + l);
            //System.out.println("S = " + s);

            //System.out.println("L1 = " + l1);
            //System.out.println("S1 = " + s1);

            double Aim = ppp(x, y, z, Aim1, parasize);
            double lcal = (P(x, y, z, a) + ppp(x, y, z, Aim1, parasize));
            double scal = (P(x, y, z, c) + ppp(x, y, z, Cim1, parasize));
            double lcomp = (P(x, y, z, a) + ppp(x, y, z, Aim1, parasize)) / (P(x, y, z, b));
            double scomp = (P(x, y, z, c) + ppp(x, y, z, Cim1, parasize)) / (P(x, y, z, d));
            double l1cal = (P(x, y, z, a1) + ppp(x, y, z, Aim2, parasize));
            double s1cal = (P(x, y, z, c1) + ppp(x, y, z, Cim2, parasize));
            double l1comp = (P(x, y, z, a1) + ppp(x, y, z, Aim2, parasize)) / (P(x, y, z, b1));
            double s1comp = (P(x, y, z, c1) + ppp(x, y, z, Cim2, parasize)) / (P(x, y, z, d1));

            double y11 = l - lcomp;
            double y12 = s - scomp;
            double y21 = l1 - l1comp;
            double y22 = s1 - s1comp;

            y1List.add(y11);
            y1List.add(y12);
            y1List.add(y21);
            y1List.add(y22);
            /* System.out.println("Lmea = " + " " + lmea);
            System.out.println("Lmeb = " + " " + lmeb);
            System.out.println("Lmec = " + " " + lmec);
            System.out.println("Lmed = " + " " + lmed);
            System.out.println("Lme = " + lme);
            System.out.println("sme = " + sme);
            System.out.println("Lmea1 = " + " " + lmea1);
            System.out.println("Lmeb1 = " + " " + lmeb1);
            System.out.println("Lmec1 = " + " " + lmec1);

```

```

        System.out.println("lmed1 = " + " " + lmed1);
    System.out.println("Lme1 = " + lme1);
        System.out.println("sme1 = " + sme1);
    System.out.println("Aim = " + " " + Aim);
    System.out.println("lcal = " + " " + lcal);
    System.out.println("scal = " + " " + scal);
    System.out.println("l1cal = " + " " + l1cal);
    System.out.println("s1cal = " + " " + s1cal);
        System.out.println("Lcomp = " + " " + lcomp);
        System.out.println("Scomp = " + " " + scomp);
    System.out.println("L1comp = " + " " + l1comp);
        System.out.println("S1comp = " + " " + s1comp);
        System.out.println("Y11 = " + y11);
        System.out.println("Y12 = " + y12);
        System.out.println("Y21 = " + y21);
        System.out.println("Y22 = " + y22);*/

    double dlai=0;
    double dla1i=0;
    double dlbi=0;
    double dlbi1=0;
    double dlci=0 ;
    double dldi=0;

        double dlci1=0 ;

    double dld1i=0;

    double dsci=0 ;
    double dski=0;

        double dsci1=0 ;

    double dsd1i=0;

for (int i = 0; i < nrParams; ++i)
{

    dlai= dLdA(x,y,z,Aim1,Bim1,a,b,i,nrParams);
    //System.out.println("i"+" "+i);
    //System.out.println("dla"+" "+i+" "+dlai);

    AA.set(matrixRow, i*2, dlai);
        AA.set(matrixRow, (i*2)+1, 0);
        AA.set(matrixRow, (i*2)+2, 0);
        AA.set(matrixRow, (i*2)+3, 0);
}

        ++matrixRow;

        for (int i = 0; i < nrParams; ++i)

        {

            dsci= dLdA(x,y,z,Cim1,Dim1,c,d,i,nrParams);

            //System.out.println("i"+" "+i);
            //System.out.println("dsc"+" "+i+" "+dsci);

            AA.set(matrixRow, i*2, 0);
                AA.set(matrixRow, (i*2)+1, dsci);
                AA.set(matrixRow, (i*2)+2, 0);
                AA.set(matrixRow, (i*2)+3, 0);
        }

            ++matrixRow;

        for (int i =0; i < nrParams; i++)
        {

            dla1i= dLdA(x,y,z,Aim2,Bim2,a1,b1,i,nrParams);;

```

```

//System.out.println("i  "+i);

//System.out.println("dla1"+i+" "+dla1i);

AA.set(matrixRow, (i*2), 0);
    AA.set(matrixRow, (i*2)+1, 0);
    AA.set(matrixRow, (i*2)+2, dla1i);
    AA.set(matrixRow, (i*2)+3, 0);
}

                                ++matrixRow;

    for (int i =0; i < nrParams; i++)
    {

        dsc1i= dLdA(x,y,z,Cim2,Dim2,c1,d1,i,nrParams);
        //System.out.println("i"+" "+i);
        //System.out.println("dsc1"+" "+i+" "+dsc1i);

AA.set(matrixRow, (i*2), 0);
        AA.set(matrixRow, (i*2)+1, 0);
        AA.set(matrixRow, (i*2)+2, 0);
        AA.set(matrixRow, (i*2)+3, dsc1i);

        }
        ++matrixRow;
    }

System.out.println("The matrix of partial differntiation is as follows");
//AA.print(AA.getRowDimension(), AA.getColumnDimension());
//FOS.close();
int size=y1List.size() + y2List.size();
    Matrix Y_ = new Matrix(y1List.size(), 1);
    int k=0;
        for (int i = 0; i < y1List.size(); ++i,k++)
            Y_.set(k, 0, ((Double)(y1List.get(i))).doubleValue());

        //for (int i = 0; i < y2List.size(); ++i,k++)
            //Y_.set(k, 0, ((Double)(y2List.get(i))).doubleValue());

/* double arr[] = new double[size];
System.out.println("size  "+size);
int i=0,k=0;
for ( i = 0,k=0; i < y1List.size(); ++i,k++)
    arr[k]= ((Double) (y1List.get(i))).doubleValue();
for (i=0; i < y2List.size(); ++i,k++)
    arr[k]= ((Double) (y2List.get(i))).doubleValue();
for (i=0; i < yt1List.size(); ++i,k++)
    arr[k]= ((Double) (yt1List.get(i))).doubleValue();
for (i=0; i < yt2List.size(); ++i,k++)
    arr[k]= ((Double) (yt2List.get(i))).doubleValue();

for (i=0;i<size;i++)
{
    System.out.println(arr[i]);
}

*/

//old code
System.out.println("The Y matrix is as follows");
//Y_.print(y1List.size(), 1);

//for (int i = 0; i < Y_.getRowDimension(); ++i)

```

```

//System.out.println("Y_["+i+",0]="+Y_.get(i, 0));

//for calculation of final formula
System.out.println("A transpose matrix is");
Matrix At = AA.transpose();
// At.print(At.getRowDimension(), At.getColumnDimension());
//System.out.println("A transpose into A matrix is");
Matrix AtA = At.times(AA);
//AtA.print(AtA.getRowDimension(), AtA.getColumnDimension());
//System.out.println("A transpose into A whole inverse matrix is");
Matrix inv = AtA.inverse();
inv.print(inv.getRowDimension(), inv.getColumnDimension());

System.out.println("Identity check");
Matrix Iden = inv.times(AtA);
//Iden.print(Iden.getRowDimension(), Iden.getColumnDimension());

System.out.println("Inverse matrix into linear matrix Y_ is:");
Matrix invAt = inv.times(At);
//invAt.print(invAt.getRowDimension(), invAt.getColumnDimension());
System.out.println("The final solution matrix is");
Matrix sol = invAt.times(Y_);
sol.print(sol.getRowDimension(), sol.getColumnDimension());

double soll[] = new double[sol.getRowDimension()];
for(int i = 0; i < sol.getRowDimension(); i++)
{
    int jj = 0;

    soll[i] = sol.get(i,jj);
}

for(int i = 0; i < sol.getRowDimension(); i++){
System.out.println("soll" +i+" "+soll[i]);}

Aim1[0] = soll[0] + Aim1[0];
Cim1[0] = soll[1] + Cim1[0];

Aim2[0] = soll[2] + Aim2[0];
Cim2[0] = soll[3] + Cim2[0];

/*System.out.println("Aim1"+0+" "+Aim1[0]);

System.out.println("Cim1"+0+" "+Cim1[0]);
System.out.println("Aim2"+0+" "+Aim2[0]);
System.out.println("Cim2"+0+" "+Cim2[0]);*/

}

chr=readInt("How many times you want to run the loop ",in);
int counter = 0;
while(counter <= chr)

{

counter = counter+1;

y1List = new ArrayList<Double>();
//System.out.println("size of list"+" "+y1List.size());
matrixRow = 0;

for (Integer id : gpsIDs) {
    Point3d point = ht.get(id);
    double x = point.x;
    double y = point.y;
    double z = point.z;
    System.out.println("Processing id="+id+" x="+x+" y="+y+" z="+z);

```

```

double lme = P(x, y, z, a) / P(x, y, z, b);
double sme = P(x, y, z, c) / P(x, y, z, d);

double l = imc0.get(id).x;
double s = imc0.get(id).y;
double lmea = P(x, y, z, a);
double lmeb = P(x, y, z, b);

double lmec = P(x, y, z, c);
double lmed = P(x, y, z, d);
for(int dd = 0; dd < 20; dd++)
System.out.println(" a1[d] " + " dd" + " " + a1[dd]);
double lmea1 = P(x, y, z, a1);
double lmeb1 = P(x, y, z, b1);
double lmec1 = P(x, y, z, c1);
double lmed1 = P(x, y, z, d1);
//double l1 = P(x, y, z, a1) / P(x, y, z, b1);
//double s1 = P(x, y, z, c1) / P(x, y, z, d1);
double l1 = imc1.get(id).x;
double s1 = imc1.get(id).y;
double lme1 = P(x, y, z, a1) / P(x, y, z, b1);
double sme1 = P(x, y, z, c1) / P(x, y, z, d1);

//System.out.println("L = " + l);
//System.out.println("S = " + s);

//System.out.println("L1 = " + l1);
//System.out.println("S1 = " + s1);

double Aim = ppp(x, y, z, Aim1, parasize);
double lcal = (P(x, y, z, a) + ppp(x, y, z, Aim1, parasize));
double scal = (P(x, y, z, c) + ppp(x, y, z, Cim1, parasize));
double lcomp = (P(x, y, z, a) + ppp(x, y, z, Aim1, parasize)) / (P(x, y, z, b));
double scomp = (P(x, y, z, c) + ppp(x, y, z, Cim1, parasize)) / (P(x, y, z, d));
double l1cal = (P(x, y, z, a1) + ppp(x, y, z, Aim2, parasize));
double s1cal = (P(x, y, z, c1) + ppp(x, y, z, Cim2, parasize));
double l1comp = (P(x, y, z, a1) + ppp(x, y, z, Aim2, parasize)) / (P(x, y, z, b1));
double s1comp = (P(x, y, z, c1) + ppp(x, y, z, Cim2, parasize)) / (P(x, y, z, d1));

double y11 = l - lcomp;
double y12 = s - scomp;
double y21 = l1 - l1comp;
double y22 = s1 - s1comp;

y1List.add(y11);
y1List.add(y12);
y1List.add(y21);
y1List.add(y22);
/*System.out.println("Lmea = " + " " + lmea);
System.out.println("Lmeb = " + " " + lmeb);
System.out.println("Lmec = " + " " + lmec);
System.out.println("Lmed = " + " " + lmed);
System.out.println("Lme = " + lme);
System.out.println("sme = " + sme);
System.out.println("Lmea1 = " + " " + lmea1);
System.out.println("Lmeb1 = " + " " + lmeb1);
System.out.println("Lmec1 = " + " " + lmec1);
System.out.println("Lmed1 = " + " " + lmed1);
System.out.println("Lme1 = " + lme1);
System.out.println("sme1 = " + sme1);
System.out.println("Aim = " + " " + Aim);
System.out.println("lcal = " + " " + lcal);
System.out.println("scal = " + " " + scal);
System.out.println("l1cal = " + " " + l1cal);
System.out.println("s1cal = " + " " + s1cal);
System.out.println("Lcomp = " + " " + lcomp);
System.out.println("Scomp = " + " " + scomp);
System.out.println("L1comp = " + " " + l1comp);
System.out.println("S1comp = " + " " + s1comp);
System.out.println("Y11 = " + y11);
System.out.println("Y12 = " + y12);
System.out.println("Y21 = " + y21);
System.out.println("Y22 = " + y22);*/
double dlai=0;

```

```

//System.out.println("dsc1"+" "+i+" "+dsc1i);

AA.set(matrixRow, (i*2), 0);
AA.set(matrixRow, (i*2)+1, 0);
AA.set(matrixRow, (i*2)+2, 0);
AA.set(matrixRow, (i*2)+3, dsc1i);

    }
    ++matrixRow;
}
System.out.println("The matrix of partial differntiation is as follows");
//AA.print(AA.getRowDimension(), AA.getColumnDimension());
int size=y1List.size() + y2List.size();
Matrix Y_ = new Matrix(y1List.size(), 1);
int k=0;
for (int i = 0; i < y1List.size(); ++i,k++)
    Y_.set(k, 0, ((Double)(y1List.get(i))).doubleValue());

//for (int i = 0; i < y2List.size(); ++i,k++)
//Y_.set(k, 0, ((Double)(y2List.get(i))).doubleValue());

/* double arr[] = new double[size];
System.out.println("size "+size);
int i=0,k=0;
for ( i = 0,k=0; i < y1List.size(); ++i,k++)
    arr[k]= ((Double) (y1List.get(i))).doubleValue();
for (i=0; i < y2List.size(); ++i,k++)
    arr[k]= ((Double) (y2List.get(i))).doubleValue();
for (i=0; i < yt1List.size(); ++i,k++)
    arr[k]= ((Double) (yt1List.get(i))).doubleValue();
for (i=0; i < yt2List.size(); ++i,k++)
    arr[k]= ((Double) (yt2List.get(i))).doubleValue();

for (i=0;i<size;i++)
{
    System.out.println(arr[i]);
}

*/

//old code
System.out.println("The Y matrix is as follows");
//Y_.print(y1List.size(), 1);

//for (int i = 0; i < Y_.getRowDimension(); ++i)
//System.out.println("Y_["+i+",0]="+Y_.get(i, 0));

//for calculation of final formula
System.out.println("A transpose matrix is");
Matrix At = AA.transpose();
//At.print(At.getRowDimension(), At.getColumnDimension());
System.out.println("A transpose into A matrix is");
Matrix AtA = At.times(AA);
//AtA.print(AtA.getRowDimension(), AtA.getColumnDimension());
System.out.println("A transpose into A whole inverse matrix is");
Matrix inv = AtA.inverse();
//inv.print(inv.getRowDimension(), inv.getColumnDimension());

System.out.println("Identity check");
Matrix Iden = inv.times(AtA);
//Iden.print(Iden.getRowDimension(), Iden.getColumnDimension());

System.out.println("Inverse matrix into linear matix Y_ is:");
Matrix invAt = inv.times(At);
//invAt.print(invAt.getRowDimension(), invAt.getColumnDimension());

```

```

        System.out.println("The final solution matrix is");
        Matrix sol = invAt.times(Y_);
        sol.print(sol.getRowDimension(), sol.getColumnDimension());

        System.out.println("program end");

        double soll[] = new double[sol.getRowDimension()];
        for(int i = 0; i < sol.getRowDimension(); i++)
        {
            int jj = 0;

            soll[i] = sol.get(i,jj);
        }
        for(int i = 0; i < sol.getRowDimension(); i++){
            //System.out.println("soll" +i+" "+soll[i]);}

        Aim1[0] = soll[0] + Aim1[0];
        Cim1[0] = soll[1] + Cim1[0];

        Aim2[0] = soll[2] + Aim2[0];
        Cim2[0] = soll[3] + Cim2[0];

        /*System.out.println("Aim1"+0+" "+Aim1[0]);

        System.out.println("Cim1"+0+" "+Cim1[0]);
        System.out.println("Aim2"+0+" "+Aim2[0]);
        System.out.println("Cim2"+0+" "+Cim2[0]);*/

    }

    chh = readInt("enter choice 1 for printing value of changed RPC ",in);

    switch(chh)
    {
case 1:
        {
            System.out.println("program works");

            a[0] = Aim1[0] + a[0];
            c[0] = Cim1[0] + c[0];

            a1[0] = Aim2[0] + a1[0];
            c1[0] = Cim2[0] + c1[0];

            for(int i = 0;i < 20;i++)
            {
                System.out.println("l_NUM_COEFF_" +i+" : " +a[i]);
            }

            for(int i = 0;i < 20;i++)
            {
                System.out.println("l_DEN_COEFF_" +i+" : " +b[i]);
            }
            for(int i = 0;i < 20;i++)
            {
                System.out.println("l_NUM_COEFF_" +i+" : " +c[i]);
            }
            for(int i = 0;i < 20;i++)
            {
                System.out.println("l_DEN_COEFF_" +i+" : " +d[i]);
            }
            for(int i = 0;i < 20;i++)
            {
                System.out.println("l_NUM_COEFF_" +i+" : " +a1[i]);
            }
            for(int i = 0;i < 20;i++)

```

```

{
    System.out.println("l_DEN_COEFF_"+i+" : "+b1[i]);
}
for(int i = 0;i < 20;i++)
{
    System.out.println("l_NUM_COEFF_"+i+" : "+c1[i]);
}
for(int i = 0;i < 20;i++)
{
    System.out.println("l_DEN_COEFF_"+i+" : "+d1[i]);
}
String name = "mod1_den";
display(a,b,c,d,name);
String name2 = "mod2_den";
display(a1,b1,c1,d1,name2);
String mod1 = "";
try
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the name of the file of image 1 for modified RPC");
    mod1 = br.readLine();
}
catch(Exception e)
{
    System.out.println("Error" +e);
}
String mod2 = "";
try
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the name of the file of image 2 for modified RPC");
    mod2 = br.readLine();
}
catch(Exception e)
{
    System.out.println("Error" +e);
}
func last = new func();
last.norm(name,im0,mod1);
last.norm(name2,im1,mod2);

String gcps = "";
try
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the name of the file to display modified RPC derieved GCPs");
    gcps = br.readLine();
}
catch(Exception e)
{
    System.out.println("Error" +e);
}
func acc = new func();
acc.accuracy(mod1,imcd0,mod2,imcd1,gcps);
System.out.println("Thankyou for using this software hope you enjoy the modified RPC");

break;
}

}
}

```

```

}

else if( nrParams > 1)
{
    ArrayList y1List = new ArrayList<Double>();
    ArrayList y2List = new ArrayList<Double>();
    ArrayList yt1List = new ArrayList<Double>();
    ArrayList yt2List = new ArrayList<Double>();

    PointReader pr = new PointReader();
    Hashtable<Integer, Point3d> ht = pr.readPoints3d(gps);
    ArrayList<Integer> gpsIDs = readArray("Enter the ID's of the GPS points needed with a space between:", in);
    Hashtable<Integer, Point2d> imc0 = pr.readPoints2d(imcd0);
    Hashtable<Integer, Point2d> imc1 = pr.readPoints2d(imcd1);
    Hashtable<Integer, Point2d> tiec0 = pr.readPoints2d("tiecord0.txt");
    Hashtable<Integer, Point2d> tiec1 = pr.readPoints2d("tieCord1.txt");
    Matrix A = new Matrix(gpsIDs.size() * 4, (((nrParams - 1) * 8) + 4));
    int matrixRoww = 0;
    if (gpsIDs != null) {
        for (Integer id : gpsIDs) {
            Point3d point = ht.get(id);
            double x = point.x;
            double y = point.y;
            double z = point.z;
            System.out.println("Processing id="+id+" x="+x+" y="+y+" z="+z);

            double lme = P(x, y, z, a1) / P(x, y, z, b1);
            double sme = P(x, y, z, c1) / P(x, y, z, d1);
            double l = imc0.get(id).x;
            double s = imc0.get(id).y;
            double lmea = P(x, y, z, a1);
            double lmeb = P(x, y, z, b1);

            double lmec = P(x, y, z, c1);
            double lmed = P(x, y, z, d1);

            //double l1 = P(x, y, z, a1) / P(x, y, z, b1);
            //double s1 = P(x, y, z, c1) / P(x, y, z, d1);
            double l1 = imc1.get(id).x;
            double s1 = imc1.get(id).y;

            double lme1 = P(x, y, z, a1) / P(x, y, z, b1);
            double sme1 = P(x, y, z, c1) / P(x, y, z, d1);

            //System.out.println("L = " + l);
            //System.out.println("S = " + s);

            //System.out.println("L1 = " + l1);
            //System.out.println("S1 = " + s1);
            double Aim = ppp(x, y, z, Aim1, parasite);
            double la1cal = (P(x, y, z, a1) + ppp(x, y, z, Aim2, parasite));
            double lb1cal = (P(x, y, z, b1) + ppp(x, y, z, Bim2, parasite));
            double sc1cal = (P(x, y, z, c1) + ppp(x, y, z, Cim2, parasite));
            double sd1cal = (P(x, y, z, d1) + ppp(x, y, z, Dim2, parasite));

            double lcomp = (P(x, y, z, a) + ppp(x, y, z, Aim1, parasite)) / (P(x, y, z, b) + ppp(x, y, z,
Bim1, parasite));
            double scomp = (P(x, y, z, c) + ppp(x, y, z, Cim1, parasite)) / (P(x, y, z, d) + ppp(x, y, z,
Dim1, parasite));

            double l1comp = (P(x, y, z, a1) + ppp(x, y, z, Aim2, parasite)) / (P(x, y, z, b1) + ppp(x, y, z, Bim2,
parasite));
            double s1comp = (P(x, y, z, c1) + ppp(x, y, z, Cim2, parasite)) / (P(x, y, z, d1) + ppp(x, y, z,
Dim2, parasite));

            double y11 = l - lcomp;
            double y12 = s - scomp;
            double y21 = l1 - l1comp;

```

```

        double y22 = s1 - s1comp;

        y1List.add(y11);
        y1List.add(y12);
        y1List.add(y21);
        y1List.add(y22);
        /*System.out.println("Lmea = " + lmea);
        System.out.println("Lmeb = " + lmeb);

        System.out.println("Lmec = " + lmeb);
        System.out.println("lmed = " + lmed);
        System.out.println("Lme = " + lme);
        System.out.println("sme = " + sme);
        System.out.println("La1cal = " + la1cal);
        System.out.println("lb1cal = " + lb1cal);
        System.out.println("sc1cal = " + sc1cal);
        System.out.println("sdical = " + sd1cal);
        System.out.println("L1comp = " + l1comp);
        System.out.println("S1comp = " + s1comp);
        System.out.println("Y11 = " + y11);
        System.out.println("Y12 = " + y12);
        System.out.println("Y21 = " + y21);
        System.out.println("Y22 = " + y22);*/

        double dlai=0;
        double dla1i=0;
        double dlbi=0;
        double dlbi=0;
        double dlci=0 ;
        double dldi=0;

        double dlc1i=0 ;

        double dld1i=0;

        double dsci=0 ;
        double dsdi=0;

        double dsc1i=0 ;

        double dsd1i=0;
        int zx = 0;
        int zz = 0;

        int i = 0;
    {

        dlai= dLdAAA(x,y,z,Aim1,Bim1,a,b,i,nrParams);

        //System.out.println("i"+" "+i);
        //System.out.println("dla"+" "+i+" "+dlai);

        A.set(matrixRoww, i, dlai);
        A.set(matrixRoww, i+1, 0);

    }

    for(i = 1; i < nrParams; i++)
    {

        dlai= dLdAAA(x,y,z,Aim1,Bim1,a,b,i,nrParams);

        dlbi= dLdBB(x,y,z,Aim1,Bim1,a,b,i,nrParams);

        //System.out.println("i"+" "+i);
        //System.out.println("dla"+" "+i+" "+dlai);
        //System.out.println("dlb"+" "+i+" "+dlbi);

        A.set(matrixRoww,(i*2)+zx, dlai);
        A.set(matrixRoww, (i*2)+zx+1, dlbi);
        A.set(matrixRoww, (i*2)+zx+2, 0);
        A.set(matrixRoww, (i*2)+zx+3, 0);
    }

```

```

int r = ((i * 2) + (4 + (2 * zz)));
//System.out.println("r" +r);

for(int w = 0; w < r; w++)

{
    A.set(matrixRoww,(i*2)+zx+4+w, 0);

}

zx = zx + 2;
zz = zz + 1;
}

++matrixRoww;

int zw = 0;
int zy = 0;
int ii = 0;
{

    dsci= dLdAAA(x,y,z,Cim1,Dim1,c,d,ii,nrParams);

    //System.out.println("ii"+" "+ii);
    //System.out.println("dsc"+" "+ii+" " +dsci);

    A.set(matrixRoww, ii, 0);
    A.set(matrixRoww, ii+1, dsci);

}

for(ii = 1;ii < nrParams;ii++)
{

    dsci= dLdAAA(x,y,z,Cim1,Dim1,c,d,ii,nrParams);

    dsdi= dLdBB(x,y,z,Cim1,Dim1,c,d,ii,nrParams);

    //System.out.println("ii"+" "+ii);
    //System.out.println("dsc"+" "+ii+" " +dsci);
    //System.out.println("dsd"+" "+ii+" " +dsdi);

    A.set(matrixRoww,(ii*2)+zw, 0);
    A.set(matrixRoww, (ii*2)+zw+1, 0);
    A.set(matrixRoww, (ii*2)+zw+2, dsci);
    A.set(matrixRoww, (ii*2)+zw+3, dsdi);

int r = ((ii * 2) + (4 + (2 * zy)));
System.out.println("r" +r);

for(int w = 0; w < r; w++)

{
    A.set(matrixRoww,(ii*2)+zw+4+w, 0);

}

zw = zw +2;
zy = zy +1;
}

++matrixRoww;

```

```

        int wq;
        int rr;
        int kk = 0;
        int ee = 6;
        for(rr = 0; rr < (nrParams - 1); rr++)
        {
            for(wq = 0; wq < ee; wq++)
            {
                A.set(matrixRoww,rr + wq, 0);

            }
            ee = ee + 3;

            kk = rr + wq;
        }
        //System.out.println("ee" +ee);
        //System.out.println("kk" +kk);

        int iii = 0;
        int zxx = 0;
        {

            dla1i= dLdAAA(x,y,z,Aim2,Bim2,a1,b1,iii,nrParams);

            //System.out.println("iii"+" "+iii);
            //System.out.println("dla1"+" "+iii+" "+dla1i);

            A.set(matrixRoww, iii + kk, dla1i);
            A.set(matrixRoww, iii + kk + 1, 0);

        }

        for(iii = 1;iii < nrParams;iii++)
        {

            dla1i= dLdAAA(x,y,z,Aim2,Bim2,a1,b1,iii,nrParams);

            dlb1i=dLdBB(x,y,z,Aim2,Bim2,a1,b1,iii,nrParams);
            //System.out.println("iii"+" "+iii);
            //System.out.println("dla1"+" "+iii+" "+dla1i);
            //System.out.println("dlb1"+" "+iii+" "+dlb1i);

            A.set(matrixRoww,(iii*2)+zxx+kk, dla1i);
            A.set(matrixRoww, (iii*2)+zxx+1+kk, dlb1i);
            A.set(matrixRoww, (iii*2)+zxx+2+kk, 0);
            A.set(matrixRoww, (iii*2)+zxx+3+kk, 0);

            zxx = zxx +2;

        }

        ++matrixRoww;

        int wqq;
        int rrr;
        int kkk = 0;
        int eee = 6;
        for(rrr = 0; rrr < (nrParams - 1); rrr++)
        {
            for(wqq = 0; wqq < eee; wqq++)
            {

```

```
A.set(matrixRoww,rrr + wqq, 0);

}
eee = eee + 3;

kkk = rrr + wqq;
}
//System.out.println("eee" +eee);
//System.out.println("kkk" +kkk);

int iii = 0;
int zxxx = 0;
{

    dsc1i= dLdAAA(x,y,z,Cim2,Dim2,c1,d1,iiii,nrParams);

    //System.out.println("iii"+" "+iiii);
    //System.out.println("dlc1"+" "+iiii+" " +dsc1i);

    A.set(matrixRoww, iii + kkk, 0);
    A.set(matrixRoww, iii + kkk + 1, dsc1i);

}

for(iiii = 1;iiii < nrParams;iiii++)
{

    dsc1i= dLdAAA(x,y,z,Cim2,Dim2,c1,d1,iiii,nrParams);

    dsd1i=dLdBB(x,y,z,Cim2,Dim2,c1,d1,iiii,nrParams);
    //System.out.println("iii"+" "+iiii);
    //System.out.println("dsc1"+" "+iiii+" " +dsc1i);
    //System.out.println("dsd1"+" "+iiii+" " +dsd1i);

    A.set(matrixRoww,(iiii*2)+zxxx+kkk, 0);
    A.set(matrixRoww, (iiii*2)+zxxx+1+kkk, 0);
    A.set(matrixRoww, (iiii*2)+zxxx+2+kkk, dsc1i);
    A.set(matrixRoww, (iiii*2)+zxxx+3+kkk, dsd1i);

    zxxx = zxxx +2;

}
    ++matrixRoww;
}

System.out.println("The matrix of partial differntiation: A is as follows");
// A.print(A.getRowDimension(), A.getColumnDimension());

int sizeg=y1List.size() + y2List.size();
    Matrix Y_ = new Matrix(y1List.size(), 1);

int k=0;
```

```

        for (int i = 0; i < y1List.size(); ++i,k++)
            Y_.set(k, 0, ((Double)(y1List.get(i))).doubleValue());

        //for (int i = 0; i < y2List.size(); ++i,k++)
        //Y_.set(k, 0, ((Double)(y2List.get(i))).doubleValue());

/* double arr[] = new double[sizeg];
System.out.println("size  "+sizeg);
int i=0,k=0;
for ( i = 0,k=0; i < y1List.size(); ++i,k++)
    arr[k]= ((Double) (y1List.get(i))).doubleValue();
for (i=0; i < y2List.size(); ++i,k++)
    arr[k]= ((Double) (y2List.get(i))).doubleValue();
for (i=0; i < yt1List.size(); ++i,k++)
    arr[k]= ((Double) (yt1List.get(i))).doubleValue();
for (i=0; i < yt2List.size(); ++i,k++)
    arr[k]= ((Double) (yt2List.get(i))).doubleValue();

for (i=0;i<sizeg;i++)
{
    System.out.println(arr[i]);
}

    */

//old code
    System.out.println("The Linear matrix: L is as follows");
    //Y_.print(y1List.size(), 1);

    //for (int i = 0; i < Y_.getRowDimension(); ++i)
    //System.out.println("Y_["+i+",0]="+Y_.get(i, 0));

//for calculation of final formula
System.out.println("A transpose matrix is");
Matrix Att = A.transpose();
//Att.print(Att.getRowDimension(), Att.getColumnDimension());
System.out.println("A transpose into A matrix is");
Matrix AtAt = Att.times(A);
//AtAt.print(AtAt.getRowDimension(), AtAt.getColumnDimension());
System.out.println("A transpose into A whole inverse matrix is");
Matrix invt = AtAt.inverse();
//invt.print(invt.getRowDimension(), invt.getColumnDimension());

System.out.println("Identity check");
Matrix Ident = invt.times(AtAt);
//Ident.print(Ident.getRowDimension(), Ident.getColumnDimension());

System.out.println("Inverse matrix into linear matix Y_ is:");
Matrix invAtt = invt.times(Att);
//invAtt.print(invAtt.getRowDimension(), invAtt.getColumnDimension());
System.out.println("The final solution matrix is as follows");
Matrix solt = invAtt.times(Y_);
solt.print(solt.getRowDimension(), solt.getColumnDimension());

double sol[] = new double[solt.getRowDimension()];
for(int i = 0; i < solt.getRowDimension(); i++)
{
    int jj = 0;

    sol[i] = solt.get(i,jj);
}

for(int i = 0; i < solt.getRowDimension(); i++){
    System.out.println("sol" +i+" "+solt[i]);}

int count1 = 0; int r = 0; int count2 = 2 + ((nrParams - 1) * 4);

Aim1[r] = sol[count1] + Aim1[r];
count1 = count1 +1;
Cim1[r] = sol[count1] + Cim1[r];

```

```

        count1 = count1 + 1;
        Aim2[r] = sol[count2] + Aim2[r];
        count2 = count2 + 1;
        Cim2[r] = sol[count2] + Cim2[r];
        count2 = count2 + 1;

        for(int ttt = 1; ttt < nrParams; ttt++)
        {
            Aim1[ttt] = sol[count1] + Aim1[ttt];
            count1 = count1 + 1;
            Bim1[ttt] = sol[count1] + Bim1[ttt];
            count1 = count1 + 1;
            Cim1[ttt] = sol[count1] + Cim1[ttt];
            count1 = count1 + 1;
            Dim1[ttt] = sol[count1] + Dim1[ttt];
            count1 = count1 + 1;
        }

        for(int tt = 1; tt < nrParams; tt++)
        {
            Aim2[tt] = sol[count2] + Aim2[tt];
            count2 = count2 + 1;
            Bim2[tt] = sol[count2] + Bim2[tt];
            count2 = count2 + 1;
            Cim2[tt] = sol[count2] + Cim2[tt];
            count2 = count2 + 1;
            Dim2[tt] = sol[count2] + Dim2[tt];
            count2 = count2 + 1;
        }

        int i=0;
    {
        //System.out.println("Aim1"+i+" "+Aim1[i]);

        //System.out.println("Cim1"+i+" "+Cim1[i]);

    }

    for(i = 1; i < nrParams; i++)
    {
        //System.out.println("Aim1"+i+" "+Aim1[i]);

        //System.out.println("Bim1"+i+" "+Bim1[i]);

        //System.out.println("Cim1"+i+" "+Cim1[i]);

        //System.out.println("Dim1"+i+" "+Dim1[i]);
    }

    i =0;
    {
        //System.out.println("Aim2"+i+" "+Aim2[i]);
        //System.out.println("Cim2"+i+" "+Cim2[i]);
    }
    for(i = 1; i < nrParams; i++)
    {
        //System.out.println("Aim2"+i+" "+Aim2[i]);

        //System.out.println("Bim2"+i+" "+Bim2[i]);

        //System.out.println("Cim2"+i+" "+Cim2[i]);

        //System.out.println("Dim2"+i+" "+Dim2[i]);
    }
}

```

```

}

chr=readInt("How many times you want to run the loop for converging the solution matrix ",in);
int counter = 0;
while(counter <= chr)
{

counter = counter+1;

y1List = new ArrayList<Double>();
//System.out.println("size of list"+" "+y1List.size());
matrixRoww = 0;

for (Integer id : gpsIDs) {
    Point3d point = ht.get(id);
    double x = point.x;
    double y = point.y;
    double z = point.z;
    System.out.println("Processing id="+id+" x="+x+" y="+y+" z="+z);

    double lme = P(x, y, z, a1) / P(x, y, z, b1);
    double sme = P(x, y, z, c1) / P(x, y, z, d1);
    double l = imc0.get(id).x;
    double s = imc0.get(id).y;
    double lmea = P(x, y, z, a1);
    double lmeb = P(x, y, z, b1);

    double lmec = P(x, y, z, c1);
    double lmed = P(x, y, z, d1);

    //double l1 = P(x, y, z, a1) / P(x, y, z, b1);
    //double s1 = P(x, y, z, c1) / P(x, y, z, d1);
    double l1 = imc1.get(id).x;
    double s1 = imc1.get(id).y;

    double lme1 = P(x, y, z, a1) / P(x, y, z, b1);
    double sme1 = P(x, y, z, c1) / P(x, y, z, d1);

    //System.out.println("L = " + l);
    //System.out.println("S = " + s);

    //System.out.println("L1 = " + l1);
    //System.out.println("S1 = " + s1);

    double Aim = ppp(x, y, z, Aim1, parasize);
    double la1cal = (P(x, y, z, a1) + ppp(x, y, z, Aim2, parasize));
    double lb1cal = (P(x, y, z, b1) + ppp(x, y, z, Bim2, parasize));
    double sc1cal = (P(x, y, z, c1) + ppp(x, y, z, Cim2, parasize));
    double sd1cal = (P(x, y, z, d1) + ppp(x, y, z, Dim2, parasize));

    double lcomp = (P(x, y, z, a) + ppp(x, y, z, Aim1, parasize)) / (P(x, y, z, b) + ppp(x, y, z,
Bim1, parasize));
    double scomp = (P(x, y, z, c) + ppp(x, y, z, Cim1, parasize)) / (P(x, y, z, d) + ppp(x, y, z,
Dim1, parasize));

    double l1comp = (P(x, y, z, a1) + ppp(x, y, z, Aim2, parasize)) / (P(x, y, z, b1) + ppp(x, y, z, Bim2,
parasize));
    double s1comp = (P(x, y, z, c1) + ppp(x, y, z, Cim2, parasize)) / (P(x, y, z, d1) + ppp(x, y, z,
Dim2, parasize));

    double y11 = l - lcomp;
    double y12 = s - scomp;
    double y21 = l1 - l1comp;
    double y22 = s1 - s1comp;

    y1List.add(y11);
    y1List.add(y12);
    y1List.add(y21);
    y1List.add(y22);
    /*System.out.println("Lmea = " + lmea);
    System.out.println("Lmeb = " + lmeb);

    System.out.println("Lmec = " + lmec);
    System.out.println("lmed = " + lmed);
    System.out.println("Lme = " + lme);
    System.out.println("sme = " + sme);

```

```

        System.out.println("L1comp = " + l1comp);
        System.out.println("S1comp = " + s1comp);
        System.out.println("Y11 = " + y11);
        System.out.println("Y12 = " + y12);
        System.out.println("Y21 = " + y21);
        System.out.println("Y22 = " + y22);*/

        double dlai=0;
        double dla1i=0;
        double dlbi=0;
        double dlbl1i=0;
        double dlci=0 ;
        double dldi=0;

        double dld1i=0;

        double dlcl1i=0 ;

        double dsci=0 ;
        double dsdi=0;

        double dscl1i=0 ;

        double dsd1i=0;
        int zx = 0;
        int zz = 0;

        int i = 0;
    {

        dlai= dLdAAA(x,y,z,Aim1,Bim1,a,b,i,nrParams);

        //System.out.println("i"+" "+i);
        //System.out.println("dla"+" "+i+" "+dlai);

        A.set(matrixRoww, i, dlai);
        A.set(matrixRoww, i+1, 0);

    }

    for(i = 1; i < nrParams; i++)
    {

        dlai= dLdAAA(x,y,z,Aim1,Bim1,a,b,i,nrParams);

        dlbi= dLdBB(x,y,z,Aim1,Bim1,a,b,i,nrParams);

        //System.out.println("i"+" "+i);
        //System.out.println("dla"+" "+i+" "+dlai);
        //System.out.println("dlb"+" "+i+" "+dlbi);

        A.set(matrixRoww,(i*2)+zx, dlai);
        A.set(matrixRoww, (i*2)+zx+1, dlbi);
        A.set(matrixRoww, (i*2)+zx+2, 0);
        A.set(matrixRoww, (i*2)+zx+3, 0);

        int r = ((i * 2) + (4 + (2 * zz)));
        System.out.println("r" +r);

        for(int w = 0; w < r; w++)

        {

            A.set(matrixRoww,(i*2)+zx+4+w, 0);

        }

    }

    zx = zx + 2;
    zz = zz + 1;
    }

    ++matrixRoww;

```

```

int zw = 0;
int zy = 0;
int ii = 0;
{

    dsci= dLdAAA(x,y,z,Cim1,Dim1,c,d,ii,nrParams);

    //System.out.println("ii"+" "+ii);
    //System.out.println("dsc"+" "+ii+" " +dsci);

    A.set(matrixRoww, ii, 0);
    A.set(matrixRoww, ii+1, dsci);

}

for(ii = 1;ii < nrParams;ii++)
{

    dsci= dLdAAA(x,y,z,Cim1,Dim1,c,d,ii,nrParams);

    dsdi= dLdBB(x,y,z,Cim1,Dim1,c,d,ii,nrParams);

    //System.out.println("ii"+" "+ii);
    //System.out.println("dsc"+" "+ii+" " +dsci);
    //System.out.println("dsd"+" "+ii+" " +dsdi);

    A.set(matrixRoww,(ii*2)+zw, 0);
    A.set(matrixRoww, (ii*2)+zw+1, 0);
    A.set(matrixRoww, (ii*2)+zw+2, dsci);
    A.set(matrixRoww, (ii*2)+zw+3, dsdi);

    int r = ((ii * 2) + (4 + (2 * zy)));
    //System.out.println("r" +r);

    for(int w = 0; w < r; w++)

    {

        A.set(matrixRoww,(ii*2)+zw+4+w, 0);

    }

    zw = zw +2;
    zy = zy +1;
    }

    ++matrixRoww;

    int wq;
    int rr;
    int kk = 0;
    int ee = 6;
    for(rr = 0; rr < (nrParams - 1); rr++)
    {
    for(wq = 0; wq < ee; wq++)
    {
    A.set(matrixRoww,rr + wq, 0);

    }
    ee = ee + 3;

    kk = rr + wq;
    }
    //System.out.println("ee" +ee);

```

```
//System.out.println("kk" +kk);

int iii = 0;
int zxx = 0;
{

    dla1i= dLdAAA(x,y,z,Aim2,Bim2,a1,b1,iii,nrParams);

    //System.out.println("iii"+" "+iii);
    //System.out.println("dla1"+" "+iii+" "+dla1i);

    A.set(matrixRoww, iii + kk, dla1i);
    A.set(matrixRoww, iii + kk + 1, 0);

}

for(iii = 1;iii < nrParams;iii++)
{

    dla1i= dLdAAA(x,y,z,Aim2,Bim2,a1,b1,iii,nrParams);

    dlb1i=dLdBB(x,y,z,Aim2,Bim2,a1,b1,iii,nrParams);
    //System.out.println("iii"+" "+iii);
    //System.out.println("dla1"+" "+iii+" "+dla1i);
    //System.out.println("dlb1"+" "+iii+" "+dlb1i);

    A.set(matrixRoww,(iii*2)+zxx+kk, dla1i);
    A.set(matrixRoww, (iii*2)+zxx+1+kk, dlb1i);
    A.set(matrixRoww, (iii*2)+zxx+2+kk, 0);
    A.set(matrixRoww, (iii*2)+zxx+3+kk, 0);

    zxx = zxx +2;

}

++matrixRoww;

int wqq;
int rrr;
int kkk = 0;
int eee = 6;
for(rrr = 0; rrr < (nrParams - 1); rrr++)
{
    for(wqq = 0; wqq < eee; wqq++)
    {
        A.set(matrixRoww,rrr + wqq, 0);

    }
    eee = eee + 3;

    kkk = rrr + wqq;
}
//System.out.println("eee" +eee);
//System.out.println("kkk" +kkk);

int iiii = 0;
int zxxx = 0;
{
```

```

dsc1i= dLdAAA(x,y,z,Cim2,Dim2,c1,d1,iiii,nrParams);

//System.out.println("iiii"+" "+iiii);
//System.out.println("dlc1"+" "+iiii+" " +dsc1i);

A.set(matrixRoww, iiii + kkk, 0);
A.set(matrixRoww, iiii + kkk + 1, dsc1i);

}

for(iiii = 1;iiii < nrParams;iiii++)
{

    dsc1i= dLdAAA(x,y,z,Cim2,Dim2,c1,d1,iiii,nrParams);

    dsd1i=dLdBB(x,y,z,Cim2,Dim2,c1,d1,iiii,nrParams);
    //System.out.println("iiii"+" "+iiii);
    //System.out.println("dsc1"+" "+iiii+" " +dsc1i);
    //System.out.println("dsd1"+" "+iiii+" " +dsd1i);

    A.set(matrixRoww, (iiii*2)+zxxx+kkk, 0);
    A.set(matrixRoww, (iiii*2)+zxxx+1+kkk, 0);
    A.set(matrixRoww, (iiii*2)+zxxx+2+kkk, dsc1i);
    A.set(matrixRoww, (iiii*2)+zxxx+3+kkk, dsd1i);

    zxxx = zxxx +2;

}
    ++matrixRoww;

}

System.out.println("The matrix of partial differntiation is as follows");
//A.print(A.getRowDimension(), A.getColumnDimension());

int size=y1List.size() + y2List.size();
Matrix Y_ = new Matrix(y1List.size(), 1);
int k=0;
for (int i = 0; i < y1List.size(); ++i,k++)
    Y_.set(k, 0, ((Double)(y1List.get(i))).doubleValue());

//for (int i = 0; i < y2List.size(); ++i,k++)
//Y_.set(k, 0, ((Double)(y2List.get(i))).doubleValue());

/* double arr[] = new double[size];
System.out.println("size  "+size);
int i=0,k=0;
for ( i = 0,k=0; i < y1List.size(); ++i,k++)
    arr[k]= ((Double) (y1List.get(i))).doubleValue();
for (i=0; i < y2List.size(); ++i,k++)
    arr[k]= ((Double) (y2List.get(i))).doubleValue();
for (i=0; i < yt1List.size(); ++i,k++)
    arr[k]= ((Double) (yt1List.get(i))).doubleValue();

```

```

        for (i=0; i < yt2List.size(); ++i,k++)
            arr[k]= ((Double) (yt2List.get(i))).doubleValue();

        for (i=0;i<sizeg;i++)
        {
            System.out.println(arr[i]);
        }
        */

//old code
        System.out.println("The Y matrix is as follows");
        //Y_.print(y1List.size(), 1);

        //for (int i = 0; i < Y_.getRowDimension(); ++i)
        //System.out.println("Y_["+i+",0]="+Y_.get(i, 0));

//for calculation of final formula
        System.out.println("A transpose matrix is");
        Matrix Att = A.transpose();
        //Att.print(Att.getRowDimension(), Att.getColumnDimension());
        System.out.println("A transpose into A matrix is");
        Matrix AtAt = Att.times(A);
        //AtAt.print(AtAt.getRowDimension(), AtAt.getColumnDimension());
        System.out.println("A transpose into A whole inverse matrix is");
        Matrix invt = AtAt.inverse();
        //invt.print(invt.getRowDimension(), invt.getColumnDimension());

        System.out.println("Identity check");
        Matrix Ident = invt.times(AtAt);
        //Ident.print(Ident.getRowDimension(), Ident.getColumnDimension());

        System.out.println("Inverse matrix into linear matix Y_ is:");
        Matrix invAtt = invt.times(Att);
        //invAtt.print(invAtt.getRowDimension(), invAtt.getColumnDimension());
        System.out.println("The final solution matrix is");
        Matrix solt = invAtt.times(Y_);
        solt.print(solt.getRowDimension(), solt.getColumnDimension());

        double sol[] = new double[solt.getRowDimension()];
        for(int i = 0; i < solt.getRowDimension(); i++)
        {
            int jj = 0;

            sol[i] = solt.get(i,jj);
        }

        for(int i = 0; i < solt.getRowDimension(); i++){
            System.out.println("sol" +i+" "+sol[i]);}

        int count1 = 0; int r = 0; int count2 = 2 + ((nrParams - 1) * 4);

        Aim1[r] = sol[count1] + Aim1[r];
        count1 = count1 + 1;
        Cim1[r] = sol[count1] + Cim1[r];
        count1 = count1 + 1;
        Aim2[r] = sol[count2] + Aim2[r];
        count2 = count2 + 1;
        Cim2[r] = sol[count2] + Cim2[r];
        count2 = count2 + 1;
        for(int ttt = 1; ttt < nrParams; ttt++)
        {
            Aim1[ttt] = sol[count1] + Aim1[ttt];
            count1 = count1 + 1;
            Bim1[ttt] = sol[count1] + Bim1[ttt];
            count1 = count1 + 1;
            Cim1[ttt] = sol[count1] + Cim1[ttt];
            count1 = count1 + 1;
            Dim1[ttt] = sol[count1] + Dim1[ttt];
            count1 = count1 + 1;
        }
    }

```

```

for(int tt = 1; tt < nrParams; tt++)
{
    Aim2[tt] = sol[count2] + Aim2[tt];
    count2 = count2 + 1;
    Bim2[tt] = sol[count2] + Bim2[tt];
    count2 = count2 + 1;
    Cim2[tt] = sol[count2] + Cim2[tt];
    count2 = count2 + 1;
    Dim2[tt] = sol[count2] + Dim2[tt];
    count2 = count2 + 1;
}
int i=0;
{
    //System.out.println("Aim1"+i+" " +Aim1[i]);

    //System.out.println("Cim1"+i+" " +Cim1[i]);

}

for(i = 1;i < nrParams;i++)
{
    //System.out.println("Aim1"+i+" " +Aim1[i]);
    //System.out.println("Bim1"+i+" " +Bim1[i]);
    //System.out.println("Cim1"+i+" " +Cim1[i]);
    //System.out.println("Dim1"+i+" " +Dim1[i]);
}

i =0;
{
    //System.out.println("Aim2"+i+" " +Aim2[i]);
    //System.out.println("Cim2"+i+" " +Cim2[i]);
}
for(i = 1;i < nrParams;i++)
{
    //System.out.println("Aim2"+i+" " +Aim2[i]);
    //System.out.println("Bim2"+i+" " +Bim2[i]);
    //System.out.println("Cim2"+i+" " +Cim2[i]);
    //System.out.println("Dim2"+i+" " +Dim2[i]);
}

}

}

chh = readInt("enter choice 1 for printing value of changed RPC ",in);

switch(chh)
{

case 1:
{
    System.out.println("program works");
    int y = 0;

    a[y] = Aim1[y] + a[y];
    c[y] = Cim1[y] + c[y];
    a1[y] = Aim2[y] + a1[y];
    c1[y] = Cim2[y] + c1[y];

    for(int s = 1; s < nrParams; s++)
    {
        a[s] = Aim1[s] + a[s];
        b[s] = Bim1[s] + b[s];

        c[s] = Cim1[s] + c[s];
        d[s] = Dim1[s] + d[s];

        a1[s] = Aim2[s] + a1[s];
        b1[s] = Bim2[s] + b1[s];
        c1[s] = Cim2[s] + c1[s];
        d1[s] = Dim2[s] + d1[s];
    }
}
}

```

```

}

    for(int i = 0;i < 20;i++)
{
    System.out.println("l_NUM_COEFF_"+i+": "+a[i]);

    }
    for(int i = 0;i < 20;i++)
{

    System.out.println("l_DEN_COEFF_"+i+": "+b[i]);
}
for(int i = 0;i < 20;i++)
{

    System.out.println("l_NUM_COEFF_"+i+": "+c[i]);
}
for(int i = 0;i < 20;i++)
{

    System.out.println("l_DEN_COEFF_"+i+": "+d[i]);
}
    for(int i = 0;i < 20;i++)
{
    System.out.println("l_NUM_COEFF_"+i+": "+a1[i]);
}
    for(int i = 0;i < 20;i++)
{

    System.out.println("l_DEN_COEFF_"+i+": "+b1[i]);
}
    for(int i = 0;i < 20;i++)
{

    System.out.println("l_NUM_COEFF_"+i+": "+c1[i]);
}
    for(int i = 0;i < 20;i++)
{

    System.out.println("l_DEN_COEFF_"+i+": "+d1[i]);
}
    String name = "mod1_den";

    display(a,b,c,d,name);

    String name2 = "mod2_den";

    display(a1,b1,c1,d1,name2);

    String mod1 = "";

    try
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the name of the file of image 1 for modified RPC");
    mod1 = br.readLine();
    }
catch(Exception e)
{
    System.out.println("Error" +e);
}

    String mod2 = "";

    try
{
    BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
    System.out.println("Enter the name of the file of image 2 for modified RPC");
    mod2 = br.readLine();
    }
catch(Exception e)
{
    System.out.println("Error" +e);
}

```

```

    }

    func last = new func();
    last.norm(name,im0,mod1);
    last.norm(name2,im1,mod2);

    String gcps = "";
    try
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        System.out.println("Enter the name of the file to display modified RPC derieved GCPs");
        gcps = br.readLine();
    }
    catch(Exception e)
    {
        System.out.println("Error" +e);
    }
    func acc = new func();
    acc.accuracy(mod1,imcd0,mod2,imcd1,gcps);
    System.out.println("Thankyou for using this software hope you enjoy the modified RPC");

    break;
}

}
}
}

```

```

public static double P(double x, double y, double z, double[] my_array) {
    if (my_array.length < 20) {
        System.out.println("A should have 20 elements");
        return 0;
    }

    double[] f = factors(x, y, z);

    double result = 0;

    for (int i = 0; i < f.length; ++i)
        result += my_array[i] * f[i];

    return result;
}

public static double ppp(double x, double y, double z, double[] my_array, int index) {

    //System.out.println("Index for PPP" + " " +index);

    double[] f = factors(x, y, z);

    double result = 0;

    for (int i = 0; i < index; ++i)
        result += my_array[i] * f[i];
    //for (int i = 0; i < index; ++i)
    //System.out.println("my_array for ppp"+" " + i + " " +my_array[i]);
}

```

```

        //for (int i = 0; i < index; ++i)
        //System.out.println("f_index for ppp"+" "+i+" " +f[i]);

        //System.out.println("result" +result);

        return result;
    }

    public static double pp(double x, double y, double z, double[] my_array, int index) {

        //System.out.println("Index for PP" +" "+index);

        double[] f = factors(x, y, z);

        double result = 0;

        for (int i = 0; i <=index; ++i)
            result += my_array[i] * f[i];
        //for (int i = 0; i <= index; ++i)
        //System.out.println("my_array for pp"+" "+i+" " +my_array[i]);

        //for (int i = 0; i <= index; ++i)
        //System.out.println("f_index for pp"+" "+i+" " +f[i]);

        //System.out.println("result" +result);

        return result;
    }

    public static double pd(double x, double y, double z, double[] my_array, int index) {

        //System.out.println("Index for Pd" +" "+index);

        double[] f = factors(x, y, z);

        double result = 0;

        result = my_array[index] * f[index];

        //System.out.println("my_array for pd"+" "+index+" " +my_array[index]);

        //System.out.println("f_index for pd"+" "+index+" " +f[index]);
        //System.out.println("result" +result);

        return result;
    }

    public static double[] factors(double x, double y, double z) {
        double[] factors = { 1, x, y, z, x * y, x * z, y * z, x * x, y * y,
            z * z, x * y * z, x * x * x, x * y * y, x * z * z, x * x * y,
            y * y * y, y * z * z, x * x * z, y * y * z, z * z * z };

        return factors;
    }

    public static double dLdAAA(double x, double y, double z, double [] A, double [] B, double [] a, double [] b, int index, int n) {
        double[] f = factors(x, y, z);
        double enumerator = f[index];
        double denominator = P(x, y, z, b) + ppp(x, y, z, B,n);
        return enumerator / denominator;
    }

    /*public static double differentiateB(double x, double y, double z,
        double[] B, int index) {
        double[] f = factors(x, y, z);
        return -f[index] * B[index];
    }*/

    public static double dLdBB(double x, double y, double z,double[] A,
        double[] B,double[] a,double[] b, int index,int n) {

```



```

        double[] f = factors(x, y, z);

        double enumerator = (P(x,y,z, a)+ppp(x,y,z, A,n));
        double denominator = ((P(x,y,z, b)+ppp(x,y,z, B,n)) * (P(x,y,z, b)+ppp(x,y,z, B,n)));
        return (enumerator * -f[index]) / (denominator);
    }

    public static double dLdA(double x, double y, double z, double [] A, double [] B, double [] a, double [] b, int index, int n) {
        double [] c1 = A;

        /*for(int i=0;i<parasize;i++)
        System.out.println("A "+A[i]);
        for(int i=0;i<parasize;i++)
        System.out.println("c1 "+c1[i]);*/
        double [] d1 = new double[parasize];
        d1[index] = c1[index] + eps;

        /*for(int i=0;i<parasize;i++)
        System.out.println("A "+A[i]);

        for(int i=0;i<parasize;i++)
            System.out.println("d1 "+d1[i]);*/
        double xy = P(x, y, z, a);
        double x1 = pd(x, y, z, d1,index);
        //System.out.println("xy" +xy);
        //System.out.println("x1" +x1);

        double xyy = P(x, y, z, b);
        double xll = ppp(x, y, z, A, n);
        double xlaa = (ppp(x, y, z, A, n) - pd(x, y, z, A,index));
        double xla = pd(x, y, z, A,index);
        //System.out.println("xyy" +xyy);
        //System.out.println("xla" +xla);
        double sub = d1[index] - A[index];
        //System.out.println("sub" +sub);
        double R = (xy +xlaa + x1) / xyy;

        double r = (xy + xll) / xyy;

        /*System.out.println("r "+r);
        System.out.println("R "+R);
        System.out.println("r-R "+(R-r));*/
        double result = (R-r) / sub;
        System.out.println("resuklt "+result);
        return result;
    }

    public static double dLdAA(double x, double y, double z, double [] A, double [] B, double [] a, double [] b, int index, int n) {
        double [] c1 = A;
        /*for(int i=0;i<parasize;i++)
        System.out.println("A "+A[i]);
        for(int i=0;i<parasize;i++)
        System.out.println("c1 "+c1[i]);*/
        double [] d1 = new double[parasize];
        d1[index] = c1[index] + eps;
        //System.out.println("index" +" " + index);
        //System.out.println("d1 "+ " "+d1[index]);
        double xy = P(x, y, z, a);
        double xyy = P(x, y, z, b);
        double xla = pd(x, y, z, A,index);
        double x1 = pd(x, y, z, d1,index);
        double xll = ppp(x, y, z, A, n);
        double xb = ppp(x, y, z, B,n);
        double xlaa = (ppp(x, y, z, A, n) - pd(x, y, z, A,index));
        double sub = d1[index] - A[index];
        /*System.out.println("xy"+" "+xy);
        System.out.println("xyy"+" "+xyy);
        System.out.println("xla"+" "+xla);
        System.out.println("x1"+" "+x1);
        System.out.println("xll"+" "+xll);
        System.out.println("xb"+" "+xb);
        System.out.println("xlaa"+" "+xlaa);
        System.out.println("sub" +sub);*/
    }

```

```

        double R = (xy + xlaa + xl) / (xyy + xb);

        double r = (xy + xll) / (xyy + xb);

        /*System.out.println("r " +r);
        System.out.println("R " +R);
        System.out.println("r-R " +(R-r));*/
        double result = (R-r) / sub;
        System.out.println("resuklt "+result);
        return result;
    }

    public static double dLdB(double x, double y, double z, double [] A, double [] B, double [] a, double [] b, int index, int n) {
        double [] c1 = B;
        double [] d1 = new double[parasize];
        d1[index] = c1[index] + eps;

        double xy = P(x, y, z, a);
        double xl = pd(x, y, z, d1, index);
        double xll = ppp(x, y, z, A, n);
        double xlaa = (ppp(x, y, z, A, n) - pd(x, y, z, A, index));
        double xlbb = (ppp(x, y, z, B, n) - pd(x, y, z, B, index));
        double xyy = P(x, y, z, b);
        double xla = pd(x, y, z, A, index);
        double xlb = pd(x, y, z, B, index);
        double xb = ppp(x, y, z, B, n);
        double R = (xy + xll) / (xyy + xlbb + xl);
        double r = (xy + xll) / (xyy + xb);
        double sub = d1[index] - B[index];
        //System.out.println("sub" +sub);

        double result = (R-r) / sub;
        return result;
    }

    public static void display(double b[], double t[], double w[], double z[], String n) throws IOException
    {
        String a[] = {"hShift: 0", "sShift: 0", "YShift: 0", "XShift: 0", "ZShift: 0", "lScale: 1", "sScale: 1", "YScale: 1", "XScale: 1", "ZScale: 1"};
        File f = new File(n);
        String s = "";
        String s1 = "";
        String s2 = "";
        String s3 = "";
        String s4 = "";
        if (f.exists())
        {
            f.delete();
        }

        RandomAccessFile ff = new RandomAccessFile(n, "rw");
        for (int i = 0; i < 10; i++)
        {
            s = s + a[i] + "\n";
        }
        ff.seek(ff.length());
        ff.writeBytes(s);

        for (int j = 0; j < b.length; j++)
        {
            s1 = "";
            s1 = s1 + "l_NUM_COEFF_" + j + " : " + b[j] + "\n";
            ff.seek(ff.length());
            ff.writeBytes(s1);
        }
        for (int j = 0; j < t.length; j++)
        {
            s2 = "";
            s2 = s2 + "l_DEN_COEFF_" + j + " : " + t[j] + "\n";
            ff.seek(ff.length());
            ff.writeBytes(s2);
        }
        for (int j = 0; j < w.length; j++)
    }

```

```

        {
            s3="";
            s3=s3+"s_NUM_COEFF_"+j+" : "+w[j]+"\\n";
            ff.seek(ff.length());
            ff.writeBytes(s3);
        }
        for(int j=0;j<z.length;j++)
        {
            s4="";
            s4=s4+"s_DEN_COEFF_"+j+" : "+z[j]+"\\n";
            ff.seek(ff.length());
            ff.writeBytes(s4);
        }
        ff.close();
    }
    else
    {
        RandomAccessFile ff=new RandomAccessFile(n,"rw");
        for(int i=0;i<10;i++)
        {
            s=s+a[i]+"\\n";
        }
        ff.seek(ff.length());
        ff.writeBytes(s);

        for(int j=0;j<b.length;j++)
        {
            s1="";
            s1=s1+"l_NUM_COEFF_"+j+" : "+b[j]+"\\n";
            ff.seek(ff.length());
            ff.writeBytes(s1);
        }
        for(int j=0;j<t.length;j++)
        {
            s2="";
            s2=s2+"l_DEN_COEFF_"+j+" : "+t[j]+"\\n";
            ff.seek(ff.length());
            ff.writeBytes(s2);
        }
        for(int j=0;j<w.length;j++)
        {
            s3="";
            s3=s3+"s_NUM_COEFF_"+j+" : "+w[j]+"\\n";
            ff.seek(ff.length());
            ff.writeBytes(s3);
        }
        for(int j=0;j<z.length;j++)
        {
            s4="";
            s4=s4+"s_DEN_COEFF_"+j+" : "+z[j]+"\\n";
            ff.seek(ff.length());
            ff.writeBytes(s4);
        }
        ff.close();
    }
}

public static double dLdx(double x, double y, double z, double [] A, double [] B, double [] a, double [] b, int index) {
    double R = (P(x+eps, y, z, a) + P(x+eps, y, z, A))
                / (P(x+eps, y, z, b) + P(x+eps, y, z, B));

    double r = (P(x, y, z, a) + P(x, y, z, A))
              / (P(x, y, z, b) + P(x, y, z, B));

    double result = (R-r)/eps;
    return result;
}

public static double dLdy(double x, double y, double z, double [] A, double [] B, double [] a, double [] b, int index) {
    double R = (P(x, y+eps, z, a) + P(x, y+eps, z, A))
                / (P(x, y+eps, z, b) + P(x, y+eps, z, B));

```

```

        double r = (P(x, y, z, a) + P(x, y, z, A))
                                / (P(x, y, z, b) + P(x, y, z, B));

        double result = (R-r)/eps;
        return result;
    }

    public static double dLdz(double x, double y, double z, double [] A, double [] B, double [] a, double [] b, int index) {
        double R = (P(x, y, z+eps, a) + P(x, y, z+eps, A))
                                / (P(x, y, z+eps, b) + P(x, y, z+eps, B));

        double r = (P(x, y, z, a) + P(x, y, z, A))
                                / (P(x, y, z, b) + P(x, y, z, B));

        double result = (R-r)/eps;
        return result;
    }

    public static int readInt(String message, BufferedReader in) {
        System.out.println(message);
        int value = 0;
        try {
            String line = in.readLine();
            if (line != null && line.length() > 0) {
                value = Integer.parseInt(line);
            }
        } catch (IOException e) {
        }
        System.out.println("You entered: "+value);
        return value;
    }

    public static double readDouble(String message, BufferedReader in) {
        System.out.println(message);
        double value = 0;
        try {
            String line = in.readLine();
            if (line != null && line.length() > 0) {
                value = Double.parseDouble(line);
            }
        } catch (IOException e) {
        }
        System.out.println("You entered: "+value);
        return value;
    }

    public static ArrayList<Integer> readArray(String message, BufferedReader in) {
        System.out.println(message);
        ArrayList<Integer> values = new ArrayList<Integer>();
        try {
            String line = in.readLine();
            if (line != null && line.length() > 0) {
                StringTokenizer st = new StringTokenizer(line);
                while (st.hasMoreTokens()) {
                    String token = st.nextToken();
                    Integer value = Integer.parseInt(token);
                    values.add(value);
                }
            }
        } catch (IOException e) {
        }
        System.out.println("You entered: "+values);
        return values;
    }
}

```