SKRIPSI

DESAIN ALGORITMA PENENTUAN NILAI PARAMETER EKSTERIOR ORIENTASI MENGGUNAKAN TEKNIK SPACE RESECTION DAN ABSOLUT ORIENTASI

Oleh:
SRI RAHAYU
05.25.004

PROGRAM STUDI TEKNIK GEODESI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
INSTITUT TEKNOLOGI NASIONAL
MALANG
2010

(SKRIPSI)

DESAIN ALGORITMA PENENTUAN NILAI PARAMETER EKSTERIOR ORIENTASI MENGGUNAKAN TEKNIK SPACE RESECTION DAN ABSOLUT ORIENTASI

Oleh :

SRI RAHAYU
05.25.004

PROGRAM STUDI TEKNIK GEODESI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
INSTITUT TEKNOLOGI NASIONAL
MALANG
2010

LEMBAR PENGESAHAN

Judul Skripsi:

"Desain Algorithma Penentuan Nilai Parameter Eksterior Orientasi menggunakan Teknik *Space Resection* dan *Absolute Orientation* "

Dipertahankan di hadapan Majelis Penguji Sidang Skripsi Jenjang Strata Satu (S1) pada :

Hari : Sabtu

Tanggal : 21 Agustus 2010

Dan diterima untuk memenuhi salah satu persyaratan guna memperoleh gelar Sarjana Teknik.

Disusun Oleh :

Sri Rahayu
05.25.004

Panitia Ujian Tugas Akhir

Ketua	Sekretaris
Hery Purwanto, ST, M.Sc	**Silvester Sari Sai, ST, MT**

Anggota Penguji

Penguji I	Penguji II	Penguji III
Ir. Agus Darpono, MT	**Hery Purwanto, ST, M.Sc**	**Dr. Edwin Tjahjadi, ST. M.Geom.Sc**

JURUSAN TEKNIK GEODESI

FAKULTAS TEKNIK SIPIL DAN PERENCANAAN

INSTITUT TEKNOLOGI NASIONAL

MALANG

2010

LEMBAR PERSETUJUAN

SKRIPSI

Judul Skripsi:

"Desain Algorithma Penentuan Nilai Parameter Eksterior Orientasi menggunakan Teknik *Space Resection* dan *Absolute Orientation*"

Diajukan Sebagai Salah Satu Syarat Memperoleh

Gelar Sarjana Teknik Geodesi S1

Institut Teknologi Nasional Malang

Disusun Oleh :

Sri Rahayu
05.25.004

Meyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Hery Purwanto, ST, M.Sc	**Dr. Edwin Tjahjadi, ST. M.Geom.Sc**

Mengetahui,

Ketua Jurusan Teknik Geodesi S-1

Hery Purwanto, ST, M.Sc

ABSTRAKSI

Dalam proses fotogrametri, enam parameter *eksterior orientation* (ω, φ, κ, X_L, Y_L, Z_L) merupakan salah satu hal mendasar yang harus diketahui. Pada tulisan ini berisi tentang penjelasan bagaimana mencari enam parameter *eksterior orientation* dengan menggunakan dua metode yaitu *Absolute Orientation* dan *Space Resection*. Metode *absolute orientation* digunakan untuk mendapatkan nilai koordinat objek 3D (X, Y, Z) dari foto stereo. Selanjutnya koordinat objek 3D ini akan dijadikan sebagai input untuk menghitung nilai 6 parameter *eksterior orientasi* sehingga dapat diketahui nilai posisi dan rotasi kamera. Kedua metode ini diselesaikan dengan menggunakan teknik *Least Square Adjusment* atau teknik perataan. Adapun nilai standar deviasi rata-rata dari 6 parameter tersebut adalah 0.02 rad untuk parameter rotasi sedangkan untuk parameter kamera adalah 0.17 m.

Kata Kunci *: Persamaan Kolinear, Eksterior Orientasi, Space Resection, Absolute Orientation, Least Square Adjusment.*

PERNYATAAN KEASLIAN
SKRIPSI

Saya yang bertanda tangan dibawah ini.

Nama	**: Sri Rahayu**
NIM	**: 05.25.004**
Program Studi	**: Teknik Geodesi S-1**
Fakultas	**: Fakultas Teknik Sipil Dan Perencanaan**

Menyatakan dengan sesungguhnya bahwa Skripsi saya dengan judul :

"Desain Algorithma Penentuan Nilai Parameter Eksterior Orientasi menggunakan Teknik *Space Resection* dan *Absolute Orientation*"

adalah hasil karya saya sendiri, bukan merupakan duplikat, copy, salinan maupun saduran, kecuali beberapa kalimat kutipan dan gambar yang telah disebutkan sumbernya.

Malang, Oktober 2010
Yang membuat pernyataan

Sri Rahayu
05.25.004

Alhamdulillah, ucapan syukur akan kebesaran **Allah Swt** yang tlah memberikan kesempatan pada diriku tuk menjalani dan merasakan semua proses dalam perkuliahanku hingga penulisan skripsi selesai..

Ampunilah hambamu ini yg "terkadang harus memilih jalan yang salah untuk menemukan suatu kebenaran"

Jalan panjang dan berliku, penuh halangan dan rintangan yang mengiringi penulisan skripsi ini tlah membuatku bertambah yakin akan kebesaranNya,..

"sabar dan ikhlas", dua kata yang makin aku pahami maknanya, gampang mengucapkan tapi susah diamalkan..
Hasil karya kecilku ini kan kupersembahkan untuk orang2 yang aku kasihi dan aku cintai :

1. Ibundaku tersayang "**Fahmiati S**", Ayahandaku tersayang "**Suhartono**", terima kasih buat doa, cinta dan kasih sayang kalian untuk anandamu, terima kasih untuk semua pengorbanan,keikhlasan,kesabaran,ketabahan dan support yang tak pernah henti kalian berikan kepadaku..buat ibuku,terima kasih udah mau dengerin curhat anakmu ini, semoga kalian selalu dalam lindungan ALLAH SWT...Amiinnn..LOVE U....

2. Buat my brother " **Edi Masrianto**", meskipun kita sering tengkar, tapi terima kasih buat doa dan dukunganmu. smoga kita bisa saling menyayangi ampe tua nanti,,coz sodaraku cuma kamu seorang..so,still be my little brother...
3. For my someone special "**Mohamad Tanzil**", gak ada kata yang lain yang bisa aku ucapin selain kata "TERIMA KASIH",,trima kasih sedalam-dalamnya buat bantuanmu dalam penulisan skripsiku,terima kasih buat support yg selalu engkau brikan disaat aku sedang terpuruk, terima kasih buat kesabaranmu yg selalu engkau brikan disaat aku kesal. Semoga dirimu menjadi yang terbaik untukku dan masa depanku..Amiiiinnn..
4. Laskar geo'05 "**Tanzil, Eno, Chandra, Via, Dodik, Agus, Lia, Gede, Ona, Alben, Weny**", kalian emang pejuang yang hebat, sahabat terbaik dalam hidupku. Kebersamaan dan kekompakan kita membuat aku tak pernah menganggap kalian hanya sebagai sahabat tapi juga keluarga. Canda tawa kalian membuatku mampu melupakan masalah yang datang dalam hidup. Terlalu banyak kenangan yang kita lewati bersama, semoga kenangan ini selamanya menjadi cerita indah untuk anak cucu kita nanti... I LOVE U ALL..keep contact yach..^_^
5. Penghuni Lab. SIG selain geo'05 "**Mas Akbar, Pace Yusak, Pace Roudger dan K desy**", terima kasih sudah berjuang bersama

selama 2 tahun lamanya. Banyak pengalaman yang kita alami bersama, semoga kita tetap menjaga silaturahmi..

6. Buat anak-anak BT.09 "Nova, Maria, Cucan, K nina, Iva, Jane, Lira, Mb Iin, Mb Dora", U are the best family yg pernah aku miliki..trima kasih bt kebersamaan kita di BT.09..kapan lagi yach bisa kumpul-kumpul..i'm gonna miss u all..
7. Buat teman-temandi bend.sempor 28, "Kak Dadi, Kak Mawar, Kak Fauzan, Bang thalib, opick". Beruntunglah diriku mengenal kalian, coz jadi punya banyak kakak laki-laki...hehehehe..
8. All Geodesi 06-09, terima kasih buat bantuan-bantuan yang kalian berikan kepadaku. Ku doakan semoga kalian bisa menjadi orang-orang yang sukses..AMIIINNNN..
9. Yang terakhir buat semua orang yang ada disekitarku yang gak bisa aku sebutin satu persatu, terima kasih buat doa dan dukungan kalian. Semoga kita semua selalu dalam lindungan-NYA..Amiiinnn..

"Kesuksesan Besar Itu Hanya Bisa Dicapai Dengan Mengambil Sesuatu Yang Negatif Dan Mengubahnya Menjadi Yang Positif....."

KATA PENGANTAR

Dengan memanjatkan puji syukur kehadirat Allah S.W.T atas limpahan rahmat serta hidayah-Nya, maka penulisan skripsi ini akhirnya dapat terselesaikan dengan judul "Desain Algorithma Penentuan Nilai Parameter Eksterior Orientasi menggunakan Teknik *Space Resection* dan *Absolute Orientation*".

Skripsi ini disusun untuk memenuhi salah satu syarat memperoleh gelar Sarjana Teknik Geodesi (S1) di Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.

Dalam kesempatan ini pula, penulis mengucapkan terima kasih yang sebesar-besarnya atas dukungan dan bantuan kepada yang terhormat :

1. Bapak Prof. Dr. Eng. Ir. Abraham Lomi, MSEE selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. A. Agus Santosa, MT selaku Dekan Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.
3. Bapak Heri Purwanto, ST., M.Sc. selaku Ketua Jurusan Teknik Geodesi Institut Teknologi Nasional Malang serta sebagai Dosen Pembimbing I
4. Bapak Dr. Edwin Tjahjadi, ST., MGeom.Sc. selaku Dosen Pembimbing II serta sebagai Dosen Penguji.
5. Bapak Silvester Sari Sai, ST., MT. selaku Dosen Penguji.
6. Bapak Leo Patimena, ST., M.Sc. selaku Dosen Penguji.

7. Segenap dosen, staff pengajar dan rekording Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.
8. Rekan-rekan Mahasiswa/i dan alumni Teknik Geodesi.
9. Semua pihak yang langsung maupun tidak langsung turut membantu dalam proses penelitian maupun penyusunan laporan Tugas Akhir ini.

Penulis sadari bahwa masih banyak kekurangan dalam penyusunan skripsi ini, sehingga penulis sangat mengharapkan berbagai saran dan kritik dalam perbaikan skripsi ini.

Malang, Oktober 2010

Penulis

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pada ilmu fotogrametri, posisi titik koordinat digambarkan dengan sistem koordinat kartesian tiga dimensi *(Tony, 2006)*. Pada penelitian ini, akan dilakukan pengukuran fotogrametri dengan teknik *close range* dimana pada pelaksanaannya menggunakan dua kamera *CCTV* yang ditempatkan pada sisi atas bagian mobil. Pengukuran dengan teknik ini sering disebut dengan *Mobile Mapping System*. Tidak hanya kamera yang digunakan tetapi akan dikombinasikan dengan *GPS* (*Global Positioning System*) yang berfungsi untuk mendapatkan koordinat tiga dimensi dari posisi kamera itu sendiri

Akan tetapi, dalam pelaksanaannya masih banyak terdapat kendala yang ditemukan terutama berhubungan dengan masalah orientasi kamera. Pada saat pengukuran dilaksanakan, ada beberapa faktor yang menyebabkan sinyal *GPS* tidak dapat ditangkap dengan baik, salah satunya yaitu *GPS* yang dipasang terhalang oleh bangunan atau pohon.

Pada bab selanjutnya akan dicoba di analisa dan diselesaikan dengan menggunakan teknik atau metode tertentu, sehingga koordinat pada lokasi yang tidak dapat menangkap sinyal *GPS* dengan baik, dapat diketahui nilai koordinat tiga dimensinya.

1.2 Identifikasi Masalah

Bagaimana mendapatkan nilai parameter posisi orientasi kamera (ω, φ, κ, X_L, Y_L, Z_L) dengan menggunakan teknik *Resection* dan *Absolute Orientasi*. Kedua teknik tersebut akan dihitung menggunakan metode *Least Square*, sehingga dapat diketahui seberapa besar nilai ketelitian dari masing-masing parameter.

1.3 Maksud Dan Tujuan Penelitian

Maksud dari penelitian ini adalah menganalisa setiap tahapan kerja dari teknik *resection* dan *absolute orientation* untuk proses penentuan parameter orientasi kamera, sehingga dapat diketahui nilai akurasi atau nilai ketelitian dari masing-masing teknik tersebut. Adapun tujuan dari penelitian ini adalah untuk mendapatkan algorithma dari kedua teknik tersebut serta mengetahui nilai parameter orientasi kamera, meliputi sudut rotasi kamera (ω,φ, κ) dan koordinat tiga dimensinya (X_L, Y_L, Z_L), sehingga algorithma ini dapat digunakan sebagai dasar pembuatan *software* yang berhubungan dengan penyelesaian teknik *resection* dan *absolute orientation*.

1.4 Batasan Masalah

Permasalahan yang akan dikaji pada penelitian ini adalah perhitungan menggunakan teknik *resection* dan *absolute orientation* dengan tujuan untuk mendapatkan nilai parameter orientasi kamera (ω,φ, κ, X_L, Y_L, Z_L). Hasil dari

penelitian ini adalah algorithma perhitungan *resection* dan *absolute orientation* yang telah dibuktikan kebenarannya.

1.5 Manfaat Penelitian

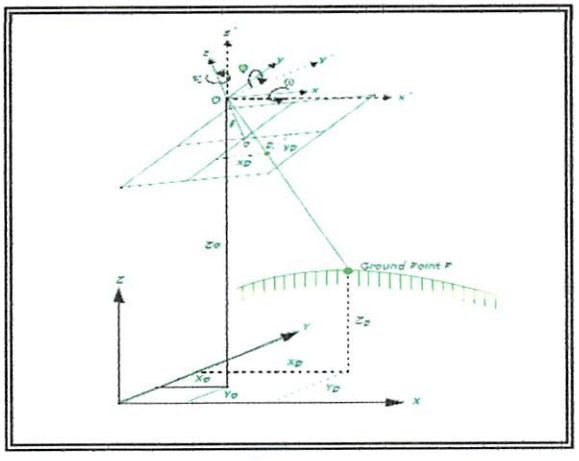
Penelitian ini dilakukan dengan harapan memberikan manfaat dalam bidang fotogrametri khususnya dalam membangun koordinat tiga dimensi dari posisi kamera serta mengetahui sudut rotasi kamera yang digunakan dalam penelitian dengan menggunakan teknik *resection* dan *absolute orientation*. Selain itu, dengan adanya penelitian ini, dapat mengetahui nilai ketelitian atau nilai akurasi dari setiap parameter eksterior orientasi kamera yang dihasilkan menggunakan teknik *least square*.

BAB II

DASAR TEORI

2.1 Space Resection

Space resection merupakan sebuah metode dalam ilmu fotogrametri yang digunakan untuk menentukan 6 elemen orientasi luar *(ω, φ, κ, X_L, Y_L, Z_L)* dari sebuah foto (*Wolf, 1993;Wolf & Dewitt, 2000)*. Menurut *(Mikhail, et al., 2001)*, *resection* merupakan sutau proses untuk penentuan posisi gambar dan orientasi parameter yang berkaitan dengan sistem koordinat objek. Sedangkan menurut (*Cooper & Robson, 2001; Zheng and Wang, 1992)*, *Space resection* merupakan sebuah metode untuk mengevaluasi secara langsung pada enam elemen orientasi bagian luar (*Exterior Orientation)* diperoleh dari diukurnya koordinat foto pada *image* dengan tiga titik kontrol non kolinear yang memerlukan beberapa nilai pendekatan.



Gambar 2.1 Elemen Orientasi Luar

Resection juga termasuk determinasi posisi dan orientasi parameter secara tidak langsung, sebagai contoh, penentuan koefisien dari penggabungan gambar *polynomial* untuk menentukan posisi dan orientasi waktu yang berkaitan dengan bermacam-macam sensor atau bentuk parameter proyeksi *(Mikhail, et al., 2001)*.

Ada banyak persamaan yang dapat digunakan untuk menyelesaikan permasalahan *resection*. Dalam kasus ini, *space resection* menggunakan persamaan kolinearitі berdasarkan pada kondisi kolinearitі yang syaratnya adalah dimana koordinat pada foto, koordinat titik kontrol berada pada satu garis lurus melewati *perspective center* pada kamera (*Jue, 2008)*. Karena ada 6 parameter Eksterior Orientasi (3 untuk orientasi sudut rotasi dan 3 untuk posisi kamera) maka minimum 3 titik kontrol dibutuhkan untuk menyelesaikan sistem ini.

Jika telah diketahui minimum tiga titik kontrol pada arah X, Y, dan Z maka *space resection* dapat dihitung untuk menentukan 6 parameter posisi orientasi luar (ω, φ, κ, X_L, Y_L, Z_L) pada sebuah *image* dengan asumsi bahwa parameter *interior orientation* juga telah diketahui. *Space resection* biasanya digunakan untuk ortorektifikasi pada satu foto , dimana proses perhitungannya dilakukan untuk setiap satu foto *(Leica, 2006)*. Apabila menggunakan lebih dari 3 titik kontrol, maka teknik perhitungan *Least Square Adjusment* dapat diaplikasikan untuk menentukan nilai yang paling mungkin bagi keenam parameter tersebut.

Meskipun persamaan kolinearitі sangat efektif untuk digunakan, *Space Resection* tetap membutuhkan nilai pendekatan awal untuk keenam parameter

Eksterior Orientasi. Oleh karena itu, untuk nilai pendekatan awal parameter *Ekterior Orientasi* dapat digunakan metode *Direct Linear Transformation* (*DLT*). Untuk lebih jelasnya metode ini akan dibahas secara singkat pada *sub bab 2.2.1*.

2.1.1 Penentuan Nilai Pendekatan Parameter Orientasi Luar

Pada proses perhitungan *resection* dibutuhkan minimal 6 parameter *Ekterior Orientasi* (ω, φ, κ, X_L, Y_L, Z_L). Oleh karena itu, pada subbab ini, akan dijelaskan bagaimana mencari nilai pendekatan paramater *Eksterior Orientasi* dengan menggunakan metode *Direct Linear transformation*.

Direct *Linear Transformation (DLT)* adalah transformasi model antara komparator atau sistem koordinat piksel foto dan sistem koordinat objek ruang sebagai fungsi linear *(Mikhail, et al., 2001)*. *DLT* dapat diturunkan dari persamaan kolinearitі standar atau sebagai kemungkinan yang lain, *DLT* dapat diasumsikan sebagai suatu implementasi geometri yang bersifat proyeksi. Koreksi kesalahan sistematis mungkin termasuk sebagai bagian dari transformasi, meskipun ini merupakan solusi nonlinear. Meskipun solusi linear lebih efisien dibandingkan dengan persamaan kolinear sepenuhnya, tetapi solusi ini masih kurang stabil apabila digunakan pada situasi yang lain. Sedikitnya 3 titik kontrol *noncoplanar* dengan tiga titik koordinat yang telah diketahui diperlukan untuk solusi linear *(Mikhail, et al., 2001)*. Untuk menurunkan persamaan *DLT* dimulai dengan persamaan kolinearitі, tetapi skala pada jarak

principal c berbeda dengan arah x dan y untuk menggambarkan *image* dari segi pembanding :

$$x+\delta x-x_0=-c_x\frac{m_{11}(X_A-X_L)+m_{12}(Y_A-Y_L)+m_{13}(Z_A-Z_L)}{m_{31}(X_A-X_L)+m_{32}(Y_A-Y_L)+m_{33}(Z_A-Z_L)}$$
$$x+\delta x-y_0=-c_y\frac{m_{21}(X_A-X_L)+m_{22}(Y_A-Y_L)+m_{23}(Z_A-Z_L)}{m_{31}(X_A-X_L)+m_{32}(Y_A-Y_L)+m_{33}(Z_A-Z_L)} \qquad [2.1]$$

Dimana x, y merupakan koordinat *image*, x_0, y_0 adalah koordinat *principal point*; c_x, c_y adalah jarak principal c yang diskalakan karena perbedaan faktor λ pada arah x dan y; X_A, Y_L, Z_L koordinat *perspective center* pada kamera; m_{ij} elemen matriks rotasi 3 x 3; δ_x, δ_y total distorsi lensa pada arah x dan y, sebagai fungsi untuk koefisien K_i *(Mikhail, et al., 2001)* :

$$\delta_x=(x-x_0)(K_1r^2+K_2r^4+\ldots)$$
$$\delta_y=(y-y_0)(K_1r^2+K_2r^4+\ldots) \qquad [2.2]$$

Menurut (*Cooper & Robson, 2001)* yang dikutip dari *(Azis & Karara, 1971),* selama kamera tidak berada pada posisi koordinat aslinya, persamaan kolineariti dapat ditransformasikan kedalam persamaan *DLT* yaitu :

$$x+\delta x=\frac{L_1X_P+L_2Y_P+L_3Z_P+L_4}{L_9X_P+L_{10}Y_P+L_{11}Z_P+1}$$
$$y+\delta y=\frac{L_5X_P+L_6Y_P+L_7Z_P+L_8}{L_9X_P+L_{10}Y_P+L_{11}Z_P+1} \qquad [2.3]$$

Dimana :

$$
\begin{aligned}
L &= -\frac{1}{m_{31}X_C + m_{32}Y_C + m_{33}Z_C} \\
L_1 &= L(x_0 m_{31} - c_x m_{11}) \\
L_2 &= L(x_0 m_{32} - c_x m_{12}) \\
L_3 &= L(x_0 m_{33} - c_x m_{13}) \\
L_4 &= x_0 + Lc_x(m_{11}X_C + m_{12}Y_C + m_{13}Z_C) \\
L_5 &= L(y_0 m_{31} - c_y m_{21}) \\
L_6 &= L(y_0 m_{32} - c_y m_{22}) \\
L_7 &= L(y_0 m_{33} - c_y m_{23}) \\
L_8 &= y_0 + Lc_y(m_{21}X_C + m_{22}Y_C + m_{23}Z_C) \\
L_9 &= Lm_{31} \\
L_{10} &= Lm_{32} \\
L_{11} &= Lm_{33}
\end{aligned}
\qquad [2.4]
$$

Perlu dicatat bahwa dalam persamaan ini, c_x, c_y, x_0, y_0, x, dan y berada pada komparator atau sistem koordinat piksel dan unit sehingga tidak membutuhkan *fiducial marks*. Untuk parameter kamera, dapat diturunkan dari parameter *DLT* dengan menggunakan hubungan sebagai berikut *(Chen, 1997))* :

$$
\begin{aligned}
L^2 &= L_9^{\,2} + L_{10}^{\,2} + L_{11}^{\,2} \\
x_0 &= L_1 L_9 + L_2 L_{10} + L_3 L_{11} \\
y_0 &= \frac{L_5 L_9 + L_6 L_{10} + L_7 L_{11}}{L^2} \\
c_x^{\,2} &= \frac{L_1^{\,2} + L_2^{\,2} + L_3^{\,2}}{L^2} - x_0^{\,2} \\
c_y^{\,2} &= \frac{L_5^{\,2} + L_6^{\,2} + L_7^{\,2}}{L^2} - y_0^{\,2}
\end{aligned}
\qquad [2.5]
$$

$$m_{31} = \frac{L_9}{L}$$

$$m_{32} = \frac{L_{10}}{L}$$

$$m_{33} = \frac{L_{11}}{L}$$

$$m_{11} = \frac{x_0 m_{31} - \frac{L_1}{L}}{c_x}$$

$$m_{12} = \frac{x_0 m_{32} - \frac{L_2}{L}}{c_x}$$

$$m_{13} = \frac{x_0 m_{33} - \frac{L_3}{L}}{c_x}$$

$$m_{21} = \frac{y_0 m_{31} - \frac{L_5}{L}}{c_y}$$

$$m_{22} = \frac{y_0 m_{32} - \frac{L_6}{L}}{c_y}$$

$$m_{23} = \frac{y_0 m_{33} - \frac{L_7}{L}}{c_y}$$

Sebagai catatan, matriks yang diturunkan atau diperoleh harus diortogonalisasi, karena kesalahan pembulatan dapat mempengaruhi nilai-nilai elemen. Dari matriks rotasi yang dihasilkan, dapat ditentukan nilai parameter sudut rotasi menggunakan persamaan sebagai berikut:

$$\tan\omega = \frac{-m_{32}}{m_{33}}$$

$$\sin\varphi = m_{31} \qquad [2.6]$$

$$\tan\kappa = \frac{-m_{21}}{m_{11}}$$

Untuk posisi kamera dapat dihitung dengan rumus sebagai berikut *(Seedahmed & Habib, 2002)*:

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = -\begin{bmatrix} L_1 & L_2 & L_3 \\ L_5 & L_6 & L_7 \\ L_9 & L_{10} & L_{11} \end{bmatrix}^{-1} \begin{bmatrix} L_4 \\ L_8 \\ 1 \end{bmatrix} \qquad [2.7]$$

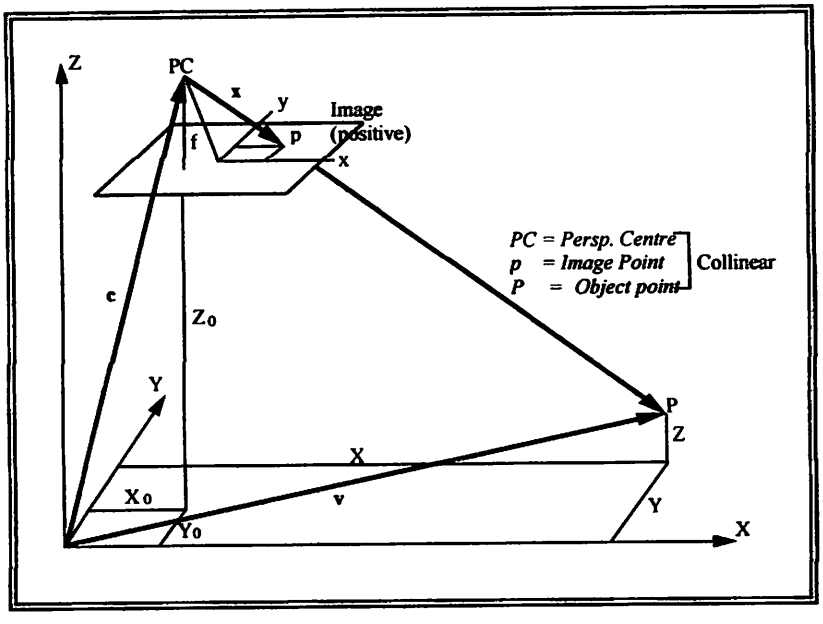
2.1.2 Perataan Nilai Parameter Orientasi Luar

Setelah mendapatkan nilai awal pendekatan untuk keenam parameter *eksterior orientasi* berupa ω, φ, κ, X_L, Y_L, Z_L, maka perhitungan *resection* dapat dilakukan dengan menggunakan persamaan kolinear. Karena pada penelitian ini akan menggunakan lebih dari 3 titik kontrol, maka metode *least square* akan diterapkan dalam proses perhitungan untuk meminimalisir nilai kesalahan dari solusi itu sendiri *(Wolf, 1993)*.

2.1.2.1 Persamaan Kolinear

Seperti yang telah diketahui, *space resection* dapat diselesaikan dengan menggunakan kondisi kolinearitas dimana koordinat *image* dan koordinat objek berada pada satu garis lurus melewati *perspective center* pada

kamera *(Leica, 2006)*. Metode ini membutuhkan minimum 3 titik kontrol dengan diketahui nilai koordinat objek yang berada pada foto. Kondisi kolinearitas dapat diilustrasikan seperti *Gambar (2.2)* dibawah ini :



Gambar 2.2 Kondisi Kolinearitas

Keterangan Gambar :

x_p, y_p : *Koordinat image*

Xo, Yo, Zo : *Koordinat kamera*

f : *Fokus kamera*

Pada umumnya persamaan dasar dari kondisi kolinearitas bersifat nonlinier sehingga persamaan ini dilinierkan menggunakan "*Teorema Taylor*". Berikut merupakan persamaan kolinear yang belum terliniearisasi *(Shih, 1996)* :

$$x_a = x_0 - f\left[\frac{m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)}\right] \quad [2.8]$$

$$y_a = y_0 - f\left[\frac{m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)}\right] \quad [2.9]$$

Dimana :

x_o, y_o	: *Koordinat foto titik a*
x_a, y_a	: *Koordinat foto yang diukur*
X_A, Y_A, Z_A	: *Koordinat Object Space untuk titik A*
X_L, Y_L, Z_L	: *Koordinat stasiun pemotretan*
f	: *Panjang fokus kamera*
m	: *3 sudut matriks rotasi ortogonal (ω, φ, κ)*

Persamaan (2.8) dan *(2.9)* diatas menunjukkan kondisi kolinearitas untuk setiap titik pada foto, satu persamaan untuk koodinat foto *x* dan persamaan yang lain untuk koordinat foto *y (Wolf, 1993)*. Dalam melinearkan persamaan kolinear, *persamaan (2.8)* dan *(2.9)* dituliskan lagi sebagai berikut :

$$F = 0 = qx_a + rf \quad [2.9]$$

$$G = 0 = qy_a + sf \quad [2.10]$$

Dimana :

$$\begin{aligned} r &= m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L) \\ s &= m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L) \\ q &= m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + r_{33}(Z_A - Z_L) \end{aligned} \quad [2.11]$$

Untuk nilai *m* merupakan nilai matriks rotasi 3x3 dimana masing-masing nilai dapat diuraikan sebagai berikut *(Kobayashi & Mori, 1997)* :

$$
\begin{aligned}
m_{11} &= \cos\varphi\cos\kappa \\
m_{12} &= \sin\omega\sin\varphi\cos\kappa + \cos\omega\sin\kappa \\
m_{13} &= -\cos\omega\sin\varphi\cos\kappa + \sin\omega\sin\kappa \\
m_{21} &= -\cos\varphi\sin\kappa \\
m_{22} &= -\sin\omega\sin\varphi\sin\kappa + \cos\omega\cos\kappa \\
m_{23} &= \cos\omega\sin\varphi\sin\kappa + \sin\omega\cos\kappa \\
m_{31} &= \sin\varphi \\
m_{32} &= -\sin\omega\cos\varphi \\
m_{33} &= \cos\omega\cos\varphi
\end{aligned}
\qquad [2.12]
$$

Menurut teori deret *Taylor*, *Persamaan (2.10)* dan *(2.11)* dapat dinyatakan dalam bentuk linier sebagai berikut ;

$$
\begin{aligned}
0 &= (F)_0 + \left(\frac{\delta F}{\delta x_a}\right)_0 dx_a + \left(\frac{\delta F}{\delta \omega}\right)_0 d\omega + \left(\frac{\delta F}{\delta \varphi}\right)_0 d\varphi + \left(\frac{\delta F}{\delta \kappa}\right)_0 d\kappa + \\
&\left(\frac{\delta F}{\delta X_L}\right)_0 dX_L + \left(\frac{\delta F}{\delta Y_L}\right)_0 dY_L + \left(\frac{\delta F}{\delta Z_L}\right)_0 dZ_L \\
0 &= (G)_0 + \left(\frac{\delta G}{\delta x_a}\right)_0 dx_a + \left(\frac{\delta G}{\delta \omega}\right)_0 d\omega + \left(\frac{\delta G}{\delta \varphi}\right)_0 d\varphi + \left(\frac{\delta F}{\delta \kappa}\right)_0 d\kappa + \\
&\left(\frac{\delta F}{\delta X_L}\right)_0 dX_L + \left(\frac{\delta F}{\delta Y_L}\right)_0 dY_L + \left(\frac{\delta F}{\delta Z_L}\right)_0 dZ_L
\end{aligned}
\qquad [2.13]
$$

Pada *Persamaan (2.13)*, F_0 dan G_0 merupakan fungsi F dan G sedangkan untuk *persamaan (2.9)* dan *(2.10)* dihitung pendekatan awal dari sembilan unsur yang tidak diketahui. Istilah $(\partial F/\partial x_a)_0$, $(\partial F/\partial \omega)_0$, $(\partial F/\partial \Phi)_0$, dan seterusnya merupakan turunan parsial dari fungsi F dan G dengan

mempertimbangkan unsur yang belum diketahui pada pendekatan awal. Unit $\partial\omega$, $\partial\Phi$, $\partial\kappa$ adalah radian. Karena dx_a dan dy_a merupakan koreksi untuk koordinat untuk koordinat foto terukur x_a dan y_a, maka dapat diinterpretasi sebagai kesalahan residual didalam pengukuran. Oleh karena itu dua istilah ini dapat diganti dengan Vx_a dan Vy_a yang merupakan simbol yang lazim digunakan untuk kesalahan residual. Perhatikan dari *Persamaan (2.9)* dan (2.10) yang jabaran parsialnya $\partial F/\partial x_a$, dan $\partial G/\partial y_a$, keduanya sama dengan q.

Dengan subtitusi q untuk istilah pada *Persamaan (2.12)* dengan memindahkan qdx_a dan qdy_a kesisi persamaan, membagi tiap persamaan, dengan q, dan mengganti dx_a dan dy_a masing-masing dengan vx_a dan vy_a. Sehingga apabila persamaan ini digunakan dalam penyelesaian secara *Least Square* maka diperoleh persamaan kolinearitas terlinearisasi dalam bentuk yang disederhanakan termasuk untuk nilai residualnya sebagai berikut *(Wolf, 1993)* :

$$\begin{aligned} b_{11}d\omega + b_{12}d\varphi + b_{13}d\kappa - b_{14}dX_L - b_{15}dY_L - b_{16}dZ_L &= J + vx_a \\ b_{21}d\omega + b_{22}d\varphi + b_{23}d\kappa - b_{24}dX_L - b_{25}dY_L - b_{26}dZ_L &= K + vy_a \end{aligned} \qquad [2.14]$$

Dimana :

$$b_{11} = \frac{f}{q^2}[r(-m_{33}\Delta Y + m_{32}\Delta Z) - q(-m_{13}\Delta Y + m_{12}\Delta Z)]$$

$$b_{12} = \frac{f}{q^2}[r(\cos\varphi\,\Delta X + \sin\omega\sin\varphi\,\Delta Y - \cos\omega\sin\varphi\,\Delta Z)$$

$$-q(-\sin\varphi\cos\kappa\,\Delta X+\sin\omega\cos\varphi\cos\kappa\,\Delta Y-\cos\omega\cos\varphi\cos\kappa\,\Delta Z)]$$

$$b_{13}=\frac{-f}{q}(m_{21}\Delta X+m_{22}\Delta Y+m_{23}\Delta Z)$$

$$b_{14}=\frac{f}{q^2}(rm_{31}-qm_{11})$$

$$b_{15}=\frac{f}{q^2}(rm_{32}-qm_{12}) \qquad [2.15]$$

$$b_{16}=\frac{f}{q^2}(rm_{33}-qm_{13})$$

$$b_{21}=\frac{f}{q^2}[s(-m_{33}\Delta Y+m_{32}\Delta Z)-q(-m_{23}\Delta Y+m_{22}\Delta Z)]$$

$$b_{22}=\frac{f}{q^2}[s(\cos\varphi\,\Delta X+\sin\omega\sin\varphi\,\Delta Y-\cos\omega\sin\varphi\,\Delta Z)$$

$$-q(\sin\varphi\cos\kappa\,\Delta X-\sin\omega\cos\varphi\sin\kappa\,\Delta Y+\cos\omega\cos\varphi\sin\kappa\,\Delta Z)]$$

$$b_{23}=\frac{f}{q}(m_{11}\Delta X+m_{12}\Delta Y+m_{13}\Delta Z)$$

$$b_{24}=\frac{f}{q^2}(sm_{31}-qm_{21})$$

$$b_{25}=\frac{f}{q^2}(sm_{32}-qm_{22})$$

$$b_{26}=\frac{f}{q^2}(sm_{33}-qm_{23})$$

$$J=xa-xo+f\frac{r}{q}$$

$$K=ya-yo+f\frac{s}{q}$$

2.1.2.2 Proses Perataan dengan *Least Square*

Least Square Adjusment atau metode kuadrat terkecil merupakan suatu prosedur untuk menyesuaikan pengamatan yang mengandung kesalahan acak *(Wolf, 1993)*. *Mikhail et al. (2001)*, mengatakan bahwa *least square adjustment* merupakan sebuah metode sistematik yang digunakan untuk menghitung nilai unik dari setiap koordinat dan elemen yang lain berdasarkan hasil jumlah yang besar disebabkan pengukuran yang berlebihan. Sedangkan menurut *Yi & Jue (2008)*, *least square* merupakan salah satu dari beberapa teknik yang digunakan untuk mengestimasi parameter linear yang telah dirancang untuk mengkompensasi kesalahan data.

Dalam ilmu fotogrametri sendiri, metode *least square adjustment* digunakan untuk beberapa proses, antara lain *(Leica, 2006)* :

1. Mengestimasi atau meratakan nilai parameter *exterior orientation*
2. Mengestimasi nilai *object space point* (X, Y, dan Z) beserta nilai keakurasiannya
3. Mengestimasi dan meratakan nilai parameter *interior orientation*
4. Meminimalisasi dan mendistribusikan errors data melalui jaringan pengamatan.

Kesalahan pada data adalah disebabkan oleh ketidaktepatan yang terkait dengan input koordinat titik kontrol tanah, pengukiran titik ikat dan koordinat titik kontrol pada *image*. Pendekatan kuadrat terkecil memerlukan pengolahan iteratif hingga solusi yang dicapai. Solusi diperoleh ketika nilai

residual atau kesalahan yang terkait dengan data *input* dapat diminimalisir *(Leica, 2006)* :

Dalam bentuk persamaan, persyaratan utama untuk *least square adjustment* dinyatakan sebagai berikut :

$$\sum (V_i)^2 = (v_1)^2 + (v_2)^2 + (v_3)^2 + ... + (v_{mi})^2 = \text{minimum} \qquad [2.16]$$

Untuk bentuk sederhana dari persamaan *least square* yang dilakukan dengan pendekatan aljabar dalam bentuk matriks dapat dituliskan sebagai berikut :

$$_m A_{n\,n} X_1 = {}_m L_1 + {}_m V_1 \qquad [2.17]$$

Atau

$$_m V_1 = {}_m A_{n\,n} X_1 - {}_n L_1 \qquad [2.18]$$

Pada persamaan ini :

V = *Matriks residu dari koordinat image*

A = *Matriks koefisien parameter unknown*

X = *Matriks koreksi dari parmater unknown*

L = *Matriks observasi / pengamatan dari data input*

Dimana untuk setiap notasi diatas diwakili oleh susunan matriks sebagai berikut :

$$A=\begin{vmatrix} a_{11a} & a_{12a} & a_{13a} & -a_{14a} & -a_{15a} & -a_{16a} \\ a_{21a} & a_{22a} & a_{23a} & -a_{24a} & -a_{25a} & -a_{26a} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{11n} & a_{12n} & a_{13n} & -a_{14n} & -a_{15n} & -a_{16n} \\ a_{21n} & a_{22n} & a_{23n} & -a_{24n} & -a_{25n} & -a_{26n} \end{vmatrix}_{m}^{n} \qquad [2.19]$$

$$X=\begin{vmatrix} X_1 \\ X_2 \\ X_3 \\ \vdots \\ \vdots \\ \vdots \\ X_n \end{vmatrix}_{n}^{1} \quad L=\begin{vmatrix} L_1 \\ L_2 \\ L_3 \\ \vdots \\ \vdots \\ \vdots \\ L_m \end{vmatrix}_{m}^{1} \quad V=\begin{vmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ \vdots \\ \vdots \\ V_m \end{vmatrix}_{m}^{1} \qquad [2.20]$$

Dengan mempelajari penyajian matriks, akan terlihat bahwa persamaan normal dapat diperoleh sebagai berikut *(Hande Demirel, 2008)* :

$$A^T AX = A^T L \qquad [2.20]$$

Pada Persamaan (2.11), A^TA adalah matriks koefisien persamaan normal dari bilangan yang tidak diketahui. Dengan mengalikan persamaan diatas dengan $A^T A$ dan kurangkan, hasilnya adalah :

$$\left(A^T A\right)^{-1}\left(A^T A\right)X = \left(A^T A\right)^{-1} A^T L$$

$$IX = \left(A^T A\right)^{-1} A^T L \qquad [2.21]$$

$$X = \left(A^T A\right)^{-1} A^T L$$

Bagi suatu sistem pengamatan terbobot, persamaan matriks berikut menyajikan matriks X bagi nilai paling mungkin untuk nilai yang tidak dikenal.

Dimana :

X = *Matriks koreksi dari parameter unknown*

A = *Matriks koefisien atau matriks Jacobian*

L = *Matriks Pengamatan / Observasi*

P = *Matriks bobot*

Di dalam persamaan matriks identik terhadap persamaan bobot, kecuali bahwa matriks P merupakan matriks diagonal bobot.

2.2 Absolut Orientasi

Absolut orientasi merupakan proses dalam ilmu fotogrametri untuk menentukan nilai skala dan elemen orientasi dari model geometri dengan membangun terlebih dahulu orientasi relatif pada setiap *image (Zhang & Yao, 2008)*. Ada tujuh parameter yang akan dicari pada metode ini yaitu satu faktor skala (s), tiga faktor rotasi (ω, φ, κ) dan faktor translasi (T_x, T_y, T_z) *(Mikhail et al., 2001)*.

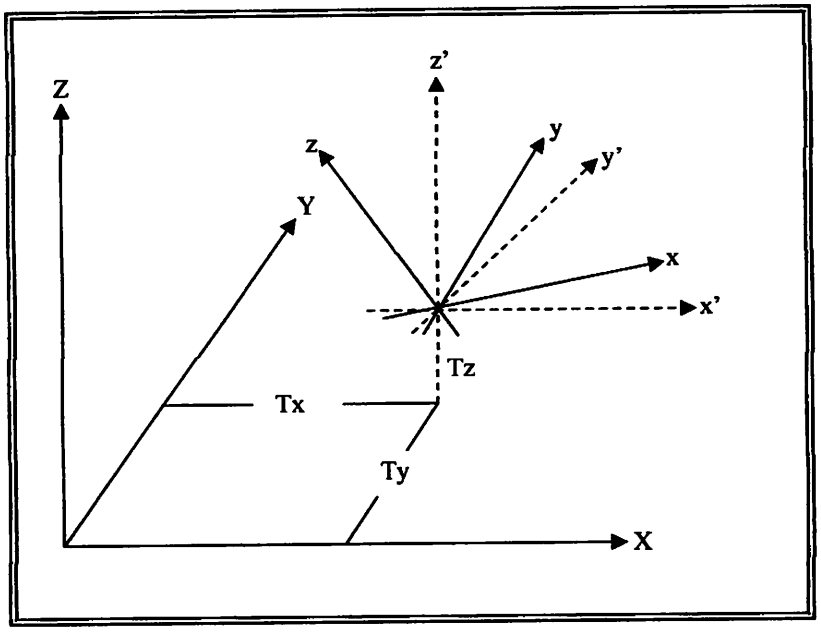
Metode ini akan digunakan untuk menentukan nilai koordinat objek 3D dengan menghitung terlebih dahulu nilai pendekatan untuk ketujuh parameter. Jika telah didapatkan nilai ketujuh parameter tersebut, maka nilai koordinat objek 3D dapat dihitung. Untuk lebih jelasnya, metode ini akan di jelaskan secara lengkap pada subbab selanjutnya.

2.2.1 Transformasi Koordinat Konform 3 Dimensi

Seperti yang tercermin oleh namanya, transformasi koordinat konform 3D meliputi pengubahan dari suatu sistem 3D ke sistem lainnya. Didalam transformasi, bentuk yang benar tetap dipertahankan. Jenis transformasi koordinat ini penting didalam fotogrametri analitik dan automatik fotogrametri sehubungan dengan dua masalah pokok *(Wolf, 1993; Wolf & Dewit, 2000)* :

1. Untuk mengubah koordinat titik-titik dari sistem koordinat foto yang mengalami kecondongan (*tilt*) ke sistem foto tegak ekuivalennya yang sejajar dengan sistem ruang medan atau sembarang.
2. Untuk membentuk model jalur 3D dari model stereo.

Persamaan transformasi koordinat konform 3D akan dibahas pada sub bab ini dan akan digunakan dalam proses penentuan parameter eksterior orientasi.



Gambar 2.3 Sistem koordinat 3D XYZ dan xyz arah kanan

Sesuai dengan *Gambar (2.3)* dilakukan pengubahan koordinat titik dari sebuah sistem *x, y, z* ke sistem *X, Y, Z*. Seperti tercermin dalam gambar tersebut, dua sitem koordinat tidak sejajar. Persamaan transformasi yang perlu dapat dinyatakan sesuai dengan tujuh parameter transformasi yang terdiri dari

tiga parameter sudut rotasi omega *(ω)*, phi *(φ)* dan kappa *(κ)*, sebuah faktor skala *(s)* dan tiga parameter translasi *(T_x, T_y, T_z)*. Sistem rotasi ω, φ, κ akan bernilai positif apabila arah rotasinya berlawanan terhadap arah jarum jam apabila diamati dari ujung positif sumbunya *(Wolf, 1993)*.

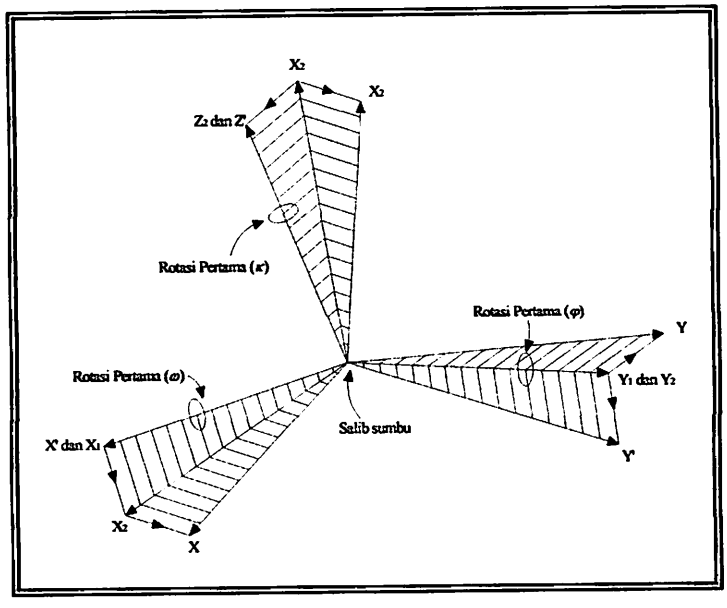
Persamaan transformasi harus dikembangkan dengan dua langkah dasar antara lain rotasi, penskalaan dan translasi. Kedua langkah tersebut akan dijelaskan secara detail pada sub bab dibawah ini.

2.2.1.1 Faktor Rotasi

Pada Gambar (2.3) dibuat sistem koordinat x', y', z' sejajar dengan sistem objek x, y, z dengan salib sumbu pada sistem salib sumbu xyz. Didalam mengembangkan formula rotasi, biasanya dipertimbangkan bahwa tiga rotasi terjadi sedemikian hingga seperti berubah dari sistem x', y', z' kesistem xyz.

Persamaan rotasi dikembangkan didalam suatu rangkaian tiga rotasi dua dimensi. Rotasi ini, seperti yang terlihat pada *Gambar (2.4)* yang pertama ialah rotasi ω pada sumbu x' yang mengubah koordinat dari sistem x', y', z' kesistem x_1, y_1, z_1, yang kedua adalah rotasi ω pada sumbu y_1 yang berputar dan mengubah koordinat dari sistem x_1, y_1, z_1 ke sistem x_2, y_2, z_2 dan ketiga, rotasi κ yang berputar mengubah sistem koordinat x_2, y_2, z_2 kesistem x, y, z. Jumlah dan arah putaran yang tepat bagi tiap transformasi koordinat tiga

dimensi akan bergantung pada hubungan arah antara sistem koordinat *x, y, z* dan *X, Y, Z*.

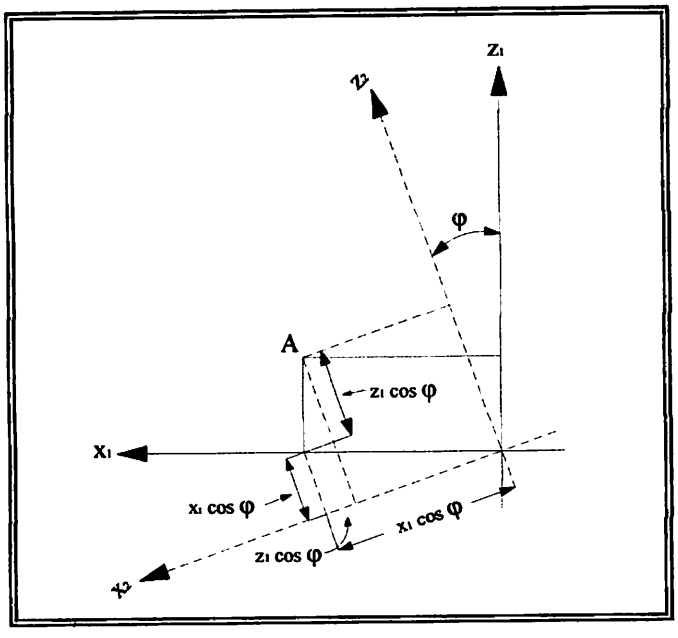


Gambar 2.4 Rotasi ketiga sudut

$$
\begin{aligned}
x_1 &= x' \\
y_1 &= y'\cos\omega + z'\sin\omega \\
z_1 &= -y'\sin\omega + z'\cos\omega
\end{aligned}
\qquad [2.22]
$$

Karena rotasi itu bersumbu putar *x'*, maka sumbu *x'* dan x_1 berhimpit dan oleh karenanya koordinat *x* bagi *A* tidak berubah.

Yang kedua, rotasi dengan sudut *phi* pada sumbu y_1 seperti tercermin pada *Gambar 2.6*. Koordinat *A* dalam sistem koordinat x_2, y_2, z_2 yang berputar, seperti yang disajikan dalam *Gambar 2.6* adalah :



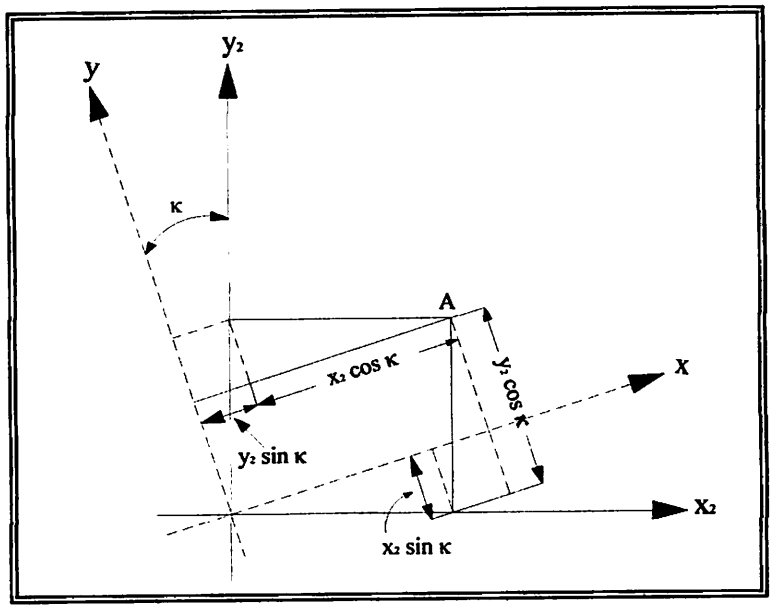
Gambar 2.5 Rotasi phi pada sumbu putar y_1

$$
\begin{aligned}
x_2 &= -z_1 \sin\varphi + x_1 \cos\varphi \\
y_2 &= y_1 \\
z_2 &= z_1 \cos\varphi + x_1 \sin\varphi
\end{aligned}
\qquad [2.23]
$$

Pada rotasi diseputar y_1, sumbu y_1 dan y_2 berimpit dan oleh karenanya koordinat y bagi titik A tidak berubah. Dengan jalan subtitusi *persamaan (2.22)* ke *(2.23)* :

$$
\begin{aligned}
x_2 &= -(-y'\sin\omega + z'\cos\omega)\sin\varphi + x'\cos\varphi \\
y_2 &= y'\cos\omega + z'\sin\omega \\
z_2 &= (-y'\sin\omega + z'\cos\omega)\cos\varphi + x'\sin\varphi
\end{aligned}
\qquad [2.24]
$$

Ketiga, rotasi melalui κ yang bersumbu putar z_2, seperti tercermin pada *Gambar 2.7*. Koordinat A didalam sistem koordinat yang telah diputar tiga kali, yang sekarang menjadi sistem x, y, z seperti yang disajikan pada *Gambar 2.7* ialah :



Gambar 2.6 Rotasi kappa pada sumbu putar z_2

$$
\begin{aligned}
x &= x_2 \cos\kappa + y_2 \sin\kappa \\
y &= -x_2 \sin\kappa + y_2 \cos\kappa \\
z &= z_2
\end{aligned}
\qquad [2.25]
$$

Didalam rotasi diseputar z_2 ini, sumbu z_2 dan z berimpit dan oleh karenanya koordinat z bagi A tidak berubah. Dengan Subtitusi *Persamaan (2.23)* ke *(2.24)* :

$$
\begin{aligned}
x &= [(y'\sin\omega - z'\cos\omega)\sin\varphi + x'\cos\varphi]\cos\kappa + \\
&\quad (y'\cos\omega + z'\sin\omega)\sin k \\
y &= [(-y'\sin\omega + z'\cos\omega)\sin\varphi - x'\cos\varphi]\sin\kappa + \\
&\quad (y'\cos\omega + z'\sin\omega)\cos k \\
z &= (-y'\sin\omega + z'\cos\omega)\cos\varphi + x'\sin\varphi
\end{aligned}
\qquad [2.26]
$$

Dengan melakukan perkalian *Persamaan (2.26)* :

$$
\begin{aligned}
x &= x'(\cos\varphi\cos\kappa) + y'(\sin\omega\sin\varphi\cos\kappa + \cos\omega\sin\kappa) \\
&\quad + z'(-\cos\omega\sin\varphi\cos\kappa + \sin\omega\sin\kappa) \\
y &= x'(-\cos\varphi\sin\kappa) + y'(-\sin\omega\sin\varphi\sin\kappa + \cos\omega\cos\kappa) \\
&\quad + z'(\cos\omega\sin\varphi\sin\kappa + \sin\omega\cos\kappa) \\
z &= x'(\sin\varphi) + y'(-\sin\omega\cos\varphi) + z'(\cos\omega\cos\varphi)
\end{aligned}
\qquad [2.27]
$$

Dengan subtitusi m untuk koefisien $x'y'z'$ pada *Persamaan (2.27)*, maka persamaan ini menjadi :

$$
\begin{aligned}
x &= m_{11}x' + m_{12}y' + m_{13}z' \\
y &= m_{21}x' + m_{22}y' + m_{23}z' \\
z &= m_{31}x' + m_{32}y' + m_{33}z
\end{aligned}
\qquad [2.28]
$$

Dimana :

$$
\begin{aligned}
m_{11} &= \cos\varphi\cos k \\
m_{12} &= \sin\omega\sin\varphi\cos\kappa + \sin\omega\sin\kappa \\
m_{13} &= -\cos\omega\sin\varphi\cos k + \sin\omega\sin\kappa \\
m_{21} &= -\cos\varphi\sin\kappa \\
m_{22} &= -\sin\omega\sin\varphi\sin\kappa + \cos\omega\cos k \\
m_{23} &= \cos\omega\sin\varphi\sin k + \sin\omega\cos k \\
m_{31} &= \sin\varphi \\
m_{32} &= -\sin\omega\cos\varphi \\
m_{33} &= \cos\omega\cos\varphi
\end{aligned}
\qquad [2.29]
$$

Persamaan (2.28) dapat dinyatakan dalam bentuk matriks sebagai :

$$X = MX' \qquad [2.30]$$

Dimana :

$$X = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \text{ dan } X' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \qquad [2.31]$$

Matriks M biasanya disebut matriks rotasi. Unsur individual matriks rotasi merupakan cosinus arah yang menghubungkan dua sistem sumbu. Unsur matriks dinyatakan didalam cosinus arah adalah :

$$M = \begin{bmatrix} \cos xx' & \cos xy' & \cos xz' \\ \cos yx' & \cos yy' & \cos yz' \\ \cos zx' & \cos zy' & \cos zz' \end{bmatrix} \qquad [2.32]$$

Pada matriks diatas, cos xx' ialah cosinus arah yang menghubungkan sumbu x dan x', cos xy' menghubungkan sumbu x dan y' dan seterusnya. Cosinus arah adalah cosinus sudut dalam ruang antara sumbu-sumbu yang bersangkutan, diukur dari 0°-180°. Merupakan satu sifat penting bahwa jumlah kuadrat tiga cosinus arah suatu garis lurus merupakan suatu kesatuan. Sifat ini dapat digunakan untuk mngecek unsur terhitung bagi matriks rotasi untuk menjaga ketelitian. Cek ini diperoleh bila jumlah kuadrat unsur-unsur suatu jalur atau lajur matriks M sama besar dengan 1. Matriks rotasi merupakan sebuah matriks *ortogonal* yang memiliki sifat bahwa kebalikannya sama besar dengan 'transposenya', atau :

$$M^{-1} = M^T \qquad [2.33]$$

Dengan menggunakan sifat ini, *Persamaan (2.30)* dapat ditulis kembali untuk menyatakan koordinat x', y', z' sesuai dengan koordinat x, y, z sebagai berikut :

$$X' = M^T X \qquad [2.34]$$

Didalam bentuk yang diperluas, persamaan ini menjadi :

$$\begin{aligned} x' &= m_{11}x + m_{21}y + m_{31}z \\ y' &= m_{12}x + m_{22}y + m_{32}z \\ z' &= m_{13}x + m_{23}y + m_{33}z \end{aligned} \qquad [2.35]$$

2.2.1.2 Skala dan Translasi

Untuk sampai pada akhir persamaan transformasi koordinat tiga dimensi, misalnya persamaan yang membuahkan koordinat didalam sistem *X, Y, Z* dalam Gambar 2.4, perlu mengalikan tiap Persamaan (2.35) dengan faktor skala *s* dan menambahkan faktor translasi T_x, T_y *dan* T_z (koordinat yang *x', y', z'* pada Persamaan (2.35) berada dalam sistem yang sejajar dengan sistem *X, Y, Z.* Ini menyebabkan panjang tiap garis sama besar pada kedua sistem koordinat dan menterjemahkan dari salib sumbu *x', y', z'* ke sistem salib sumbu *X, Y, Z.* Pelaksanaan untuk langkahnya adalah sebagai berikut *(Romsek, 2004)* :

$$\begin{aligned} X &= sx' + T_x = s(m_{11}x + m_{21}y + m_{31}z) + T_x \\ Y &= sy' + T_y = s(m_{12}x + m_{22}y + m_{32}z) + T_y \\ Z &= sz' + T_x = s(m_{13}x + m_{23}y + m_{33}z) + T_z \end{aligned} \qquad (2.36)$$

Didalam bentuk matriks, *Persamaan (2.34)* yaitu :

$$\overline{X} = sM^T X + T \qquad [2.37]$$

Didalam *Persamaan (2.37)* matriks *M* dan *X* seperti yang telah didefinisikan, *s* merupakan faktor skala, dan :

$$\overline{X} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad \text{dan} \quad T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \qquad [2.38]$$

Didalam *Persamaan (2.36)*, sembilan m tidak bebas terhadap yang lain, akan tetapi seperti terlihat pada *Persamaan (2.29)*, m tersebut merupakan fungsi tiga sudut rotasi ω, φ, κ. Sebagai tambahan terhadap tiga sudut yang tidak diketahui ini, juga ada tiga faktor translasi dan satu faktor skala yang tidak diketahui didalam *Persamaan (2.36)*, yang keseluruhannya membuahkan tujuh faktor yang tidak diketahui. Suatu penyelesaian unik diperoleh bagi yang tidak diketahui apabila koordinat x dan y dua titik mendatar dan koordinat Z bagi titik tegak diketahui, juga kedua sistem tersebut. Apabila pada kedua sistem diketahui, lebih dari tujuh koordinat, dapat ditulis persamaan ulangan yang memungkinkan perbaikan penyelesaian dengan cara kuadrat terkecil *(Wolf, 1993)*.

2.2.2 Penentuan Nilai Pendekatan Parameter Transformasi

Sebelum melakukan perhitungan untuk tujuh parameter, dilakukan terlebih dahulu perhitungan untuk nilai pendekatan terhadap ketujuh parameter tersebut. Hal ini akan diuraikan secara jelas pada subbab di bawah ini.

2.2.2.1 Nilai Pendekatan Faktor Skala

Proses perhitungan untuk faktor skala sangatlah sederhana *(Dewitt, 1996)*. Dengan memilih sepasang titik yang mewakili dua sistem yang berbeda dan dilakukan proses perhitungan rasio untuk tiap yang terdapat pada dua sistem tersebut. Dalam proses transformasi, faktor skala ini dapat dihitung dengan menggunakan persamaan sebagai berikut :

$$Skala = \frac{jarak\ pada\ sistem\ kontrol}{jarak\ pada\ sistem\ sembarang} \qquad [2.39]$$

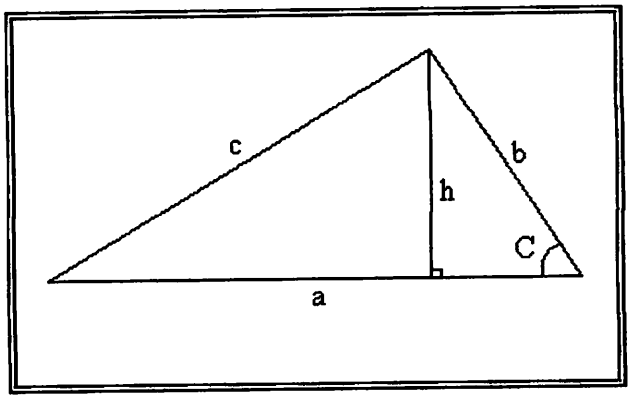
Dengan mengasumsikan tidak ada nilai kesalahan dan garis yang diwakili memiliki panjang yang cukup, maka akurasi nilai pendekatan untuk faktor skala dapat ditentukan dengan persamaan diatas.

2.2.2.2 Nilai Pendekatan Faktor Rotasi

Penentuan nilai pendekatan yang akurat untuk tiga faktor rotasi adalah persoalan yang cukup sulit untuk diselesaikan. Pada prosesnya akan dibahas dengan langkah-langkah sebagai berikut *(Dewwit, 1996)* :

Yang pertama ialah memilih tiga titik dengan bentuk geometri yang baik. Dalam proses penentuan nilai geometri yang baik, tiga titik yang tersebar pada kedua sistem harus dipilih. Ini dilakukan untuk mengurangi kemungkinan memilih tiga titik yang hampir segaris. Pemilihan tiga titik didasarkan pada pembentukan sebuah segitiga yang memiliki nilai tinggi segitiga terbesar, dimana ketinggian tersebut didefinisikan sebagai jarak tegak

lurus dari sisi terpanjang. Seperti yang diilustrasikan pada gambar dibawah ini :



Gambar 2.7 Tinggi dari sisi yang terpanjang.

Penentuan ini dapat dilakukan baik dalam sistem kontrol atau sistem sembarang dengan anggapan dari kedua sistem tersebut tidak terdapat nilai kesalahan. Pada Gambar (2.8) mengilustrasikan bagaimana tinggi dari segitiga ini didefinisikan. Dalam gambar diatas, sisi terpanjang dinotasikan dengan (a) dan dikarenakan garis ketinggian tegak lurus terhadap sisi ini, karena ditentukan ketinggian tegak lurus terhadap sisi ini, sehingga didapat dua buah segitiga yang sebangun. Persamaan untuk menentukan nilai ketinggian dapat dituliskan dalam persamaan dibawah ini :

$$h^2 = b^2 - \left(\frac{a^2 + b^2 - c^2}{2a} \right) \qquad [2.40]$$

Kedua, menghitung nilai vektor normal menggunakan *Persamaan (2.41)* untuk kombinasi titik yang memiliki nilai h terbesar. Proses ini akan

menghasilkan dua pasang koefisien *a*, *b* dan *c* yang merupakan koefisien untuk sistem sembarang dan sistem kontrol.

$$\begin{aligned} a &= (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1) \\ b &= (x_3 - x_1)(z_2 - z_1) - (x_2 - x_1)(z_3 - z_1) \\ c &= (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1) \end{aligned} \qquad [2.41]$$

Persamaan diatas merupakan nilai koefisien dari sebuah unit vektor normal yang didefinisikan dalam arah *i, j, k* dan dapat dituliskan dalam sebuah persamaan matematika sebagai berikut :

$$N = a \cdot \bar{i} + b \cdot \bar{j} + c \cdot \bar{k} \qquad [2.42]$$

Ketiga, menentukan nilai parameter sudut rotasi dalam sistem rotasi *tilt*, *swing*, dan *azimuth*. Parameter *tilt* dan *azimuth* dapat dihitung menggunakan nilai koefisien vektor normal seperti yang telah dijelaskan pada Persamaan (2.41) sedangkan nilai parameter sudut rotasi untuk *swing* dapat diberikan nilai awal nol. Adapun persamaan untuk menentukan nilai *tilt* dan *azimuth* dapat digunakan persamaan dibawah ini :

$$\begin{aligned} tilt &= \tan^{-1}\left(\frac{c}{\sqrt{a^2 + b^2}}\right) + 90° \\ azimuth &= \tan^{-1}\left(\frac{a}{b}\right) \end{aligned} \qquad [2.43]$$

Keempat, menghitung nilai sudut rotasi pada kedua sistem menggunakan nilai yang berhubungan dengan *tilt* dan *azimuth*. Rotasi ini akan menyebabkan bidang yang didefinisikan oleh tiga titik sejajar terhadap bidang *xy* pada kedua sistem. Ini diselesaikan dengan menghitung matriks rotasi pada

kedua sistem berdasarkan pada titik yang mewakili nilai untuk *tilt* dan *azimuth* dengan *swing* yang diset pada nilai 0°. Untuk lebih jelasnya, dibawah ini merupakan persamaan yang menunjukkan matriks rotasi *(Dewwit, 1996)* :

$$
\begin{aligned}
m_{11} &= -\cos(\alpha)\cos(s) - \sin(\alpha)\cos(t)\sin(s) \\
m_{12} &= \sin(\alpha)\cos(s) - \cos(\alpha)\cos(t)\sin(s) \\
m_{13} &= -\sin(t)\sin(s) \\
m_{21} &= \cos(\alpha)\sin(s) - \sin(\alpha)\cos(t)\cos(s) \\
m_{22} &= -\sin(\alpha)\sin(s) - \cos(\alpha)\cos(t)\cos(s) \\
m_{23} &= -\sin(t)\cos s \\
m_{31} &= -\sin(\alpha)\sin(t) \\
m_{32} &= -\cos(\alpha)\sin(t) \\
m_{33} &= \cos(t)
\end{aligned}
\qquad [2.44]
$$

Matriks rotasi ini lalu diterapkan terhadap dua diantara tiga titik pada setiap sistem, menghasilkan titik yang ditransformasikan pada koordinat *z*. Penyelesaian ini dapat selesaikan dengan menggunakan *Persamaan (2.35)* yang dituliskan kembali seperti dibawah ini :

$$x' = m_{11}x + m_{21}y + m_{31} \qquad [2.45]$$

$$y' = m_{12}x + m_{22}y + m_{32}z \qquad [2.46]$$

$$z' = m_{13}x + m_{23}y + m_{33}z \qquad [2.47]$$

Persamaan ini diaplikasikan sebanyak empat kali, yang pertama adalah untuk setiap dua titik pada sistem kontrol dan kedua untuk setiap dua titik yang sama pada sistem sembarang. Untuk dua aplikasi yang pertama, matriks rotasi untuk sistem kontrol menggunakan koordinat titik kontrol *x, y, z* untuk dua titik. Untuk dua aplikasi yang terakhir, matriks rotasi pada sistem

sembarang menggunakan titik yang berhubungan dengan koordinat *xyz* pada sistem sembarang *(Dewwit, 1996)*.

Koordinat utama yang dihasilkan mengartikan garis horizontal pada kedua sistem dan penerapannya memerlukan nilai *swing* pada sistem yang pertama untuk memberi arah yang sama pada sistem yang kedua. *Persamaan (2.47)* menunjukkan kejelasan, akan tetapi untuk koordinat *z'* tidak perlu dilakukan perhitungan karena tidak digunakan untuk perhitungan selanjutnya.

Proses yang kelima ialah penentuan nilai *swing* dengan nilai *azimuth* yang berbeda pada garis normal. Menggunakan koordinat yang telah ditransformasikan pada dua titik sebagai penentuan pada langkah keempat. Selanjutnya melakukan proses perhitungan pada nilai *azimuth* yang menghubungkan garis pada tiap sistem. *Swing* diperlukan untuk menyamakan sistem sembarang dengan sistem kontrol yang dapat dihitung dengan menggunakan persamaan berikut :

$$Swing = Azimuth(control) - Azimuth(sembarang) \qquad [2.48]$$

Keenam, menggabungkan dua nilai *tilt*, dua nilai *azimuth* dan satu nilai *swing* kedalam satu matriks rotasi. Matriks rotasi M_l dibentuk menggunakan *tilt* dan *azimuth* pada vektor normal di sistem sembarang, digabungkan dengan nilai *swing* yang ditentukan pada langkah kelima.

Sedangkan untuk matriks M_2, bentuk matriksnya hanya menggunakan nilai *tilt* dan *azimuth* pada vektor normal di sistem kontrol. Persamaan dibawah ini menunjukkan bagaimana titik koordinat pada dua sistem dihubungkan :

$$M_1 \cdot \begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} = M_2 \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \qquad [2.49]$$

Notasi yang terdapat pada persamaan diatas secara teknis tidaklah sama. Ini diartikan bahwa notasi di atas saling berhubungan atau mempunyai fungsi yang sama. Kedua matriks tersebut digabungkan kedalam satu matriks rotasi yang persamaannya ditunjukkan pada persamaan dibawah ini dimana tanda yang sama mempunyai arti yang sama seperti pada *Persamaan (2.49)* (*Dewitt, 1996*):

$$\begin{bmatrix} x_a \\ y_a \\ z_a \end{bmatrix} = M_1^T \cdot M_2 \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = M \cdot \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \qquad [2.50]$$

Ketujuh, dari matriks rotasi *M*, hitung nilai masing-masing untuk *omega*, *phi*, dan *kappa*. Hal ini dapat diselesaikan dengan menggunakan persamaan seperti dibawah ini (*Dewitt, 1996*) :

$$\begin{aligned} omega &= \tan^{-1}(-m_{32}/m_{33}) \\ phi &= \sin^{-1}(m_{31}) \\ kappa &= \tan^{-1}(-m_{21}/m_{11}) \end{aligned} \qquad [2.51]$$

2.2.3 *Proses Perataan Parameter Transformasi*

Ada beraneka pendekatan untuk menyelesaikan parameter transformasi yang belum diketahui. Sebuah metode yang meniadakan tiga faktor translasi dengan jalan substraksi, dan oleh karenanya mengurangi jumlah persamaan yang harus diselesaikan bagi empat persamaan. Metode ini secara bersamaan menyelesaikan tujuh faktor yang tidak diketahui, meskipun cara ini memerlukan upaya perhitungan yang lebih banyak.

Persamaan (2.36) merupakan persamaan nonlinier yang mengandung tujuh faktor yang tidak diketahui yaitu s, ω, φ, κ, T_x, T_y dan T_z. Untuk menyelesaikan persamaan ini, maka *Persamaan (2.36)* terlebih dahulu dilinierkan menggunakan *teorema Taylor*. Misalkan ada titik p koordinatnya diketahui pada kedua sistem. Sehubungan dengan *teorema Taylor*, maka dapat ditulis bentuk linier dari *persamaan (2.36)* adalah :

$$X_P = (X_P)_0 + \left(\frac{\delta X_P}{\delta s}\right)_0 ds + \left(\frac{\delta X_P}{\delta \omega}\right)_0 d\omega + \left(\frac{\delta X_P}{\delta \varphi}\right)_0 d\varphi + \left(\frac{\delta X_P}{\delta \kappa}\right)_0 d\kappa + \left(\frac{\delta X_P}{\delta T_x}\right)_0 dT_x + \left(\frac{\delta X_P}{\delta T_y}\right)_0 dT_y + \left(\frac{\delta X_P}{\delta T_z}\right)_0 dT_z$$

$$Y_P = (Y_P)_0 + \left(\frac{\delta Y_P}{\delta s}\right)_0 ds + \left(\frac{\delta Y_P}{\delta \omega}\right)_0 d\omega + \left(\frac{\delta Y_P}{\delta \varphi}\right)_0 d\varphi + \left(\frac{\delta Y_P}{\delta \kappa}\right)_0 d\kappa + \left(\frac{\delta Y_P}{\delta T_x}\right)_0 dT_x + \left(\frac{\delta Y_P}{\delta T_y}\right)_0 dT_y + \left(\frac{\delta Y_P}{\delta T_z}\right)_0 dT_z \qquad [2.52]$$

$$Y_P = (Y_P)_0 + \left(\frac{\delta Y_P}{\delta s}\right)_0 ds + \left(\frac{\delta Y_P}{\delta \omega}\right)_0 d\omega + \left(\frac{\delta Y_P}{\delta \varphi}\right)_0 d\varphi + \left(\frac{\delta Y_P}{\delta \kappa}\right)_0 d\kappa +$$
$$\left(\frac{\delta Y_P}{\delta T_x}\right)_0 dT_x + \left(\frac{\delta Y_P}{\delta T_y}\right)_0 dT_y + \left(\frac{\delta Y_P}{\delta T_z}\right)_0 dT_z$$

Secara sederhana dengan mengubah huruf-huruf, pernyataan sejenis *Persamaan (2.52)* dibuat untuk titik *q* dan *r*, sehingga membuahkan jumlah total sembilan persamaan. Didalam *Persamaan (2.52)*, $(X_p)_0$, $(Y_p)_0$ dan $(Z_p)_0$ merupakan sisi sebelah kanan segitiga persamaan pertama *persamaan (2.43)* yang dievaluasi pada awal perkiraan, $(\delta X_P/\delta s)_0$, $(\delta X_P/\delta s)_0$ dan sebagainya merupakan ubahan parsial sehubungan dengan faktor tak diketahui yang telah ditunjukkan yang dievaluasi pada awal perkiraan. Satuan $d\omega$, $d\varphi$ dan $d\kappa$ adalah radian *(Wolf, 1993)*.

Dengan subtitusi huruf bagi koefisien ubahan, sembilan persamaan linier dalam bentuk *Persamaan (2.52)* semuanya diberikan berikut :

$$\begin{aligned}
&a_{11} + a_{12}d\omega + a_{13}d\varphi + a_{14}d\kappa + a_{15}dT_x + a_{16}dT_y \\
&+ a_{17}dT_z = [X_P - (X_P)_0] + vX_P \\
&a_{21} + a_{22}d\omega + a_{23}d\varphi + a_{24}d\kappa + a_{25}dT_x + a_{26}dT_y \\
&+ a_{27}dT_z = [Y_P - (Y_P)_0] + vY_P \\
&a_{31} + a_{32}d\omega + a_{33}d\varphi + a_{34}d\kappa + a_{35}dT_x + a_{36}dT_y \\
&+ a_{37}dT_z = [Z_P - (Z_P)_0] + vZ_P
\end{aligned} \qquad [2.53]$$

Persamaan (2.50) dapat dinyatakan dalam bentuk matriks sebagai berikut :

$$AX = L + V \qquad [2.54]$$

Dimana :

$$A = \begin{vmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} & a_{16} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} & a_{26} \\ a_{31} & a_{32} & a_{34} & a_{35} & a_{36} & a_{37} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{11n} & a_{12n} & a_{13n} & a_{14n} & a_{15n} & a_{16n} \\ a_{21n} & a_{22n} & a_{23n} & a_{24n} & a_{25n} & a_{26n} \\ a_{31n} & a_{32n} & a_{33n} & a_{34n} & a_{35n} & a_{36n} \end{vmatrix}_{m}^{n} \qquad X = \begin{vmatrix} s \\ d\omega \\ d\varphi \\ d\kappa \\ dT_x \\ dT_y \\ dT_z \end{vmatrix}_{7}^{1}$$

$$L = \begin{bmatrix} X_P - (X_P)_0 \\ Y_P - (Y_P)_0 \\ Z_P - (Z_P)_0 \end{bmatrix}_{m}^{1} \qquad V = \begin{bmatrix} Vx_p \\ Vy_p \\ Vz_p \end{bmatrix}$$

Persamaan (2.54) dapat diselesaikan dengan menggunakan persamaan kuadrat terkecil. Penyelesainnya diulangi sesuai dengan penggunaan seri *Taylor*, hingga diperoleh nilai kecil yang dapat diabaikan untuk koreksi bagi perkiraan awal untuk parameter transformasi yang tidak diketahui. Setelah diperoleh faktor transformasi, koordinat terubah bagi semua titik yang koordinatnya diketahui hanya pada sistem yang asli, dapat diperoleh dengan menggunakan persamaan sejenis *Persamaan (2.36)*.

Untuk memperjelas koefisien *Persamaan (2.52)*, ubahan parsial bagi tiga persamaan pertama adalah :

$$
\begin{aligned}
a_{11} &= m_{11}(x_p) + m_{21}(y_p) + m_{31}(z_p) \\
a_{12} &= 0 \\
a_{13} &= [(-\sin\varphi\cos\kappa)(x_p) + \sin\varphi\sin\kappa(y_p) + \cos\varphi(z_p)]s \\
a_{14} &= [m_{21}(x_p) - m_{11}(y_p)]s \\
a_{15} &= a_{26} = a_{37} = 1 \\
a_{16} &= a_{17} = a_{25} = a_{27} = a_{35} = a_{36} = 0
\end{aligned}
\qquad [2.55]
$$

$$
\begin{aligned}
a_{21} &= m_{12}(x_p) + m_{22}(y_p) + m_{32}(z_p) \\
a_{22} &= [-m_{13}(x_p) - m_{23}(y_p) - m_{33}(z_p)]s \\
a_{23} &= [(\sin\omega\cos\varphi\cos\kappa)(x_p) + (-\sin\omega\cos\varphi\sin\kappa)(y_p) + (\sin\omega\sin\varphi)(z_p)]s \\
a_{24} &= [m_{22}(x_p) - m_{12}(y_p)]s \\
a_{31} &= m_{13}(x_p) + m_{23}(y_p) + m_{33}(z_p) \\
a_{32} &= [m_{12}(x_p) + m_{22}(y_p) + m_{32}(z_p)]s \\
a_{33} &= [(-\cos\omega\cos\varphi\cos\kappa)(x_p) + (\cos\omega\cos\varphi\sin\kappa)(y_p) + (-\cos\omega\sin\varphi)(z_p)]s \\
a_{34} &= [m_{23}(x_p) - m_{13}(y_p)]s
\end{aligned}
$$

Koefisien bagi enam persamaan lainnya dari *Persamaan (2.53)* persis sama dengan tiga persamaan pertama, dengan pengecualian bahwan huruf *p* diganti dengan huruf *q* dan *r*. Pada umumnya, apabila ada *n* titik yang koordinatnya *X, Y* dan *Z*-nya diketahui pda kedua sistem, maka dapat disusun 3(*n*) persamaan sejenis *Persamaan (2.53)*.

2.2.4 Penentuan Koordinat Objek 3 Dimensi berdasarkan Parameter Transformasi

Dalam proses penentuan parameter eksterior orientasi digunakan metode absolute orientasi untuk mendapatkan nilai koordinat objek tiga dimensi, dalam hal ini dibutuhkan nilai tujuh parameter transformasi. Ketujuh parameter ini

digunakan untuk menghitung nilai parameter posisi kamera dalam sistem kontrol. Adapun persamaan yang dapat digunakan ialah *Persamaan (2.36)* dan akan ditulis kembali untuk nilai koordinat kamera sebagai berikut :

$$\begin{aligned} X_{cam\_cont} &= s(m_{11}X_{cam\_arb} + m_{21}Y_{cam\_arb} + m_{31}Z_{cam\_arb}) + T_x \\ Y_{cam\_cont} &= s(m_{12}X_{cam\_arb} + m_{22}Y_{cam\_arb} + m_{32}Z_{cam\_arb}) + T_y \\ Z_{cam\_cont} &= s(m_{13}X_{cam\_arb} + m_{23}Y_{cam\_arb} + m_{33}Z_{cam\_arb}) + T_z \end{aligned} \qquad [2.56]$$

Selanjutnya, tujuh parameter dapat digunakan juga untuk menghitung parameter rotasi kamera (ω, φ, κ), ditentukan dengan beberapa tahapan antara lain, yang pertama menghitung jarak dari *perspective center* ke titik koordinat foto menggunakan rumus jarak sebagai berikut :

$$d_i = \sqrt{x_i^2 + y_i^2 + z_i^2} \qquad [2.57]$$

Dari persamaan diatas i merupakan notasi titik ke *1-3*, x dan y merupakan koordinat foto dan nilai z merupakan nilai $-f$. Dari tiap jarak tersebut dapat dihitung tiga nilai parameter cosinus jarak yang dinotasikan sebagai l, m, n. ketiga parameter ini dapat di hitung dengan rumus sebagai berikut :

$$l_i = \frac{x_i}{d_i}; \quad m_i = \frac{y_i}{d_i}; \quad n_i = \frac{z_i}{d_i} \qquad [2.58]$$

Setelah mengetahui arah kosinus jarak pada foto, selanjutnya menghitung kosinus jarak dari sistem ke titik kontrol. Adapun persamaan yang digunakan yaitu :

$$D_i = \sqrt{(X_i - X_L)^2 + (Y_i - Y_L)^2 + (Z_i - Z_L)^2} \qquad [2.59]$$

Dari nilai jarak diatas dapat dihitung parameter kosinus jarak untuk titik kontrol. Tiga parameter untuk nilai kosinus jarak ini, dapat dinotasikan dengan *L, M, N* dengan persamaan dibawah ini :

$$L_i = \frac{X_i - X_L}{D_i}; \quad M_i = \frac{Y_i - Y_L}{D_i}; \quad N_i = \frac{Z_i - Z_L}{D_i} \qquad [2.60]$$

Jika kedua parameter kosinus jarak pada sistem foto dan kontrol telah diketahui, maka nilai bantu yang akan digunakan untuk menghitung parameter rotasi dapat dihitung dengan rumus sebagai berikut :

$$\begin{aligned} e_i &= l_i m_j - l_j m_i \\ f_i &= l_i n_j - l_j n_i \\ g_i &= m_i n_j - m_j n_i \end{aligned} \qquad [2.61]$$

Dimana nilai $i = 1,2,3$ dan $j = 2,3,1$.

Dari nilai *l, m, n, L, M, N, e,f,g* dapat dihitung nilai parameter matriks rotasi sebagai berikut :

$$m_{11} = \frac{L_1 g_2 + L_2 g_3 + L_3 g_1}{\Delta}$$

$$m_{21} = \frac{L_1 f_2 + L_2 f_3 + L_3 f_1}{-\Delta}$$

$$m_{31} = \frac{L_1 e_2 + L_2 e_3 + L_3 e_1}{\Delta}$$

$$m_{12} = \frac{M_1 g_2 + M_2 g_3 + M_3 g_1}{\Delta}$$

$$m_{22} = \frac{M_1 f_2 + M_2 f_3 + L_3 f_1}{-\Delta}$$

$$m_{32} = \frac{M_1 e_2 + M_2 e_3 + M_3 e_1}{\Delta}$$

$$m_{12} = \frac{N_1 g_2 + N_2 g_3 + N_3 g_1}{\Delta}$$

$$m_{22} = \frac{N_1 f_2 + N_2 f_3 + N_3 f_1}{-\Delta}$$

$$m_{32} = \frac{N_1 e_2 + N_2 e_3 + N_3 e_1}{\Delta}$$

Dimana nilai Δ didapat menggunakan persamaan sebagai berikut :

$$\Delta = e_1 n_3 + e_2 n_1 + e_3 n_2 \qquad [2.62]$$

Maka parameter sudut rotasi dapat dihitung dengan menggunakan *persamaan (2.51)*. Selanjutnya semua parameter tersebut dapat dirata-ratakan menggunakan *least square* seperti yang dijelaskan pada sub bab sebelumnya.

BAB III

METODOLOGI PENELITIAN

Dalam pelaksanaan sebuah penelitian, ada beberapa hal-hal penting yang harus diperhatikan yaitu persiapan penelitian, materi penelitian, alat dan bahan penelitian serta diagram alir dan penjelasannya. Ini dilakukan agar pada saat pelaksanaan penelitian dapat berjalan dengan efektif.

Pada bab ini, akan menjelaskan secara lengkap hal-hal yang dibutuhkan untuk penelitian serta tahapan-tahapan kerja yang akan dilaksanakan dalam penelitian. Adapun pembahasan yang lengkapnya akan diuraikan secara jelas pada sub bab selanjutnya.

3.1 Persiapan Penelitian

Sebelum melakukan penelitian, dilakukan persiapan terlebih dahulu agar pada proses pelaksanaannya dapat berjalan dengan lancar hingga pada proses pembuatan algorithma. Persiapan-persiapan yang harus dilakukan akan dijelaskan pada sub bab dibawah ini.

3.1.1 Materi Penelitian

Materi penelitian, sebelumnya telah dijelaskan pada *bab II* yaitu dasar teori. Penelitian ini dilakukan mengacu pada dasar teori yang telah dijabarkan

secara lengkap dan jelas tentang menghitung parameter orientasi kamera menggunakan teknik *resection* dan absolute orientasi.

3.1.2 Alat penelitian

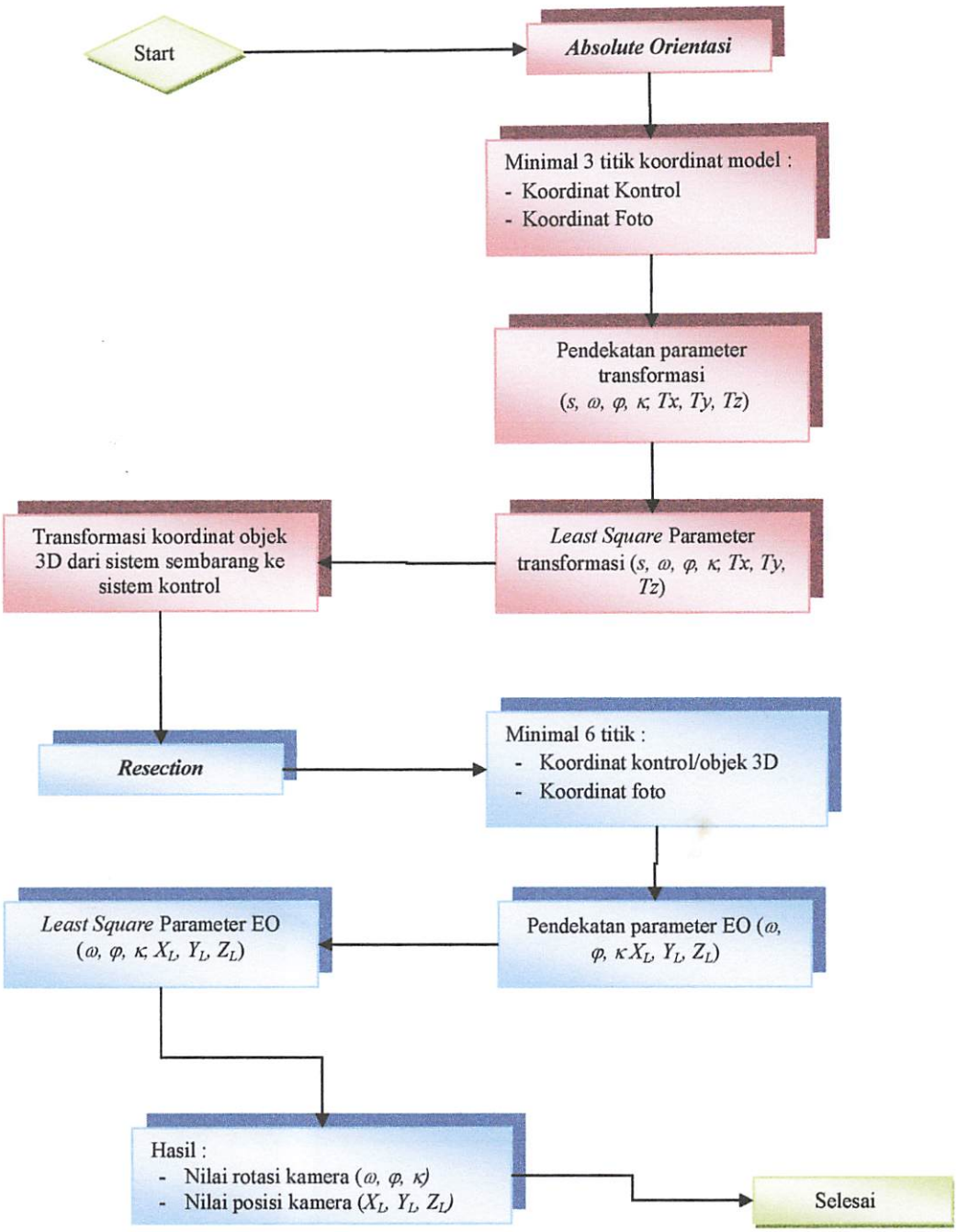
Proses penelitian ini menggunakan peralatan pendukung untuk melakukan perhitungan parameter eksterior orientasi pada *Space Resection* dan *Absolute Orientation*. Peralatan yang digunakan berupa perangkat lunak (*software*), dalam hal ini perangkat lunak yang digunakan adalah Microsoft Excel 2003 dan Matlab R2008b serta Microsoft Visual Studio 2010.

3.1.3 Bahan Penelitian

Selain memerlukan beberapa peralatan juga dibutuhkan bahan-bahan yang akan di gunakan untuk melakukan perhitungan terhadap kedua metode tersebut. Adapun bahan-bahan tersebut adalah sebagai berikut :

- *Space Resection*
  - Koordinat foto 2D (x, y)
  - Koordinat objek 3D (X, Y, Z)
  - Nilai pendekatan parameter Eksterior Orientasi
  - Nilai parameter Interior Orientasi
- *Absolute Orientation*
  - Koordinat objek 3D dari 2 sistem (Sistem Kontrol dan Sistem Sembarang)

3.2 Diagram Alir Penelitian



3.3 Penjelasan Diagram Alir

Diagram alir pada sub bab sebelumnya merupakan tahapan-tahapan kerja yang akan dilaksanakan dalam penelitian ini. Diagram ini akan dijelaskan secara lengkap dan jelas pada subbab selanjutnya.

3.3.1 *Metode Resection*

1. Mencari nilai pendekatan parameter eksterior orientasi dengan metode *DLT* (*Direct Linear Transformation*). Langkah-langkah pengerjaannya adalah sebagai berikut :

   a. Proses linearisasi untuk rumus dasar *DLT* yang terdapat pada *Persamaan (2.3)* dan dituliskan kembali sebagai berikut :

$$x + \delta x = \frac{L_1 X_P + L_2 Y_P + L_3 Z_P + L_4}{L_9 X_P + L_{10} Y_P + L_{11} Z_P + 1}$$

$$y + \delta y = \frac{L_5 X_P + L_6 Y_P + L_7 Z_P + L_8}{L_9 X_P + L_{10} Y_P + L_{11} Z_P + 1}$$

   Maka persamaan yang sudah terlinearisasi adalah :

$$x_p = L_1 X_P + L_2 Y_P + L_3 Z_P + L_4 - L_9 X_P x + L_{10} Y_P x + L_{11} Z_P x$$

$$y_p = L_5 X_P + L_6 Y_P + L_7 Z_P + L_8 - L_9 X_P y + L_{10} Y_P y + L_{11} Z_P y$$

   b. Sebelum melakukan proses *Least Square* untuk persamaan yang sudah terlinearisas terlebih dahulu menyusun matrik A atau matrik koefisien.

$$A = \begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -X_1x & -Y_1x & -Z_1x \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1y & -Y_1y & -Z_1y \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & X_n & 1 & 0 & 0 & 0 & 0 & -X_nx & -Y_nx & -Z_nx \\ 0 & 0 & 0 & 0 & X_n & Y_n & X_n & 1 & -X_1y & -Y_1y & -Z_1y \end{bmatrix}$$

c. Selanjutnya membentuk matrik L atau matrik observasi dan dari persamaan terlinearisasi.

$$L = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ \vdots \\ \vdots \\ x_n \\ y_n \end{bmatrix}$$

d. Setelah matrik A dan L telah selesai dibangun, maka lakukan proses *least square* untuk kedua matrik tersebut.

$$X = \left[A^T A\right]^{-1}\left[A^T L\right]$$

Dimana nilai x mengandung parameter

$$X^T = \left[L_1 \quad L_2 \quad L_3 \quad L_4 \quad L_5 \quad L_6 \quad L_7 \quad L_8 \quad L_9 \quad L_{10} \quad L_{11}\right]$$

e. Jika telah mendapatkan nilai matrik L, maka untuk nilai parameter posisi kamera dapat dihitung dengan persamaan sebagai berikut :

$$\begin{bmatrix} X_C \\ Y_C \\ Z_C \end{bmatrix} = -\begin{bmatrix} L_1 & L_2 & L_3 \\ L_5 & L_6 & L_7 \\ L_9 & L_{10} & L_{11} \end{bmatrix}^{-1} \begin{bmatrix} L_4 \\ L_8 \\ 1 \end{bmatrix}$$

f. Untuk langkah berikutnya, menghitung parameter sudut rotasi kamera. Sebelum menghitung parameter rotasi dilakukan terlebih dahulu perhitungan terhadap nilai konstanta.

$$L^2 = L_9^2 + L_{10}^2 + L_{11}^2$$
$$X_0 = L_1 L_9 + L_2 L_{10} + L_3 L_{11}$$
$$Y_0 = L_5 L_9 + L_6 L_{10} + L_7 L_{11}$$
$$C_x = \frac{L_1^2 + L_2^2 + L_3^2}{L^2} - X_0^2$$
$$C_y = \frac{L_5^2 + L_6^2 + L_7^2}{L^2} - Y_0^2$$

g. Setelah menghitung nilai konstanta, maka matrik rotasi dapat dihitung sehingga dapat menentukan parameter rotasi (ω, φ, κ).

$$m_{31} = \frac{L_9}{L}$$

$$m_{32} = \frac{L_{10}}{L}$$

$$m_{33} = \frac{L_{11}}{L}$$

$$m_{11} = \frac{x_0 m_{31} - \frac{L_1}{L}}{c_x} \qquad m_{21} = \frac{y_0 m_{31} - \frac{L_5}{L}}{c_y}$$

$$m_{12} = \frac{x_0 m_{32} - \frac{L_2}{L}}{c_x} \qquad m_{22} = \frac{y_0 m_{32} - \frac{L_6}{L}}{c_y}$$

$$m_{13} = \frac{x_0 m_{33} - \frac{L_3}{L}}{c_x} \qquad m_{23} = \frac{y_0 m_{33} - \frac{L_7}{L}}{c_y}$$

Maka, untuk parameter rotasi, digunakan rumus :

$$\tan \omega = \frac{-m_{32}}{m_{33}}$$

$$\sin \varphi = m_{31}$$

$$\tan \kappa = \frac{-m_{21}}{m_{11}}$$

2. Perataan parameter eksterior orientasi menggunakan teknik *Least Square*. Tahapan pengerjaannya seperti dibawah ini :

   a. Mencari nilai matrik rotasi atau matrik *M* dari nilai parameter yang telah diketahui.

$$M = \begin{bmatrix} \cos\varphi\cos\kappa & \sin\omega\sin\varphi\cos\kappa + \cos\omega\sin\kappa & -\cos\omega\sin\varphi\cos\kappa + \sin\omega\sin\kappa \\ -\cos\varphi\sin\kappa & -\sin\omega\sin\varphi\sin\kappa + \cos\omega\cos\kappa & \cos\omega\sin\varphi\sin\kappa + \sin\omega\cos\kappa \\ \sin\varphi & -\sin\omega\cos\varphi & \cos\omega\cos\varphi \end{bmatrix}$$

b. Setelah menghitung matrik *M*, maka lanjutkan perhitungan untuk nilai r, s, q.

$$r = m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L)$$
$$s = m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L)$$
$$q = m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + r_{33}(Z_A - Z_L)$$

c. Jika semua nilai telah dihitung, matrik *A* dapat disusun sebagai berikut :

$$A = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ m \end{array} \begin{vmatrix} a_{11a} & a_{12a} & a_{13a} & -a_{14a} & -a_{15a} & -a_{16a} \\ a_{21a} & a_{22a} & a_{23a} & -a_{24a} & -a_{25a} & -a_{26a} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{11n} & a_{12n} & a_{13n} & -a_{14n} & -a_{15n} & -a_{16n} \\ a_{21n} & a_{22n} & a_{23n} & -a_{24n} & -a_{25n} & -a_{26n} \end{vmatrix}^{n}$$

Dimana untuk nilai a_{11}, a_{12}, a_{13}, ...

$$b_{11} = \frac{f}{q^2}[r(-m_{33}\Delta Y + m_{32}\Delta Z) - q(-m_{13}\Delta Y + m_{12}\Delta Z)]$$

$$b_{12} = \frac{f}{q^2}[r(\cos\varphi\,\Delta X + \sin\omega\sin\varphi\,\Delta Y - \cos\omega\sin\varphi\,\Delta Z)$$

$$-q(-\sin\varphi\cos\kappa\,\Delta X + \sin\omega\cos\varphi\cos\kappa\,\Delta Y - \cos\omega\cos\varphi\cos\kappa\,\Delta Z)]$$

$$b_{13} = \frac{-f}{q}(m_{21}\Delta X + m_{22}\Delta Y + m_{23}\Delta Z)$$

$$b_{14} = \frac{f}{q^2}(rm_{31} - qm_{11})$$

$$b_{15} = \frac{f}{q^2}(rm_{32} - qm_{12})$$

$$b_{16} = \frac{f}{q^2}(rm_{33} - qm_{13})$$

$$b_{21} = \frac{f}{q^2}[s(-m_{33}\Delta Y + m_{32}\Delta Z) - q(-m_{23}\Delta Y + m_{22}\Delta Z)]$$

$$b_{22} = \frac{f}{q^2}[s(\cos\varphi\,\Delta X + \sin\omega\sin\varphi\,\Delta Y - \cos\omega\sin\varphi\,\Delta Z)$$

$$-q(\sin\varphi\cos\kappa\,\Delta X - \sin\omega\cos\varphi\sin\kappa\,\Delta Y + \cos\omega\cos\varphi\sin\kappa\,\Delta Z)]$$

$$b_{23} = \frac{f}{q}(m_{11}\Delta X + m_{12}\Delta Y + m_{13}\Delta Z)$$

$$b_{24} = \frac{f}{q^2}(sm_{31} - qm_{21})$$

$$b_{25} = \frac{f}{q^2}(sm_{32} - qm_{22})$$

$$b_{26} = \frac{f}{q^2}(sm_{33} - qm_{23})$$

d. Lalu bentuk matrik L sebagai matrik observasi atau pengamatan menggunakan rumus dibawah ini.

$$J = xa - xo + f\frac{r}{q}$$

$$K = ya - yo + f\frac{s}{q}$$

Dimana matrik L dapat disusun sebagai berikut :

$$L = \begin{bmatrix} J_1 \\ K_1 \\ J_2 \\ K_2 \\ \vdots \\ \vdots \\ J_n \\ K_n \end{bmatrix}$$

e. Jika matrik A dan L telah terbentuk, maka kedua matrik tersebut dapat diselesaikan dengan teknik *least square* untuk mendapatkan nilai koreksi dari masing-masing parameter yaitu matrik X.

$$X = \left[A^T A\right]^{-1}\left[A^T L\right]$$

$$X = \begin{bmatrix} d\omega \\ d\varphi \\ d\kappa \\ dX_L \\ dY_L \\ dZ_L \end{bmatrix}$$

f. Selanjutnya mencari nilai matrik residu atau matrik V dan nilai standar deviasi dari parameter eksterior orientasi.

$$V = AX - L$$

$$S_0 = \sqrt{\frac{V^T V}{r}}$$

3.3.1 *Metode Absolut Orientation*

1. Untuk langkah pertama, lakukan perhitungan terhadap nilai pendekatan parameter transformasi. Tahap-tahap yang dilakukan untuk menghitung nilai pendekatan masing-masing parameter transformasi menggunakan rumus sebagai berikut : :

   a. Faktor skala :

   $$skala = \frac{jarak\ pada\ sistem\ kontrol}{jarak\ pada\ sistem\ sembarang}$$

   b. Faktor Rotasi :

   1) Menghitung nilai koefisien *a, b, c* dari vetor normal untuk system kontrol dan sembarang.

   $$a = (y_2 - y_1)(z_3 - z_1) - (y_3 - y_1)(z_2 - z_1)$$
   $$b = (x_3 - x_1)(z_2 - z_1) - (x_2 - x_1)(z_3 - z_1)$$
   $$c = (x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)$$

   2) Menentukan nilai geometri terbaik dari tiga titik sekutu pada salah satu sistem.

   $$h^2 = b^2 - \left(\frac{a^2 + b^2 - c^2}{2a}\right)$$

   3) Mencari nilai sudut rotasi dalam sistem rotasi *tilt* dan *azimuth* untuk kedua sistem.

$$tilt = \tan^{-1}\left(\frac{c}{\sqrt{a^2+b^2}}\right) + 90°$$

$$azimuth = \tan^{-1}\left(\frac{a}{b}\right)$$

4) Menghitung nilai matrik rotasi dari *tilt* dan *azimuth* dengan nilai *swing* 0 untuk kedua sistem koordinat ($M_{1control}$, $M_{2sembarang}$).

$$m_{11} = -\cos(\alpha)\cos(s) - \sin(\alpha)\cos(t)\sin(s)$$
$$m_{12} = \sin(\alpha)\cos(s) - \cos(\alpha)\cos(t)\sin(s)$$
$$m_{13} = -\sin(t)\sin(s)$$
$$m_{21} = \cos(\alpha)\sin(s) - \sin(\alpha)\cos(t)\cos(s)$$
$$m_{22} = -\sin(\alpha)\sin(s) - \cos(\alpha)\cos(t)\cos(s)$$
$$m_{23} = -\sin(t)\cos s$$
$$m_{31} = -\sin(\alpha)\sin(t)$$
$$m_{32} = -\cos(\alpha)\sin(t)$$
$$m_{33} = \cos(t)$$

5) Lakukan transformasi untuk dua titik yang sama pada sistem kontrol dan sembarang menggunakan masing-masing nilai matrik rotasi (t, s, α) dengan persamaan sebagai berikut :

$$x' = m_{11}x + m_{21}y + m_{31}$$

$$y' = m_{12}x + m_{22}y + m_{32}z$$

6) Setelah mendapatkan nilai koordinat dari titik yang telah ditransformasikan diatas, dapat dihitung nilai dari *azimuth* untuk kedua sistem dengan rumus sebagai berikut :

$$\alpha_{control} = \tan^{-1}\left(\frac{x'_{2control} - x'_{1control}}{y'_{2control} - y'_{1control}}\right)$$

$$\alpha_{sembarang} = \tan^{-1}\left(\frac{x'_{2sembarang} - x'_{1sembarang}}{y'_{2sembarang} - y'_{1sembarang}}\right)$$

7) Jika nilai *azimuth* dari kedua system telah diketahui maka dapat ditentukan nilai *swing* dengan menggunakan persamaan sebagai berikut :

$$Swing = Azimuth(control) - Azimuth(sembarang)$$

8) Selanjutnya, lakukan kembali langkah (4) untuk menghitung nilai matrik rotasi dengan nilai *swing* yang telah diketahui pada sistem sembarang ($M_{2sembarang}$).

9) Untuk mendapatkan nilai matrik rotasi yang sebenarnya dapat digunakan perkalian antar matrik pada rotasi yang dihasilkan pada Point 8 ($M_{2sembarang}$) dengan matrik rotasi pada Point 4 ($M_{1control}$) atau dapat di tulis dalam hubungan matemetika sebagai berikut :

$$M = M^T_{2sembarang} . M_{1control}$$

10) Selanjutnya nilai *omega, phi* dan *kappa* dapat ditentukan dengan menggunakan matrik M pada Point (9) dengan persamaan sebagai berikut :

$$omega = \tan^{-1}(-m_{32}/m_{33})$$
$$phi = \sin^{-1}(m_{31})$$
$$kappa = \tan^{-1}(-m_{21}/m_{11})$$

c. Faktor Translasi :

Faktor translasi dapat dihitung menggunakan rumus :

$$X_{cont} = s(m_{11}X_{arb} + m_{21}Y_{arb} + m_{31}Z_{arb}) + T_x$$
$$Y_{cont} = s(m_{12}X_{arb} + m_{22}Y_{arb} + m_{32}Z_{arb}) + T_y$$
$$Z_{cont} = s(m_{13}X_{arb} + m_{23}Y_{arb} + m_{33}Z_{arb}) + T_z$$

Dimana :

$X_{control}$, $Y_{control}$, $Z_{control}$: *Koordinat titik pada sistem kontrol*

X_{arb}, Y_{arb}, Z_{arb} : *Koordinat titik pada sistem sembarang*

s : *faktor skala*

m_{11}, m_{12}, ... m_{32}, m_{33} : *elemen matrik rotasi*

2. Apabila nilai pendekatan telah diketahui, maka lakukan perataan terhadap nilai ketujuh parameter dengan menggunakan rumus *least square*.

a. Mencari matrik rotasi atau matrik M dari nilai parameter pendekatan.

$$M = \begin{bmatrix} \cos\varphi\cos\kappa & \sin\omega\sin\varphi\cos\kappa + \cos\omega\sin\kappa & -\cos\omega\sin\varphi\cos\kappa + \sin\omega\sin\kappa \\ -\cos\varphi\sin\kappa & -\sin\omega\sin\varphi\sin\kappa + \cos\omega\cos\kappa & \cos\omega\sin\varphi\sin\kappa + \sin\omega\cos\kappa \\ \sin\varphi & -\sin\omega\cos\varphi & \cos\omega\cos\varphi \end{bmatrix}$$

b. Menyusun matrik A dengan bentuk matrik seperti dibawah ini.

$$A = \begin{bmatrix} a_{11a} & a_{12a} & a_{13a} & a_{14a} & a_{15a} & a_{16a} & a_{17a} \\ a_{21a} & a_{22a} & a_{23a} & a_{24a} & a_{25a} & a_{26a} & a_{27a} \\ a_{31a} & a_{32a} & a_{33a} & a_{34a} & a_{35a} & a_{36a} & a_{37a} \\ a_{11b} & a_{12b} & a_{13b} & a_{14b} & a_{15b} & a_{16b} & a_{17b} \\ a_{21b} & a_{22b} & a_{23b} & a_{24b} & a_{25b} & a_{26b} & a_{27b} \\ a_{31b} & a_{32b} & a_{33b} & a_{34b} & a_{35b} & a_{36b} & a_{37b} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{11n} & a_{12n} & a_{13n} & a_{14n} & a_{15n} & a_{16n} & a_{17n} \\ a_{21n} & a_{22n} & a_{23n} & a_{24n} & a_{25n} & a_{26n} & a_{27n} \\ a_{31n} & a_{32n} & a_{33n} & a_{34n} & a_{35n} & a_{36n} & a_{37n} \end{bmatrix}$$

Dimana untuk nilai koefisien a_{11}, a_{12}, a_{13},.....

$$a_{11} = m_{11}x_P + m_{12}y_p + m_{13}z_P$$
$$a_{12} = 0$$
$$a_{13} = \left[(-\sin\varphi\cos\kappa)x_P + (\sin\varphi\sin\kappa)y_P + (\cos\varphi)z_P\right]s$$
$$a_{14} = (m_{21}x_P - m_{11}y_p)s$$
$$a_{15} = a_{26} = a_{27} = 1$$
$$a_{16} = a_{17} = a_{25} = a_{27} = a_{35} = a_{36} = 0$$
$$a_{21} = m_{12}x_P + m_{22}y_p + m_{32}z_P$$
$$a_{22} = (-m_{13}x_P - m_{23}y_p - m_{33}z_P)s$$
$$a_{23} = \left[(\sin\omega\cos\varphi\cos\kappa)x_P + (-\sin\omega\cos\varphi\sin\kappa)y_P + (\sin\omega\sin\varphi)z_P\right]s$$
$$a_{24} = (m_{22}x_P - m_{12}y_p)s$$
$$a_{31} = m_{13}x_P + m_{23}y_p + m_{33}z_P$$
$$a_{32} = (m_{12}x_P + m_{22}y_P + m_{32}z_p)s$$
$$a_{33} = \left[(-\cos\omega\cos\varphi\cos\kappa)x_P + (\cos\omega\cos\varphi\sin\kappa)y_P + (-\cos\omega\sin\varphi)z_P\right]s$$
$$a_{34} = (m_{23}x_P - m_{13}y_p)s$$

d. Menbentuk matrik L sebagai matrik pengamatan atau matrik observasi dengan rumus sebagai berikut :

$$L = \begin{bmatrix} X_1 - (X_1)_0 \\ Y_1 - (Y_1)_0 \\ Z_1 - (Z_1)_0 \\ X_2 - (X_2)_0 \\ Y_2 - (Y_2)_0 \\ Z_2 - (Z_2)_0 \\ \vdots \\ X_n - (X_n)_0 \\ X_n - (X_n)_0 \\ X_n - (X_n)_0 \end{bmatrix}$$

e. Apabila matrik A dan Matrik L telah diketahui, maka perhitungan *least square* dapat dilakukan untuk mencari nilai koreksi dari tiap parameter yaitu matrik X.

$$X = \left[A^T A\right]^{-1}\left[A^T L\right]$$

$$X = \begin{bmatrix} ds \\ d\omega \\ d\varphi \\ d\kappa \\ dT_X \\ dT_Y \\ dT_Z \end{bmatrix}$$

1. Jika telah mendapatkan nilai parameter transformasi, maka lakukan proses tranformasi koordinat objek tiga dimensi dari sistem sembarang ke sistem kontrol dengan rumus sebagai berikut :

$$X_{cont} = s(m_{11}X_{arb} + m_{21}Y_{arb} + m_{31}Z_{arb}) + T_x$$
$$Y_{cont} = s(m_{12}X_{arb} + m_{22}Y_{arb} + m_{32}Z_{arb}) + T_y$$
$$Z_{cont} = s(m_{13}X_{arb} + m_{23}Y_{arb} + m_{33}Z_{arb}) + T_z$$

BAB 1V

HASIL DAN PEMBAHASAN

4.1 Hasil Absolut Orientasi

Proses absolut orientasi dihitung menggunakan data *input* dari pasangan foto stereo hasil pengukuran dilapangan. Pada penelitian ini, dilakukan perhitungan terhadap empat pasang foto stereo. Setiap pasangan foto memiliki sistem yang berbeda-beda, oleh karena itu teknik ini digunakan untuk melakukan penggabungan semua pasangan foto menjadi satu sistem yang sama. Adapun proses penggabungannya menggunakan tujuh parameter transformasi yaitu s, ω, φ, κ, Tx, Ty, Tz. Pada sub bab selanjutnya akan di tampilkan hasil perhitungan terhadap ketujuh parameter transformasi hingga pada nilai akurasi untuk tiap parameter.

4.1.1 Nilai Pendekatan Awal Parameter Absolut Orientasi

Nilai pendekatan absolut orientasi dihitung menggunakan data awal dari koordinat sistem pada tiap pasangan foto stereo. Masing-masing pasangan foto akan diolah untuk menghasilkan tujuh parameter transformasi. Nilai ketujuh parameter ini dapat dilihat pada *Tabel 4.1* dibawah ini :

Tabel 4.1 Nilai Tujuh Parameter Pendekatan

No	Transformasi Pasangan Stereo	Nilai Pendekatan Awal Parameter Transformasi						
		S	ω (°)	φ (°)	κ (°)	Tx (m)	Ty (m)	Tz (m)
1	2-1	1.03157	77.6327	-1.8976	-2.6041	-4.144	188.205	-18.929
2	3-2	1.03438	79.3300	-8.5050	-0.0050	13.280	206.624	-23.323
3	4-3	1.00033	79.0193	-5.6713	0.2646	11.387	203.586	-19.321
4	5-4	1.0233	78.7879	-7.504	-0.2247	12.301	213.308	-22.142

4.1.2 Proses Perataan Nilai Pendekatan

Setelah melakukan perhitungan terhadap nilai pendekatan untuk ketujuh parameter transformasi, maka lakukan proses perataan untuk setiap nilai parameter transformasi. Proses perataan ini dilakukan dengan tujuan untuk mendapatkan nilai parameter tranformasi yang terkoreksi. Adapun hasil untuk parameter yang terkoreksi disajikan dalam tabel dibawah ini :

Tabel 4.2 Nilai Hasil Perataan Parameter Transformasi

Parameter	Transformasi Pasangan Stereo							
	2-1		3-2		4-3		5-4	
	Hasil	Nilai koreksi	Hasil	Nilai koreksi	Hasil	Nilai koreksi	Hasil	Nilai koreksi
S	1.03151	0.00053	1.03432	0.00072	1.00003	0.00100	1.02357	0.00066
ω (°)	77.6100°	0.0325°	79.3489°	0.0451°	79.0322°	0.0642°	78.7903°	0.0418°
φ (°)	-1.9175°	0.0328°	-8.6647°	0.0430°	-5.7200°	0.0617°	-7.5202°	0.0401°
κ(°)	-2.6241°	0.0678°	-0.0216°	0.0919°	0.2613°	0.1291°	-0.2060°	0.0831°
Tx (m)	-4.135	0.161	13.577	0.220	11.5	0.300	12.342	0.199
Ty (m)	188.198	0.096	206.911	0.131	203.664	0.182	213.35	0.120
Tz (m)	-18.925	0.153	-23.301	0.187	-19.274	0.263	-22.208	0.171

Dari tabel diatas, dapat diketahui bahwa ketujuh parameter mempunyai nilai akurasi yang cukup teliti. Ini dapat dilihat dari kecilnya nilai koreksi dari tiap parameter transformasi.

4.1.3 Transformasi Koordinat Objek 3D

Sebagaimana tujuan dari proses absolut orientasi yaitu menentukan posisi koordinat objek tiga dimensi dari sistem yang lama ke sistem yang baru, maka proses perhitungannya dapat dilakukan dengan menggunakan parameter transformasi yang telah diketahui. Dalam hal ini, parameter koordinat tiga dimensi pasangan foto satu merupakan sistem referensi yang digunakan untuk melakukan transformasi terhadap pasangan foto stereo selanjutnya.

Tabel dibawah ini menunjukkan hasil proses transformasi koordinat objek tiga dimensi pada tiap pasang foto terhadap sistem referensi :

Tabel 4.3 Transformasi Koordinat dari Pasangan Stereo 2 ke 1

Id	SISTEM KONTROL (m)			HASIL TRANSFORMASI (m)			STANDAR DEVIASI (m)		
	X	Y	Z	X	Y	Z	STDx	STDy	STDz
1	1000.0000	1000.0000	1000.0000						
2	1001.5592	1000.0000	1000.0000						
3	1002.8010	1000.0904	1000.0104						
4	999.8796	1004.9985	1000.0000	999.8796	1004.9985	1000.0000			
5	1001.3401	1005.0857	1000.0486	1001.3401	1005.0857	1000.0486			
6	1002.9039	1005.0258	1000.0675	1002.9039	1005.0258	1000.0675			
7	999.8207	1010.0611	1000.0981	999.8207	1010.0611	1000.0981			
8	1001.4057	1010.1513	1000.1275	1001.4057	1010.1513	1000.1275			
9	1002.9759	1010.1102	1000.1557	1002.9759	1010.1102	1000.1557			
10				999.8910	1015.1780	1000.1740	0.012	0.012	0.014
11				1001.4350	1015.1760	1000.1880	0.012	0.012	0.013
12				1003.0370	1015.2150	1000.1960	0.012	0.012	0.014

Tabel 4.4 Transformasi Koordinat dari Pasangan Stereo 3 ke 2

ID	SISTEM KONTROL (m)			HASIL TRANSFORMASI (m)			STANDAR DEVIASI (m)		
	X	Y	Z	X	Y	Z	STDx	STDy	STDz
7	999.8207	1010.0611	1000.0981	999.8207	1010.0611	1000.0981			
8	1001.4057	1010.1513	1000.1275	1001.4057	1010.1513	1000.1275			
9	1002.9759	1010.1102	1000.1557	1002.9759	1010.1102	1000.1557			
10	999.8910	1015.1780	1000.1740	999.8910	1015.1780	1000.1740			
11	1001.4350	1015.1760	1000.1880	1001.4350	1015.1760	1000.1880			
12	1003.0370	1015.2150	1000.1960	1003.0370	1015.2150	1000.1960			
13				999.8640	1020.2030	1000.1770	0.029	0.029	0.035
14				1001.4280	1020.1830	1000.2300	0.029	0.029	0.032
15				1003.0450	1020.1510	1000.2270	0.029	0.029	0.034

Tabel 4.5 Transformasi Koordinat dari Pasangan Stereo 4 ke 3

ID	SISTEM KONTROL (m)			HASIL TRANSFORMASI (m)			STANDAR DEVIASI (m)		
	X	Y	Z	X	Y	Z	STDx	STDy	STDz
10	999.8910	1015.1780	1000.1740	999.8640	1020.2030	1000.1770			
11	1001.4350	1015.1760	1000.1880	1001.4280	1020.1830	1000.2300			
12	1003.0370	1015.2150	1000.1960	1003.0450	1020.1510	1000.2270			
13	999.8640	1020.2030	1000.1770	999.8430	1025.2530	1000.2580			
14	1001.4280	1020.1830	1000.2300	1001.4440	1025.2340	1000.3040			
15	1003.0450	1020.1510	1000.2270	1003.0450	1025.1720	1000.3270			
16				999.843	1025.253	1000.258	0.017	0.017	0.020
17				1001.444	1025.234	1000.304	0.017	0.017	0.019
18				1003.045	1025.172	1000.327	0.017	0.017	0.020

Tabel 4.6 Transformasi Koordinat dari Pasangan Stereo 5 ke 4

ID	SISTEM KONTROL (m)			HASIL TRANSFORMASI (m)			STANDAR DEVIASI (m)		
	X	Y	Z	X	Y	Z	STDx	STDy	STDz
13	999.8640	1020.2030	1000.1770	999.8640	1020.2030	1000.1770			
14	1001.4280	1020.1830	1000.2300	1001.4280	1020.1830	1000.2300			
15	1003.0450	1020.1510	1000.2270	1003.0450	1020.1510	1000.2270			
16	999.8430	1025.2530	1000.2580	999.8430	1025.2530	1000.2580			
17	1001.4440	1025.2340	1000.3040	1001.4440	1025.2340	1000.3040			
18	1003.0450	1025.1720	1000.3270	1003.0450	1025.1720	1000.3270			
19				999.8190	1030.3090	1000.2970	0.022	0.022	0.026
20				1001.4230	1030.2230	1000.3540	0.022	0.022	0.024
21				1002.9500	1030.2630	1000.3650	0.022	0.022	0.026

Dari beberapa tabel diatas, dapat dilihat hasil transformasi koordinat tiga dimensi dari 5 pasang foto dengan nilai standar deviasi cukup akurat. Dari keseluruhan titik hasil transformasi memiliki standar deviasi rata-rata sebagai berikut :

Tabel 4.7 Nilai Standar Deviasi Rata-rata

TRANSFORMASI PASANGAN STEREO	STANDAR DEVIASI RATA-RATA (m)		
	STDx	STDy	STDz
2-1	0.012	0.012	0.0137
3-2	0.029	0.029	0.0337
4-3	0.017	0.017	0.0197
5-4	0.022	0.022	0.0253

4.2 Hasil Space Resection

Pada perhitungan parameter eksteror orientasi (EO), data *input* yang diperlukan adalah nilai koordinat objek hasil dari proses absolut orientasi dan koordinat foto. Selain itu, untuk tiap foto harus diketahui nilai *interior orientasi* (x_0, y_0, f) dan nilai awal pendekatan dari parameter eksterior orientasi sebagai data dalam proses perataan. Proses *resection* ini dilakukan untuk setiap satu foto yang akan ditentukan nilai parameter posisi dan orientasinya. Untuk lebih jelasnya, hasil dari nilai parameter ekterior orientasi dapat di sajikan pada sub bab selanjutnya.

4.2.1 Nilai Pendekatan Awal Parameter Eksterior Orientasi

Nilai pendekatan awal parameter eksterior orientasi membutuhkan nilai koordinat objek dan koordinat foto sebagai data awal perhitungan. Metode yang digunakan untuk menentukan nilai ini adalah metode *DLT* (*Direct Linear Transformation*). Adapun hasil perhitungan untuk nilai pendekatan eksterior orientasi dapat dilihat pada *Tabel 4.8* dibawah ini :

Tabel 4.8 Nilai Awal Parameter Pendekatan Eksterior Orientasi

PARAMETER	PASANGAN STEREO 2		PASANGAN STEREO 3		PASANGAN STEREO 4		PASANGAN STEREO 5	
	KIRI	KANAN	KIRI	KANAN	KIRI	KANAN	KIRI	KANAN
ω (°)	13.030	14.020	10.231	2.331	-1.615	3.336	14.494	14.417
φ (°)	-1.457	-0.696	-1.228	-0.945	-0.425	-0.404	-1.676	-0.875
κ (°)	-1.512	-4.773	-8.160	16.675	169.596	-7.151	-0.729	-3.908
X_L (m)	1001.803	1002.160	1003.328	1000.867	1001.102	1002.605	1001.475	1001.797
Y_L (m)	1006.499	1006.171	1016.180	1014.829	1022.600	1019.445	1020.742	1020.594
Z_L (m)	1001.299	1001.325	1001.961	1000.451	999.818	1000.638	1001.352	1001.323

Data diatas akan digunakan sebagai data *input* untuk proses perhitungan perataan terhadap parameter eksterior orientasi sehingga akan didapatkan nilai parameter yang terkoreksi.

4.2.2 Proses Perataan Nilai Eksterior Orientasi

Proses ini dilakukan untuk mendapatkan nilai parameter eksterior orientasi yang terkoreksi dan hasil dari perataan merupakan nilai eksterior orientasi yang sebenarnya. Berikut merupakan hasil proses perataan yang ditampilkan dalam bentuk tabel :

Tabel 4.9 Parameter Eksterior Orientasi dari Pasangan Stereo 2 dan 3

PARAMETER	PASANGAN STEREO 2				PASANGAN STEREO 3			
	KIRI	STD	KANAN	STD	KIRI	STD	KANAN	STD
ω (°)	-2.94809	0.02047	-2.96068	0.02841	-2.93941	0.01921	-2.95018	0.02948
φ (°)	-0.55330	0.00995	-0.59419	0.01249	-0.69768	0.00800	-0.73777	0.01109
κ (°)	-0.14489	0.04679	-0.16258	0.06132	-0.09509	0.03922	-0.11206	0.05691
X_L (m)	99.38117	0.07491	99.20719	0.11548	98.58352	0.07959	98.36305	0.13182
Y_L (m)	102.53852	0.17598	102.59942	0.22566	102.44062	0.13606	102.53044	0.19440
Z_L (m)	92.19662	0.29339	92.07002	0.37171	91.91515	0.22828	91.74846	0.31733

Tabel 4.10 Parameter Eksterior Orientasi dari Pasangan Stereo 4 dan 5

PARAMETER	PASANGAN STEREO 4				PASANGAN STEREO 5			
	KIRI	STD	KANAN	STD	KIRI	STD	KANAN	STD
ω (°)	-0.24882	0.02929	-0.23249	0.01463	-0.21625	0.01005	-0.22643	0.01543
φ (°)	-0.31321	0.02396	-0.12290	0.01414	-0.15741	0.00933	-0.10683	0.01473
κ (°)	-0.03265	0.03082	-0.03066	0.01673	-0.02941	0.01095	-0.03032	0.01712
X_L (m)	101.84114	0.28128	103.59388	0.12462	103.01109	0.07909	103.70104	0.12496
Y_L (m)	103.28609	0.29930	101.31520	0.12369	101.31151	0.07976	101.31113	0.12487
Z_L (m)	110.06578	0.25852	108.34672	0.14916	108.04083	0.09187	108.02849	0.15054

Tabel 4.9 dan *4.10* merupakan nilai final parameter eksterior orientasi (ω, φ, κ, X_L, Y_L, Z_L) dari foto kiri dan foto kanan pada tiap pasangan foto. Setiap foto memiliki masing-masing nilai parameter eksterior orientasi dan nilai standar deviasinya.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan pada judul penelitian ini, "*Desain Algoritma Penentuan Nilai Parameter Eksterior Orientasi Menggunakan Teknik Space Resection dan Absolute Orientation*", maka secara keseluruhan dapat disimpulkan bahwa :

1. Untuk menentukan parameter eksterior orientasi (ω, φ, κ, X_L, Y_L, Z_L) digunakan dua metode yaitu Absolut Orientasi dan *Space Resection*. Dalam hal ini, teknik absolut orientasi menghasilkan koordinat objek tiga dimensi dari hasil transformasi koordinat ke sistem referensi dimana koordinat ini akan digunakan sebagai data *input* untuk perhitungan *space resection*.
2. Setiap parameter yang dihasilkan dari teknik absolut orientasi dan *space resection*, membutuhkan nilai pendekatan sebagai data awal untuk proses perhitungan perataan untuk mendapatkan nilai parameter yang terkoreksi.
3. Dengan menggunakan teknik *least square*, maka proses perhitungan perataan terhadap parameter eksterior orientasi dapat dilakukan sehingga diketahui nilai ketelitian atau nilai akurasi dari tiap parameter.
4. Proses iterasi yang dilakukan pada perhitungan parameter eksterior orientasi menghasilkan nilai standar deviasi yang semakin kecil, dimana hal tersebut

menunjukkan bahwa nilai yang dihasilkan semakin teliti. Dengan standar deviasi rata-rata untuk parameter rotasi = 0.02rad dan untuk parameter posisi kamera = 0.17 m.

5. Dari kedua teknik tersebut, dapat dibangun sebuah algorithma yang dapat diimplementasikan untuk pembuatan sebuah program atau *software* khususnya pada bidang fotogrametri yang berkaitan dengan teknik absolut orientasi dan *resection*.
6. Pembuatan algorithma untuk teknik absolut orientasi dan *space resection* menggunakan program Matlab 2008b.

5.2 Saran

Pada penelitian ini, masih banyak hal yang belum terpenuhi secara keseluruhan. Oleh karena itu, saran yang dapat dianjurkan untuk penelitian ini agar mendapatkan hasil yang lebih maksimal adalah :

1. Agar dapat meminimalisir proses iterasi pada perhitungan perataan parameter eksterior orientasi, disarankan untuk mencoba atau menggunakan metode lain dalam proses penentuan nilai pendekatan awal parameter eksterior orientasi.
2. Untuk mendapatkan panjang basis yang sama dengan jarak yang sebenarnya maka dianjurkan untuk menambah jarak basis antara dua posisi kamera.
3. Hasil koordinat tiga dimensi pada proses absolut orientasi, akan lebih baik jika dilakukan proses perataan bersama parameter eksterior orientasi

menggunakan teknik *relative* orientation atau teknik *bundle adjustment* untuk meningkatkan ketelitian dari nilai koordinat tersebut.

DAFTAR PUSTAKA

Azis, Y. A. & Karara, H. M. (1971) Direct linear transformation from comparator coordinates into object space coordinates in close range photogrammetry. *American Society of Photogrammetry.*

Chen, F.-J. (1997) Application of least-squares adjustment technique to geometric camera calibration and photogrammetric flow visualization. *International Instrumentation Symposium.*

Dewwit, B. A. (1996) Initial approximations for the three-dimensional conformal coordinate transformation. *Photogrammetric Engineering & Remote Sensing,* 62(79-83.

Garcia, A. M. & Tozzi, C. L. (1996) A recursive approach to space resection using straight lines. *Photogrammetric Engineering & Remote Sensing,* 62

Hande Demirel, P. (2008) Analytical photgrammetry.

Jue, L. (2008) Research on close range photogrammetry with big rotation angle. *IAPRS,* 37

Kobayashi, K. & Mori, C. (1997) Relations between the coefficients in the projective transformation equations and the orientation elements of a photograph. *ASPRS,* 63

Leica (2006) *Leica photogrammetry suite : Project manager,* (Norcross, United States of America, Leica Geosystem Geospatial Imaging).

Mikhail, E. M., Bethel, J. S. & Mcglone, J. C. (2001) *Introduction to modern photogrammetry* (United States of America, John Willey & Sons).

Romsek, B. (2004) Coordinat transformations paper presented at the *3D Transformations,* City.

Seedahmed, G. H. & Habib, A. F. (2002) Linear recovery of the exterior orientation parameters in a planar object space. *The Pacific Northwest National Lab.*

Shih, T. Y. (1996) The sign permutation in the rotation matrix and the formulation of the collinearity and coplanarity equations. *ASPRS,* 62

Tony, W. C. S. (2006) *Development of land based mobile mapping system using gps and close range photogrammetry techniques.* Universiti Teknologi Malaysia.

Wolf, P. R. (1993) *Elemen fotogrametri dengan interpretasi foto udara dan penginderaan jauh : Edisi kedua,* (D. Gunadi, M. S. Drs. Totok Gunawan & D. Zuharnen, Trans.) (Bulaksumur, Yogyakarta, Gadjah Mada University Press).

Wolf, P. R. & Dewit, B. A. (2000) *Element of photogrammetry with applications in gis : Third edition,* (Madison, McGraw-Hill Companies).

Yi, C. & Jue, L. (2008) Performing space resection using total least square. *IAPRS,* XXXVII

Zhang, C. & Yao, W. (2008) The comparison of 3d anlysis between photogrammetry and computer vision. *IAPRS,* 37

LAMPIRAN A

DATA FOTO

A.1 Data Foto Stereo Pair One

Gambar 1.a Foto jalan diambil dari sisi kiri

Gambar 2.a Foto jalan diambil dari sisi kanan

A.2 Data Foto Stereo Pair Two

Gambar 3.a Foto jalan di ambil dari sisi kiri

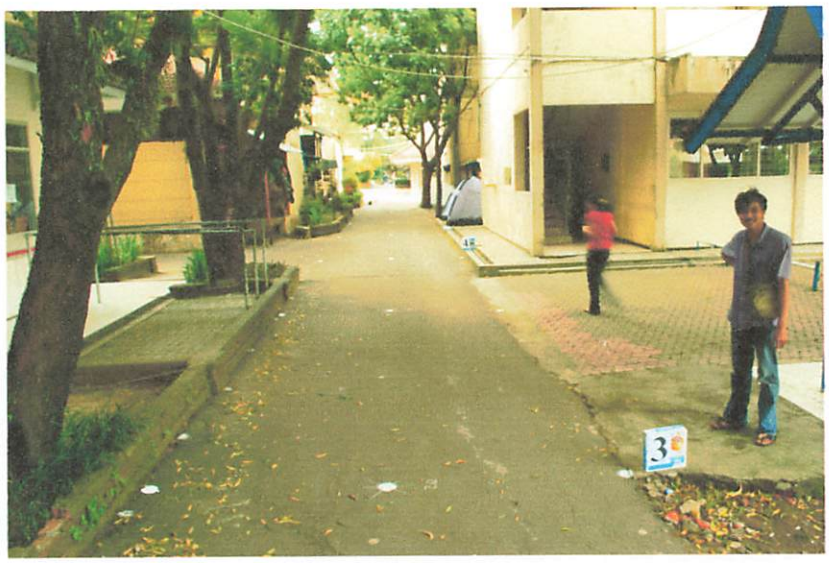
Gambar 4.a Foto jalan di ambil dari sisi kanan

A.3 Data Foto Stereo Pair Three

Gambar 5.a Foto jalan di ambil dari sisi kiri

Gambar 6.a Foto jalan di ambil dari sisi kanan

A.4 Data Foto Stereo Pair Four



Gambar 7.a Foto jalan di ambil dari sisi kiri

Gambar 8.a Foto jalan di ambil dari sisi kanan

A.5 Data Foto Stereo Pair Five

Gambar 9.a Foto jalan di ambil dari sisi kiri

Gambar 10.a Foto jalan di ambil dari sisi kanan

LAMPIRAN B

LISTING CODE ABSOLUTE ORIENTATION

```
ing System;
ing System.Collections.Generic;
ing System.Linq;
sing System.IO;
sing System.Text;
sing System.Diagnostics;

amespace AbsOrientation

    class Program
    {
        private const string File_Name = "First Data.txt";
        static void Main(string[] args)
        {
            string[] readText = File.ReadAllLines(File_Name);
            int rows = readText.Count() - 1;
            //Trace.WriteLine("Rows : " + rows.ToString());

            int numCommon = 0;
            int numPoint = 0;
            Funct.ReadData(File_Name, ref numCommon, ref numPoint);
            //Trace.WriteLine("Common, points : " + numCommon.ToString() + ", " +
    numPoint.ToString());

            double[,] common = new double[numCommon, 6];
            double[,] points = new double[numPoint, 3];
            int[] label = new int[rows];
            double[] param = new double[7];
            Funct.ReadData(File_Name, label, common, points, numCommon);
            //Funct.PrintMatrixDebug(ref common, "Common");

            AbsOrientation.AbsoluteOrientation method = new AbsOrientation.
    AbsoluteOrientation();
            method.AbsoluteOrientationAll(common, points, ref param);
        }
    }
}
```

```
    public static void PrintMatrixDebug(ref double[,] a, string name)
    {
        int lower1 = a.GetLowerBound(0);
        int upper1 = a.GetUpperBound(0);
        int lower2 = a.GetLowerBound(1);
        int upper2 = a.GetUpperBound(1);
        int length = a.Length;
        System.Diagnostics.Trace.WriteLine("\n" + name + ":");
        for (int i = lower1; i <= upper1; i++)
        {
            for (int j = lower2; j <= upper2; j++)
            {
                System.Diagnostics.Trace.Write(a[i, j].ToString("0.0#####",
CultureInfo.InvariantCulture) + "\t");
            }
            System.Diagnostics.Trace.WriteLine("");
        }
    }

    public static void PrintVectorDebug(ref double[] v, string name)
    {
        int lower = v.GetLowerBound(0);
        int upper = v.GetUpperBound(0);
        int length = v.Length;
        System.Diagnostics.Trace.WriteLine("\n" + name + ":");
        for (int i = lower; i <= upper; i++)
        {
            System.Diagnostics.Trace.Write(v[i].ToString() + "\t");
        }

        System.Diagnostics.Trace.WriteLine("");
    }
  }
}
```

```
ing System;
sing System.Collections.Generic;
sing System.Linq;
sing System.Text;
sing System.IO;
sing System.Diagnostics;
sing System.Globalization;

amespace AbsOrientation

    public static class Funct
    {
        public static void ReadData(string file, int[] lbl, double[,] common, double[,]
    point, int commonNum)
        {
            if (!File.Exists(file))
            {
                Console.WriteLine("{0} does not exist.", file);
                return;
            }

            string[] readText = File.ReadAllLines(file);
            int rows = readText.Count();

            for (int i = 1; i < rows; i++)
            {
                String[] text = readText[i].Split(new char[] { ' ', '\t' });
                if (i <= commonNum)
                {
                    lbl[i - 1] = Int32.Parse(text[0]);
                    common[i - 1, 0] = Double.Parse(text[1]);
                    common[i - 1, 1] = Double.Parse(text[2]);
                    common[i - 1, 2] = Double.Parse(text[3]);
                    common[i - 1, 3] = Double.Parse(text[4]);
                    common[i - 1, 4] = Double.Parse(text[5]);
                    common[i - 1, 5] = Double.Parse(text[6]);
                }
                else
                {
                    lbl[i - 1] = Int32.Parse(text[0]);
                    point[i - (commonNum + 1), 0] = Double.Parse(text[1]);
                    point[i - (commonNum + 1), 1] = Double.Parse(text[2]);
                    point[i - (commonNum + 1), 2] = Double.Parse(text[3]);
                }
            }
        }

        public static void ReadData(string file, ref int commonNum, ref int pointNum)
        {
            if (!File.Exists(file))
            {
                Console.WriteLine("{0} does not exist.", file);
                return;
            }

            string[] readText = File.ReadAllLines(file);
            int rows = readText.Count();

            for (int i = 0; i < 1; i++)
            {
                String[] text = readText[i].Split(new char[] { ' ', '\t' });
                commonNum = Int32.Parse(text[0]);
                pointNum = Int32.Parse(text[1]);
            }
        }
```

```
ing System;
sing System.Collections.Generic;
sing System.Linq;
sing System.Text;
sing System.Diagnostics;
sing Accord.Math;

namespace AbsOrientation
{
    class AbsoluteOrientation
    {
        double[] xarb; double[] yarb; double[] zarb;
        double[] xcon; double[] ycon; double[] zcon;
        double[] x; double[] y; double[] z;

        #region Public Methods
        public double[,] AbsoluteOrientationAll(double[,] common, double[,] points, ref ↙
    double[] sevenparam)
        {
            // Common points
            this.xarb = Matrix.GetColumn(common, 0);
            this.yarb = Matrix.GetColumn(common, 1);
            this.zarb = Matrix.GetColumn(common, 2);
            this.xcon = Matrix.GetColumn(common, 3);
            this.ycon = Matrix.GetColumn(common, 4);
            this.zcon = Matrix.GetColumn(common, 5);

            // points transformation
            this.x = points.GetColumn(0);
            this.y = points.GetColumn(1);
            this.z = points.GetColumn(2);

            // number data points transformation
            int n = x.Length;
            double[,] pointTrans = new double[n, 3];

            // Get initial approximation value
            sevenparam = InitialAbsoluteParameters(xarb, yarb, zarb, xcon, ycon, zcon);
            Funct.PrintVectorDebug(ref sevenparam, "Initial Value");

            // Adjustment initial value
            double std = 0.0;
            sevenparam = AdjustmentAbsoluteParameters(sevenparam, xarb, yarb, zarb, xcon,↙
     ycon, zcon, ref std);
            Funct.PrintVectorDebug(ref sevenparam, "Adjustment Value");

            // Points transformation
            pointTrans = PointTransformation(sevenparam, x, y, z);
            Funct.PrintMatrixDebug(ref pointTrans, "Points Transformation");

            return pointTrans;
        }

        public double[] InitialAbsoluteParameters(double[] x1, double[] y1, double[] z1, ↙
    double[] x2, double[] y2, double[] z2)
        {
            // Deklarasi dimensi nilai parameter transformasi
            double[] param = new double[7];

            // Jumlah data
            int n = x1.Length;

            // Menghitung nilai parameter skala (scale)
            int numscale = 0;
            double sumscale = 0;
            for (int i = 0; i < n -1 ; i++)
```

```
        {
            for (int j = i + 1; j < n; j++)
            {
                double distcon = Math.Sqrt(Math.Pow(x2[i] - x2[j], 2) + Math.Pow(y2 ↙
[i] - y2[j], 2) +
                                           Math.Pow(z2[i] - z2[j], 2));
                double distarb = Math.Sqrt(Math.Pow(x1[i] - x1[j], 2) + Math.Pow(y1 ↙
[i] - y1[j], 2) +
                                           Math.Pow(z1[i] - z1[j], 2));
                sumscale += distcon / distarb;
                numscale++;
            }
        }
        param[0] = sumscale / numscale;

        // Menentukan kombinasi titik dengan kekuatan geometri terbaik
        int[] p = StrongGeometry(x2, y2, z2);
        int pt1 = p[0]; int pt2 = p[1]; int pt3 = p[2];

        /* Compute coefficients of equation of plane through the 3 points
        in the arbitrary system */
        double[] P12arb = new double[3] { x1[pt1] - x1[pt2], y1[pt1] - y1[pt2], z1 ↙
[pt1] - z1[pt2] };
        double[] P13arb = new double[3] { x1[pt1] - x1[pt3], y1[pt1] - y1[pt3], z1 ↙
[pt1] - z1[pt3] };
        double[] N_arb = Matrix.VectorProduct(P12arb, P13arb);

        /* Compute coefficients of equation of plane through the 3 points
        in the control system */
        double[] P12con = new double[3] { x2[pt1] - x2[pt2], y2[pt1] - y2[pt2], z2 ↙
[pt1] - z2[pt2] };
        double[] P13con = new double[3] { x2[pt1] - x2[pt3], y2[pt1] - y2[pt3], z2 ↙
[pt1] - z2[pt3] };
        double[] N_con = Matrix.VectorProduct(P12con, P13con);

        /* Compute tilt & azimuth of plane in arbitrary system */
        double arb_tilt = Math.Atan2(N_arb[2], Tools.Hypotenuse(N_arb[0], N_arb[1])) ↙
+ Math.PI / 2;
        double arb_az = Math.Atan2(N_arb[0], N_arb[1]);

        /* Compute tilt & azimuth of plane in control system */
        double con_tilt = Math.Atan2(N_con[2], Tools.Hypotenuse(N_con[0], N_con[1])) ↙
+ Math.PI / 2;
        double con_az = Math.Atan2(N_con[0], N_con[1]);

        /* Compute the corresponding rotation matrices with swing = 0 */
        double[,] Arb_Rot = RotationMatrixTSA(arb_tilt, 0.0, arb_az);
        double[,] Con_Rot = RotationMatrixTSA(con_tilt, 0.0, con_az);

        /* Rotate arbitrary and control coordinates for points 1 and 2
        to get a line in the arbitrary system and the corresponding
        line in the control system. Don't need to rotate Z because
        with these rotations all of the z values must be the same. */
        double x_arb1 = Arb_Rot[0, 0] * x1[pt1] + Arb_Rot[0, 1] * y1[pt1] + Arb_Rot[0↙
, 2] * z1[pt1];
        double y_arb1 = Arb_Rot[1, 0] * x1[pt1] + Arb_Rot[1, 1] * y1[pt1] + Arb_Rot[1↙
, 2] * z1[pt1];
        double x_arb2 = Arb_Rot[0, 0] * x1[pt2] + Arb_Rot[0, 1] * y1[pt2] + Arb_Rot[0↙
, 2] * z1[pt2];
        double y_arb2 = Arb_Rot[1, 0] * x1[pt2] + Arb_Rot[1, 1] * y1[pt2] + Arb_Rot[1↙
, 2] * z1[pt2];
        double x_con1 = Con_Rot[0, 0] * x2[pt1] + Con_Rot[0, 1] * y2[pt1] + Con_Rot[0↙
, 2] * z2[pt1];
        double y_con1 = Con_Rot[1, 0] * x2[pt1] + Con_Rot[1, 1] * y2[pt1] + Con_Rot[1↙
, 2] * z2[pt1];
        double x_con2 = Con_Rot[0, 0] * x2[pt2] + Con_Rot[0, 1] * y2[pt2] + Con_Rot[0↙
```

```
, 2] * z2[pt2];
        double y_con2 = Con_Rot[1, 0] * x2[pt2] + Con_Rot[1, 1] * y2[pt2] + Con_Rot[1
, 2] * z2[pt2];

        /* Get swing by subtracting azimuths */
        double azimuth_con = Math.Atan2(x_con2 - x_con1, y_con2 - y_con1);
        double azimuth_arb = Math.Atan2(x_arb2 - x_arb1, y_arb2 - y_arb1);
        double swing = azimuth_con - azimuth_arb;
        Arb_Rot = RotationMatrixTSA(arb_tilt, swing, arb_az);

        ///* Now compute (ConRotMat:transpose * ArbRotMat):transpose
        //This is overall rotation matrix */
        double[,] RotMat = ((Con_Rot.Transpose()).Multiply(Arb_Rot)).Transpose();

        /* Compute omega, phi, kappa from rotation matrix */
        param[2] = Math.Asin(RotMat[2, 0]);          // phi
        if (Math.Abs(RotMat[2, 0]) < 1.0)
        {
            param[1] = Math.Atan2(-RotMat[2, 1], RotMat[2, 2]);      //omega
            param[3] = Math.Atan2(-RotMat[1, 0], RotMat[0, 0]);      //kappa
        }
        else
        {
            // omega and kappa are undefined, so define them
            param[1] = 0.0;                                                  //omega
            param[3] = Math.Atan2(RotMat[0, 1], RotMat[1, 1]);  //kappa
        }

        /* Compute average Tx, Ty, and Tz using all common points */
        double txsum = 0.0;
        double tysum = 0.0;
        double tzsum = 0.0;
        for (int i = 0; i < n; i++)
        {
            txsum += xcon[i] - param[0] * (RotMat[0, 0] * x1[i] + RotMat[1, 0] * y1
[i] + RotMat[2, 0] * z1[i]);
            tysum += ycon[i] - param[0] * (RotMat[0, 1] * x1[i] + RotMat[1, 1] * y1
[i] + RotMat[2, 1] * z1[i]);
            tzsum += zcon[i] - param[0] * (RotMat[0, 2] * x1[i] + RotMat[1, 2] * y1
[i] + RotMat[2, 2] * z1[i]);
        }
        param[4] = txsum / n;
        param[5] = tysum / n;
        param[6] = tzsum / n;
        return param;
    }

    public double[] AdjustmentAbsoluteParameters(double[] initparam, double[] x1,
double[] y1, double[] z1, double[] x2, double[] y2, double[] z2, ref double std)
    {
        double[] param = initparam;
        int n = x1.Length;
        double[,] A = new double[3 * n, 7];
        double[] L = new double[3 * n];
        double[] cor = new double[7];
        double convergen = 1.0;
        for (int iter = 1; iter <= 20; iter++)
        {
            for (int i = 0; i < n; i++)
            {
                int eqn = i * 3;
                double[] Lo = FormMatrixL(param, x1[i], y1[i], z1[i], x2[i], y2[i],
z2[i]);
                for (int j = 0; j < 7; j++)
                {
                    double[,] Ao = FormMatrixA(param, x1[i], y1[i], z1[i]);
```

```
                    A[eqn, j] = Ao[0, j];
                    A[eqn + 1, j] = Ao[1, j];
                    A[eqn + 2, j] = Ao[2, j];
                }
                L[eqn] = Lo[0];
                L[eqn + 1] = Lo[1];
                L[eqn + 2] = Lo[2];
            }
            cor = (Matrix.PseudoInverse(A)).Multiply(L);
            param = param.Add(cor);
            double[] res = (A.Multiply(cor)).Subtract(L);
            std = SumSquare(res) / (3 * n - 7);
            if (SumAbs(cor) < 1.0e-5)
            {
                break;
            }
            else
            {
                if (SumAbs(cor) < convergen)
                {
                    convergen = SumAbs(cor);
                }
                else
                {
                    break;
                }

            }
        }
        return param;
    }

    public double[,] PointTransformation(double[] p, double[] x1, double[] y1, double↙
[] z1)
    {
        double[,] point = new double[3, 3];
        double[,] R = RotationMatrixOPK(p[1], p[2], p[3]);
        for (int i = 0; i < x1.Length; i++)
        {
            point[i, 0] = p[0] * (R[0, 0] * x1[i] + R[1, 0] * y[i] + R[2, 0] * z1[i])↙
 + p[4];
            point[i, 1] = p[0] * (R[0, 1] * x1[i] + R[1, 1] * y[i] + R[2, 1] * z1[i])↙
 + p[5];
            point[i, 2] = p[0] * (R[0, 2] * x1[i] + R[1, 2] * y[i] + R[2, 2] * z1[i])↙
 + p[6];
        }
        return point;
    }
    #endregion

    #region Private Methods
    private double[,] FormMatrixA(double[] par, double x1, double y1, double z1)
    {
        double[,] A = new double[3, 7];
        double[,] R = RotationMatrixOPK(par[1], par[2], par[3]);

        A[0, 0] = R[0, 0] * x1 + R[1, 0] * y1 + R[2, 0] * z1;
        A[1, 0] = R[0, 1] * x1 + R[1, 1] * y1 + R[2, 1] * z1;
        A[2, 0] = R[0, 2] * x1 + R[1, 2] * y1 + R[2, 2] * z1;

        A[0, 1] = 0.0;
        A[1, 1] = -1.0 * par[0] * A[2, 0];
        A[2, 1] = par[0] * A[1, 0];

        A[0, 2] = par[0] * (-Math.Sin(par[2]) * Math.Cos(par[3]) * x1 + Math.Sin(par ↙
[2]) * Math.Sin(par[3]) * y1 +
```

```
                    Math.Cos(par[2]) * z1);
        A[1, 2] = par[0] * (Math.Sin(par[1]) * Math.Cos(par[1]) * Math.Cos(par[3]) * ↙
x1 + (-1.0) * Math.Sin(par[1]) * Math.Cos(par[2]) * Math.Sin(par[3]) * y1 +
                    Math.Sin(par[1]) * Math.Sin(par[2]) * z1);
        A[2, 2] = par[0] * (-1.0 * Math.Cos(par[1]) * Math.Cos(par[2]) * Math.Cos(par↙
[3]) * x1 + Math.Cos(par[1]) * Math.Cos(par[2]) * Math.Sin(par[3]) * y1 +
                    -1.0 * Math.Cos(par[1]) * Math.Sin(par[2]) * z1);

        A[0, 3] = par[0] * (R[1, 0] * x1 - R[0, 0] * y1);
        A[1, 3] = par[0] * (R[1, 1] * x1 - R[0, 1] * y1);
        A[2, 3] = par[0] * (R[1, 2] * x1 - R[0, 2] * y1);

        A[0, 4] = 1.0;
        A[1, 4] = 0.0;
        A[2, 4] = 0.0;
        A[0, 5] = 0.0;
        A[1, 5] = 1.0;
        A[2, 5] = 0.0;
        A[0, 6] = 0.0;
        A[1, 6] = 0.0;
        A[2, 6] = 1.0;

        return A;
    }
    private double[] FormMatrixL(double[] par, double x1,double y1, double z1, double↙
x2, double y2, double z2)
    {
        double[] L = new double[3];
        double[,] R = RotationMatrixOPK(par[1], par[2], par[3]);

        L[0] = x2 - par[0] * (R[0, 0] * x1 + R[1, 0] * y1 + R[2, 0] * z1) - par[4];
        L[1] = y2 - par[0] * (R[0, 1] * x1 + R[1, 1] * y1 + R[2, 1] * z1) - par[5];
        L[2] = z2 - par[0] * (R[0, 2] * x1 + R[1, 2] * y1 + R[2, 2] * z1) - par[6];

        return L;
    }
    private int[] StrongGeometry(double[] x1, double[] y1, double[] z1)
    {
        int[] combPoints = new int[3];
        int conNum = x1.Length;
        double maxaltsq = 0.0;
        for (int ind1 = 0; ind1 < conNum - 2; ind1++)
        {
            for (int ind2 = ind1 + 1; ind2 < conNum - 1; ind2++)
            {
                double a = Math.Pow(x1[ind1] - x1[ind2], 2);
                double b = Math.Pow(y1[ind1] - y1[ind2], 2);
                double c = Math.Pow(z1[ind1] - z1[ind2], 2);
                double dsq12 = a + b + c;
                for (int ind3 = ind2 + 1; ind3 < conNum; ind3++)
                {
                    double d = Math.Pow(x1[ind1] - x1[ind3], 2);
                    double e = Math.Pow(y1[ind1] - y1[ind3], 2);
                    double f = Math.Pow(z1[ind1] - z1[ind3], 2);
                    double dsq13 = d + e + c;

                    double g = Math.Pow(x1[ind2] - x1[ind3], 2);
                    double h = Math.Pow(y1[ind2] - y1[ind3], 2);
                    double i = Math.Pow(z1[ind2] - z1[ind3], 2);
                    double dsq23 = g + h + i;

                    double a2, b2, c2;
                    if ((dsq12 >= dsq13) && (dsq12 >= dsq23))
                    {
                        c2 = dsq12;
                        a2 = dsq13;
```

```
                        b2 = dsq23;
                    }
                    else
                    {
                        if ((dsq13 >= dsq12) && (dsq13 >= dsq23))
                        {
                            c2 = dsq13;
                            a2 = dsq12;
                            b2 = dsq23;
                        }
                        else
                        {
                            c2 = dsq23;
                            a2 = dsq12;
                            b2 = dsq13;
                        }
                    }
                    double h2 = (2 * (c2 * (a2 + b2) + a2 * b2) - a2 * a2 - b2 * b2 -
c2 * c2) / (4 * c2);
                    if (h2 < 0)
                    {
                        throw new Exception("Error in Altitude Computation");
                    }
                    if (h2 > maxaltsq)
                    {
                        combPoints[0] = ind1;
                        combPoints[1] = ind2;
                        combPoints[2] = ind3;
                        maxaltsq = h2;
                    }
                }
            }
        }
        return combPoints;
    }
    private double[,] RotationMatrixOPK(double om, double ph, double kp)
    {
        double[,] M = new double[3, 3];
        M[0, 0] = Math.Cos(ph) * Math.Cos(kp);
        M[0, 1] = Math.Cos(om) * Math.Sin(kp) + Math.Sin(om) * Math.Sin(ph) * Math.
Cos(kp);
        M[0, 2] = Math.Sin(om) * Math.Sin(kp) - Math.Cos(om) * Math.Sin(ph) * Math.
Cos(kp);

        M[1, 0] = -Math.Cos(ph) * Math.Sin(kp);
        M[1, 1] = Math.Cos(om) * Math.Cos(kp) - Math.Sin(om) * Math.Sin(ph) * Math.
Sin(kp);
        M[1, 2] = Math.Sin(om) * Math.Cos(kp) + Math.Cos(om) * Math.Sin(ph) * Math.
Sin(kp);

        M[2, 0] = Math.Sin(ph);
        M[2, 1] = -Math.Sin(om) * Math.Cos(ph);
        M[2, 2] = Math.Cos(om) * Math.Cos(ph);
        return M;
    }
    private double[,] RotationMatrixTSA(double t, double s, double a)
    {
        double[,] R = new double[3, 3];

        R[0, 0] = -Math.Cos(a) * Math.Cos(s) - Math.Sin(a) * Math.Cos(t) * Math.Sin
(s);
        R[0, 1] = Math.Sin(a) * Math.Cos(s) - Math.Cos(a) * Math.Cos(t) * Math.Sin(s)
;
        R[0, 2] = -Math.Sin(t) * Math.Sin(s);

        R[1, 0] = Math.Cos(a) * Math.Sin(s) - Math.Sin(a) * Math.Cos(t) * Math.Cos(s)
```

```
* ;
            R[1, 1] = -Math.Sin(a) * Math.Sin(s) - Math.Cos(a) * Math.Cos(t) * Math.Cos
  (s);
            R[1, 2] = -Math.Sin(t) * Math.Cos(s);

            R[2, 0] = -Math.Sin(a) * Math.Sin(t);
            R[2, 1] = -Math.Cos(a) * Math.Sin(t);
            R[2, 2] = Math.Cos(t);

            return R;
        }
        private double SumSquare(double[] a)
        {
            int dim = a.Length;
            double b = 0.0;
            for (int i = 0; i < dim; i++)
            {
                b += Math.Pow(a[i], 2);
            }
            return b;
        }
        private double SumAbs(double[] a)
        {
            int dim = a.Length;
            double b = 0.0;
            for (int i = 0; i < dim; i++)
            {
                b += Math.Abs(a[i]);
            }
            return b;
        }
        #endregion
    }
}
```

LAMPIRAN C

LISTING CODE LINEAR SPACE RESECTION

```
ing System;
sing System.Collections.Generic;
sing System.Linq;
sing System.Text;
sing System.IO;
sing System.Diagnostics;

namespace GeneralSpaceResection
{
    class Program
    {
        private const string File_Name = "DSC_44_NonPlanarDLT.txt";
        static void Main(string[] args)
        {
            #region Fields
            string[] readText = File.ReadAllLines(File_Name);
            int rows = readText.Count() - 2;
            //Trace.WriteLine("Rows : " + rows.ToString());

            double pixelSize = 0.0;
            double[] IntOr = new double[3];
            double[] Resolution = new double[2];
            int[] Label = new int[rows];
            double[,] ImageCord = new double[rows, 2];
            double[,] ObjectCord = new double[rows, 3];
            #endregion

            Funct.ReadData(File_Name, Label, IntOr, ImageCord, ObjectCord, Resolution,
    ref pixelSize);
            //Funct.PrintMatrixDebug(ref ImageCord, "Image Coord ");
            //Funct.PrintMatrixDebug(ref ObjectCord, "Control Coord ");
            ////Trace.WriteLine("Pixel Size : " + pixelSize.ToString());

            double[,] grid = Funct.PixelToGrid(ImageCord, pixelSize);
            //Funct.PrintMatrixDebug(ref grid, "Image Coord ");

            double[,] cart = Funct.PixelToCartesian(ImageCord, Resolution, pixelSize);
            //Funct.PrintMatrixDebug(ref cart, "Image Coord ");

            GeneralSpaceResection.SpaceResection methods = new SpaceResection();
            //double[] SolOne = methods.MullerMethods(IntOr, ImageCord, ObjectCord);
            double[] SolTwo = methods.NonPlanarDLT(grid, ObjectCord, Resolution, IntOr[0]
    , pixelSize);
        }
    }
}
```

```
ing System;
ing System.Collections.Generic;
sing System.Linq;
sing System.Text;
sing System.IO;
sing System.Diagnostics;
sing System.Globalization;
sing Accord.Math;

amespace GeneralSpaceResection

    public static class Funct
    {
        #region Public Methods
        public static void ReadData(string file, int[] label, double[] IO, double[,] im, ↙
    double[,] con, double[] imgResol,ref double pixSize)
        {
            if (!File.Exists(file))
            {
                Console.WriteLine("{0} does not exist.", file);
                return;
            }

            string[] readText = File.ReadAllLines(file);
            int rows = readText.Count();

            for (int i = 0; i < rows; i++)
            {
                String[] text = readText[i].Split(new char[] { ' ', '\t' });
                if (i == 0)
                {
                    IO[0] = Double.Parse(text[0]);
                    IO[1] = Double.Parse(text[1]);
                    IO[2] = Double.Parse(text[2]);
                    //Trace.WriteLine(IntOrient[2].ToString());
                }
                else if (i == 1)
                {
                    imgResol[0] = Double.Parse(text[0]);
                    imgResol[1] = Double.Parse(text[1]);
                    pixSize = Double.Parse(text[2]);
                }
                else
                {
                    label[i - 2] = Int32.Parse(text[0]);
                    im[i - 2, 0] = Double.Parse(text[1]);
                    im[i - 2, 1] = Double.Parse(text[2]);
                    con[i - 2, 0] = Double.Parse(text[3]);
                    con[i - 2, 1] = Double.Parse(text[4]);
                    con[i - 2, 2] = Double.Parse(text[5]);
                }
            }
        }

        public static bool IsEmpty(ref double[] m)
        {
            int row = m.Length;
            for (int i = 1; i <= row; i++)
                if (m.Euclidean() != 0.0)
                    return false;
            return true;
        }

        public static double Distance(double x, double y, double z)
        {
            double d = 0;
```

```
        d = Math.Sqrt(Math.Pow(x, 2) + Math.Pow(y, 2) + Math.Pow(z, 2));
        return d;
    }

    public static double[] Distance(double[] x, double[] y, double[] z, int n)
    {
        int ndist = 0;
        if (n == 4)
        {
            ndist = 6;
        }

        if (n == 5)
        {
            ndist = 10;
        }

        double[] D = new double[ndist];
        if (n == 4)
        {
            D[0] = Distance(x[1] - x[0], y[1] - y[0], z[1] - z[0]);      //a
            D[1] = Distance(x[2] - x[0], y[2] - y[0], z[2] - z[0]);      //c
            D[2] = Distance(x[3] - x[0], y[3] - y[0], z[3] - z[0]);
            D[3] = Distance(x[2] - x[1], y[2] - y[1], z[2] - z[1]);      //b
            D[4] = Distance(x[3] - x[1], y[3] - y[1], z[3] - z[1]);
            D[5] = Distance(x[3] - x[2], y[3] - y[2], z[3] - z[2]);
        }

        if (n == 5)
        {
            D[0] = Distance(x[1] - x[0], y[1] - y[0], z[1] - z[0]);      //a
            D[1] = Distance(x[2] - x[0], y[2] - y[0], z[2] - z[0]);      //c
            D[2] = Distance(x[3] - x[0], y[3] - y[0], z[3] - z[0]);
            D[3] = Distance(x[4] - x[0], y[4] - y[0], z[4] - z[0]);
            D[4] = Distance(x[2] - x[1], y[2] - y[1], z[2] - z[1]);      //b
            D[5] = Distance(x[3] - x[1], y[3] - y[1], z[3] - z[1]);
            D[6] = Distance(x[4] - x[1], y[4] - y[1], z[4] - z[1]);
            D[7] = Distance(x[3] - x[2], y[3] - y[2], z[3] - z[2]);
            D[8] = Distance(x[4] - x[2], y[4] - y[2], z[4] - z[2]);
            D[9] = Distance(x[4] - x[3], y[4] - y[3], z[4] - z[3]);
        }
        return D;
    }

    public static int[] SmallErr(double[] x)
    {
        int n = x.Length;
        List<int> indx = new List<int>();
        for (int i = 0; i < n; i++)
        {
            if (Math.Abs(x[i]) < 0.025)
                indx.Add(i);
        }
        int[] ind = indx.ToArray();
        return ind;
    }

    public static int MostSmallErr(double[] x)
    {
        int n = x.Length;
        double c = Matrix.Min(x);
        int indx = 0;
        for (int i = 0; i < n; i++)
        {
            if (Math.Abs(x[i]) == c)
                indx = i;
```

```
        }
        return indx;
    }

    public static double[,] DeleteDuplicate(double[,] p)
    {
        int row = p.GetUpperBound(0)+1;
        int col = p.GetUpperBound(1)+1;
        double[,] q = new double[row, col];
        //List<double[]> temp = new List<double[]>();
        for (int i = 0; i < col; i++)
        {
            bool duplicate = false;
            double[] temp1 = p.GetColumn(i);
            for (int j = 0; j < col; j++)
            {
                double[] temp2 = q.GetColumn(j);
                double[] sub = Matrix.Subtract(temp1, temp2);
                double sum = Matrix.Sum(Matrix.Abs(sub));
                if (sum < 1.0e-5)
                {
                    duplicate = true;
                    break;
                }
            }
            if (!duplicate)
            {
                q = q.SetColumn(i, temp1);
            }
        }

        List<double[]> t = new List<double[]>();
        for (int k = 0; k < col; k++)
        {
            double[] tempdel = q.GetColumn(k);
            double s = Matrix.Sum(tempdel);
            if (s != 0.0)
                t.Add(tempdel);
        }

        double[][] convt = t.ToArray();
        q = convt.ToMatrix().Transpose();
        return q;
    }

    public static double SumSquare(double[] a)
    {
        int dim = a.Length;
        double b = 0.0;
        for (int i = 0; i < dim; i++)
        {
            b += Math.Pow(a[i], 2);
        }
        return b;
    }

    public static double SumAbs(double[] a)
    {
        int dim = a.Length;
        double b = 0.0;
        for (int i = 0; i < dim; i++)
        {
            b += Math.Abs(a[i]);
        }
        return b;
    }
```

```
    public static double[] ArrayToRow(double[,] m)
    {
        int col = m.Length;
        double[] t = new double[col];
        for (int i = 0; i < col; i++)
        {
            t[i] = m[i, 0];
        }
        return t;
    }

    public static void PrintMatrixDebug(ref double[,] a, string name)
    {
        int lower1 = a.GetLowerBound(0);
        int upper1 = a.GetUpperBound(0);
        int lower2 = a.GetLowerBound(1);
        int upper2 = a.GetUpperBound(1);
        int length = a.Length;
        System.Diagnostics.Trace.WriteLine("\n" + name + ":");
        for (int i = lower1; i <= upper1; i++)
        {
            for (int j = lower2; j <= upper2; j++)
            {
                System.Diagnostics.Trace.Write(a[i, j].ToString("0.0######",
CultureInfo.InvariantCulture) + "\t");
            }
            System.Diagnostics.Trace.WriteLine("");
        }
    }

    public static void PrintVectorDebug(ref double[] v, string name)
    {
        int lower = v.GetLowerBound(0);
        int upper = v.GetUpperBound(0);
        int length = v.Length;
        System.Diagnostics.Trace.WriteLine("\n" + name + ":");
        for (int i = lower; i <= upper; i++)
        {
            System.Diagnostics.Trace.Write(v[i].ToString() + "\t");
        }

        System.Diagnostics.Trace.WriteLine("");
    }

    public static double[,] PixelToGrid(double[,] p, double pixelSize)
    {
        int n = p.GetUpperBound(0) + 1;
        double[,] grid = new double[n, 2];
        for (int i = 0; i < n; i++)
        {
            grid[i, 0] = p[i, 0] * pixelSize;
            grid[i, 1] = p[i, 1] * pixelSize;
        }
        return grid;
    }

    public static double[,] PixelToCartesian(double[,] p, double[] ImageResolution,
double pixelSize)
    {
        int n = p.GetUpperBound(0) + 1;
        double Xc = ImageResolution[0] / 2 - 0.5;
        double Yc = ImageResolution[1] / 2 - 0.5;
        double[,] cart = new double[n, 2];
        for (int i = 0; i < n; i++)
        {
```

```
            cart[i, 0] = (p[i, 0] - Xc) * pixelSize;
            cart[i, 1] = (Yc - p[i, 1]) * pixelSize;
        }
        return cart;
    }
    #endregion
}
```

```
ing System;
sing System.Collections.Generic;
sing System.Diagnostics;
sing System.Linq;
sing System.Text;
sing Accord.Math;

namespace GeneralSpaceResection
{
    public class SpaceResection
    {
        #region Fields
        double[] x; double[] y;
        double[] X; double[] Y; double[] Z;
        #endregion

        #region Public Methods
        #region Church Adjustment
        public double[] ChurchAdjustment(double[] Initpose, double[] x, double[] y,
    double[] X, double[] Y, double[] Z, double f, ref double err)
        {
            int ndata = x.Length;
            double[] pose = new double[3];

            double[] d = new double[ndata];
            double[] l = new double[ndata];
            double[] m = new double[ndata];
            double[] n = new double[ndata];
            for (int i = 0; i < ndata; i++)
            {
                d[i] = Funct.Distance(x[i], y[i], f);
                l[i] = x[i] / d[i];
                m[i] = y[i] / d[i];
                n[i] = -f / d[i];
            }

            double[] k = new double[ndata];
            k[0] = l[0] * l[1] + m[0] * m[1] + n[0] * n[1];
            k[1] = l[1] * l[2] + m[1] * m[2] + n[1] * n[2];
            k[2] = l[2] * l[0] + m[2] * m[0] + n[2] * n[0];


            double[] D = new double[ndata];
            double[] L = new double[ndata];
            double[] M = new double[ndata];
            double[] N = new double[ndata];
            double[] K = new double[ndata];
            double[] I = new double[ndata];
            double[] J = new double[ndata];
            double[] A = new double[ndata];
            double[] B = new double[ndata];
            double[] C = new double[ndata];
            double[] deltaK = new double[ndata];
            double[] E = new double[ndata];
            double[] F = new double[ndata];
            double[] G = new double[ndata];
            double delta = 0.0;
            double[] cor = new double[3];
            for (int iter = 1; iter <= 10; iter++)
            {
                //Trace.WriteLine("Iterasi : " + iter.ToString());
                for (int i = 0; i < ndata; i++)
                {
                    D[i] = Funct.Distance(X[i] - Initpose[0], Y[i] - Initpose[1], Z[i] -
    Initpose[2]);
                    L[i] = (X[i] - Initpose[0]) / (D[i]);
```

```
                M[i] = (Y[i] - Initpose[1]) / (D[i]);
                N[i] = (Z[i] - Initpose[2]) / (D[i]);
            }

            I[0] = (k[0] / D[0]) - (1 / D[1]);
            I[1] = (k[1] / D[1]) - (1 / D[2]);
            I[2] = (k[2] / D[2]) - (1 / D[0]);

            J[0] = (k[0] / D[1]) - (1 / D[0]);
            J[1] = (k[1] / D[2]) - (1 / D[1]);
            J[2] = (k[2] / D[0]) - (1 / D[2]);

            K[0] = L[0] * L[1] + M[0] * M[1] + N[0] * N[1];
            K[1] = L[1] * L[2] + M[1] * M[2] + N[1] * N[2];
            K[2] = L[2] * L[0] + M[2] * M[0] + N[2] * N[0];

            A[0] = L[0] * I[0] + L[1] * J[0];
            A[1] = L[1] * I[1] + L[2] * J[1];
            A[2] = L[2] * I[2] + L[0] * J[2];

            B[0] = M[0] * I[0] + M[1] * J[0];
            B[1] = M[1] * I[1] + M[2] * J[1];
            B[2] = M[2] * I[2] + M[0] * J[2];

            C[0] = N[0] * I[0] + N[1] * J[0];
            C[1] = N[1] * I[1] + N[2] * J[1];
            C[2] = N[2] * I[2] + N[0] * J[2];

            deltaK[0] = k[0] - K[0];
            deltaK[1] = k[1] - K[1];
            deltaK[2] = k[2] - K[2];

            E[0] = A[0] * B[1] - A[1] * B[0];
            E[1] = A[1] * B[2] - A[2] * B[1];
            E[2] = A[2] * B[0] - A[0] * B[2];

            F[0] = A[0] * C[1] - A[1] * C[0];
            F[1] = A[1] * C[2] - A[2] * C[1];
            F[2] = A[2] * C[0] - A[0] * C[2];

            G[0] = B[0] * C[1] - B[1] * C[0];
            G[1] = B[1] * C[2] - B[2] * C[1];
            G[2] = B[2] * C[0] - B[0] * C[2];

            delta = E[0] * C[2] + E[2] * C[1] + E[1] * C[0];
            //Trace.WriteLine("delta : " + delta.ToString());

            cor[0] = (G[1] * deltaK[0] + G[2] * deltaK[1] + G[0] * deltaK[2]) /
(delta);
            cor[1] = (F[1] * deltaK[0] + F[2] * deltaK[1] + F[0] * deltaK[2]) / (-
delta);
            cor[2] = (E[1] * deltaK[0] + E[2] * deltaK[1] + E[0] * deltaK[2]) /
(delta);
            //Funct.PrintVectorDebug(ref cor, "Corrections");

            Initpose[0] = Initpose[0] + cor[0];
            Initpose[1] = Initpose[1] + cor[1];
            Initpose[2] = Initpose[2] + cor[2];

            err = Math.Abs(delta) + Math.Abs(cor[0]) + Math.Abs(cor[1]) + Math.Abs
(cor[2]);

            if (Matrix.Max(Matrix.Abs(cor)) < 1.0e-20)
                break;
        }
        pose[0] = Initpose[0];
```

```
            return pose;
        }
        #endregion

        #region Linear Least Square Adjustment
        public double[] LeastSquareAdjustment(double[] InitPose, double[] io, double[] x, ↙
    double[] y, double[] X, double[] Y, double[] Z, ref double cor, ref double std)
        {
            int ndata = x.Length;
            int numpar = 6;
            int numeqn = ndata * 2;

            double[] EksOr = InitPose;
            double[,] A = new double[numeqn, numpar];
            double[] L = new double[numeqn];
            double[] del = new double[numpar];
            double[] res = new double[numeqn];
            double convergen = 10.0;
            for (int iter = 1; iter <= 25; iter++)
            {
                for (int i = 0; i < ndata; i++)
                {
                    int eqn = i * 2;
                    double[] Lo = FormL(EksOr, io, x[i], y[i], X[i], Y[i], Z[i]);
                    for (int j = 0; j < numpar; j++)
                    {
                        double[,] Ao = FormA(EksOr, io, X[i], Y[i], Z[i]);
                        A[eqn, j] = Ao[0, j];
                        A[eqn + 1, j] = Ao[1, j];
                    }
                    L[eqn] = Lo[0];
                    L[eqn + 1] = Lo[1];
                }
                del = (Matrix.PseudoInverse(A)).Multiply(L);
                EksOr = EksOr.Add(del);
                res = (A.Multiply(del)).Subtract(L);
                std = Funct.SumSquare(res) / (numeqn - numpar);
                cor = Funct.SumAbs(del);
                if (cor < 1.0e-5)
                {
                    break;
                }
                else
                {
                    if (cor < convergen)
                    {
                        convergen = cor;
                    }
                    else
                    {
                        break;
                    }
                }
            }
            return EksOr;
        }
        #endregion
        #endregion

        #region Private Methods
        #region Least Square Functions
        private double[,] FormA(double[] eo, double[] io, double X, double Y, double Z)
        {
            double[,] A = new double[2, 6];
            double xo = io[1]; double yo = io[2]; double f = io[0];
            double om = eo[0]; double ph = eo[1]; double kp = eo[2];
```

```
        double Xo = eo[3]; double Yo = eo[4]; double Zo = eo[5];
        double[,] M = RotationMatrix(om, ph, kp);
        double[] uvw = UVWcoef(eo, X, Y, Z);

        A[0, 0] = f / Math.Pow(uvw[2], 2) * (uvw[0] * (-M[2, 2] * (Y - Yo) + M[2, 1] *
 (Z - Zo)) - uvw[2] * (-M[0, 2] * (Y - Yo) + M[0, 1] * (Z - Zo)));
        A[0, 1] = f / Math.Pow(uvw[2], 2) * (uvw[0] * (Math.Cos(ph) * (X - Xo) + Math.
Sin(om) * Math.Sin(ph) * (Y - Yo) - Math.Cos(om) * Math.Sin(ph) * (Z - Zo)) -
                    uvw[2] * (-Math.Sin(ph) * Math.Cos(kp) * (X - Xo) + Math.Sin(om) *
Math.Cos(ph) * Math.Cos(kp) * (Y - Yo) - Math.Cos(om) * Math.Cos(ph) * Math.Cos(kp) *
(Z - Zo)));
        A[0, 2] = -f / uvw[2] * (M[1, 0] * (X - Xo) + M[1, 1] * (Y - Yo) + M[1, 2] *
(Z - Zo));
        A[0, 3] = -(f / Math.Pow(uvw[2], 2) * (uvw[0] * M[2, 0] - uvw[2] * M[0, 0]));
        A[0, 4] = -(f / Math.Pow(uvw[2], 2) * (uvw[0] * M[2, 1] - uvw[2] * M[0, 1]));
        A[0, 5] = -(f / Math.Pow(uvw[2], 2) * (uvw[0] * M[2, 2] - uvw[2] * M[0, 2]));

        A[1, 0] = f / Math.Pow(uvw[2], 2) * (uvw[1] * (-M[2, 2] * (Y - Yo) + M[2, 1] *
 (Z - Zo)) - uvw[2] * (-M[1, 2] * (Y - Yo) + M[1, 1] * (Z - Zo)));
        A[1, 1] = (f / Math.Pow(uvw[2], 2)) * ((uvw[1] * (Math.Cos(ph) * (X - Xo) +
Math.Sin(om) * Math.Sin(ph) * (Y - Yo) - Math.Cos(om) * Math.Sin(ph) * (Z - Zo))) -
                    (uvw[2] * (Math.Sin(ph) * Math.Sin(kp) * (X - Xo) - Math.Sin(om) *
Math.Cos(ph) * Math.Sin(kp) * (Y - Yo) + Math.Cos(om) * Math.Cos(ph) * Math.Sin(kp) *
(Z - Zo))));
        A[1, 2] = f / uvw[2] * (M[0, 0] * (X - Xo) + M[0, 1] * (Y - Yo) + M[0, 2] * (Z
 - Zo));
        A[1, 3] = -(f / Math.Pow(uvw[2], 2) * (uvw[1] * M[2, 0] - uvw[2] * M[1, 0]));
        A[1, 4] = -(f / Math.Pow(uvw[2], 2) * (uvw[1] * M[2, 1] - uvw[2] * M[1, 1]));
        A[1, 5] = -(f / Math.Pow(uvw[2], 2) * (uvw[1] * M[2, 2] - uvw[2] * M[1, 2]));
        return A;
    }

    private double[] FormL(double[] eo, double[] io, double x, double y, double X,
double Y, double Z)
    {
        double[] F = new double[2];
        double f = io[0]; double xo = io[1]; double yo = io[2];
        double[] uvw = UVWcoef(eo, X, Y, Z);
        F[0] = x - xo + f * (uvw[0] / uvw[2]);
        F[1] = y - yo + f * (uvw[1] / uvw[2]);
        return F;
    }

    private double[] UVWcoef(double[] eo, double X, double Y, double Z)
    {
        double[] uvw = new double[3];
        double[,] M = RotationMatrix(eo[0], eo[1], eo[2]);
        uvw[0] = M[0, 0] * (X - eo[3]) + M[0, 1] * (Y - eo[4]) + M[0, 2] * (Z - eo[5])
;
        uvw[1] = M[1, 0] * (X - eo[3]) + M[1, 1] * (Y - eo[4]) + M[1, 2] * (Z - eo[5])
;
        uvw[2] = M[2, 0] * (X - eo[3]) + M[2, 1] * (Y - eo[4]) + M[2, 2] * (Z - eo[5])
;
        //PrintVectorDebug(ref delta_pose, "delta_pose");
        return uvw;
    }

    private double[,] RotationMatrix(double om, double ph, double kp)
    {
        double[,] M = new double[3, 3];
        M[0, 0] = Math.Cos(ph) * Math.Cos(kp);
        M[0, 1] = Math.Cos(om) * Math.Sin(kp) + Math.Sin(om) * Math.Sin(ph) * Math.Cos
(kp);
        M[0, 2] = Math.Sin(om) * Math.Sin(kp) - Math.Cos(om) * Math.Sin(ph) * Math.Cos
(kp);

        M[1, 0] = -Math.Cos(ph) * Math.Sin(kp);
```