SKRIPSI

PEMBUATAN PROGRAM SELF-CALIBRATING BUNDLE ADJUSTMENT UNTUK JARINGAN PEMOTRETAN KONVERGEN DENGAN BAHASA C#

(Studi kasus : Foto Terrestrial)

Disusun Oleh :

ERNESTO DOS SANTOS

08.25.002

JURUSAN TEKNIK GEODESI

FAKULTAS TEKNIK SIPIL DAN PERENCANAAN

INSTITUT TEKNOLOGI NASIONAL

MALANG

2013

PENGUJIAN PROGRAM SELF-CALIBRATING RUNNUL ADJUSTMENT UNTUK JARINGAN PEMOTRETAN MONUMENTOGEN DENGAN KAMERA CCD

(Studi Kasus : Foto Terrestrial)

Disusun Oleh :

SANDOYO DOS SANTOS

04.25.004

JURUSAN TEKNIK GEODESI

FAKULTAS TEKNIK SIPIL DAN PERENCANAAN

INSTITUT TEKNOLOGI NASIONAL

MALANG

2013

LEMBAR PERSETUJUAN

SKRIPSI

Judul Skripsi:

"Pembuatan Program *Self Calibrating Bundle Adjusment* Menggunakan Software Matlab Untuk Aplikasi Foto Udara UAV"

Diajukan Sebagai Salah Satu Syarat Memperoleh

Gelar Sarjana Teknik Geodesi S1

Institut Teknologi Nasional Malang

Disusun Oleh :

Ernesto Dos Santos
08.25.002

Meyetujui,

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

Mengetahui,

Ketua Jurusan Teknik Geodesi S-1

Ir. Agus Darpono, MT

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
T. BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2 Telp. (0341) 551431 (Hunting), Fax. (0341) 553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341) 417636 Fax. (0341) 417634 Malang

LEMBAR PENGESAHAN

Judul Skripsi:

"Pembuatan Program *Self-Calibrating Bundle Adjustment* untuk Jaringan Pemotretan Konvergen dengan Bahasa C#".

Dipertahankan di hadapan Majelis Penguji Sidang Skripsi Jenjang Starata Satu (S1) pada :
Hari : Jumat
Tanggal : 22 Februari 2013
Dan diterima untuk memenuhi salah satu persyaratan guna memperoleh gelar Sarjana Teknik.

Disusun Oleh :

Ernesto Dos Santos
08.25.002

Panitia Ujian Tugas Akhir

Ketua	Sekretaris
Ir. Agus Darpono, MT	Silvester Sari Sai, ST, MT

Anggota Penguji

Penguji I	Penguji II	Penguji III
Silvester Sari Sai, ST, MT	Heri Purwanto, ST., M.Sc	Ir. D.K. Sunaryo, MS.Tis

JURUSAN TEKNIK GEODESI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
INSTITUT TEKNOLOGI NASIONAL
MALANG
2013

ABSTRAKSI

Proses perhitungan kalibrasi kamera non metrik, memiliki peranan penting dalam fotogrametri untuk melakukan proses transformasi perspective 2D foto ke 3D koordinat *object space*. Sehingga pada tulisan kali ini akan diuraikan sebuah teknik untuk mendapatkan nilai parameter kalibrasi kamera non metrik yaitu menggunakan metode *Self Calibrating Bundle Adjusment*. Untuk mendapatkan nilai parameter kalibrasi diperlukan data pendekatan awal berupa parameter *interior orientation, exterior orientation, object space point* dan koordinat foto, yang akan meminimalisir kesalahan sistematis pada kamera non metrik yang disebabkan tidak stabilnya bidang CCD kamera. Hasil dari proses *Self Calibrating Bundle Adjusment* ini adalah parameter *interior orientation* terkoreksi, parameter *exterior orientation* terkoreksi, parameter *object space point* terkoreksi yang nantinya akan ditularkan ke tiap-tiap foto.

Kata kunci *: Interior Orientation, Exterior Orientation, Object Space Point, Self Calibrating Bundle Adjusment.*

PERNYATAAN KEASLIAN
SKRIPSI

Saya yang bertanda tangan dibawah ini :

Nama	**: Ernesto Dos Santos**
NIM	**: 08.25.002**
Program Studi	**: Teknik Geodesi S-1**
Fakultas	**: Fakultas Teknik Sipil Dan Perencanaan**

Menyatakan dengan sesungguhnya bahwa Skripsi saya dengan judul :

"Pembuatan Program *Self-Calibrating Bundle Adjustment* untuk Jaringan Pemotretan Konvergen dengan Bahasa C#".

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

Malang, 23 Februari 2013
Yang membuat pernyataan

Ernesto Dos Santos
08.25.002

LEMBAR PERSEMBAHAN

Kupersembahkan Skripsi ini

Dengan memanjatkan puji syukur kehadirat Tuhan yang Maha Kuasa yang telah melimpahkan rahmatnya kepada penulis sehingga, dapat terselesaikan penulisan skripsi ini.

Kepada bapak dan ibu Di Kampung Halaman Terima kasih kepada papa dan mamaku atas cinta dan doa tulus yang tak pernah putus sehinga saya dapat menyelesaikan kuliah ku ini.

Kepada saudara-saudaraku, terima kasih telah banyak mengajariku dan membatu ku selama saya kuliah t-a darimu-, pelajaran berharga untuk bertahan dalam segala kondisi.

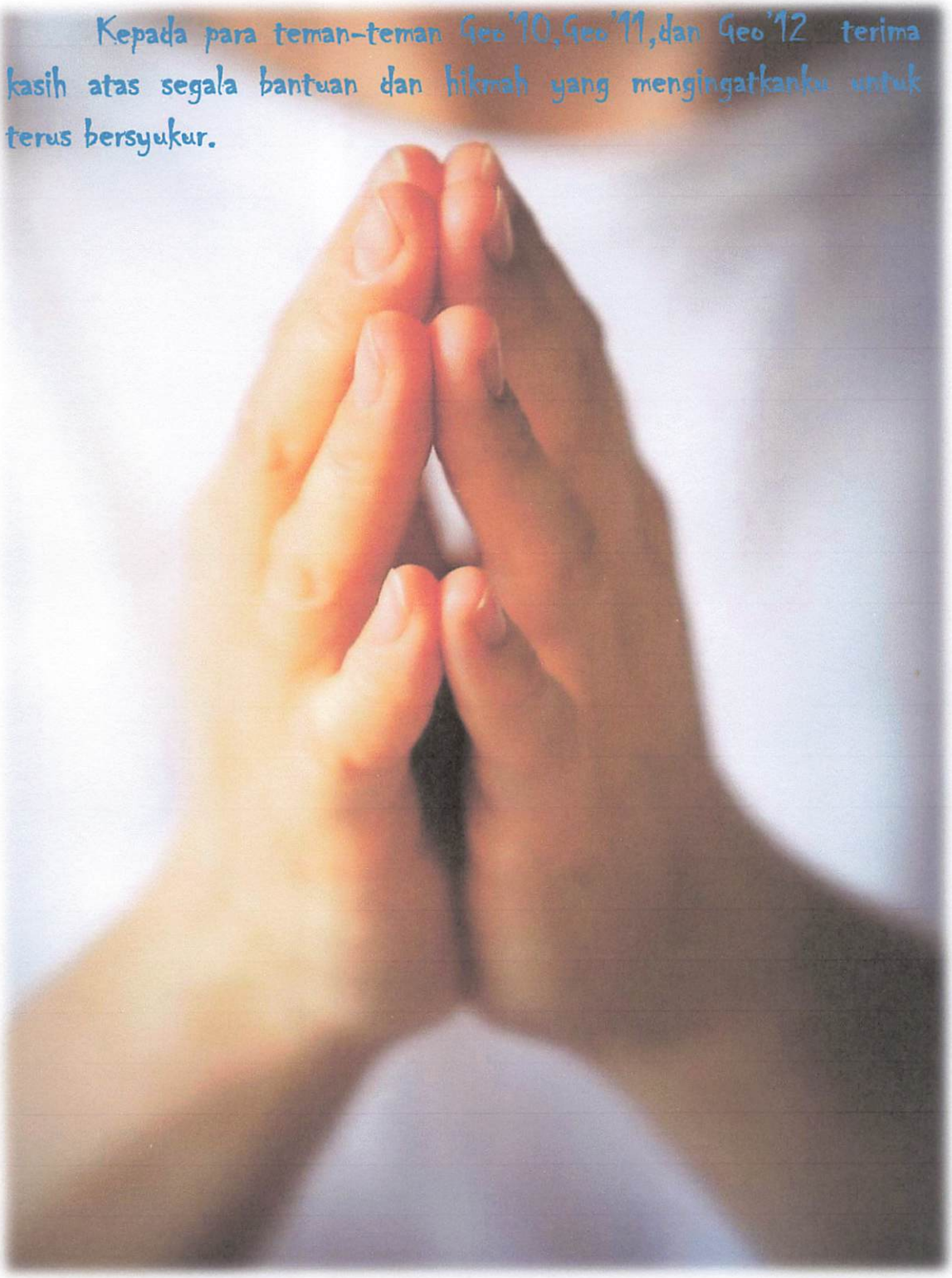
Kepada para teman-teman Geo'08 yang saya tidak bisa sebut namanya satu2, terimakasih atas kerjasamanya, kemauan saling berbagi, dan canda yang membekas di hati dan inspirasi tentang kebaikan dan kegigihan yang membekas kuat di ingatanku.

. Kepada para teman-teman Geo'06, terima kasih atas segala bantuan dan hikmah yang mengingatkanku untuk terus bersyukur.

Kepada para teman-teman Geo'07, terima kasih atas segala bantuan dan hikmah yang mengingatkanku untuk terus bersyukur.

Kepada para teman-teman Geo'09, terima kasih atas segala bantuan dan hikmah yang mengingatkanku untuk terus bersyukur.

Kepada para teman-teman Geo'10, Geo'11, dan Geo'12 terima kasih atas segala bantuan dan hikmah yang mengingatkanku untuk terus bersyukur.



KATA PENGANTAR

Dengan memanjatkan puji syukur kehadirat Tuhan yang Maha Kuasa yang telah melimpahkan rahmatnya kepada penulis sehingga, dapat terselesaikan penulisan ini skripsi dengan judul "**Pembuatan Program *Self-Calibrating Bundle Adjustment* untuk Jaringan Pemotretan Konvergen dengan Bahasa C#.**".

Skripsi ini disusun untuk memenuhi salah satu syarat memperoleh gelas 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 Ir. Soeparno Djwo, MT 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 Ir. Agus Darpono, MT. selaku Ketua Jurusan Teknik Geodesi Institut Teknologi Nasional Malang.
4. Bapak Dr. Edwin Tjahjadi, ST., MGeom.Sc serta sebagai Dosen Pembimbing I.
5. Bapak. Heri Purwanto, ST., M.Sc. selaku Dosen Pembimbing II.
6. Bapak Silvester Sari Sai, ST., MT. selaku Dosen Penguji.
7. Bapak Ir. D.K. Sunaryo, MS. Tis. selaku Dosen Penguji.
8. Segenap dosen, staff pengajar dan rekording Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.
9. Rekan-rekan Mahasiswa/i dan alumni Teknik Geodesi.

10. 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, 22 Februari 2013

Penulis

DAFTAR ISI

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1 Latar Belakang

Self-Calibrating Bundle adjustment merupakan salah satu bagian penting dalam *photogrammetry* untuk mengkalibrasi kamera yang digunakan dalam pemotretan(*Yilmazturk.,2011*). Dalam beberapa tahun terakhir hampir semuanya menggunakan kamera digital *non-metric* dalam proses *close-range photogrammetry* oleh karena itu untuk mendapatkan tingkat keakurasian yang tinggi maka kamera *digital Non-metric* yang diggunakan harus dikalibrasi(*Yilmazturk.,2011*). Perhitungan *Self-Calibrating Bundle Adjustment* dalam *Photogrammetry* tidak terlepas dari dari proses *Relaive Orientation, resection* dan *Intersection*(*Mikhail et al., 2001*). Relatif Orientasi merupakan proses untuk menentukan nilai perputaran sudut rotasi dan pergeseran posisi antara dua foto(*wolf and dewitt, 2000*). proses *Resection* merupakan proses penentuan posisi dan orientasi luar dari tiap foto (*wolf and dewitt, 2000*). Proses *Intersection* merupakan teknik untuk menentukan koordinat titik-titik objek pada dua buah foto atau lebih yang saling bertampalan sehingga dapat diketahui posisi secara 3D (*Mikhail et al., 2001*). Sehingga keberadaan proses *Self-calibrating bundle adjustment* sangat dibutuhkan untuk memperoleh hasil *Xi, Yi, Zi* yang akurat dan parameter orientasi kamera yang teliti. Melihat peranan penting dari *bundle adjustment* dalam mendukung proses *photogrammetry*, maka penelitian ini

bermaksud untuk menghitung *Self-Calibrating Bundle Adjustment* untuk jaringan pemotretan konvergen dengan bahasa pemograman Visual Studio C# 2010.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas dalam skripsi ini adalah bagaimana merancang algoritma dan *program C# Self-Calibrating bundle adjustment* dalam *photogrammetry.*

1.3 Batasan Masalah

Batasan masalah penelitian ini yaitu mendesain program *Self-Calibrating bundle adjustment* dan parameter EO pendekatan, parameter kalibrasi dan parameter IO telah diketahui dari perangkat lunak fotogrametri seperti Australis dan Photomodeler, parameter x_0, y_0, dan c dianggap tetap untuk semua foto.

1.4 Manfaat Penelitian

Adapun manfaat yang diharapkan dalam penelitian ini adalah :

1. Memperoleh program *Self-Calibrating bundle adjustment* berupa program console dalam Visual C# yang kompatibel pada windows.
2. Dari hasil penelitian ini diharapkan dapat memberikan kontribusi terhadap ilmu pengtahuan khususnya yang berkaitan dengan *Self-Calibrating Bundle adjustment.*

1.5 Tujuan Penelitian

Maksud dan tujuan dari penulisan skripsi ini adalah membuat program dengan pengembangan perangkat lunak *visual studio C#* dalam bentuk *console application*

untuk proses perhitungan *Self-Calibrating bundle adjustment* dalam metode *close range photogrammetry,* untuk jaringan pemotretan multi foto jaringan konvergen.

1.6 Tinjauan Pustaka

Beberapa tinjauan pustaka telah dilakukan dalam menyusun penelitian, guna mengumpulkan informasi mengenai proses *self-calibrating bundle adjustment* dalam *close range photogrammetry* yang didasarkan atas berbagai riset oleh para ilmuan dalam bidang *close range photogrammetry* antara lain :

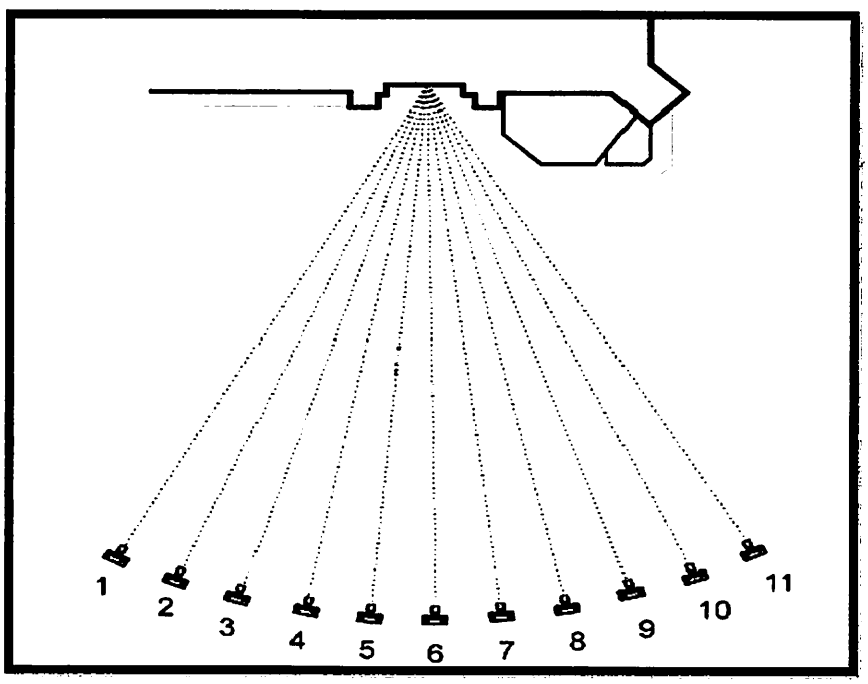
Menurut (*Fraser, Shortis, Ganci, 1995*) *Self-calibrating Bundle adjustment* sudah diperkenalkan dalam fotogrametry sejak dua decade lalu sebagai salah satu metode yang sangat berguna dan sangat teliti untuk melakukan proses transformasi perspective 2D foto ke 3D koordinat *object space*.

Dalam *Close-range photogrammetry* sekarang hampir semuanya menggunakan kamera non-metrik oleh karena itu kamera harus dimodelkan dan dikalibrasi untuk menjamin keakurasiannya supaya bisa digunakan dalam aplikasi-aplikasi *photogrammetric*. Sebuah kamera dapat dikalibrasi apabila *principal distance* (c), *principal point offset* (x_0,y_0), parameter dirtorsi lensa(K_1, K_2, K_3, P_1, P_2, b_1, b_2) diketahui (*Yilmaztürk,2011*).

BAB II

DASAR TEORI

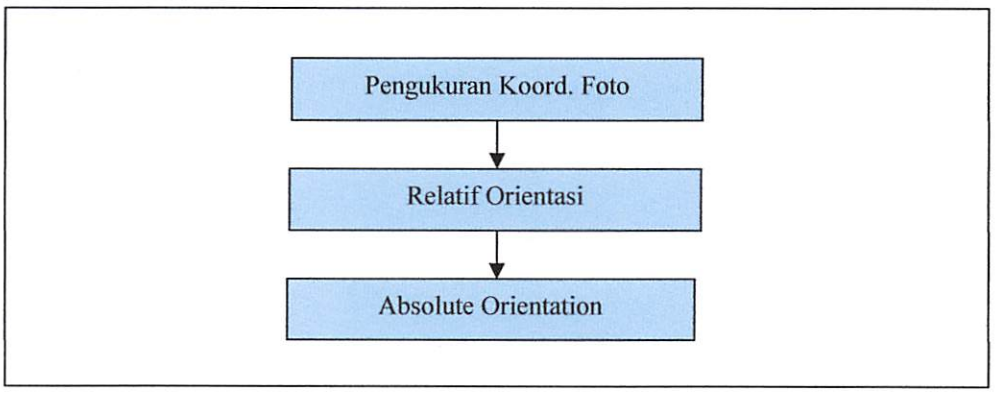
Fotogrametri dengan jaringan pemotretan konvergen memiliki beberapa keunggulan dibandingkan geometri pemotretan normal. Salah satunya adalah penurunan jumlah gambar yang diperlukan untuk mencover seluruh permukaan objek. Keuntungan ini menyederhanakan desain jaringan dan meningkatkan secara teoritis keakuratan metode dan dapat diandalkan (*Mason, 1994*).



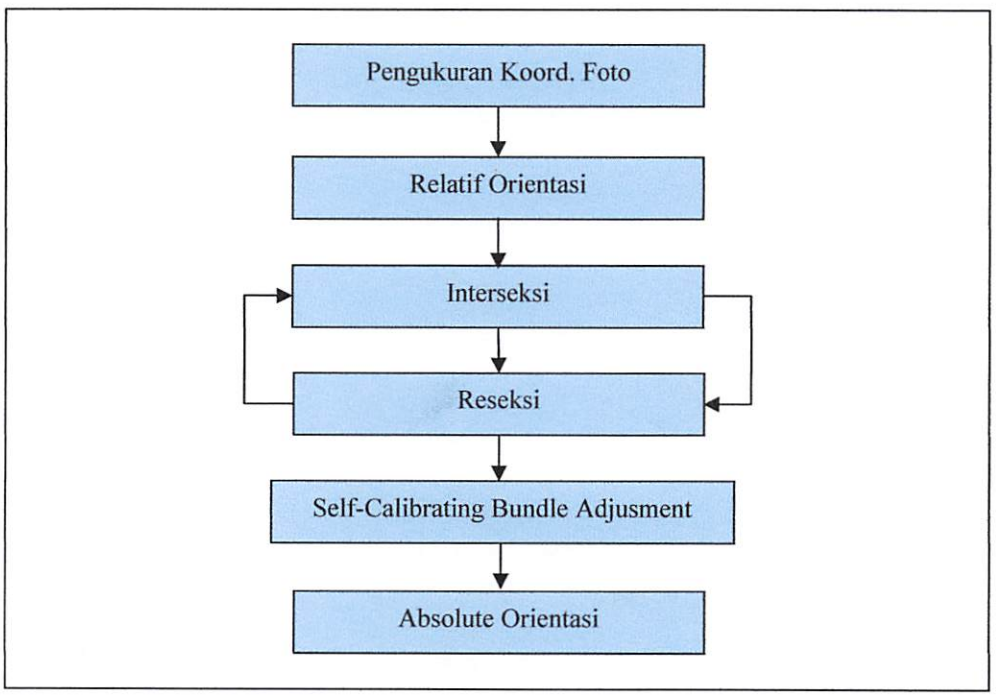
Gambar. 2.1 Contoh konfigurasi jaringan pemotretan konvergen(J. García-León et al, 2007)

Dari sisi metode perhitungan, cara analitik merupakan cara yang sangat populer di dalam fotogrametri. Hampir keseluruhan proses fotogrametri diselesaikan

dengan metode analitik seperti proses relatif orientasi, reseksi, interseksi, bundle adjustment dan yang terakhir absolute orientasi. Proses-proses fotogrametri tersebut dapat disusun dalam sebuah skema sebagai berikut (*Fraser, 2006a*) :



Gambar 2.2 Skema proses fotogrametri analog



Gambar 2.3 Skema proses fotogrametri analitik

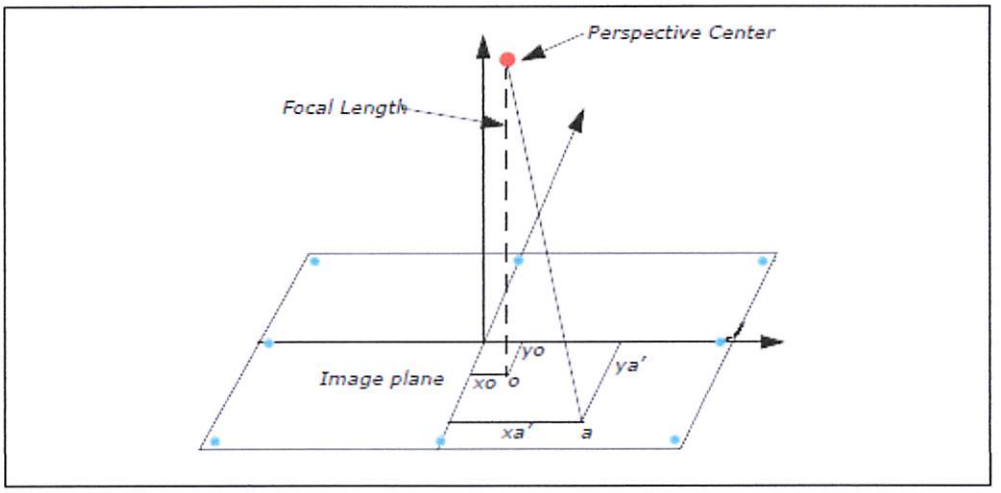
Seluruh rangkaian proses pada fotogrametri analitik seperti yang digambarkan pada *gambar 2.2* dapat diselesaikan secara langsung menggunakan metode perataan kuadrat terkecil (*least square adjustment*). Metode kuadrat terkecil biasanya digunakan untuk memecahkan masalah penentuan enam parameter ekstrinsik $(X_L, Y_L, Z_L, \omega, \varphi, \kappa)$ pada proses reseksi dan tiga parameter koordinat objek (X, Y, Z) dalam ruang tiga-dimensi pada proses interseksi.

2.1 Parameter Kamera

Dalam fotogrametri maupun komputer vision, terdapat dua parameter penting yang digunakan dalam berbagai persamaan untuk merekonstruksi objek tiga-dimensi menggunakan fotografi (*Geosystem, 2006a;Trucco & Verri, 1998*). Parameter tersebut terdiri dari parameter interinsik (*Interior Orientation*) dan parameter ekstrinsik (*Exterior Orientation*). Parameter interinsik biasanya dipakai sebagai parameter untuk mendefinisikan nilai geometri dari kamera, sedangkan parameter ekstrinsik digunakan untuk menjelaskan hubungan secara geometri posisi kamera dan objek dalam suatu sistem koordinat referensi.

2.1.1 Parameter Interinsik (*Interior Orientation*)

Parameter interinsik merupakan parameter yang mendefinisikan geometri dari sensor kamera pada saat melakukan pengambilan foto. Parameter insterinsik kamera ini terdiri dari tiga parameter pokok yaitu c atau f yang merupakan panjang fokus dan dua parameter *principle point* (x_0, y_0).



Gambar 2.4 Interior orientasi kamera (Geosystem, 2006a)

Parameter *principle point* secara matematika dapat didefinisikan sebagai perpotongan garis tegak lurus yang melalui *perspective center* ke bidang foto. Panjang dari *principle point* ke *perspective center* disebut sebagai panjang fokus (*Wang, 1990*).

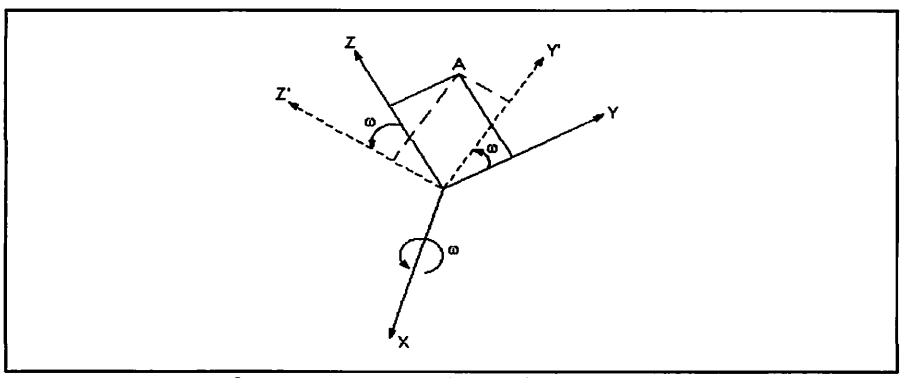
Pada umumnya parameter ini juga dapat dikatakan sebagai parameter transformasi, dimana pusat sistem koordinat terdapat pada *principle point (Geosystem, 2006a)*. Persamaan transformasi ini dapat dituliskan dalam persamaan matematika sebagai berikut (*Fraser, 2006b*) :

$$\begin{vmatrix} x - x_0 \\ y - y_0 \\ -f \end{vmatrix} = \begin{vmatrix} 1 & 0 & -x_0 \\ 0 & 1 & -y_0 \\ 0 & 0 & -f \end{vmatrix} \begin{vmatrix} x \\ y \\ 1 \end{vmatrix} \quad \text{.....(2.1)}$$

2.1.2 Parameter Ekstrinsik (*Exterior Orientation*)

Parameter ekstrinsik atau *exterior orientation* merupakan parameter posisi dan orientasi sudut kamera pada saat pengambilan foto. Parameter posisi kamera didefinisikan dalam ruang tiga-dimensi yang merupakan posisi dari *perspektif center* (pusat kamera) X_o, Y_o, Z_o X_0, Y_0, Z_0. Sedangkan parameter orientasi sudut berfungsi sebagai penghubung antara sistem koordinat kamera (x, y, z) dengan sistem koordinat referensi (X, Y, Z). Sesuai dengan (*Elias, 2007;Fraser, 2006b;Mikhail et al., 2001;Shih, 1994;Wolf & Dewitt, 2000*) orientasi sudut dapat didefinisikan dalam sistem rotasi ω, φ, κ (*omega, phi, kappa*) atau t, α, s (*tilt, azimuth, swing*).t, α, s

Dari masing-masing sistem rotasi, dapat dibangun sebuah matrik rotasi yang digunakan untuk menghubungkan kedua sistem koordinat. Sebagai contoh, akan dijelaskan proses penurunan sebuah persamaan pada matrik rotasi dalam sudut rotasi *omega* untuk rotasi sumbu *x*, *phi* untuk rotasi sumbu *y* dan *kappa* sebagai rotasi sumbu *z*. Dari ketiga sudut diatas didefinisikan dalam sistem rotasi tangan kanan, dimana apabila berlawanan dengan putaran arah jarum jam akan bernilai negatif dan sebaliknya.



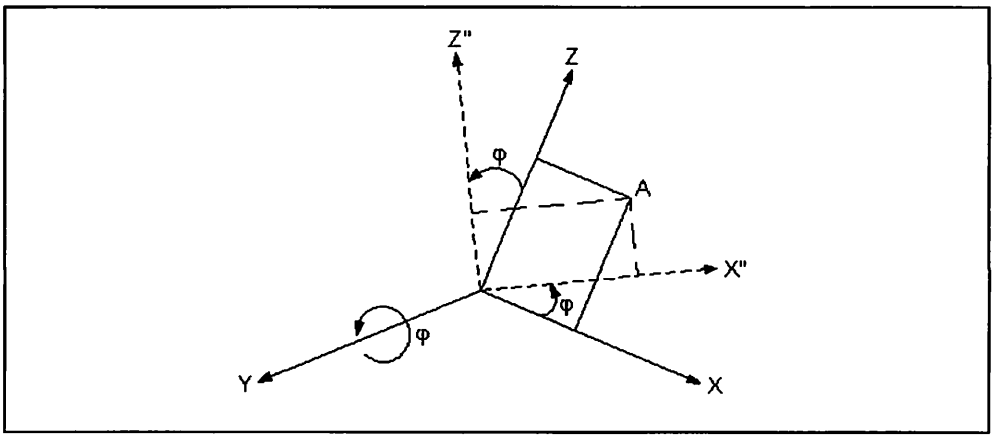
Gambar 2.5 Rotasi pada sumbu x sebesar ω

Dari gambar diatas, sumbu *x* positif diputar searah jarum jam, sehingga posisi titik *A* dalam sistem koordinat yang terotasi dapat ditulis dalam sebuah persamaan.

$$\begin{vmatrix} X' \\ Y' \\ Z' \end{vmatrix} = R_\omega \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \quad \text{.....(2.2)}$$

Dimana parameter R_ω didefinisikan sebagai :

$$R_\omega = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{vmatrix} \quad \text{..... (2.3)}$$



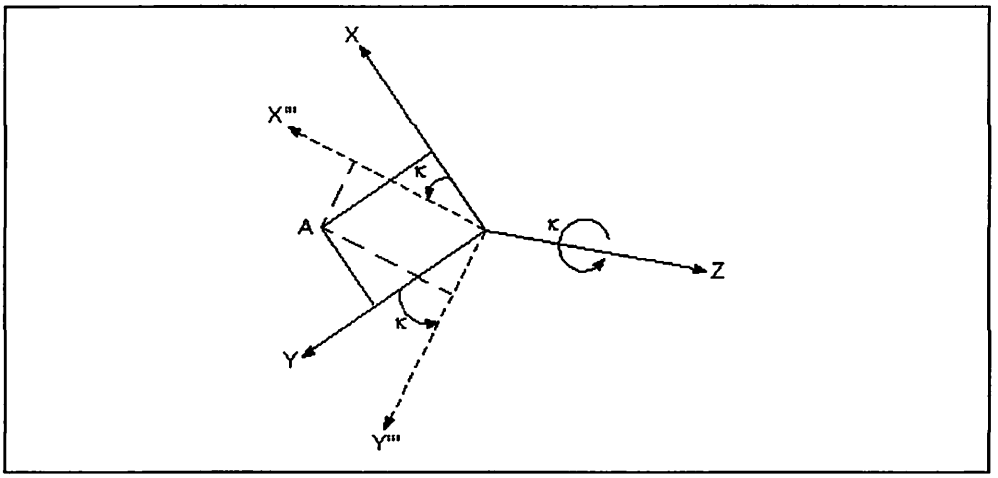
Gambar 2.6 Rotasi sumbu y sebesar φ

Selanjutnya, dilakukan rotasi terhadap sumbu *y* positif dengan arah rotasi positif searah dengan jarum jam seperti pada *gambar 2.7*. Dari hasil rotasi didapat nilai posisi titik *A* dalam sistem koordinat terotasi sebagai berikut.

$$\begin{vmatrix} X'' \\ Y'' \\ Z'' \end{vmatrix} = R_\varphi \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \quad \text{..... (2.4)}$$

Dengan parameter R_φ didapat dari persamaan :

$$R_\varphi = \begin{vmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{vmatrix} \quad \ldots\ldots (2.5)$$



Gambar 2.7 Rotasi sumbu z sebesar κ

Dan yang terakhir, sumbu positif z dirotasi positif searah jarum jam, sebagaimana diilustrasikan pada *gambar 2.9* diatas. Sehingga posisi titik *A* pada sistem rotasi sumbu z dinyatakan dalam sebuah persamaan sebagai berikut :

$$\begin{vmatrix} X''' \\ Y''' \\ Z''' \end{vmatrix} = R_\kappa \begin{vmatrix} X \\ Y \\ Z \end{vmatrix} \quad \ldots\ldots (2.6)$$

Dimana parameter R_κ didefinisikan sebagai :

$$R_\kappa = \begin{vmatrix} \cos\kappa & \sin\kappa & 0 \\ -\sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{vmatrix} \quad \ldots\ldots (2.7)$$

Dari perkalian $R_\omega R_\varphi R_\kappa$ yang merupakan matrik rotasi dari parameter ω, φ dan κ didapat nilai matrik rotasi secara utuh sebagai berikut (*Cooper & Robson, 2001*) :

$$R_{\omega\varphi\kappa} = \begin{vmatrix} \cos\varphi\cos\kappa & \sin\omega\sin\varphi\cos\kappa + \cos\omega\sin\kappa & -\cos\omega\sin\varphi\cos\kappa + \sin\omega\sin\kappa \\ -\cos\varphi\sin\kappa & -\sin\omega\sin\varphi\sin\kappa + \cos\omega\cos\kappa & \cos\omega\sin\varphi\sin\kappa + \sin\omega\cos\kappa \\ \sin\varphi & -\sin\omega\cos\varphi & \cos\omega\cos\varphi \end{vmatrix} \quad ...(2.8)$$

atau

$$R = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{vmatrix} \quad .....(2.9)$$

Dan apabila nilai matrik rotasi R telah diketahui, parameter sudut rotasi berupa ω, φ dan κ dapat ditentukan pula dengan menggunakan hubungan sebagai berikut (*Slabaugh, 2004;Wolf, 1993*) :

$$\sin\varphi = m_{31} \;\; ; \;\; \tan\omega = {}^{-m_{32}}\!/_{m_{33}} \;\; ; \;\; \tan\kappa = {}^{-m_{21}}\!/_{m_{11}} \quad .....(2.10)$$

Pada dasarnya, matrik rotasi merupakan matrik ortogonal. Kondisi ini dapat dibuktikan dengan melakukan perkalian antara matrik tersebut dengan nilai transposnya sehingga akan menghasilkan sebuah matrik identitas (*Stefanovic, 1973;Thompson, 1959*) :

$$R^T R = RR^T = I \quad .....(2.11)$$

Kondisi keortogonalan matrik rotasi dapat dibuktikan juga dengan cara melakukan *invers* terhadap matrik tersebut. Apabila nilai *invers* dari matrik tersebut sama dengan nilai transposnya, maka matrik tersebut dapat dikatakan ortogonal sebagaimana yang dikemukakan oleh *Cooper & Robson, 2001* sebagai berikut :

$$R^{-1} = R^T \quad \text{.....(2.12)}$$

Matrik rotasi dapat pula dibangun dengan menggunakan sistem rotasi t, α, s (*tilt, azimuth, swing*) sebagaimana yang telah dijelaskan diatas dengan menggunakan persamaan sebagai berikut :

$$R = \begin{vmatrix} -\cos\alpha\cos s - \sin\alpha\cos t\sin s & \sin\alpha\cos s - \cos\alpha\cos t\sin s & -\sin t\sin s \\ \cos\alpha\sin s - \sin\alpha\cos t\cos s & -\sin\alpha\sin s - \cos\alpha\cos t\cos s & -\sin t\cos s \\ -\sin\alpha\sin t & -\cos\alpha\sin t & \cos t \end{vmatrix} \quad \text{.....(2.13)}$$

Atau dengan menggunakan tiga parameter independent yaitu a, b dan c sebagai element untuk membentuk matrik rotasi *Rodrigues* sebagai berikut (*Faig, 1984*) :

$$R = \Delta^{-1} \begin{vmatrix} 1 + \frac{1}{4}(a^2 - b^2 - c^2) & -c + \frac{1}{2}ab & b + \frac{1}{2}ac \\ c + \frac{1}{2}ba & 1 + \frac{1}{4}(-a^2 + b^2 - c^2) & -a + \frac{1}{2}bc \\ -b + \frac{1}{2}ca & a + \frac{1}{2}cb & 1 + \frac{1}{4}(-a^2 - b^2 + c^2) \end{vmatrix} \quad \text{.....(2.14)}$$

Dimana $\Delta = 1 + \frac{1}{4}(a^2 + b^2 + c^2)$.

Masih terdapat dua persamaan lainnya untuk membentuk suatu matrik rotasi yaitu matrik *Pope-Hinsken* yang telah di apalikasikan oleh *Zheng & Wang, (1992)* dalam proses reseksi dan matrik *Quaternion* yang pertama kali dikenalkan oleh *Hamilton (1983)* dan telah banyak digunakan dalam aplikasi komputer vision maupun fotogrametri.

2.2 Persamaan Umum Fotogrametri

Dalam fotogrametri dikenal dua persamaan yaitu persamaan kolinear dan persamaan koplanar. Menurut (*Fraser, 2006d*), kedua persamaan tersebut diadopsi dari sebuah persamaan dasar transformasi similariti tiga-dimensi sebagai berikut :

$$x = \lambda R(X - X_c) \quad \text{.....(2.15)}$$

Apabila dijabarkan dalam bentuk matrik, persamaan diatas dapat ditulis sebagai berikut :

$$\begin{vmatrix} x \\ y \\ -f \end{vmatrix} = \lambda R \begin{vmatrix} X_i - X_c \\ Y_i - Y_c \\ Z_i - Z_c \end{vmatrix} \quad \text{.....(2.16)}$$

Dimana :

x, y : *Koordinat foto.*

f : *Panjang fokus kamera.*

λ : *Parameter skala.*

R : *Parameter matrik rotasi dengan dimensi 3x3.*

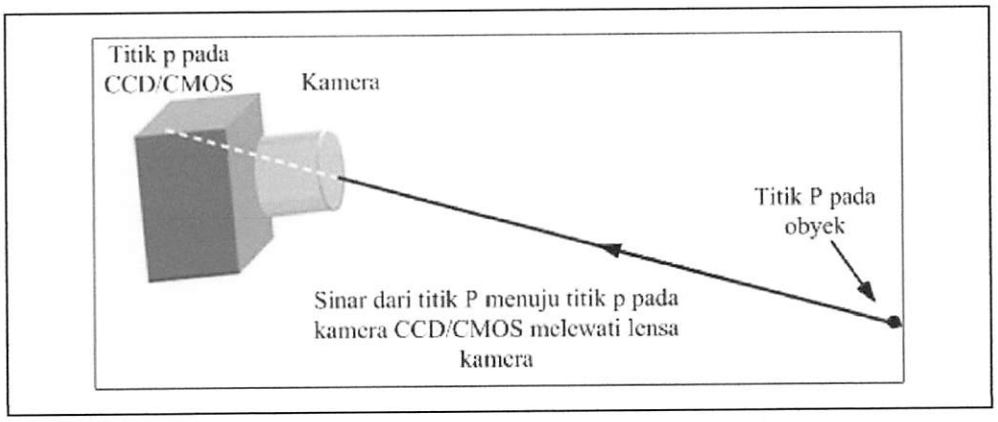
X_i, Y_i, Z_i : *parameter koordinat objek dalam ruang tiga-dimensi.*

X_c, Y_c, Z_c : *Parameter posisi kamera pada saat pengambilan gambar objek.*

Kedua persamaan tersebut dapat digunakan dalam berbagai kasus fotogrametri. Akan tetapi, penggunaannya disesuaikan terhadap parameter yang akan dicari dan data observasi yang dimiliki. Untuk lebih jelasnya, penggunaan kedua persamaan tersebut akan dijelaskan secara rinci pada tiap sub-bab dibawah ini.

2.2.1 Persamaan Kolinear *(Collinearity Equation)*

Kebersamaan garis (kolinear) merupakan kondisi kedudukan titik pemotretan, titik objek dan gambaran titik pada foto seluruhnya terletak pada suatu garis lurus (*Fraser, 2006a;Mikhail et al., 2001;Wolf & Dewitt, 2000*). Kedudukan dalam kebersamaan garis tersebut dapat dilihat pada *gambar 2.10* dibawah ini.



Gambar 2.8 Kondisi kesegarisan atau kolinear (Tjahjadi, 2009)

Dua buah persamaan menyatakan kondisi kebersamaan garis bagi sembarang titik diatas foto udara, diantaranya sebuah persamaan bagi koordinat foto x dan lainnya untuk koordinat foto y. Asal persamaan tersebut ialah *persamaan 2.16* dan dijabarkan sebagai berikut (*Geosystem, 2006b;Karara, 1989*) :

$$x_a - x_0 = -f\left|\frac{m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)}\right| \quad \text{.....(2.17)}$$

$$y_a - y_0 = -f\left|\frac{m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L)}{m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)}\right| \quad \text{.....(2.18)}$$

Dalam hal ini :

x_a, y_a : *Koordinat foto dari gambaran satu objek.*

x_0, y_0 : *Pusat sistem koordinat foto (Principle point).*

X_A, Y_A, Z_A : *Koordinat titik objek pada ruang tiga-dimensi.*

X_L, Y_L, Z_L : *Koordinat posisi kamera pada saat melakukan pemotretan.*

f : *Panjang fokus kamera .*

$m_{11}, m_{12}, \ldots, m_{33}$: *Elemen matrik rotasi .*

Persamaan kolinear diatas bukanlah persamaan linear, sehingga persamaan tersebut harus dilinearisasi terlebih dahulu menggunakan teorema *Taylor* (*Wolf, 1980*). Pada persamaan diatas, apabila telah mengalami proses linearisasi akan berbentuk sebagai berikut :

$$v_x + J = b_{11}d\omega + b_{12}d\varphi + b_{13}d\kappa - b_{14}dX_L - b_{15}dY_L - b_{16}dZ_L + b_{14}dX_A + b_{15}dY_A + b_{16}dZ_A \quad ..(2.19)$$

$$v_x + K = b_{11}d\omega + b_{12}d\varphi + b_{13}d\kappa - b_{14}dX_L - b_{15}dY_L - b_{16}dZ_L + b_{14}dX_A + b_{15}dY_A + b_{16}dZ_A \quad .....(2.20)$$

Dari *persamaan 2.19* dan *2.20* diatas :

v_x, v_y : *Kesalahan residual dalam koordinat foto yang terukur.*

$d\omega, d\varphi, d\kappa$: *Koreksi nilai pendekatan awal bagi parameter rotasi.*

dX_L, dY_L, dZ_L : *Koreksi terhadap nilai awal bagi koordinat titik pemotretan.*

dX_A, dY_A, dZ_A : *Koreksi terhadap nilai awal bagi koordinat ruang titik objek.*

Sedangkan nilai koefisien $b_{11}, b_{12}, \cdots\cdots, b_{25}, b_{26}$ dapat ditentukan menggunakan persamaan sebagai berikut :

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

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

$$\cos\varphi\cos\kappa\Delta Y - \cos\omega\cos\varphi\cos\kappa\Delta Z)$$

$$b_{13} = \frac{-f}{q}\left(m_{12}\Delta X + m_{22}\Delta Y + m_{23}\Delta Z\right)$$

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

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

$$b_{16} = \frac{f}{q^2}(rm_{32} - qm_{23}) \quad \text{.....(2.21)}$$

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

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

$$\cos\varphi\sin\kappa\Delta Y - \cos\omega\cos\varphi\sin\kappa\Delta Z)$$

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

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

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

$$b_{26} = \frac{f}{q^2}(sm_{32} - qm_{23})$$

Dengan parameter r, s, q sebagai berikut :

$$r = m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L) \quad \text{.....(2.22)}$$

$$s = m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L) \quad \text{.....(2.23)}$$

$$q = m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L) \quad \text{.....(2.24)}$$

Dan koefisien J, K yang merupakan parameter observasi pada *persamaan 2.19* dan *2.20* dapat ditentukan dengan menggunakan persamaan sebagai berikut :

$$J = x_a - x_0 + f\frac{r}{q} \quad \text{.....(2.25)}$$

$$K = y_a - y_0 + f\frac{s}{q} \quad \text{.....(2.26)}$$

Dimana :

x_a, y_a : *Koordinat foto dari gambaran satu objek.*

x_a, y_a : *Pusat sistem koordinat foto (Principle point).*

f : *Panjang fokus kamera .*

r, s, q : *Nilai dari persamaan 2.22-2.24.*

Keseluruhan persamaan diatas dapat digunakan dalam proses iterasi untuk menentukan nilai koreksi tiap parameter pendekatan awal. Proses iterasi tersebut akan dihentikan apabila besarnya nilai koreksi dapat diabaikan.

2.2.2 Sistem Persamaan *Least Square Adjustment*

Untuk menerapkan persamaan (2.19) dan (2.20) dalam bentuk sederhana dari persamaan kolinearitas, pendekatan aljabar dalam bentuk matriks dan dituliskan sebagai berikut :

$$_mV_1 = {}_mA_n \, {}_nX_1 - {}_mL_1 \quad \text{.....(2.27)}$$

Dimana m adalah jumlah persamaan, n adalah jumlah unsur yang tidak diketahui, V merupakan matriks kesalahan residual dalam koordinat foto x dan y terukur, A ialah matriks b, koefisien unsur yang tidak diketahui, X ialah matriks koreksi unsur tidak diketahui untuk perkiraan awal, dan L ialah matriks tetapan J dan K.

Dengan memperoleh penyajian matrik penyelesaian kuadrat terkecil bagi unsur yang tidak diketahui, akan terlihat bahwa persamaan normal dapat diperoleh dengan sebagai berikut :

$A^TAX = A^TL$ ... . *(2.28)*

Menjadi

$X = (A^TA)^{-1} A^TL$ ..... *(2.29)*

Bagi suatu sistem pengamatan berbobot :

$X= (A^TPA)^{-1} A^TPL$ ..... *(2.30)*

Didalam persamaan matriks identik terhadap persamaan bobot, kecuali bahwa matriks *P* merupakan matriks diagonal bobot. Melakukan proses iterasi apabila nilai residu belum sesuai. Adapun persamaan matrik untuk menghitung nilai residu setelah penyesuaian, sebagai berikut :

$V = AX - L$ .....*(2.31)*

Rumus standar deviasinya adalah :

$So=\frac{\sqrt{V^TV}}{r}$ .....*(2.32)*

Iterasi berhenti apabila besarnya nilai koreksi parameter dicari (*unknown*) yang didapat paling kecil.

2.3 Ekstraksi Data Koordinat Foto

Suatu foto dalam format dijital merupakan kuantitas nilai-nilai tingkat keabuan (*greyscale*) yang ditampilkan dalam sebuah susunan matriks atau *array*, dimana nilai baris dan kolom dari matriks tersebut merupakan koordinat piksel. Dengan kelebihan yang dimiliki oleh foto dalam format digital ini, maka dapat dengan mudah menentukan nilai suatu koordinat objek dalam suatu sistem koordinat

foto. Secara umum metode penetuaan nilai koordinat objek pada foto dijital yang sering digunakan dalam prsoses fotogrametri antara lain sebagai berikut.

2.3.1 Metode *Centroid*

Dalam penentuan koordinat foto menggunakan teknik ini, hanya terbatas pada target atau objek yang berbentuk lingkaran atau elips. Hal ini dikarenakan, pada metode ini bertujuan menghitung nilai tengah (*center of gravity*) atau pusat *centroid* pada sebuah target dengan memperhitungkan nilai tingkat keabuan pada tiap piksel. Seperti yang dikemukakan oleh *Ganci* dan *Shortis* dalam *Ahmad*, metode ini dapat digunakan untuk menentukan sentroid hingga ketepatan ±0,03 piksel.

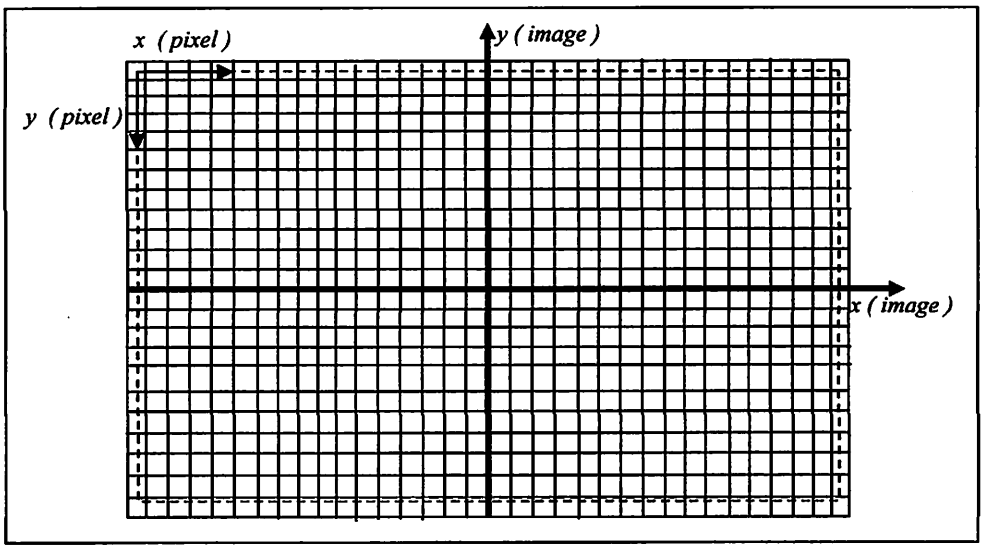
Untuk lebih jelasnya persamaan penentuan *centroid* secara umum dijabarkan sebagai berikut (*Shortis, et al, 1994*):

$$x=\frac{\sum_{j=1}^{n}\sum_{i=1}^{m} j \times l_{ij}}{\sum_{j=1}^{n}\sum_{i=1}^{m} l_{ij}} ; y=\frac{\sum_{j=1}^{n}\sum_{i=1}^{m} i \times l_{ij}}{\sum_{j=1}^{n}\sum_{i=1}^{m} l_{ij}} \quad \text{..... (2.33)}$$

Dimana l_{ij} merupakan nilai tingkat keabuan (*greyscale*) piksel dalam arah x dan y, m jumlah kolom, n jumlah baris dan masinng-masing i dan j adalah nilai baris dan kolom piksel.

2.3.2 Konversi Koordinat Piksel Ke Foto

Pada kamera dijital sistem koordinat yang dipakai adalah sistem koordinat piksel, sedangkan dalam proses perhitungan secara analitik, sistem yang dipakai adalah sistem koordinat kartesian (metrik). Sehingga dalam hal ini harus dilakukan transformasi koordinat dari sistem piksel kedalam sistem kartesian foto. Adapun persamaan yang digunakan adalah (*Australis, User Manual, 2004*).



Gambar 2.9 Sistem koordinat piksel dan sistem koordinat foto

$$x = (x' - x_c') \times x_{PixelSize}$$

$$y = (y_c' - y') \times y_{PixelSize} \quad \text{..... (2.34)}$$

Dimana :

$$x_c' = \left(\frac{nx'}{2}\right) - 0{,}5$$

$$y_c' = \left(\frac{ny'}{2}\right) - 0{,}5 \quad \text{..... (2.35)}$$

Dalam hal ini x, y merupakan koordinat foto dalam sistem koordinat metrik, x', y' koordinat dalam piksel, xc, yc, *principle point* dalam piksel, $x_{PixelSize}$, $y_{PixelSize}$ ukuran satu piksel dalam metrik dan nx', ny' merupakan resolusi dari foto dalam piksel.

2.4 Kalibrasi Kamera

Fotogrametri merupakan teknik untuk memperoleh informasi mengenai posisi, ukuran dan bentuk objek melalui suatu pengukuran foto sebagai pengganti pengukuran yang dilakukan secara langsung (*Atkinson, 2001*). Istilah fotogrametri berasal dari kata *photos* (sinar), *gramma* (sesuatu yang tergambar) dan *metron*

(mengukur). Secara sederhana maka fotogrametri dapat diartikan sebagai pengukuran secara grafis dengan menggunakan sinar. Dari definisi tersebut dapat dimengerti bahwa fotogrametri meliputi (*Wolf dan Dewitt, 2000*) :

- Perekaman objek (pemotretan).
- Pengukuran gambar objek pada foto udara.
- Pemotretan hasil ukuran untuk dijadikan bentuk yang bermanfaat (peta).

Salah satu karakteristik fotogrametri adalah pengukuran terhadap objek yang dilakukan tanpa berhubungan, perlu berhubungan, ataupun bersentuhan secara langsung dengannya. Pengukuran terhadap objek tersebut dilakukan melalui data yang diperoleh pada sistem sensor yang digunakan.

Dalam bidang fotogrametri, terdapat proses kalibrasi kamera yang perlu dilakukan sebelum kamera tersebut digunakan. Menurut *Fraser* (2000), kalibrasi kamera dilakukan untuk menentukan parameter internal kamera (IO) yang meliputi *principle distance* (f), titik pusat fidusial foto (x_0, y_0), distorsi lensa (K_1, K_2, K_3, P_1 dan P_2), serta distorsi akibat perbedaan penyekalaan dan ketidak ortogonal antara sumbu X dan Y (b_1, b_2).

Kalibrasi kamera dapat dilakukan dengan berbagai metode. Secara umum kalibrasi kamera biasa dilakukan dengan tiga metode, yaitu *laboratory calibration, on the job calibration* dan *self calibration* (*Atkinson, 1987*). *Laboratory calibration* dilakukan di laboratorium, terpisah dengan proses pemotretan objek. Metode yang termasuk di dalamnya antara lain *optical laboratory* dan *test range calibration.* Secara umum metode ini sesuai untuk kamera jenis metrik. *On the job calibration*

merupakan teknik penentuan parameter kalibrasi lensa dan kamera dilakukan bersamaan dengan pelaksanaan pemotretan objek. Pada *self calibration* pengukuran titik-titik target pada objek pengamatan digunakan sebagai data untuk penentuan titik objek sekaligus untuk menentukan parameter kalibrasi kamera.

Dalam penelitian ini kalibrasi kamera akan dilakukan dengan menggunakan metode *self calibrating bundle adjustment* yang berdasarkan pada persaman kolinear (*Fraser, 1995*) :

$$x - x_0 + \Delta x = -c\frac{R_1}{R_3}$$

$$y - y_0 + \Delta y = -c\frac{R_2}{R_3} \quad \text{..... (2.36)}$$

Dimana :

$$R_3 = m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)$$

$$R_2 = m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L) \quad \text{..... (2.37)}$$

$$R_1 = m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L)$$

Dimana :

$$\Delta x = -x_0 - \frac{x}{c}\Delta c + \bar{x}r^2k_1 + \bar{x}r^4k_2 + \bar{x}r^6k_3 + (r^2 + 2\bar{x}^2)p_1 + 2p_2\bar{x}\bar{y} +$$

$$b_1\bar{x} + b_2\bar{y} \quad \text{..... (2.38)}$$

$$\Delta y = -y_0 - \frac{y}{c}\Delta c + \bar{y}r^2k_1 + \bar{y}r^4k_2 + \bar{y}r^6k_3 + 2p_1\bar{x}\bar{y} + (r^2 + 2\bar{y}^2)p_2$$

Dimana :

$$r = \sqrt{\bar{x}^2 + \bar{y}^2}$$

$$\bar{x} = x - x_0 \quad \text{..... (2.39)}$$

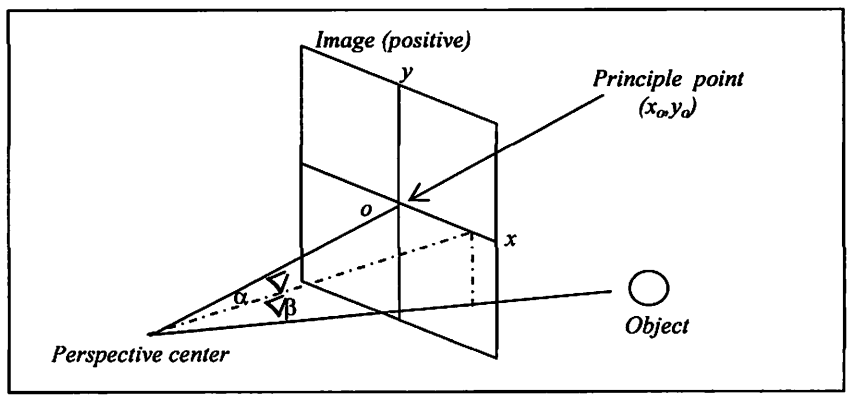
$$\bar{y} = y - y_0$$

2.5 Kalibrasi Kamera Non Metrik

Kamera non-metrik tidak mempunyai lensa yang sempurna, sehingga proses perekaman yang dilakukan akan memiliki kesalahan (*Fraser, 2000*). Oleh karena itu perlu dilakukan kalibrasi kamera untuk dapat menentukan besarnya penyimpangan-penyimpangan yang terjadi. Kalibrasi kamera ditujukan untuk memodelkan dan menentukan nilai distorsi dan konstanta sistem optik yang ada pada kamera. Kalibrasi kamera dilakukan untuk menentukan parameter distorsi, meliputi distorsi radial dan distorsi tangensial (*decentring*), serta parameter-parameter lensa lainnya, termasuk juga *principal distance* (*c*), serta titik pusat fidusial foto.

2.5.1 Parameter x_0, y_0, dan Fokus (*c*)

Dalam berbagai kasus fotogrametri, elemen dari *principle point* (x_o, y_o) dan *perspektif distance* (panjang fokus) harus ditentukan, hal ini dikarenakan semua sistem persamaan matematis yang digunakan dalam fotogrametri bergantung dari ketiga parameter ini. Secara geometris hubungan antara ketiga parameter ini dapat di lihat pada gambar dibawah ini.



Gambar 2.10 Geometri foto

Dari gambar diatas posisi *principle point* (x_o, y_o) merupakan proyeksi garis lurus dari letak *perspective center* ke bidang foto dan jarak dari *principle point* ke *perspective center* merupakan panjang fokus (c). Secara praktis panjang fokus kamera dan letak *principle point* tidak mutlak berada di tengah-tengah pusat foto, permasalahan ini disebabkan oleh kurang stabilnya susunan lensa dan CCD yang berguna untuk merekam bayangan objek pada saat perakitan. Sehingga perubahan posisi *principle point* dan panjang fokus dapat dimodelkan menggunakan persamaan sebagai berikut (*Dorstel, 2004*) :

$$\Delta x_i = \Delta x_p - \frac{\bar{x}}{\bar{z}}\Delta f$$
$$\Delta y_i = \Delta y_p - \frac{\bar{y}}{\bar{z}}\Delta f \quad \text{..... (2.40)}$$

Dimana $\Delta x_i, \Delta y_i$ merupakan total koreksi dari parameter (x_0, y_0) dan fokus, $\Delta x_p, \Delta y_p$ koreksi untuk parameter *principle point* dan Δf koreksi untuk nilai parameter fokus, dengan nilai koordinat foto didefinisikan sebagai berikut (*Dorstel, 2004*) :

$$\bar{x} = x - x_p$$
$$\bar{y} = y - y_p$$
$$\bar{z} = -c \quad \text{..... (2.41)}$$

2.5.2 Distorsi Radial

Distorsi radial menyebabkan semua bagian gambar diubah letaknya menurut jari-jari, bermula dari sumbu optik. Keadaan ini disebabkan oleh kesalahan dalam pengasahan bagian-bagian lensa. Nilai distorsi radial merupakan perpidahan secara

radial suatu titik dari posisi sebenarnya terhadap posisi dari *pronciple point* (x_0, y_0). dengan indikator apabila nilainya positif maka pergeserannya mengarah keluar dan jika nilainya negatif maka pergeserannya mengarah kedalam (*Wolf dan Dewitt, 2000*).

Secara teoritik koreksi radial dilakukan setelah dilakukan reduksi gambar ke titik utama dan koreksi oleh kesalahan pengkerutan atau pemekaran. Jumlah pengkerutan dan pemekaran pada foto ditentukan di dalam kalibrasi kamera. Sehingga koordinat foto dapat dikoreksi. Bila x_m dan y_c merupakan ukuran fidusial pada positif, sedangkan x_c dan y_c merupakan jarak fidusial terkalibrasi, maka koordinat foto pada titik "*a*" dapat dihitung sebagai berikut (*Wolf dan Dewitt, 2000*) :

$$x'_a = \left(\frac{x_c}{x_m}\right) x_a$$

$$y'_a = \left(\frac{y_c}{y_m}\right) y_a \quad \text{..... (2.42)}$$

Dimana x_a' dan y_a' adalah koordinat foto terkoreksi dan x_a dan y_a merupakan koordinat terukur. Persamaan polinomial berdasarkan pada teori desain kamera adalah sebagai berikut (*Wolf dan Dewitt, 2000*) :

$$\Delta xr = k_1 r^3 + k_2 r^5 + k_3 r^7$$
$$\Delta yr = k_1 r^3 + k_2 r^5 + k_3 r^7 \quad \text{..... (2.43)}$$

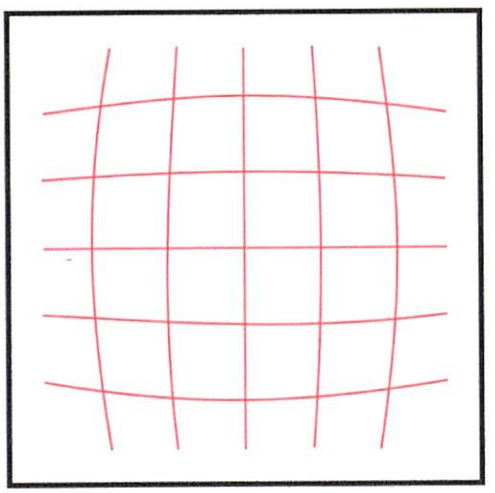
Atau

$$d_r = k_1 r^3 + k_2 r^5 + k_3 r^7 + \cdots \quad \text{..... (2.44)}$$

Dimana istilah dari K_1, K_2, dan K_3 merupakan koefisien dari distorsi radial lensa dan *r* merupakan jarak radial dari pusat foto terkalibrasi yang didapat dari (*Wolf dan Dewitt, 2000*) :

$$r = \sqrt{\bar{x}^2 + \bar{y}^2} \quad \text{..... (2.45)}$$

Efek distorsi radial adalah sekitar 1 sampai 2 piksel di perbatasan CCD sensor. Dalam kaitannya dengan definisi distorsi radial, ada korelasi besar antara koefisien distorsi itu sendiri K_1, K_2, K_3 dan antara *principle distance*. Hubungan antara distorsi radial dengan *principle distance* adalah dalam kaitannya sesuai *principle distance* dengan deviasi rata-rata akan dihitung menjadi minimum.



Gambar 2.11. Distorsi Radial

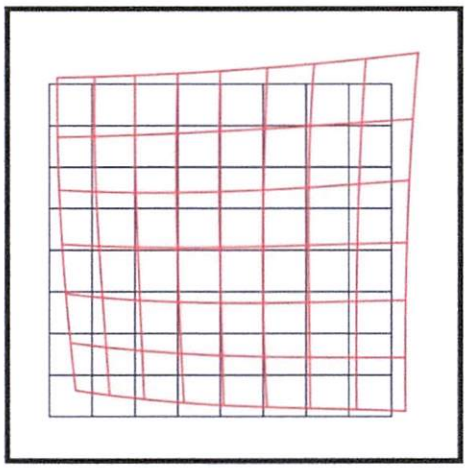
2.5.3 Distorsi *Decentring* (Distorsi tangensial)

Distorsi tangensial atau distorsi *decentring* adalah pergeseran linier titik di foto pada arah normal (tegak lurus) garis radial melalui titik foto tersebut. Distorsi tangensial disebabkan kesalahan *centering* elemen-elemen lensa dalam satu gabungan lensa dimana titik pusat elemen-elemen lensa dalam gabungan lensa tersebut tidak terletak pada satu garis lurus. Pergeseran ini biasa dideskripsikan dengan 2 persamaan polonomial untuk pergeseran pada arah x (dx) dan y (dy) (*Atkinson, 2001*).

$$d_x = P_1\,[r^2 + 2(x - x_0)^2] + 2P_2(x - x_0)(y - y_0)$$

$$d_y = P_2\,[r^2 + 2(y - y_0)^2] + 2P_1(x - x_0)(y - y_0) \quad \text{..... (2.46)}$$

Dimana P_1 dan P_2 merupakan koefesien dari parameter distorsi *decentring* yang nilainya tergantung dari nilai panjang fokus kamera. Efek dari distorsi *decentring* ini akan menyebabkan kesan hiperbolik pada foto yang terekam oleh kamera.



Gambar 2.12 Distorsi decentring (Atkinson, 2001)

2.5.4 Distorsi *Affinity*

Parameter distorsi *affinity* secara umum dapat dimodelkan dalam bentuk persamaan polinimial dan gabungan dari koreksi koordinat foto (*Fraser, 1995*).

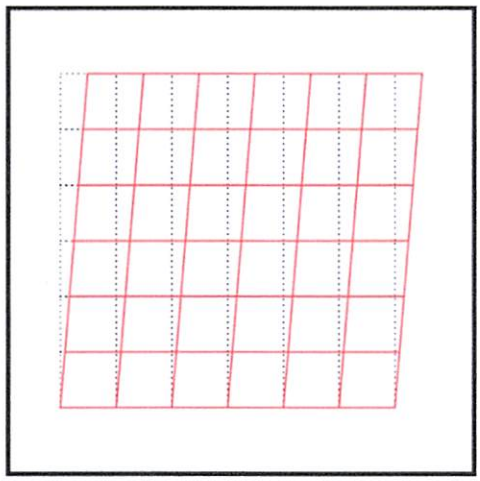
$$\begin{Bmatrix}\Delta x_u \\ \Delta x_u\end{Bmatrix}\begin{Bmatrix}\bar{x}/r \\ \bar{y}/r\end{Bmatrix}\sum_{i=0}^{n}\sum_{j=0}^{i} a_{ij}\bar{x}^{(i-j)}\bar{y}^{(j)} \quad \text{..... (2.47)}$$

Dari persamaan (2.31) dapat digunakan untuk memodelkan bidang yang tidak rata, dimana model koreksi pada bidang distorsi dapat dikurangi menjadi dua aturan hanya untuk koordinat *x*, satu untuk menghitung perbedaan skala antar selang piksel

horisontal dan vertikal, dan satu untuk memodelkan *non-ortogonality* antara sumbu *x* dan sumbu *y* (*Fraser, 1995*) :

$$\begin{aligned} \Delta x_f &= b_1 x + b_2 y \\ \Delta y_f &= 0 \end{aligned} \quad \text{..... (2.48)}$$

Distorsi *affinity* ini terjadi akibat kurang sikunya bidang CCD atau CMOS yang digunakan untuk merekam bayangan objek, sehingga frame dari foto tidak akan benar-benar berbentuk sebuah bujur sangkar ataupun persegi panjang akan tetapi akan membentuk sebuah jajaran genjang.



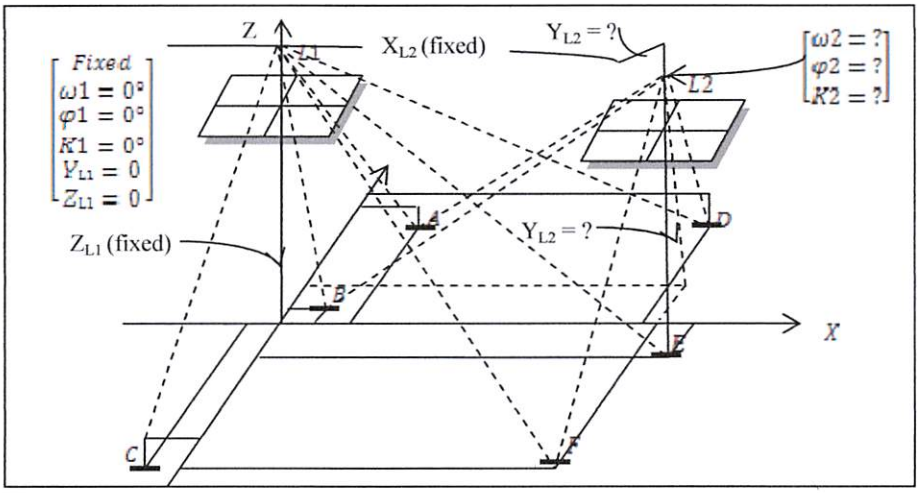
Gambar 2.13 Distorsi Affinity

2.6 *Relative Orientation*

Relatif orientasi merupakan proses untuk menentukan nilai perputaran sudut rotasi dan pergeseran posisi antar foto yang diambil (*Wolf dan Dewitt, 2000*). Proses ini dilakukan dengan cara memberikan nilai posisi dan orientasi untuk foto pertama, kemudian dilakukan proses perhitungan nilai posisi dan orientasi pada foto kedua menggunakan parameter dari posisi kamera pertama dan koordinat foto dari kedua

buah foto. Dalam proses relatif orientasi ini tidak menghasilkan nilai posisi dan orientasi dari foto yang sebenarnya, akan tetapi menghasilkan sebuah nilai relatif antara dua buah foto tersebut. Yaitu menetapkan beberapa parameter eksterior orientasi (EO) ω, φ, k, Y_L, Z_L dari foto kanan (foto kedua) dari pertemuan 5 berkas sinar dari koordinat obyek 3D (X_i, Y_i, Z_i) yang ada (*Wolf dan Dewitt, 2000*).

Dengan cara dijital, relatif orientasi dapat menggunakan syarat kesegarisan (*colinearity condition*) atau syarat kesebidangan (*coplanarity condition*). Dimana kondisi kesegarisan antar foto dapat dilukiskan seperti pada gambar dibawah ini :



Gambar 2.14 Relatif orientasi secara analitik

Proses penentuan relatif orientasi dapat dilukiskan dalam gambar (2.16), sinar-sinar yang berkaitan dengan enam titik dari A hingga F tampak memenuhi kondisi tersebut. Keenam buah titik tersebut pada dasarnya terletak pada bagian daerah yang sama pada kedua foto (*Wolf dan Dewitt, 2000*).

Sehingga dapat dituliskan persamaan-persamaan kebersamaan garis untuk kedua foto, dan minimal untuk lima buah titik objek. Persamaan dari kedua foto

tersebut mengandung koordinat keruangan yang sama dan sistem persamaan kebersamaan garis yang dirumuskan terdapat lima buah parameter orientasi luar foto kanan (foto kedua) yaitu ω_2, φ_2, κ_2, Y_{L2} dan Z_{L2} yang belum diketahui dan ditambah bentuk 3D koordinat objek yang belum diketahui (X_i, Y_i, Z_i) untuk masing-masing titik yang digunakan dalam pemecahan masalah sehingga parameter orientasi luar yang diperoleh nantinya akan dikoreksi pada relatif antara kedua foto (*Wolf dan Dewitt, 2000*).

Pada relatif orientasi analitik, biasanya parameter EO (ω, φ, κ, X_L, Y_L) dari foto kiri sama dengan nol. Dan juga untuk Z_L pada foto kiri (Z_{L1}) ditetapkan secara sembarang pada harga bulat dan sebagai alternatif yang nyaman maka nilai dari Z_{L1} tepat pada angka nol, dan X_L pada foto kanan (X_{L2}) ditetapkan pada harga mendekati basis foto (jarak difoto pada kedua foto) yang mendekati nol dan harus ditentukan 5 parameter *unknown* pada foto kanan. Hal ini akan mempermudah dalam perhitungan koordinat objek X_i, Y_i, Z_i sehingga mendekati satuan koordinat foto yang terukur.

Metode yang digunakan sebagai solusi untuk mendapatkan parameter yang dicari adalah menggunakan teknik kuadrat terkecil (*Wolf dan Dewitt, 2000*).

$$b_{11}d\omega + b_{12}d\varphi + b_{13}d\kappa - b_{14}dX_L - b_{15}dY_L - b_{16}dZ_L + b_{14}dX_A + b_{15}dY_A + b_{16}dZ_A = J + v_{xa}$$

$$b_{21}d\omega + b_{22}d\varphi + b_{23}d\kappa - b_{24}dX_L - b_{25}dY_L - b_{26}dZ_L + b_{24}dX_A + b_{25}dY_A + b_{26}dZ_A = K + v_{ya} \quad \text{.....(2.49)}$$

Bentuk matriks A yaitu :

$$A=\begin{bmatrix}
(b_{a11})_1 & (b_{a12})_1 & (b_{a13})_1 & (-b_{a15})_1 & (-b_{a16})_1 & (b_{a14})_1 & (b_{a15})_1 & (b_{a16})_1 & & & & 0 & 0 & 0 \\
(b_{a21})_1 & (b_{a22})_1 & (b_{a23})_1 & (-b_{a25})_1 & (-b_{a26})_1 & (b_{a24})_1 & (b_{a25})_1 & (b_{a26})_1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\
(b_{n11})_1 & (b_{n12})_1 & (b_{n13})_1 & (-b_{n15})_1 & (-b_{n16})_1 & 0 & 0 & 0 & \ddots & \ddots & \ddots & (b_{n14})_1 & (b_{n15})_1 & (b_{n16})_1 \\
(b_{n21})_1 & (b_{n22})_1 & (b_{n23})_1 & (-b_{n25})_1 & (-b_{n26})_1 & 0 & 0 & 0 & 0 & 0 & 0 & (b_{n24})_1 & (b_{n25})_1 & (b_{n26})_1 \\
(b_{a11})_2 & (b_{a12})_2 & (b_{a13})_2 & (-b_{a15})_2 & (-b_{a16})_2 & (b_{a14})_2 & (b_{a15})_2 & (b_{a16})_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
(b_{a21})_2 & (b_{a22})_2 & (b_{a23})_2 & (-b_{a25})_2 & (-b_{a26})_2 & (b_{a24})_2 & (b_{a25})_2 & (b_{a26})_2 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & 0 & 0 & 0 & \ddots & \ddots & \ddots & 0 & 0 & 0 \\
(b_{n11})_2 & (b_{n12})_2 & (b_{n13})_2 & (-b_{n15})_2 & (-b_{n16})_2 & 0 & 0 & 0 & 0 & 0 & 0 & (b_{n14})_2 & (b_{n15})_2 & (b_{n16})_2 \\
(b_{n21})_2 & (b_{n22})_2 & (b_{n23})_2 & (-b_{n25})_2 & (-b_{n26})_2 & 0 & 0 & 0 & 0 & 0 & 0 & (b_{n24})_2 & (b_{n25})_2 & (b_{n26})_2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{bmatrix}$$

Bentuk matriks *X*, *L*, dan *V* sebagai berikut :

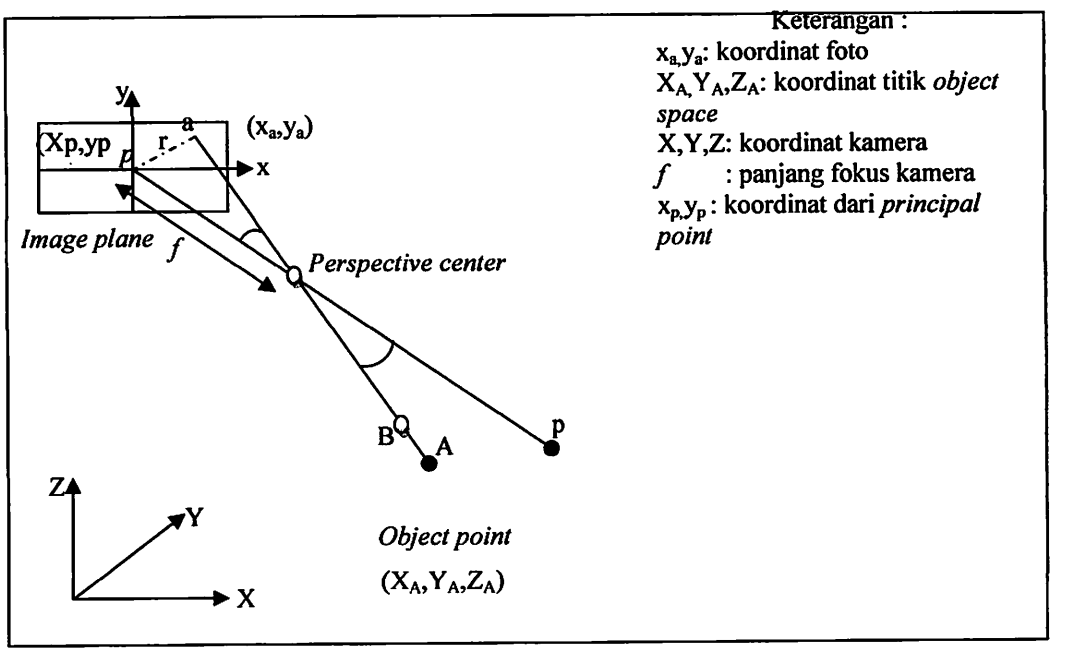
$$X=\begin{bmatrix} d\omega_2 \\ d\varphi_2 \\ d\kappa_2 \\ dY_2 \\ dZ_2 \\ dX_A \\ dY_A \\ dZ_A \\ \vdots \\ \vdots \\ dX_n \\ dY_n \\ dZ_n \end{bmatrix} \quad L=\begin{bmatrix} (J_a)_1 \\ (K_a)_1 \\ \vdots \\ \vdots \\ \vdots \\ (J_n)_1 \\ (K_n)_1 \\ (J_a)_2 \\ (K_a)_2 \\ \vdots \\ \vdots \\ (J_n)_2 \\ (K_n)_2 \end{bmatrix} \quad V=\begin{bmatrix} (V_{xa})_1 \\ (V_{ya})_1 \\ \vdots \\ \vdots \\ \vdots \\ (V_{xn})_1 \\ (V_{yn})_1 \\ (V_{xa})_2 \\ (V_{ya})_2 \\ \vdots \\ \vdots \\ (V_{xn})_2 \\ (V_{yn})_2 \end{bmatrix} \qquad \text{.....(2.50)}$$

2.7 *Resection*

Space Resection atau reseksi ruang dengan kolinearitas merupakan metode numerik murni yang secara serentak menghasilkan enam unsur orientasi luar (EO). Biasanya nilai sudut XL, YL, ZL, ω, φ, κ diperoleh dengan penyelesaian itu. *Space Resection* dengan kolinearitas memungkinkan penggunaan ulang sejumlah titik kontrol medan. Oleh karena itu dapat digunakan cara perhitungan kuadrat terkecil untuk menentukan nilai yang paling mungkin bagi keenam unsur itu. Meskipun perhitungannya panjang dapat dilakukan secara rutin. *Space Resection* dengan

kolinearitas merupakan metode yang lebih disukai untuk menentukan unsur orientasi luar (*wolf, 2000*).

Space Resection dengan kolinearitas meliputi formulasi yang disebut dengan Persamaan Kolinearitas (*Collinearity equation*) untuk sejumlah titik kontrol yang koordinat medannya X, Y dan Z diketahui dan yang gambarnya tampak pada foto. Kemudian persamaan itu diselesaikan untuk enam unsur orientasi luar yang belum diketahui dan tampak pada foto. Kolinearitas di deskripsikan sebagai kondisi dimana stasiun pemotretan, beberapa titik objek, dan foto berada pada satu garis lurus pada *space* 3D. Kondisi kolinearitas di ilustrasikan seperti gambar di bawah ini dimana A, o dan a terletak pada satu garis lurus :



Gambar 2.15. Kondisi kolinearitasi

Space Resection merupakan suatu proses untuk menentukan elemen *Exterior Orientation* (EO) dan posisi sensor dari titik kontrol tanah dan koordinat image. Metode perhitungan yang paling biasa digunakan adalah persamaan kolineariti, dimana prinsip dari persamaan tersebut adalah titik kontrol, titik pada image, dan proyeksi pusat terletak pada satu garis lurus. Untuk setiap titik kontrol, dapat diperoleh dua persamaan. Karena terdapat 6 parameter EO, sedikitnya tiga titik kontrol dibutuhkan untuk memecahkan masalah resection. Metode perhitungan dengan menggunakan teknik Least Square akan diterapkan pada penelitian ini untuk menentukan nilai yang paling mungkin pada enam parameter EO (*Yao Jianchao and Chia Chern, 2001*).

Ukuran koordinat foto x_a dan y_a (menyuling dan mengoreksi untuk distorsi lensa jika sesuai) *image* sasaran memberi kenaikan ke dua persamaan kolineariti. Jika tiga *elemen Interior Orientation* (c, x_o, and y_o) diberikan oleh kalibrasi kamera dan koordinat (X_A, Y_A, Z_A) dititik A pada sistem koordinat *object space* maka dikenal dua persamaan dengan 6 nilai yang belum diketahui yaitu rotasi ω, φ, κ dan koordinat (X_O, Y_O, Z_O) pada *perspective center.* Sedikitnya 3 target *non-collinear* seperti titik kontrol diperlukan untuk *resection* dari kamera. Metode ini digunakan untuk mengevaluasi elemen EO yang bergantung pada tujuan fotogrametri *(Cooper, 1987).*

Metode untuk evaluasi secara berlangsung pada enam elemen orientasi bagian luar (*Eksterior Orientation*) diperoleh dari diukurnya koordinat foto pada image dengan tiga titik kontrol non kolinear yang tidak memerlukan beberapa

nilai pendekatan *(Zeng and Wang, 1992 dalam Cooper et al, 1987)*. Prosedur ini memberikan koordinat secara langsung dari *perspective center*. Bentuk secara aljabar akan digunakan pada matriks rotasinya. Jika diperlukan, nilai untuk rotasi ω, φ, dan κ dapat dicari dari 9 elemen matrik rotasi *(Cooper, 1987)*.

Jika perhitungan resection secara statistik lebih teliti diperlukan, maka persamaan kolineariti dapat dilinearisasikan dan proses *least square* dapat digunakan untuk mengevaluasi 6 *elemen Eksterior Orientation*. Untuk mendapat nilai yang *resection* yang teliti perlu mendapat nilai pendekatan untuk unsur orientasi yang cukup dekat dengan nilai akhir untuk proses iterasi agar lebih teliti. Biasanya nilai yang tepat untuk koordinat (X_O, Y_O, Z_O) dapat langsung diperoleh, tetapi tidak untuk nilai sudut rotasinya. *Resection* hanya tingkat menengah pada prosedur fotogrametri, serigkali diikuti oleh *intersection* atau *bundle adjustment* dengan *multistation* yang teliti dimana menggunakan nilai unsur EO sebagai nilai awal pendekatan (*Cooper, 1987*).

Dalam metode persamaan pengamatan bagi penyesuaian least square, ditulis persamaan pengamatan yang berkaitan dengan nilai terukur terhadap kesalahan residual dan parameter unknown. Untuk pemecahan yang unik maka jumlah persamaan harus sama besar dengan jumlah unknown. Bila dilakukan pengamatan berulang, maka dapat ditulis persamaan pengamtan yang lebih banyak dari yang diperlukan untuk pemecahan yang unik. Dan nilai yang paling mungkin dapat ditentukan dengan metode *least square*.

Dalam penyelesaian secara *least square* maka diperoleh persamaan untuk proses resection dalam bentuk persamaan kolinearitas terlinearisasi yang disederhanakan termasuk untuk nilai residualnya sebagai berikut *(wolf, 2000)* :

$b_{11}d\omega + b_{12}d\varphi + b_{13}k - b_{14}dX_L - b_{15}Y_L - b_{16}Z_L = J + vx_a$

$b_{21}d\omega + b_{22}d\varphi + b_{23}k - b_{24}dX_L - b_{25}Y_L - b_{26}Z_L = K + vy_a$ ..... *(2.51)*

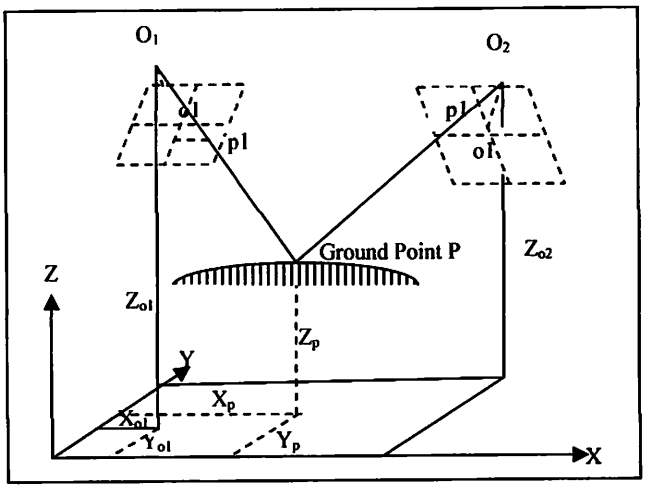
Dimana setiap notasi diatas diwakili oleh susunan matriks sebagai berikut :

$$A = \begin{bmatrix} b_{11_a} & b_{12_a} & b_{13_a} & -b_{14_a} & -b_{15_a} & -b_{16_a} \\ b_{21_a} & b_{22_a} & b_{23_a} & -b_{24_a} & -b_{25_a} & -b_{26_a} \\ b_{11_b} & b_{12_b} & b_{13_b} & -b_{14_b} & -b_{15_b} & -b_{16_b} \\ b_{21_b} & b_{22_b} & b_{23_b} & -b_{24_b} & -b_{25_b} & -b_{26_b} \\ b_{11_c} & b_{12_c} & b_{13_c} & -b_{14_c} & -b_{15_c} & -b_{16_c} \\ b_{21_c} & b_{22_c} & b_{23_c} & -b_{24_c} & -b_{25_c} & -b_{26_c} \\ b_{11_d} & b_{12_d} & b_{13_d} & -b_{14_d} & -b_{15_d} & -b_{16_d} \\ b_{21_d} & b_{22_d} & b_{23_d} & -b_{24_d} & -b_{25_d} & -b_{26_d} \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} d\omega \\ d\varphi \\ dk \\ dX_L \\ dY_L \\ dZ_L \end{bmatrix} \quad \mathrm{L} = \begin{bmatrix} J_A \\ K_A \\ J_B \\ K_B \\ J_C \\ K_C \\ J_D \\ K_D \\ \vdots \\ J_n \\ K_n \end{bmatrix} \quad \mathbf{v} = \begin{bmatrix} V_{x_a} \\ V_{y_a} \\ V_{x_b} \\ V_{y_b} \\ V_{x_c} \\ V_{y_c} \\ V_{x_d} \\ Vy_d \\ \vdots \\ Vx_n \\ Vy_n \end{bmatrix}$$

.....*(2.52)*

2.8 *Intersection*

Intersection merupakan suatu teknik menentukan koordinat titik-titik objek pada dua gambar atau lebih yang saling bertampalan sehingga diketahui posisi secara 3D (X_i, Y_i, Z_i). Proses ini membutuhkan enam parameter orientasi luar yang diketahui (ω, φ, κ, X_L, Y_L, Z_L) untuk dua foto yang bertampalan. Nilai koordinat objek dalam ruang tiga dimensi ini dapat dihitung menggunakan persamaan kolinier yang telah dilinierisasi(*Wolf and Dewitt, 2000*).



Gambar 2.16. proses Intersection

Intersection mengacu kepada determinasi posisi titik pada ruang objek dengan dua persamaan untuk setiap titik pada foto. Jika terdapat dua foto, total ada empat persamaan yang terdiri dari tiga persamaan yang tidak diketahui, titik koordinat ruang objek yang diperoleh. Ada satu derajat bebas, dan satuan persamaan linier dimana dapat dipecahkan dengan metode least square. Dengan menambahkan beberapa foto, meningkatkan jumlah derajat kebebasan dengan demikian akan meningkatkan solusinya (*Mikhail, Bethel et al, 2001*).

Karena enam unsur orientasi sudah diketahui, yang tidak diketahui pada persamaan ialah dX_A, dY_A, dan dZ_A. Ini merupakan koreksi yang harus diterapkan bagi pendekatan awal untuk masing-masing koordinat object space X_A, Y_A, Z_A, untuk titik A. Bentuk persamaan intersection yang diliniearkan sebagai berikut (*Wolf and Dewitt, 2000*):

$$b_{14}dX_A + b_{15}dY_A + b_{16}dZ_A = J + V_{xa}$$

$$b_{24}dX_A+b_{25}dY_A+b_{26}dZ_A=K+V_{ya} \quad \text{.....(2.53)}$$

Dengan demikian dapat dibuat empat persamaan seperti persamaan di atas, dan nilai dX_A, dY_A, dan dZ_A dapat diselesaikan melalui perhitungan least square. Koreksi ini

diterapkan bagi pendekatan awal untuk memperoleh nilai revisi untuk X_A, Y_A, Z_A. Penyelesaian ini kemudian diulang lagi atau proses iterasi hingga nilai residu sesuai.

Dalam bentuk matriks dapat dinyatakan sebagai berikut (*Wolf and Dewit, 2000)*:

$$A = \begin{bmatrix} (b14_a)_1 & (b15_a)_1 & (b16_a)_1 & \cdots & \cdots & \cdots \\ (b24_a)_1 & (b25_a)_1 & (b26_a)_1 & \cdots & \cdots & \cdots \\ (b14_a)_2 & (b15_a)_2 & (b16_a)_2 & \cdots & \cdots & \cdots \\ (b24_a)_2 & (b25_a)_2 & (b26_a)_2 & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & (b14_b)_1 & (b15_b)_1 & (b26_b)_1 \\ \cdots & \cdots & \cdots & (b24_b)_1 & (b25_b)_1 & (b26_b)_1 \\ \cdots & \cdots & \cdots & (b14_b)_2 & (b15_b)_2 & (b26_b)_2 \\ \cdots & \cdots & \cdots & (b24_b)_2 & (b25_b)_2 & (b26_b)_2 \end{bmatrix} \quad L = \begin{bmatrix} J_a \\ K_a \\ J_b \\ K_b \\ J_c \\ K_c \\ J_d \\ K_d \end{bmatrix}$$

$$X = \begin{bmatrix} dX_A \\ dY_A \\ dZ_A \end{bmatrix} \quad V = \begin{bmatrix} Vx_a \\ Vy_a \\ Vx_b \\ Vy_b \\ Vx_c \\ Vy_c \\ Vx_d \\ Vy_d \end{bmatrix} \qquad \text{.....38}$$

Untuk proses perhitungan dari intersection dapat dilihat pada pembahasan sebelumnya (Sistem Persamaan Dalam Fotogrametri). Dimana iterasi berhenti apabila nilai residu sudah sesuai. Jadi nilai akhir untuk proses intersection menggunakan metode least square adalah nilai (3) parameter (XA, YA, ZA), yang sudah diiterasi berulang kali dengan nilai residu yang sesuai dan seminimal mungkin (*Wolf and Dewit, 2000*).

2.9 *Self Calibrating Bundle Adjustment*

2.9.1 Persamaan Observasi

Metode gabungan kuadrat terkecil mensyaratkan agar persamaan dituliskan untuk masing-masing observasi yang secara fisik dibuat dan untuk masing-masing parameter yang digunakan dalam model fungsional. Persamaan ini disebut dengan

persamaan observasi. Persamaan observasi dikembangkan dalam tiga kelompok variabel, yaitu (*King, 1993*) :

1. *The plate coordinates*
2. The camera's exterior orientation parameters
3. The object point coordinates

2.9.2 Matrik Bobot Observasi

Salah satu keuntungan dari penerapan metode gabungan kuadrat terkecil adalah memungkinkan untuk menggabungkan parameter dan observasi yang digunakan dalam proses penyesuaian dengan mempertimbangkan bobot yang sesuai. Parameter atau observasi yang nilainya reliabel dapat digabungkan dengan bobot yang tinggi dengan sebaliknya nilai yang tidak reliabel digabungkan dengan bobot yang rendah.

Bobot dari sebuah observasi bernilai proporsional dengan varian observasi (σ^2). Varian utama untuk tiap bobot dilambangkan dengan (σ_0^2). Bobot dari observasi dinyatakan sebagai berikut (*King, 1993*) :

$$W = \sigma_0^2 / \sigma^2 \quad \text{.....(2.39)}$$

Matrik bobot untuk *plate coordinates observation* dalam *i* pada foto *j* sebagai berikut (*Fraser, 1997*) :

$$W_{ij} = \begin{pmatrix} 1/\sigma_x^2 & 0 \\ 0 & 1/\sigma_y^2 \end{pmatrix}_{ij} \quad \text{.....(2.40)}$$

Pembahasan mengenai matrik bobot untuk kamera dan objek sebagai berikut (*King, 1993*).

Matrik bobot diagonal untuk kamera *j* sebagai berikut :

$$W_j = \begin{bmatrix} 1/\sigma_\omega^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1/\sigma_\varphi^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/\sigma_\kappa^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/\sigma_{XL}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/\sigma_{YL}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/\sigma_{ZL}^2 \end{bmatrix} \quad \text{.....(2.41)}$$

untuk keseluruhan *n* foto sebagai berikut :

$$W = \begin{bmatrix} W_1 & & & \\ & \ddots & & \\ & & W_j & \\ & & & \ddots \\ & & & & W_n \end{bmatrix} \quad \text{.....(2.42)}$$

Matrik diagonal untuk titik objek *i* sebagai berikut :

$$W = \begin{bmatrix} 1/\sigma_X^2 & 0 & 0 \\ 0 & 1/\sigma_Y^2 & 0 \\ 0 & 0 & 1/\sigma_Z^2 \end{bmatrix} \quad \text{.....(2.43)}$$

dan untuk keseluruhan *m* titik objek sebagai berikut :

$$W = \begin{bmatrix} W_1 & & & \\ & \ddots & & \\ & & W_j & \\ & & & \ddots \\ & & & & W_m \end{bmatrix} \quad \text{.....(2.44)}$$

2.9.3 Model Matematika

Untuk m titik didalam n foto dengan mengacu pada persamaan observasi digabungkan untuk membentuk persamaan pengamatan sebagai berikut (*King, 1993*) :

$$V + B\Delta = f \quad \text{.....(2.45)}$$

Dimana Δ merupakan matrik koreksi parameter yang dicari, V adalah matrik residu, B merupakan matrik koefisien dan f yaitu matrik observasi. Jika sekumpulan persamaan dikombinasikan dengan persamaan matrik bobotnya (W), solusi kuadrat terkecilnya sebagai berikut (King, 1993) :

$$\begin{aligned} \Delta &= \left(B^T W B\right)^{-1} B^T W f \\ &= N^{-1} t \end{aligned} \quad \text{.....(2.46)}$$

2.9.4 Linierisasi Persamaan Kolinier

Persamaan dasar dari *bundle adjustment* adalah persamaan kolinear, yang mendiskripsikan satuan dasar dari *photogrammetry*. Dengan menggunakan persamaan kolinear membuat bentuk persamaan dan proses penyelesaiannya akan lebih efisien (*Mikhail et al., 2001*). Bentuk linier dari persamaan kolinear untuk foto i dan titik j. Karena persamaan pengamatan garis lurus pada Persamaan (2.19) tidak

linier, maka untuk menghitung parameter didalamnya harus dilinierkan terlebih dahulu dengan menggunakan deret Taylor sebagai berikut (*Stewart, 1999*):

$$\left(df/d\underline{x}.dx\right)_{ij}+\left(df/d\underline{O}.d\underline{O}\right)_{ij}+\left(df/d\underline{x}.d\underline{X}\right)_{ij}+f\left(\underline{x},\underline{O}^0,\underline{X}^0\right)_{ij}=0 \quad \text{.....(2.47)}$$

Bentuk linier dari persamaan pada Persamaan (2.28) ini setara dengan persamaan hitung kuadrat terkecil (*least square observation*) menurut Mikhail et al. (2001) yaitu :

$$\underset{(2,1)}{v}+\underset{(2,6)(6,1)}{B_1}\ \underset{}{\delta_1}+\underset{(2,3)(3,1)}{B_2}\ \underset{}{\delta_2}+\underset{(2,10)(10,1)}{B_3}\ \underset{}{\delta_3}=\underset{(2,1)}{f} \quad \text{.....(2.48)}$$

dengan :

$$O=O^0+\delta_1,\quad X=X^0+\delta_2 \quad \text{.....(2.49)}$$

Keterangan :

$v=\left[v_x\ \ v_y\right]^T_{ij}$ = Residu dari koordinat foto

$f=f\left(\underline{x},\underline{O}^0,\underline{X}^0\right)_{ij}$ = Observasi

$\delta_1=\left[d\omega_i\ d\varphi_i\ d\kappa_i\ dXL_i\ dYL_i\ dZL_i\right]^T$ =Koreksi parameter EO 6m vektor)

$\delta_2=\left[dX_j,dY_j,dZ_j\right]^T$ =Koreksi Koordinat koreksi XYZ (3n vektor)

2.9.5 Desain Matrik untuk *Self-Calibrating Bundle Adjustment*

Menurut Mikhail et al. (2001) dan Wolf dan Dewitt (2000), jika sebuah titik objek (objek ke-j) terekam pada sebuah foto (foto ke-i), maka dimensi-dimensi matrik didalam Persamaan (2.29) adalah sebagai berikut :

$$B_{\underset{(2,2)}{ij}} = \begin{pmatrix} \frac{\partial f_x}{dx} & \frac{\partial f_y}{dy} \\ \frac{\partial f_x}{dx} & \frac{\partial f_y}{dy} \end{pmatrix}_{ij} \quad \text{.....(2.50)}$$

$$v_{\underset{(2,1)}{ij}} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}_{ij} \quad \text{.....(2.51)}$$

Susunan matrik B_1 sebagai berikut (*Fraser, 1997*):

$$B_{\underset{(2,6)}{1ij}} = \begin{bmatrix} \left(\frac{\partial f_x}{\partial \omega}\right)_0 & \left(\frac{\partial f_x}{\partial \varphi}\right)_0 & \left(\frac{\partial f_x}{\partial \kappa}\right)_0 & \left(\frac{\partial f_x}{\partial X_L}\right)_0 & \left(\frac{\partial f_x}{\partial Y_L}\right)_0 & \left(\frac{\partial f_x}{\partial Z_L}\right)_0 \\ \left(\frac{\partial f_y}{\partial \omega}\right)_0 & \left(\frac{\partial f_y}{\partial \varphi}\right)_0 & \left(\frac{\partial f_y}{\partial \kappa}\right)_0 & \left(\frac{\partial f_y}{\partial X_L}\right)_0 & \left(\frac{\partial f_y}{\partial Y_L}\right)_0 & \left(\frac{\partial f_y}{\partial Z_L}\right)_0 \end{bmatrix}_{ij} \text{..(2.52)}$$

atau secara sederhana sebagai berikut (*Wolf dan Dewitt, 2000*).

$$B_{\underset{(2,6)}{1ij}} = \begin{bmatrix} (b_{11})_0 & (b_{12})_0 & (b_{13})_0 & (-b_{14})_0 & (-b_{15})_0 & (-b_{16})_0 \\ (b_{21})_0 & (a_{22})_0 & (b_{23})_0 & (-b_{24})_0 & (-b_{25})_0 & (-b_{26})_0 \end{bmatrix}_{ij} \quad \text{.....(2.53)}$$

Susunan matrik B_2 sebagai berikut (*Fraser, 1997*) :

$$B_{\underset{(2,3)}{2ij}} = \begin{bmatrix} \left(\frac{\partial f_x}{\partial X_i}\right)_0 & \left(\frac{\partial f_x}{\partial X_i}\right)_0 & \left(\frac{\partial f_x}{\partial X_i}\right)_0 \\ \left(\frac{\partial f_x}{\partial X_i}\right)_0 & \left(\frac{\partial f_x}{\partial X_i}\right)_0 & \left(\frac{\partial f_x}{\partial X_i}\right)_0 \end{bmatrix}_{ij} \quad \text{.....(2.54)}$$

atau (*Wolf dan Dewitt, 2000*)

$$B_{\underset{(2,3)}{2ij}} = \begin{bmatrix} (b_{14})_0 & (b_{15})_0 & (b_{16})_0 \\ (b_{24})_0 & (b_{25})_0 & (b_{26})_0 \end{bmatrix}_{ij} \quad \text{.....(2.55)}$$

B_1 mempunyai dimensi 2x6 sedangkan B_2 mempunyai dimensi 2x3 dimana untuk tiap titik objek ke-j yang terekam pada foto ke-i.

Susunan matrik f sebagai berikut (*Fraser, 1997*) :

$$f_{ij} = \begin{bmatrix} J \\ K \end{bmatrix}_{ij} \quad \text{.....(2.56)}$$

Penjelasan elemen matrik B dan matrik f seperti yang ada pada persamaan (2.21 dan 2.25) (*Wolf and Dewitt, 2000*) :

Untuk matrik δ_{1i} dan δ_{2j} struktur matriknya sebagai berikut (*Mikhail et al. 2001*) :

$$\underset{(6m,1)}{\delta_{1i}} = \begin{bmatrix} \delta\omega \\ \delta\varphi \\ \delta\kappa \\ \delta X_L \\ \delta Y_L \\ \delta Z_L \end{bmatrix}_i \quad \text{.....(2.57)}$$

$$\underset{(3n,1)}{\delta_{2j}} = \begin{bmatrix} \delta X \\ \delta Y \\ \delta Z \end{bmatrix}_j \quad \text{.....(2.58)}$$

Dimana δ_1 besar dimensinya 6mx1 dan δ_2 besar dimensinya 3nx1

Didalam kamera dijital SLR terdapat bidang sensor CCD/CMOS dan susunan lensa, dimana susunan lensa ini mentransformasikan (transformasi perspektif) gambaran suatu titik obyek di permukaan bumi ke bidang foto dijital. Karena adanya kesalahan sistematis pada bidang sensor dan susunan lensa ini, maka gambaran titik pada bidang foto mengalami distorsi. Karena koordinat titik-titik obyek ini menjadi fokus pembahasan, maka untuk mendapatkan nilai koordinat dengan akurasi tinggi, maka kesalahan sistematis ini harus dihilangkan atau diminimalkan pengaruhnya terhadap ketelitian titik-titik obyek. Teknik "*Self-Calibrating*" merupakan cara yang paling umum untuk meminimalkan kesalahan sistematis ini yang diperkenalkan oleh

Brown di akhir tahun 1960 (*1974*) dan sudah dianggap teknik baku sejak tahun 1980an (*Gruen, 1985a*). Pada prinsipnya, teknik *Self-Calibrating* menggunakan parameter kalibrasi ke dalam model fungsi Δx dan Δy didalam sistem persamaan kolinier. *Forstner et al.* (*2004*) menyebutkan model fisik kesalahan sistematis ini dapat dituliskan sebagai:

$$B_3\delta_3 = \begin{pmatrix} -1 & 0 & -\frac{\bar{x}}{c} & \bar{x}r^2 & \bar{x}r^4 & \bar{x}r^6 & 2\bar{x}^2+r & 2\bar{x}\bar{y} & \bar{x} & \bar{y} \\ 0 & -1 & -\frac{\bar{y}}{c} & \bar{y}r^2 & \bar{y}r^4 & \bar{y}r^6 & 2\bar{x}\bar{y} & 2\bar{y}^2+r & 0 & 0 \end{pmatrix} \begin{pmatrix} \delta x_0 \\ \delta y_0 \\ \delta c \\ \delta k_1 \\ \delta k_2 \\ \delta k_3 \\ \delta p_1 \\ \delta p_2 \\ \delta b_1 \\ \delta b_2 \end{pmatrix}$$

.....*(2.59)*

Dimana rumusan ini adalah elemen A_3, δ_3 pada persamaan 2.48 merupakan parameter-parameter yang dikenal sebagai parameter tambahan di dalam teknik *Bundle Adjustment*.

$$\begin{pmatrix} B_1^T PB_1 & B_1^T PB_3 & B_1^T PB_2 \\ B_3^T PB_1 & B_3^T PB_3 & B_3^T PB_2 \\ B_2^T PB_1 & B_2^T PB_3 & B_2^T PB_2 \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_3 \\ \delta_2 \end{pmatrix} + \begin{pmatrix} B_1^T Pw \\ B_3^T Pw \\ B_2^T Pw \end{pmatrix} = 0 \quad \text{..... (2.60)}$$

Atau

$$\begin{pmatrix} \dot{N}_{ij} & \tilde{N}_i & \bar{N}_{ij} \\ \tilde{N}_i^T & \ddot{N}_p & \hat{N}_j \\ \bar{N}_{ij}^T & \hat{N}_j^T & \ddot{N}_{ij} \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_3 \\ \delta_2 \end{pmatrix} + \begin{pmatrix} \dot{C}_i \\ \ddot{C}_p \\ \ddot{C}_j \end{pmatrix} = 0 \quad \text{..... (2.61)}$$

Atau

$$N\delta + C = 0 \quad ; \quad C_x = \sigma_0^2 N^{-1} = \sigma_0^2 Q_x \quad \text{.....(2.62)}$$

Dimana

$$\underset{(6x6)}{\dot{N}_{ij}} = \sum_{j=1}^{n} B_{1ij}^{T} W_{ij} B_{1ij} \; ; \; \underset{(6x1)}{\dot{C}} = \sum_{j=1}^{n} B_{1ij}^{T} W_{ij} f_{ij}$$

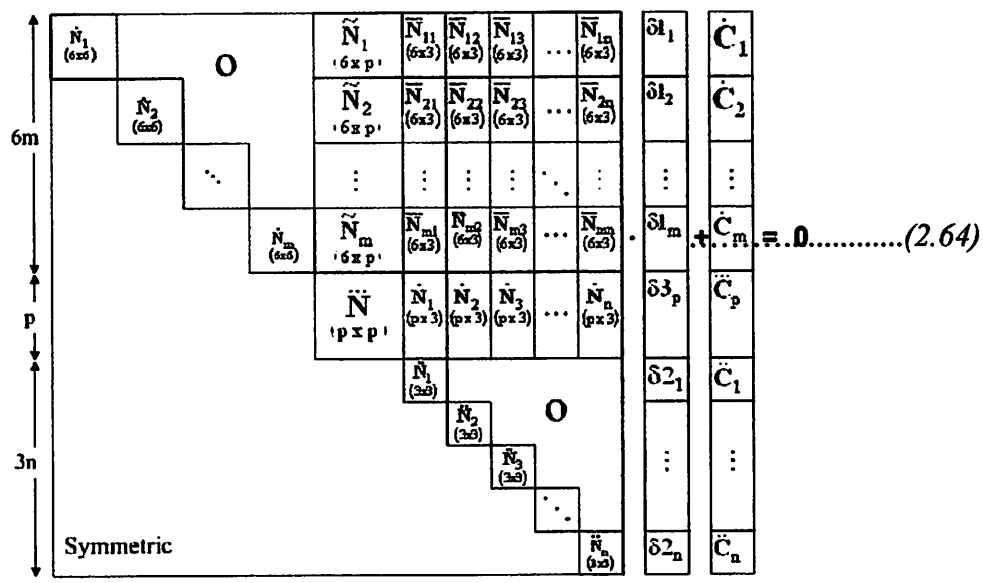
$$\underset{(3x3)}{\ddot{N}_{ij}} = \sum_{i=1}^{m} B_{2ij}^{T} W_{ij} B_{2ij} \; ; \; \underset{(3x1)}{\ddot{C}_{j}} = \sum_{i=1}^{m} B_{2ij}^{T} W_{ij} f_{ij}$$

$$\underset{(6x3)}{\overline{N}_{ij}} = B_{1\ ij}^{T} W_{ij} B_{2ij} \qquad \text{.....(2.63)}$$

$$\underset{(6x6)}{\tilde{N}_{i}} = B_{1ij}{}^{T} P_{ij} B_{3} \qquad \underset{(p x p)}{\dddot{N}_{p}} = \sum_{i=1}^{m}\sum_{j=1}^{n} B_{3}{}^{T} P_{ij} B_{3}$$

$$\underset{(p x n)}{\hat{N}_{j}} = B_{3}{}^{T} P_{ij} B_{2ij} \qquad \underset{(p x 1)}{\dddot{C}_{p}} = \sum_{i=1}^{m}\sum_{j=1}^{n} B_{3}{}^{T} P_{ij} w_{ij}$$

Dimana *Cx* dan *Qx* adalah matrik kovarian dan kofaktor dari parameter tambahan yang berjumlah 10 buah.



$$\qquad \text{.....(2.64)}$$

Solusi hitung kuadrat terkecil dapat ditentukan dengan menggunakan matrik inverse Cayley N-1. Tetapi matrik N memiliki rank defect sebesar parameter penentuan datum (*Cooper and Cross, 1988; 1991*), yaitu 7 (*Granshaw, 1980*).

Singularitas matrik N ini dapat diselesaikan dengan menambahkan parameter datum secara implisit untuk membentuk persamaan normal yang non-singular seperti:

$$\begin{pmatrix} \dot{N}_{ij} & \tilde{N}_i & \overline{N}_{ij} & 0 \\ \tilde{N}_i^T & \ddot{N}_p & \hat{N}_j & 0 \\ \overline{N}_{ij}^T & \hat{N}_j^T & \ddot{N}_{ij} & G \\ 0 & 0 & G^T & 0 \end{pmatrix} \begin{pmatrix} \delta_1 \\ \delta_3 \\ \delta_2 \\ k \end{pmatrix} + \begin{pmatrix} \dot{C}_i \\ \ddot{C}_p \\ \ddot{C}_j \\ 0 \end{pmatrix} = 0 \quad ; \text{atau } N\delta + C = 0 \text{ .....(2.65)}$$

Dimana G adalah matrik transformasi *Helmert* dan k adalah faktor pengali *Lagrangian*. Matrik G didefinisikan sebagai:

$$G = \begin{pmatrix} 1 & 0 & 0 & 0 & Z_j & -Y_j & X_j \\ 0 & 1 & 0 & -Z_j & 0 & X_j & Y_j \\ 0 & 0 & 1 & Y_j & -X_j & 0 & Z_j \end{pmatrix} \text{.....(2.66)}$$

Teknik hitungan yang dipakai untuk menyelesaikan Persamaan (2.61) ada beberapa macam, seperti misalnya teknik *minimum constraint, S-transformation, Pseudo Inverse, dan Free-Net Adjustment (Tjahjadi, 2008b)*. Tetapi, teknik yang paling sesuai didalam kasus ini adalah teknik yang dapat mengoptimalkan tingkat keakurasian koordinat titik-titik obyek. Dengan kata lain, harus dipilih suatu teknik yang dapat meminimumkan matrik kovarian dari titik-titik obyek.

2.10 Pemrograman menggunakan bahasa C#

Bahasa pemrograman C# dikembangkan oleh Microsoft sebagai bahasa yang simple, modern, *general-purpose*, dan berorientasi obyek (*SmithDev*, 2009). Kehadiran C# memeberi suntikan optimisme bagi para programmer untuk dapat mengembangkan aplikasi yang bedaya guna dengan lebih cepat dan mudah. Bahasa *C* merupakan *general-purpose language*, yaitu bahasa pemrograman yang dapat digunakan untuk tujuan apa saja. *C* merupakan *industrial-strenght language*. Dengan

bahasa *C*, kita dapat membangun beragam aplikasi, mulai dari pemrograman sistem, aplikasi cerdas (*artificial intelligence*), sistem pakar, *utility*, *driver*, database, *browser*, *network programming, sistem operasi, game, virus,* dan lainnya.

Untuk menjawab semua permasalahan dan kebutuhan diatas pada tahun 2000 *Microsoft* meluncurkan bahasa pemrograman baru yang diberi nama *C# Programming Language*. *C#* dikembangkan oleh Microsoft oleh tim yang dipimpin oleh *Anders Hejlsberg* dan *Scott Wiltamuth*. *C#* memiliki kesamaan bahasa dengan *C, C++,* dan *Java,* sehingga memudahkan *developer* yang sudah terbiasa dengan bahasa *C* untuk menggunakannya, *C#* mengambil fitur-fitur terbaik dari ketiga bahasa tersebut dan juga menambahkan fitur-fitur baru. *C#* merupakan bahasa pemrograman *Object Oriented* dan memiliki *class library* yang sangat lengkap yang berisi *pre-built* component sehingga memudahkan programer untuk men-*develop* program lebih cepat. *C#* juga distandarkan oleh *Ecma International* pada bulan desember 2002.

Pada akhir tahun 2005 *Microsot* merilis *.NET Framework* 2.0 bersamaan dengan paket *Visual* Studio. Otomatis versi dari *C#* juga diperbaharui menjadi *C# 2.0* yang berjalan diatas *.NET Framework 2.0*. Pada versi baru ini banyak sekali fitur-fitur yang ditambahkan terutama pada pengembangan aplikasi berbasis *web* dengan *ASP.NET* seperti (*master page, site map control, user login,* dll), juga penambahan *generic collection* yang sangat membantu programer bekerja dengan *object-object collection* dan *list* (*Budi Hartanto, 2008*).

2.10.1 Definisi dan deklarasi *variabel*

Variabel adalah suatu pengenal (*identifier*) yang digunakan untuk mewakili suatu nilai tertentu di dalam proses program. Berbeda dengan konstanta yang nilainya selalu tetap, nilai dari suatu *variable* bisa diubah-ubah sesuai kebutuhan. Nama dari suatu variable dapat ditentukan sendiri oleh pemrogram dengan aturan sebagai berikut :

1. Terdiri dari gabungan huruf dan angka dengan karakter pertama harus berupa huruf.

2. Bahasa *C* bersifat *case-sensitive* artinya huruf besar dan kecil dianggap berbeda. Jadi antara Metal, dengan metal itu berbeda.

3. Tidak boleh mengandung spasi.

4. Tidak boleh mengandung simbol-simbol khusus, kecuali garis bawah (*underscore*), seperti : *$, ?, %, #, !, &, *, (,), -, +,* dsb.

5. Panjangnya bebas, tetapi hanya 32 karakter pertama yang terpakai.

Contoh penamaan yang salah : NIM, a, x, nama_mhs, f3098, f4, nilai, budi, dsb.

2.10.2 Definisi dan deklarasi *method* dan *class*

Method disebut fungsi atau *subroutine*). Setiap program C# harus memiliki method ini. Anggaplah *method Main*() sebagai pintu masuk program anda. Kalau anda mencoba mengganti nama *Main*() dengan nama lain, *compiler* akan mengeluh bahwa tidak ada *entry point* atau pintu masuk. Perlu juga diketahui bahwa isi atau

tubuh method harus diawali dengan { dan diakhiri dengan }. Ini adalah kutipan *method Main*() yang lengkap:

```
static void Main()
{
// tubuh atau isi method
}
```

Mungkin tidak mudah untuk mengerti arti dari sebuah class dan object tetap penulis akan mencoba menerangkannya , sebenarnya konsep class dan object adalah menghubungkan satu dan yang lainnya dan biasanya para pemula dalam pemprogramman tidak mengerti dan tidak peduli dan hanya berpikir hal ini menghabiskan waktu dalam mempelajari *C#.*

2.10.3 Definisi dan deklarasi *Array*

Array adalah kumpulan elemen *bertype* sama, yang mempunyai sebuah nama (nama *Array*) dan setiap elemen dapat diacu melalui indeksnya. *Array* dengan satu *indeks* disebut *array* berdimensi satu, *vektor*, larik atau *table* sedangkan *array* dengan dua indeks disebut array dua dimensi atau *matrik. Array* dapat mempunyai dimensi lebih dari dua. Yang harus diperhatikan adalah :

- Nama *Array* (seluruh elemen)
- Dimensi *Array* (banyak indeks)
- Ukuran *Array* atau batas nilai indeks

Array dapat didefinisikan secara statik atau secara dinamik. *Array statik* adalah *array* yang ukurannya ditentukan saat kompilasi, sedangkan *array dinamik* adalah *array* yang ukurannya didefinisikan pada saat run time dengan perintah alokasi memori.

2.10.4 Build dan debug

Build adalah tingkat-tinggi membangun sistem untuk mengelola C# dan proyek *C*. Hal ini didasarkan pada file-file konfigurasi yang adalah file teks sederhana. Dalam bertentangan dengan lain membangun sistem file konfigurasi compiler dan *platform*-independen dan hanya berisi sebanyak mungkin informasi mutlak diperlukan untuk membangun sistem untuk mengetahui bagaimana untuk menghasilkan binari. Sebagai *build* mendukung banyak kompiler dan platform keluar dari kotak pilihan dan diterjemahkan dalam file konfigurasi untuk opsi baris perintah yang diharapkan oleh sebuah kompiler yang dipilih file-file konfigurasi ditulis sekali dan digunakan di mana-mana. Program dilaksanakan di standar C# atau C standar dapat dengan mudah dibangun dengan build pada *platform* yang berbeda tidak peduli apa *compiler* yang digunakan.

Debug adalah proses menemukan dan memperbaiki atau melewati *bugs (error)* dalam program kode komputer atau rekayasa perangkat keras. Untuk *debug* atau perangkat keras program ini adalah untuk mulai dengan masalah, mengisolasi sumber masalah, dan kemudian memperbaikinya.

BAB III

PELAKSANAAN PENELITIAN

3.1 Persiapan

Sebelum melakukan sebuah penelitian diperlukan persiapan yang cukup untuk mendapatkan hasil yang optimal dalam proses penelitian, yaitu :

3.1.1 Materi Penelitian

Adapun materi yang digunakan sebagai bahan dalam penelitian ini meliputi data interior orientasi, data parameter eksterior orientasi yang disesuaikan dengan batasan penelitian ini :

1. Data Parameter Intrinsik

   Parameter interior mendefinisikan geometri internal sebuah kamera antara lain *perspektif center*, *principle point*, serta panjang fokus dari kamera.

2. Data Parameter Ekstrinsik

   Parameter eksterior terdiri dari 6 parameter diantaranya 3 parameter rotasi yaitu rotasi *omega* terhadap sumbu x, rotasi *phi* terhadap sumbu y, dan rotasi *kappa* terhadap sumbu z. Selain itu terdapat 3 parameter posisi kamera yaitu X, Y, dan Z.

3. Data Parameter Distorsi

   Parameter distorsi teridiri 7 parameter yaitu 3 parameter distorsi radial (K_1, K_2, K_3), 2 parameter Distorsi dicentring (P_1, P_2), dan 2 parameter Affinity (b_1, b_2).

3.1.2 Alat Penelitian

Adapun alat dan bahan yang dibutuhkan dalam proses penelitian ini baik itu perangkat lunak (*software*) maupun perangkat keras (*hardware*) antara lain :

1. Perangkat Lunak

- Microsoft Visual Studio 2010
- Photomodeler Professional v5.2.32.
- Australis V6.5 atau 7.06

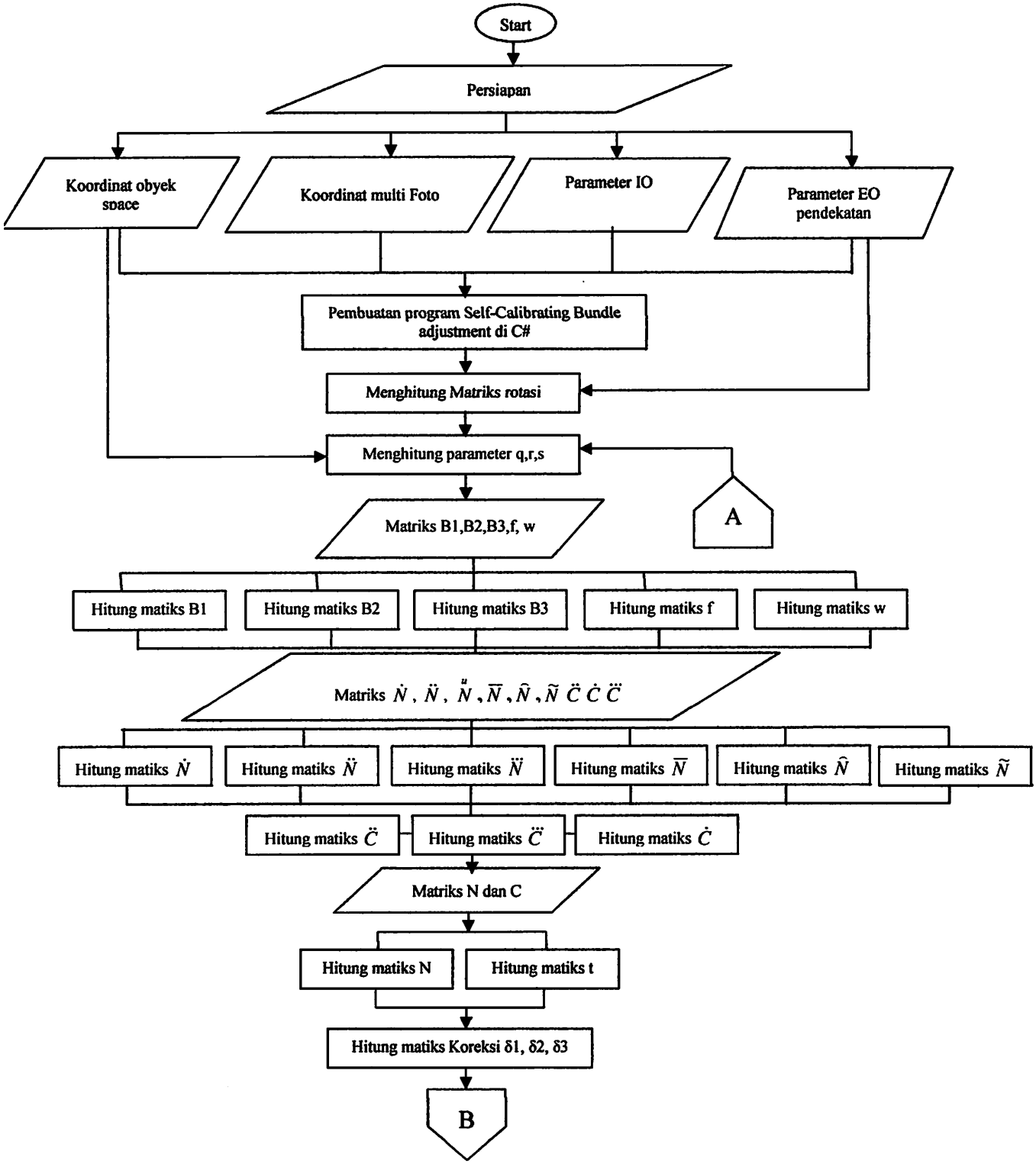
2. Perangkat Keras

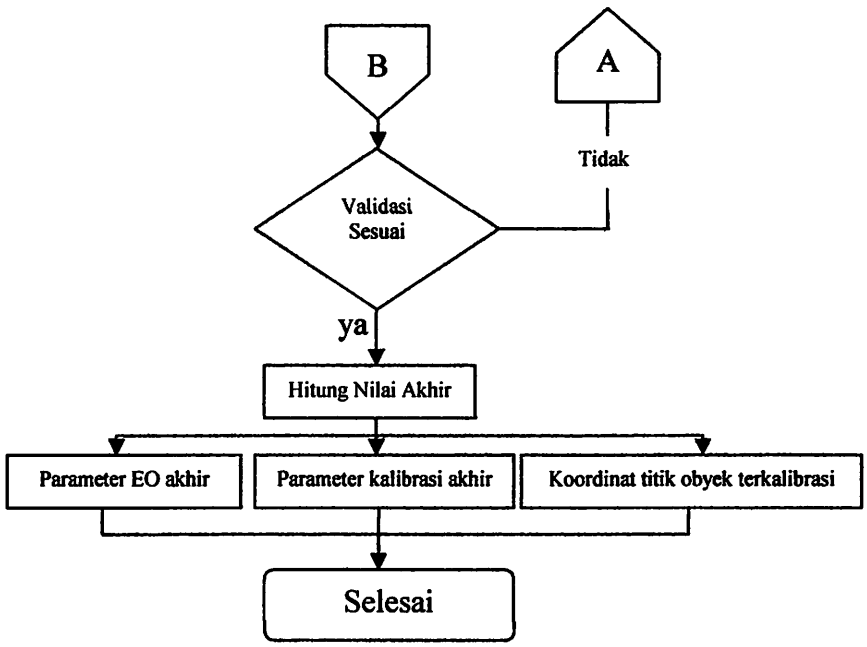
- Perangkat komputer Intel Core i-3

3.2 Langkah Penelitian

Dalam proses penelitian haruslah dibuat suatu kerangka pekerjaan yang sistematis agar mudah dipahami dan mempermudah dalam penelitian. Adapun langkah atau alur penelitian yang akan dilakukan sebagai berikut :

Diagram alir penelitian :


Start
Persiapan
Koordinat obyek space
Koordinat multi Foto
Parameter IO
Parameter EO pendekatan
Pembuatan program Self-Calibrating Bundle adjustment di C#
Menghitung Matriks rotasi
Menghitung parameter q,r,s
A
Matriks B1,B2,B3,f, w
Hitung matiks B1
Hitung matiks B2
Hitung matiks B3
Hitung matiks f
Hitung matiks w
Matriks $\dot{N}$, $\ddot{N}$, $\overset{u}{N}$, $\overline{N}$, $\hat{N}$, $\tilde{N}$ $\ddot{C}$ $\dot{C}$ $\ddot{C}$
Hitung matiks $\dot{N}$
Hitung matiks $\ddot{N}$
Hitung matiks $\ddot{N}$
Hitung matiks $\overline{N}$
Hitung matiks $\hat{N}$
Hitung matiks $\tilde{N}$
Hitung matiks $\ddot{C}$
Hitung matiks $\ddot{C}$
Hitung matiks $\dot{C}$
Matriks N dan C
Hitung matiks N
Hitung matiks t
Hitung matiks Koreksi δ1, δ2, δ3
B



3.3 Penjelasan Diagram Alir (*Flow Chart*) Penelitian

Aktivitas penelitian ini dimulai dengan tahap persiapan alat, bahandan perijinan guna untuk kelancaran dalam proses penelitian. Pengambilandata foto / akusisi data foto sebagai data dasar dalam proses pendesainan perhitungan bundle adjustment dalam close range photogrammetry Langkah selanjutnya, dilakukan sesuai dengan diagram alir penelitian ini.

3.3.1 Persiapan Data Parameter Awal.

Mengumpulkan dan menghitung data parameter-parameter awal guna memperlancar proses *Self-calibrating bundle adjustment*, dimana diantaranya adalah.

1 Koordinat multi foto (x,y) dan ketelitian (sx, sy) Proses ekstrasi data foto dilakukan untuk mendapatkan nilai-nilai parameter antara lain : koordinat foto (x,y) serta keakurasian titik-titik foto (sx, sy) menggunakan Software fotogrammetry yaitu Software Australis v6.5.

2 Parameter interior orientation (IO) dan Parameter Kalibrasi parameter kalibrasi dan parameter IO telah diketahui dari perangkat lunak fotogrametri seperti Australis dan photomodeler, parameter x_0, y_0, dan c dianggap fix untuk semua foto.

3 Parameter exterior orientation (EO) pendekatan untuk 2 foto yaitu omega (ω),phi(φ), kappa (κ), XL, YL, ZL, dan parameter koordinat titik-titik objek 3D untuk 2 foto yaitu Xi, Yi, dan Zi Diperoleh melalui proses Relatif orientasi(RO) di Australis V6.5.

4 Parameter exterior orientation (EO) pendekatan untuk multi foto yaitu omega (ω),phi(φ), kappa (κ), XL, YL, ZL,. Diperoleh melalui proses Resection (RO) di Australis V6.5. Koordinat multi foto titik-titik objek(X,Y,Z)i pendekatan. Diperoleh menggunakan proses intersection(Triangulate) di Australis V6.5.

3.3.2 Menghitung Matrik Rotasi (R)

Untuk menghitung matrik rotasi ini, peneliti menggunakan data omega omega (ω),phi(φ), kappa (κ), dan diproses menggunakan Persamaan (2.8) berikut:

$$R_{\omega\varphi\kappa} = \begin{vmatrix} \cos\varphi\cos\kappa & \sin\omega\sin\varphi\cos\kappa+\cos\omega\sin\kappa & -\cos\omega\sin\varphi\cos\kappa+\sin\omega\sin\kappa \\ -\cos\varphi\sin\kappa & -\sin\omega\sin\varphi\sin\kappa+\cos\omega\cos\kappa & \cos\omega\sin\varphi\sin\kappa+\sin\omega\cos\kappa \\ \sin\varphi & -\sin\omega\cos\varphi & \cos\omega\cos\varphi \end{vmatrix}$$

3.3.3 Menghitung Parameter q,r,s

Parameter qrs dihitung menggunakan data parameter matrik rotasi (R) dan koordinat titik objek pendekatan, dengan menggunakan Persamaan (2.22, 2.23, 2.24) sebagai berikut :

$$r = m_{11}(X_A - X_L) + m_{12}(Y_A - Y_L) + m_{13}(Z_A - Z_L)$$

$$s = m_{21}(X_A - X_L) + m_{22}(Y_A - Y_L) + m_{23}(Z_A - Z_L)$$

$$q = m_{31}(X_A - X_L) + m_{32}(Y_A - Y_L) + m_{33}(Z_A - Z_L)$$

3.3.4 Menghitung Matrix B1, B2, B3, f, w

Data yang digunakan dalam menghitung matrik B1, B2, B3, f, w, adalah parameter EO, koordinat titik objek pendekatan(Xi,Yi,Zi) koordinat foto (x,y) dan ketelitian (sx, sy) serta parameter IO (x_0,y_0, focus),matrik rotasi (R) , parameter q,r,s dan Parameter. Dimana proses perhitungannya menggunakan Persamaan (2.53, 2.55, 2.56, 2.59, 2.40) sebagai berikut :

$$\underset{(2,6)}{B_{1ij}} = \begin{bmatrix} (b_{11})_0 & (b_{12})_0 & (b_{13})_0 & (-b_{14})_0 & (-b_{15})_0 & (-b_{16})_0 \\ (b_{21})_0 & (a_{22})_0 & (b_{23})_0 & (-b_{24})_0 & (-b_{25})_0 & (-b_{26})_0 \end{bmatrix}_{ij}$$

$$\underset{(2,3)}{B_{2ij}} = \begin{bmatrix} (b_{14})_0 & (b_{15})_0 & (b_{16})_0 \\ (b_{24})_0 & (b_{25})_0 & (b_{26})_0 \end{bmatrix}_{ij}$$

$$B_3 = \begin{pmatrix} -1 & 0 & -\frac{\bar{x}}{c} & \bar{x}r^2 & \bar{x}r^4 & \bar{x}r^6 & 2\bar{x}^2 + r & 2\bar{x}\bar{y} & \bar{x} & \bar{y} \\ 0 & -1 & -\frac{\bar{y}}{c} & \bar{y}r^2 & \bar{y}r^4 & \bar{y}r^6 & 2\bar{x}\bar{y} & 2\bar{y}^2 + r & 0 & 0 \end{pmatrix}$$

$$f_{ij} = \begin{bmatrix} J \\ K \end{bmatrix}_{ij}$$

$$W_{ij} = \begin{pmatrix} 1/\sigma_x^2 & 0 \\ 0 & 1/\sigma_y^2 \end{pmatrix}_{ij}$$

3.3.5 *Menghitung Matriks* $\dot{N}, \ddot{N}, \overset{u}{N}, \overline{N}, \ddot{\overline{N}}, \tilde{N}, \ddot{C}, \dot{C}, \ddot{C}$.

Data yang digunakan dalam proses ini adalah matrik B1, B2, B3, f, w yang akan diproses dengan Persamaan (2.63) berikut :

Dimana

$$\underset{(6x6)}{\dot{N}_{ij}} = \sum_{j=1}^{n} B_{1ij}^{T} W_{ij} B_{1ij} \; ; \quad \underset{(6x1)}{\dot{C}_{i}} = \sum_{j=1}^{n} B_{1ij}^{T} W_{ij} f_{ij}$$

$$\underset{(3x3)}{\ddot{N}_{ij}} = \sum_{i=1}^{m} B_{2ij}^{T} W_{ij} B_{2ij} \; ; \quad \underset{(3x1)}{\ddot{C}_{j}} = \sum_{i=1}^{m} B_{2ij}^{T} W_{ij} f_{ij}$$

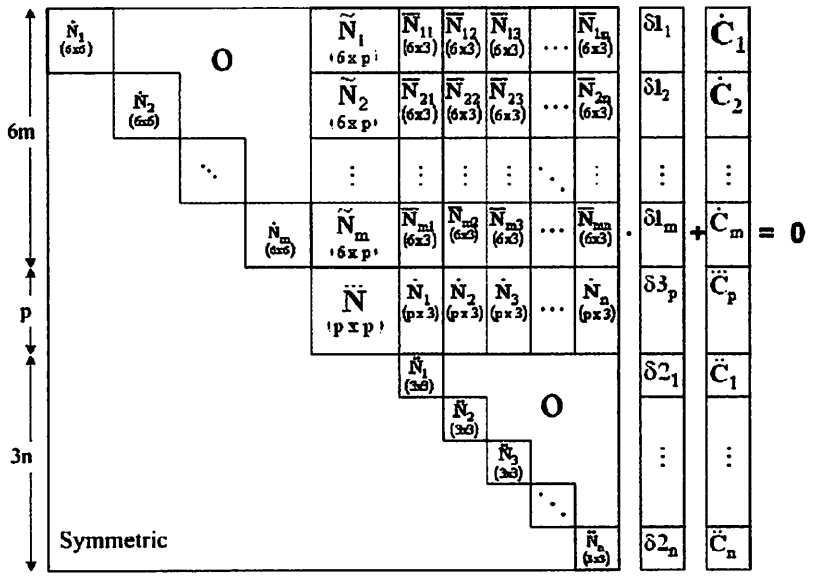
$$\underset{(6x3)}{\overline{N}_{ij}} = B_{1ij}^{T} W_{ij} B_{2ij}$$

$$\underset{(6x p)}{\tilde{N}_{i}} = B_{1ij}{}^{T} P_{ij} B_{3} \qquad \underset{(p x p)}{\ddot{\overline{N}}_{p}} = \sum_{i=1}^{m} \sum_{j=1}^{n} B_{3}{}^{T} P_{ij} B_{3}$$

$$\underset{(p x n)}{\overset{u}{N}_{j}} = B_{3}{}^{T} P_{ij} B_{2ij} \qquad \underset{(p x 1)}{\ddot{C}_{p}} = \sum_{i=1}^{m} \sum_{j=1}^{n} B_{3}{}^{T} P_{ij} w_{ij}$$

3.3.6 Menyusun Matrik N dan C

Menyusun Matrik Ndan C untuk memudahkan dalam perhitungan. Matrik N dan C dapat disusun dengan menggabungkan data $\ddot{N}_{ij}$ $\dot{N}_{ij}$ $\overline{N}_{ij}$, $\tilde{N}_{ij}$ $\ddot{\overline{N}}_{p}$ $\dot{C}_{ij}$ $\overset{u}{N}_{ij}$ $\ddot{C}_{ij}$ $\ddot{C}_{p}$ Struktur persamaannya dapat dinyatakan sebagai berikut (Fraser, 1997) :



3.3.7 Menghitung matrik Koreksi (δ1, δ2 dan δ3)

Setelah disusun matrik N dan C, maka nilai koreksi untuk tiap-tiap Parameternya dapat dihitung. Dengan menggunakan Persamaan (2.61) sebagai berikut :

$$\begin{pmatrix}\delta_1\\ \delta_3\\ \delta_2\end{pmatrix} = \begin{pmatrix}\dot{N}_{ij} & \tilde{N}_i & \overline{N}_{ij}\\ \tilde{N}_i^{\,T} & \ddot{N}_p & \hat{N}_j\\ \overline{N}_{ij}^{\,T} & \hat{N}_j^{\,T} & \ddot{N}_{ij}\end{pmatrix}^{-1} - \begin{pmatrix}\dot{C}_i\\ \ddot{C}_p\\ \ddot{C}_j\end{pmatrix}$$

maka akan diperoleh nilai koreksi δ1 yaitu nilai koreksi untuk parameter EO tiap-tiap foto , δ2 yaitu koreksi untuk koordinat titik-titik objek dan δ3 yaitu koreksi untuk parameter kalibrasi.

3.3.8 Validasi

Validasi data hasil Perhintungan *Self-calibrating Bundle Adjustment* dengan menggunakan bahasa pemrograman *Visual studio C# 2010* yaitu melakukan perbandingan antara hasil perhitungan yang didapat dengan data input dan standart deviasi dari perhitungan itu sendiri.

3.3.9 Menghitung Nilai Akhir.

Menghitung nilai akhir, dimana proses Self-Calibrating bundle adjustment akan mendapatkan nilai parameter EO, koordinat titik objek yang sudah terkoreksi dan parameter kalibrasi. Dimana penyelesaiannya menambahkahkan parameter koreksi terhadap parameter awal, seperti berikut:

$X = X^{O} + \delta.$

BAB IV

HASIL DAN PEMBAHASAN

Untuk mengetahui tingkat keberhasilan dari penelitian, perlu dilakukan proses analisa atau pembahasan dari sebuah hasil yang telah dicapai selama proses pelaksanaan penelitian. Parameter keberhasilan dapat diukur dengan membandingkan tujuan dari penelitian dan hasil yang dicapai. Adapun beberapa parameter yang telah dihasilkan selama pelaksanaan penelitian akan disajikan secara jelas dalam bab ini.

4.1. Data Parameter Awal *Self-Calibrating Bundle Adjustment*

4.1.1 Parameter IO (*Interior Orientation*) dan Koordinat foto

Parameter IO (*interior orientation*) yaitu x_0, y_0, dan panjang focus(f) koordinat foto (x, y) dan ketelitian (sx,sy) ditunjukkan pada tabel 4.1, 4.2, 4.3, 4.4, dan tabel 4.5.

Tabel 4.1 parameter IO

Parameter	Nilai
F	18.556
Xo	0
Yo	0

Tabel 4.2 Data koordinat foto1(x,y) dan ketelian koordinat foto1(sx,sy)

No.	x(mm)	y(mm)	sx(mm)	sy(mm)
1	-4.98644	2.562080	0.00001	1.2
2	-1.07506	2.686070	0.00001	0.00001
3	3.324880	2.466660	-0.5	-3.1
4	-5.49710	1.142620	0.00001	0.4
5	-1.52819	1.395720	0.00001	0.2
6	3.130430	1.340450	-0.2	-1.2
7	-3.62480	-0.11628	0.1	0.2
8	2.789350	-0.37067	0.00001	1
9	-4.34598	-2.01395	0.00001	-0.3
10	2.249910	-3.53328	0.00001	0.1

Tabel 4.3 Data koordinat foto2(x,y) dan ketelian koordinat foto2(sx,sy)

No.	x(mm)	y(mm)	sx(mm)	sy(mm)
1	-5.44213	3.229290	0.00001	1.2
2	-2.59421	0.795600	0.00001	0.00001
3	0.299030	-2.09597	-0.5	-3.1
4	-5.79764	1.808920	0.00001	0.4
5	-3.04766	-0.43694	0.00001	0.2
6	-0.05915	-3.08251	-0.2	-1.2
7	-4.47723	-0.62055	0.1	0.2
8	-0.49305	-4.23726	0.00001	1
9	-4.90060	-1.77552	0.00001	-0.3
10	-1.05342	-5.95397	0.00001	0.1

Tabel 4.4 Data koordinat foto3(x,y) dan ketelian koordinat foto3(sx,sy)

No.	x(mm)	y(mm)	sx(mm)	sy(mm)
1	-6.55729	1.92161	0.00001	1.1
2	1.05927	2.33825	0.007	0.08
3	4.62660	2.23324	-0.5	0.5
4	-5.12756	-0.99907	0.00001	0.4
5	1.97363	0.80649	0.00001	0.2
6	5.17800	1.34483	0.08	0.5
7	0.59782	-1.73893	0.1	0.2
8	5.69702	0.03863	0.00001	0.003
9	0.97763	-4.66638	-0.3	0.00001
10	6.37421	-2.30980	0.00001	0.1

Tabel 4.5 Data koordinat foto4(x,y) dan ketelian koordinat foto4(sx,sy)

No.	x(mm)	y(mm)	sx(mm)	sy(mm)
1	-5.57849	-3.20283	1.02	0.008
2	0.44613	-0.60749	0.0001	0.00001
3	3.99047	0.62584	-0.5	-3.1
4	-4.57613	-4.98377	0.0001	0.4
5	0.96635	-1.86894	0.00001	0.2
6	4.27488	-0.16436	-0.2	-1.2
7	-0.54182	-4.20498	0.1	0.03
8	4.44164	-1.21684	0.0001	0.07
9	-0.52412	-5.90092	0.00001	-0.3
10	4.54693	-2.87018	0.02	0.2

4.1.2 Parameter *object space point*

Parameter object space point yaitu *X, Y, dan Z.* dan ketelitian *object space* (*SX,SY SZ*) ditunjukkan pada tabel 4.6.

Tabel 4.6 parameter koordinat *object space*

No.	X(mm)	Y(mm)	Z(mm)	SX(mm)	SY(mm)	SZ(mm)
1	-0.248212	0.127585	-0.924662	0.00001	1.2	0.00001
2	-0.043758	0.108611	-0.750067	0.00001	0.00001	0.00001
3	0.109730	0.081328	-0.610558	-0.5	-3.1	-0.5
4	-0.264773	0.055156	-0.894553	0.00001	0.4	0.00001
5	-0.060100	0.054513	-0.725221	0.00001	0.2	0.00001
6	0.100197	0.042822	-0.593114	-0.2	-1.2	-0.2
7	-0.151294	-0.004943	-0.773228	0.1	0.2	0.1
8	0.087079	-0.011657	-0.579469	0.00006	0.002	0.00002
9	-0.180456	-0.083628	-0.770431	0.00004	-0.3	0.00001
10	0.068379	-0.107470	-0.564724	0.00001	0.1	0.00001

4.1.3 Parameter EO (*Exterion Orientation*)

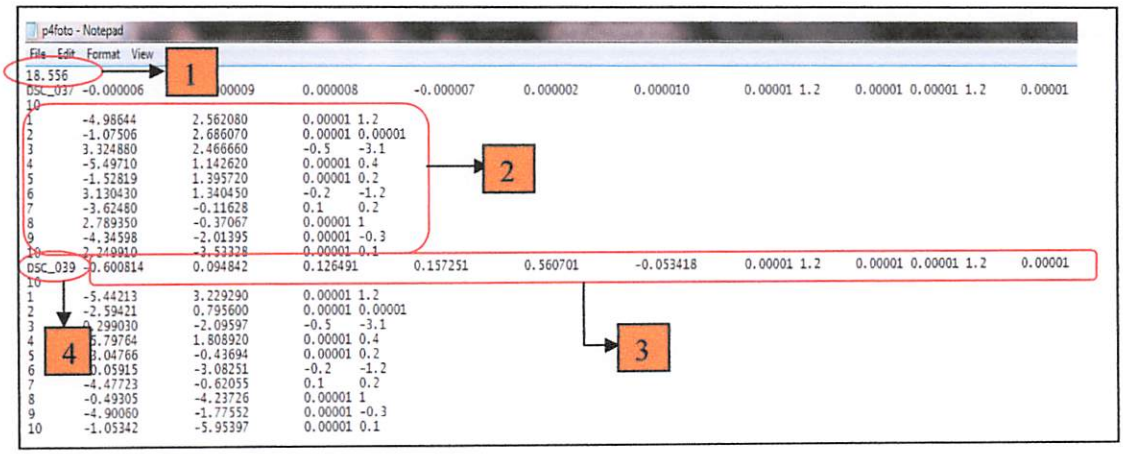
Parameter EO (*exterion orientation*) yaitu ω, φ, κ, X_L, Y_L, Z_L. ditunjukkan pada tabel 4.7.

Tabel 4.7. parameter EO (*exterior Orientation*)

Photo	ω (rad)	φ (rad)	κ (rad)	X_L (mm)	Y_L(mm)	Z_L(mm)
Img1	-0.000006	-0.000009	0.000008	-0.000007	0.000002	0.000010
Img2	0.600814	0.094842	0.126491	0.157251	0.560701	-0.053418
Img3	-0.889227	-1.386234	-1.013069	-0.680837	0.125193	-0.721659
Img4	-1.214648	-0.869058	-1.187392	-0.640585	0.611842	-0.592235

4.1.4 Format Data Input

Data inputan pada Program ini adalah panjang focus, parameter EO pendekatan, *object space* pendekatan, dan titik-titik koordinat foto. Keempat data ini akan menjadi data inputan dalam program self-calibrating bundle adjustment dalam bentuk notepad (*file* dalam bentuk format *.txt*), seperti dibawah ini:



Gambar 4.1 Format data input (panjang fokus, parameter EO, dan koordinat foto).

Keterangan :

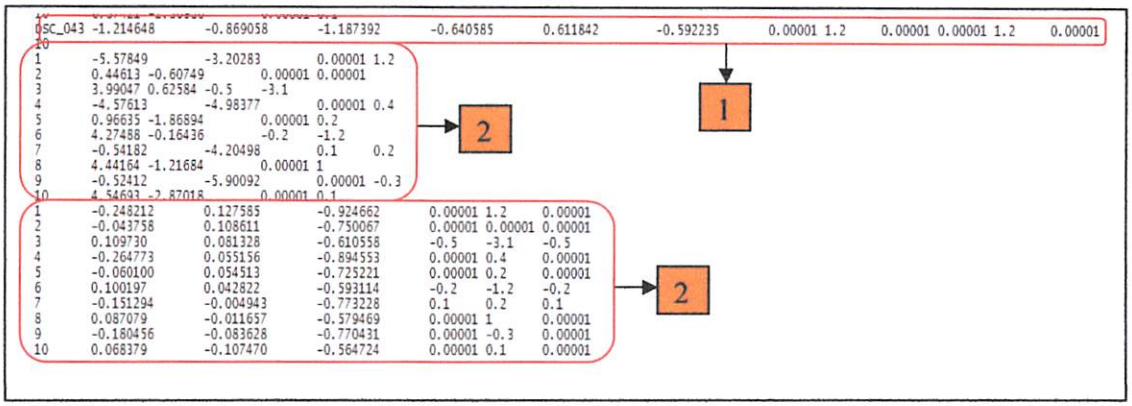
1 = Panjang focus.

2 = Koordinat foto (Id, x,y, sx,sy).

3 = Parameter EO (ω, φ, κ, X_L, Y_L, Z_L, $s\omega$, $s\varphi$, $s\kappa$, sX_L, sY_L, sZ_L).

4 = ID foto.

Dari Gbr 4.1 setelah data panjang fokus, data parameter EO dan koordinat foto beserta standar error di inputkan maka yang teakhir inputkan data koordinat object space point 3D pada bagian akhir. Seperti pada gambar dibawah ini:



Gambar 4.2 parameter koordinat object space point 3D, parameter EO dan koordinat Foto.

Keterangan :

1 = Parameter EO (ω, φ, κ, X_L, Y_L, Z_L, $s\omega$, $s\varphi$, $s\kappa$, sX_L, sY_L, sZ_L).

2 = Koordinat foto (Id, x,y, sx,sy).

3 = Koordinat Object Space point 3D (Xi,Yi,Zi,SXi, SYi, SZi)

4.2. Hasil Penelitian

Dari hasil pelaksanaan penelitian yang telah dilakukan baik dilapangan maupun dilaboratorium dan dengan menggunakan beberapa jenis data yang telah dikumpulkan. Didapat beberapa parameter berupa parameter kalibrasi kamera, parameter *Exterior Orientation* (EO) yaitu (X_L, Y_L, Z_L, ω, φ, κ) dan parameter *Object space* yaitu (X_A, Y_A, Z_A). Selain parameter-parameter tersebut dihasilkan pula suatu fungsi matematika yang ditulis dalam sebuah bahasa pemrograman *Microsoft Visual Studio C# 2010* untuk dapat dipergunakan secara fleksibel dengan berbagai kondisi data. Keseluruhan data yang dihasilkan, akan ditampilkan pada tiap sub-bab dibawah ini.

4.2.1 Aplikasi dan Listing Kode Pemrograman

Dari setiap tahapan proses penelitian dan perhitungan yang dijelaskan pada *bab-3*, dapat diaplikasikan dalam sebuah bahasa pemrograman. Adapun perangkat lunak yang digunakan dalam pembuatan aplikasi *Program Self-calibrating bundle adjustment* sederhana ini ialah *Visual Studio C# 2010*. Untuk memudahkan dalam proses pembangunan aplikasi perhitungan, maka setiap tahapan dibagi dalam beberapa class yang memiliki fungsi yang berbeda-beda. Beberapa contoh Class tersebut disajikan dalam bentuk tabel seperti yang terlihat dibawah ini.

Tabel 4.8 Nama class, fungsi dan Output Data yang Telah Dibuat

No.	Nama class	Fungsi Class	Data
1	Program.cs	Class utama atau main class untuk menjalankan seluruh class yang ada.	-
2	ParameterEO.cs	Class yang berfungsi untuk menampung semua parameter EO yang diinputkan dan hasil akhir parameter EO.	ω, φ, κ, X_L, Y_L, Z_L.
3	Point3D.cs	Class ini berfungsi untuk menampung semua koordinat *objcet space point 3D.*	*Xi, Yi, Zi*
4	Point2D.cs	Class ini berfungsi untuk menampung semua koordinat *foto 2D.*	*x, y,*
5	SCBA.cs	Class ini berfungsi untuk menghitun data inputan	-

6	Paramkalib.cs	Class ini berfungsi untuk menampung semua Data parameter kalibrasi	x_0, y_0, f, K_1, K_2, K_3, P_1, P_2, b_1, b_2

Keseluruhan fungsi yang telah dibuat dapat dilihat pula pada lampiran. Adapun isi masing-masing fungsi tersebut merupakan suatu kode logika dalam bahasa pemrograman *Visual studio C# 2010.* Dibawah ini adalah tampilan kode dari class *Point2D.cs* sebagai bagian dari Program *Self-Calibrating Bundle Adjustment*

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace SelfCalibrating
{
    class Point2D
    {
        private string label;
        private double x;
        private double y;
        private double sx;
        private double sy;

        public string Label
        {
            get { return label; }
            set { label = value; }
        }
        public double X
        {
            get { return x; }
            set { x = value; }
        }

        public double Y
        {
            get { return y; }
            set { y = value; }
        }
        public double SX
        {
            get { return sx; }
            set { sx = value; }
        }
        public double SY
        {
            get { return sy; }
```

```
            set { sy = value; }
        }
    }
}
```

Dari keseluruhan fungsi tersebut disatukan dalam Class utama atau main class yaitu *Program.Cs* yang menjadi penggerak utama aplikasi atau program yang dibuat. Berikut adalah code dari class utama dalam program perhitungan *Self-Calibrating Bundle Adjustment*.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace SelfCalibrating
{
    class Program
    {
        private const string FILE_NAME = "P4foto.txt";
        static void Main(string[] args)
        {
            SCBA scba = new SCBA(FILE_NAME);
            scba.adjusting();
            Console.ReadLine();
        }
    }
}
```

4.2.2 Hasil Perhitungan *Self-Calibrating Bundle Adjustment*

Pada perhitungan *Self-Calibrating Bundle Adjustment* menggunakan bahasa pemrograman *Visual studio C# 2010* menampilkan hasil yang diperoleh sebagai berikut.

1. PerhitunganKoreksi δ1, δ2 dan δ3

Nilai masing-masing koreksi δ1, δ2 dan δ3 yang diperoleh dari perhitungan menggunakan *console application* dalam bahasa pemograman C# sebagai berikut :

Tabel 4.9 Koreksi Parameter EO Menggunakan *Console Application* Pemrograman C# .

Photo	Koreksi Parameter EO (δ1)					
	ω(rad)	*φ(rad)*	*κ(rad)*	X_L*(mm)*	Y_L*(mm)*	Z_L*(mm)*
1	0.000110	-0.02460	0.00095	-0.019369	-0.0066	-0.002588
2	-0.00225	-0.0029	0.00368	0.00070	-0.00746	-0.0046
3	-0.01312	0.00625	-0.0147	0.001503	-0.00054	0.00351
4	-0.0099	0.0079	-0.0040	0.00657	-0.00149	-0.0009

Tabel 4.10 Koreksi Parameter Kalibrasi Menggunakan *Console Application.*

Koreksi Parameter Kalibrasi Kamera(δ3)	
Parameter	Koreksi(mm)
x_0	-0.0655510571042726
y_0	0.123785338335779
fokus	-0.00556961752359551
K_1	-0.000197810068265215
K_2	-4.13269128492775E-06
K_3	5.70691328083499E-08
P_1	0.000373815760261895
P_2	-0.000356122105269799
b_1	0.00293170603872639
b_2	-0.00306740788908044

Tabel 4.11 Koreksi Parameter Koordinat *Object Space Point*

Koreksi Parameter Koordinat Object(δ2)			
Titik	X	Y	Z
1	-0.000619	0.00052044	-0.000697075
2	0.00065363	-0.0002112	0.0004522
3	-0.0001369	0.0001738	0.000102356
4	-0.0004239	-0.0001568	0.000128223
5	0.0006887	-0.0001684	0.00015399
6	-0.0002513	0.0001178	-0.0001666
7	0.000748	-7.990E-05	0.0003586
8	-0.0003614	9.4559E-05	-0.00036
9	0.000165	-0.00023	0.00024447
10	-0.0004620	-5.927E-05	-0.000216

Sebagaimana dalam Tabel 4.9, Tabel 4.10 dan tabel 4.11, didapat nilai selisih terbesar untuk parameter EO yaitu 0.0246 mm dan yang terkecil 0.00011 mm dengan rata-rata selisih sebesar 0.00609 mm. Sedangkan selisih terbesar untuk parameter OSP yaitu 0.00074 mm dan yang terkecil 0.000059 mm dengan rata-rata selisih sebesar 0.000306 mm.

2. Hasil pehitungan Nilai akhir parameter

Nilai masing-masing koreksi δ1, δ2 dan δ3 yang diperoleh dari perhitungan menggunakan *console application* dalam bahasa pemograman C# dijumlahkan dengan parameter awal sebagai hasil akhir Parameter EO(ω, φ, κ, X_L, Y_L, Z_L), Parameter Kalibrasi(x_0, y_0,

f, K_1, K_2, K_3, P_1, P_2, b_1, b_2) dan parameter *Object space point* (Xi, Yi , Zi) menggunakan *console application* dalam bahasa pemograman C# sebagai berikut :

Tabel 4.12 Hasil Akhir Parameter Exterior Orientation (EO)

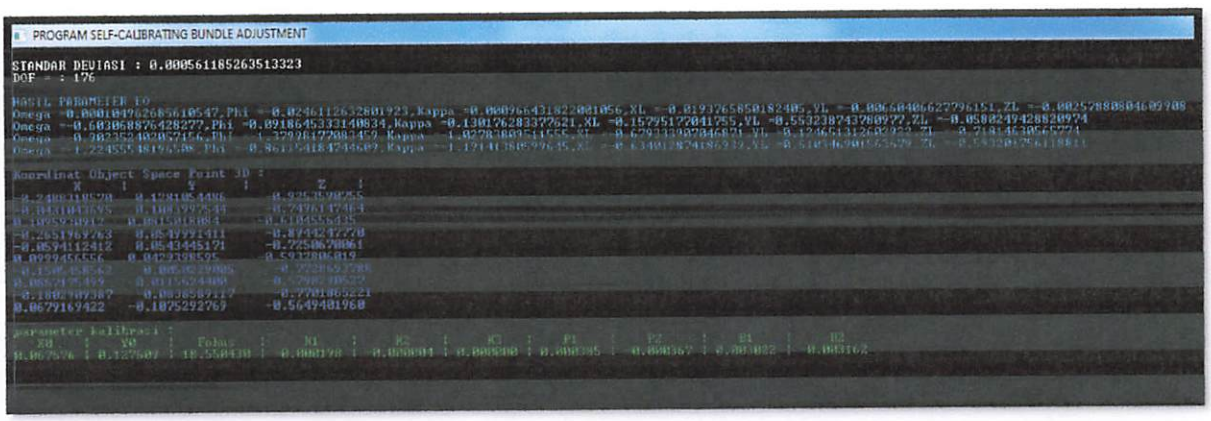
Foto	Omega(rad)	Phi(rad)	Kappa(rad)	XL(mm)	YL(mm)	ZL(mm)
1	0.0001049	-0.02461	0.00096	0.019376	-0.00660	-0.0025
2	-0.603068	0.09186	0.130176	0.15795	0.55323	-0.05802
3	-0.90235	-1.37998	-1.027838	0.67933	0.124651	-0.71814
4	-1.224555	-0.86115	-1.191413	0.6340128	0.6103469	-0.59320

Tabel 4.13 Hasil Akhir Parameter Kalibrasi

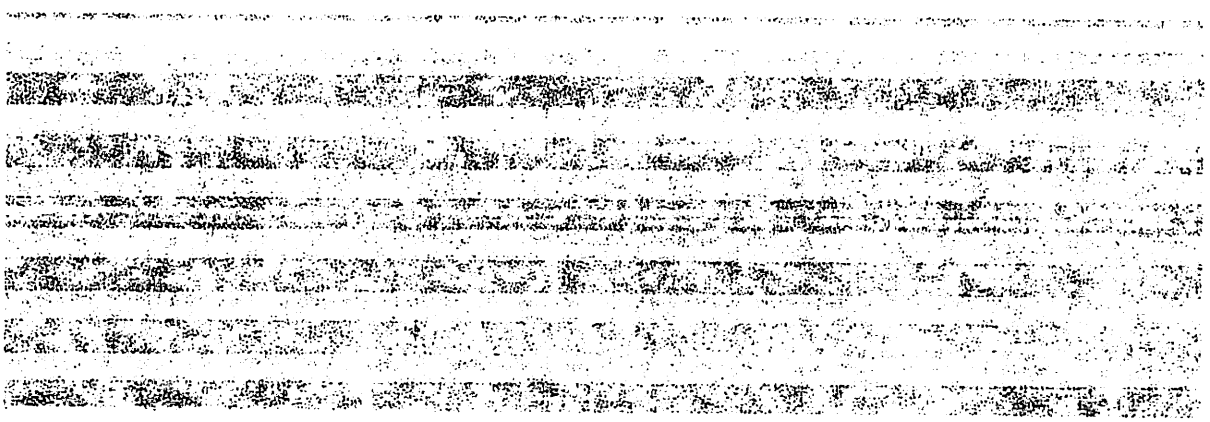
Parameter Kalibrasi Akhir	
Parameter	Nilai akhir(mm)
x_0	-0.0655510571042726
y_0	0.127608930018452
fokus	18.5504303824764
K_1	-0.000191883014985692
K_2	-4.0088619994253E-06
K_3	5.70691328083499E-08
P_1	0.000373815760261895
P_2	-0.000356122105269799
b_1	0.00293170603872639
b_2	-0.00306740788908044

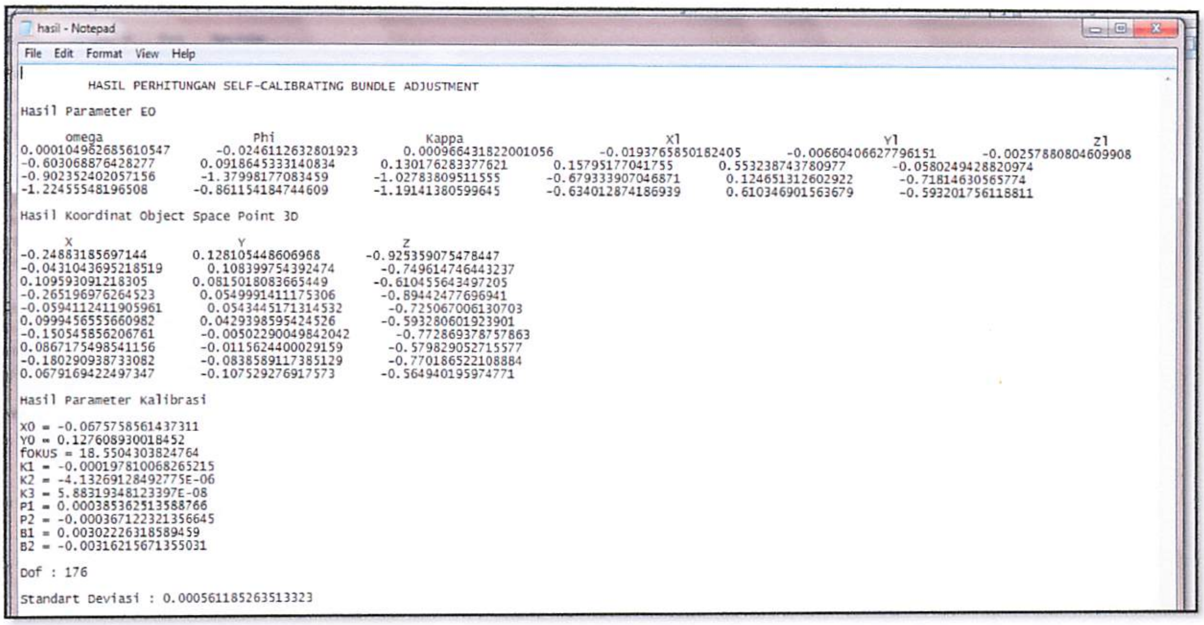
Tabel 4.14 Hasil Akhir Parameter *object space point*

Titik	X(mm)	Y(mm)	Z(mm)
1	-0.2488318	0.1281054	-0.925359
2	-0.0431043	0.1083997	-0.7496147
3	0.109593	0.0815018	-0.6104556
4	-0.2651969	0.05499914	-0.89442
5	-0.0594112	0.0543445	-0.725067
6	0.0999456	0.042939	-0.59328
7	-0.1505458	-0.00502	-0.772869
8	0.0867175	-0.011562	-0.579829
9	-0.1802909	-0.0838589	-0.7701865
10	0.0679169	-0.10752927	-0.564940



Gambar. 4.3 Hasil tampilan data output pada Console Application.





hasil - Notepad

File Edit Format View Help

HASIL PERHITUNGAN SELF-CALIBRATING BUNDLE ADJUSTMENT

Hasil Parameter EO

omega	Phi	Kappa	X1	Y1	Z1
0.000104962685610547	-0.0246112632801923	0.000966431822001056	-0.0193765850182405	-0.00660406627796151	-0.00257880804609908
-0.603068876428277	0.0918645333140834	0.130176283377621	0.15795177041755	0.553238743780977	-0.0580249428820974
-0.902352402057156	-1.37998177083459	-1.02783809511555	-0.679333907046871	0.124651312602922	-0.71814630565774
-1.22455548196508	-0.861154184744609	-1.19141380599645	-0.634012874186939	0.610346901563679	-0.593201756118811

Hasil Koordinat Object Space Point 3D

X	Y	Z
-0.24883185697144	0.128105448606968	-0.925359075478447
-0.0431043695218519	0.108399754392474	-0.749614746443237
0.109593091218305	0.0815018083665449	-0.610455643497205
-0.265196976264523	0.0549991411175306	-0.89442477696941
-0.0594112411905961	0.0543445171314532	-0.725067006130703
0.0999456555660982	0.0429398595424526	-0.593280601923901
-0.150545856206761	-0.00502290049842042	-0.772869378757863
0.0867175498541156	-0.0115624400029159	-0.579829052715577
-0.180290938733082	-0.0838589117385129	-0.770186522108884
0.0679169422497347	-0.107529276917573	-0.564940195974771

Hasil Parameter Kalibrasi

X0 = -0.0675758561437311
Y0 = 0.127608930018452
fOKUS = 18.5504303824764
K1 = -0.000197810068265215
K2 = -4.13269128492775E-06
K3 = 5.88319348123397E-08
P1 = 0.000385362513588766
P2 = -0.000367122321356645
B1 = 0.00302226318589459
B2 = -0.00316215671355031

Dof : 176

Standart Deviasi : 0.000561185263513323

Gambar. 4.4 Hasil tampilan data output yang di save dalam notepad

4.3 Pembahasan

Pembuatan program *Self-Calibrating bundle adjustment* yang dibuat dengan menggunakan bahasa C# akan berbeda hasil perhitungannya dengan hasil perhitungan *Excel, software Australis* maupun dengan mengunakan bahasa pemrograman yang lain serta algoritma pembuatan program pun jika berbeda akan berbeda pula hasil yang diperoleh. Program *Self-Calibrating bundle adjustment* akan menghasilkan parameter EO terkoreksi parameter kalibrasi dan parameter koordinat *object space point* terkoreksi. Parameter tersebut perlu dilakukan analisis untuk mengetahui selisih antara parameter pendekatan awal dengan parameter yang telah dikoreksi seperti yang telihat pada table 4.9, 4.10 dan 4.11.

Solusi hitung kuadrat terkecil dapat ditentukan dengan menggunakan matriks *inverse Cayley N-1*. Tetapi matrik *N* memiliki *rank defect* sebesar parameter penentuan datum.

Teknik yang paling sesuai didalam kasus ini adalah teknik yang dapat mengoptimalkan tingkat keakurasian koordinat titik-titik obyek. Dengan kata lain, harus dipilih suatu teknik yang dapat meminimumkan matrik kovarian dari titik-titik objek.

4.4. Keunggulan dan Kelemahan Program *Self-Calibrating Bundle Adjustment.*

Keunggulan menggunakan Program *Self-Calibrating Bundle Adjustment* ini adalah sebagai berikut :

1. Mudah untuk melakukan perhitungan *Self-Calibrating Bundle Adjustment* karena telah tersedia algoritma didalam program ini yang mudah digunakan untuk perhitungan *Self-Calibrating Bundle Adjustment.*
2. Penggunaan waktu yang lebih efisien saat proses perhitungan karena keseluruhan tahap dapat dihitung secara otomatis setelah listing kode disusun secara benar.

Sedangkan kelemahan dari pengunaan program *Self-Calibrating Bundle Adjustment* ini yaitu harus menyesuaikan data input sesuai dengan format data input dan juga parameter EO (ω, φ, κ) sudah harus dalam satuan radian.

BAB V

PENUTUP

5.1 Kesimpulan

Setelah penulis melakukan seluruh rangkaian penelitian dan menyelesaikan penulisan laporan dalam rangkaian "Pembuatan Program *Self-calibrating Bundle Adjustment Multi Photo Konvergen* Dengan Bahasa C#" dapat ditarik beberapa kesimpulan sebagai berikut:

1. Program *Self-Calibrating Bundle Adjustment* membutuhkan data input awal yaitu parameter IO (x_0, y_0, fokus) yang di anggap tetap untuk semua foto, parameter EO masing-masing kamera, parameter koordinat *object space point*, dan koordinat foto masing-masing foto.
2. Program *Self-Calibrating Bundle Adjustment* menghasilkan parameter EO terkoreksi, parameter koordinat *object space point* terkoreksi, dan parameter kalibrasi.
3. Penggunaan waktu yang lebih efisien saat proses perhitungan karena keseluruhan tahap dapat dihitung secara otomatis setelah listing kode disusun secara benar.

5.2 Saran

Dalam pengerjaan penelitian Tugas Akhir ini tentu nya penulis banyak mendapat masalah teknis maupun non-teknis didalam pembuatan program, adapun saran-saran jika sekiranya program perhitungan *Self-Calibrating Bundle*

Adjustment ini akan coba disempurnakan oleh perancang program yang lain, antara lain:

1. Pembuatan program *Self-Calibrating Bundle Adjustment* multi photo dengan menambahkan parameter datum untuk menstabilkan hasil perhitungan.
2. Untuk tujuan memperoleh ketelitian yang relatif lebih tinggi, hendaknya bagi peneliti yang berminat melanjutkan penelitian ini dapat mengembangkan lagi algoritma ini, sehingga mendapatkan nilai presisi yang minimal.
3. Pembuatan program dalam C# ini perlu dikembangkan untuk mendapatkan hasil yang lebih stabil dan mendapatkan presisi minimal dikarenakan didalam program C# ini masih banyak logika-logika yang perlu disempurnakan.

DAFTAR PUSTAKA

Atkinson, K. B. 1987. *Developments in Close Range Photogrammetry-1*. Applied Science Publishers. London.

Atkinson, K. B. 2001. *Close Range Photogrammetry and Mechine Vision.* Whittles Publishing. Scotland, UK.

Australis. 2004. *Users Manual.* Photometrix, Australia.

Bradski, G. and A. Keahler. 2008. *Learning OpenCV.* Gravenstein Highway North, Sebastopol.

Cooper, M.A.R. and Robson, S. 2001. Theory Of Close Range Photogrammetry. Wittles Publishing, London.

Dorstel, C., Jacobsen, K. and Stallmann, D. 2004. *DMC – Photogrammetric Accuracy – Calibration Aspect And Generation Of Synthetic DMC Images.* University of Hannover, Germany.

F. Yilmaztürk, 2011. Full-automatic self-caliration of color digital cameras using color targets. Aksaray University, Geomatics Engineering, Aksaray, Turkey.

Fraser, C. S., M. R. Shortis and G. Ganci. 1995. *Multi-sensor system self-calibration.* Department of Geomatics, University of Melbourne, Parkville 3052, Australia.

Fraser, C. S., 1997. *Digital Camera Self-Calibration.* ISPRS Journal Of Photogrammetry and Remote Sensing, 52(4) : 149-159.

Fraser, C. S., Kenneth L. E. 2000. *Design and Implementation of a Computational Processing System for Off-line Digital Close Range Photogrammetry.* ISPRS Journal of Photogrammetry and Remote Sensing, 55(2): 94-104.

Fraser, C. S. 2006a. *Evolution of Network Orientation Procedures.* ISPRS. XXXVI. 114-120.

Fraser, C. S. 2006b. *Interior Orientation and Network Design.* Lecture Notes 2. University of Melbourne.

Geosystem, L. 2006.Leica *Photogrammetry Suite Project Manager, Leica Geosystems Geospatial Imaging*, United States of America.

J. García-León, A. M. Felicísimo a, J. J. Martínez. 2007 First Experiments With Convergent Multi-Images Photogrammetry With Automatic Correlation Applied To Differential Rectification Of Architectural Façades.

Mason, S.O., 1994. Expert system-based design of photogrammetric networks. Institute for Geodesy and Photogrammetry, Swiss Federal Institute of Technology.

Hanifa, N. R. 2007. *Studi Penggunaan Kamera Digital Low-Cost Non Metric Auto-Focus Untuk Pemantauan Deformasi.* Program Studi Teknik Geodesi dan Geoinformatika, Fakultas Teknik Sipil dan Lingkungan, Institut Teknologi Bandung.

Karara, H. M. 1989 *Non-Topographic Photogrammetry : Second Edition,* American Society for Photogrammetry and Remote Sensing.

King, B. A. 1993. *Methods For The Photogrammetric adjustment of Bundles of Constrained Streospairs.* Departement of Civil Engineering And Surveying. New South Wales, The University of Newcastle.

Mikhail, J. S. Bethel, et al. 2001. *Introduction To Modern Photogrammetry.* New York, John Wiley and Sons, Inc.

Pullivelli, Anoop. 2005. *Low-Cost Digital Cameras: Calibration, Stability Analysis, and Applications.* Tesis Magister Departemen of Geomatics Engineering. University of Calgary.

Shortis, M. R. Clarke, T. A., and Short, T. 1994. *A Comparison Of Some Techniques For The Subpixel Location Of Discrete Target Images.* Videometrics III, SPIE Vol. 2350 239-250.

SmithDev. *Cara Mudah Menguasai Microsoft C# 2008.* Penerbit ANDI OFFSET, Yogyakarta, 2009.

Tjahjadi, M. E. 2008. *Pemetaan Cepat Profil Jalan Dengan Mengintegrasikan Kamera CCTV dan DGPS.* ITN, Malang.

Tjahjadi, M. E. 2008b. *Precesion Feature Extraction Form Unmanned Aerialplatforms*. ITN, Malang.

Tjahjadi, E. 2009. *Precision Feature Extraction from Unmanned Aerial Platforms*. Jurusan Teknik Geodesi, Fakultas Teknik Sipil dan Perencanaan.

Wang, X. and T. A. Clarke. 1998. *Separate Adjustment of Close Range Photogrammetric Measurements*. ISPRS XXXII.

Wolf, P. R. 1980. *Adjustment Computations (Practical Least Square for Surveyors)*. Madison, Wisconsin

Wolf, P. R. 1993. *Element of Photogrammetry, Dengan Interpretasi Foto Udara dan Penginderaan Jauh*. UGM, Yogyakarta.

Wolf, P. R. and Dewitt, B. A. 2000. *Element Of Photogrammetry with Application in GIS*. Mc Graw Hill, New York.

LAMPIRAN A
Data Parameter Awal

Data Parameter Awal

18.556												
DSC_037	-0.000006	-0.000009	0.000008	-0.000007	0.000002	0.00001	0.00001	1.2	0.00001	0.00001	1.2	0.00001
10												
1	-4.98644	2.56208	0.00001	1.2								
2	-1.07506	2.68607	0.00001	0.00001								
3	3.32488	2.46666	-0.5	-3.1								
4	-5.4971	1.14262	0.00001	0.4								
5	-1.52819	1.39572	0.00001	0.2								
6	3.13043	1.34045	-0.2	-1.2								
7	-3.6248	-0.11628	0.1	0.2								
8	2.78935	-0.37067	0.00001	1								
9	-4.34598	-2.01395	0.00001	-0.3								
10	2.24991	-3.53328	0.00001	0.1								
DSC_039	-0.600814	0.094842	0.126491	0.157251	0.560701	-0.053418	0.00001	1.2	0.00001	0.00001	1.2	0.00001
10												
1	-5.44213	3.22929	0.00001	1.2								
2	-2.59421	0.7956	0.00001	0.00001								
3	0.29903	-2.09597	-0.5	-3.1								
4	-5.79764	1.80892	0.00001	0.4								
5	-3.04766	-0.43694	0.00001	0.2								
6	-0.05915	-3.08251	-0.2	-1.2								
7	-4.47723	-0.62055	0.1	0.2								
8	-0.49305	-4.23726	0.00001	1								
9	-4.9006	-1.77552	0.00001	-0.3								
10	-1.05342	-5.95397	0.00001	0.1								
DSC_041	-0.889227	-1.386234	-1.013069	-0.680837	0.125193	-0.721659	0.00001	1.2	0.00001	0.00001	1.2	0.00001
10												
1	-6.55729	1.92161	0.00001	1.2								
2	1.05927	2.33825	0.00001	0.00001								
3	4.6266	2.23324	-0.5	-3.1								
4	-5.12756	-0.99907	0.00001	0.4								
5	1.97363	0.80649	0.00001	0.2								
6	5.178	1.34483	-0.2	-1.2								
7	0.59782	-1.73893	0.1	0.2								
8	5.69702	0.03863	0.00001	1								
9	0.97763	-4.66638	0.00001	-0.3								
10	6.37421	-2.3098	0.00001	0.1								
DSC_043	-1.214648	-0.869058	-1.187392	-0.640585	0.611842	-0.592235	0.00001	1.2	0.00001	0.00001	1.2	0.00001
10												
1	-5.57849	-3.20283	0.00001	1.2								
2	0.44613	-0.60749	0.00001	0.00001								
3	3.99047	0.62584	-0.5	-3.1								
4	-4.57613	-4.98377	0.00001	0.4								
5	0.96635	-1.86894	0.00001	0.2								
6	4.27488	-0.16436	-0.2	-1.2								
7	-0.54182	-4.20498	0.1	0.2								
8	4.44164	-1.21684	0.00001	1								
9	-0.52412	-5.90092	0.00001	-0.3								
10	4.54693	-2.87018	0.00001	0.1								
1	-0.248212	0.127585	-0.924662	0.00001	1.2	0.00001						
2	-0.043758	0.108611	-0.750067	0.00001	0.00001	0.00001						
3	0.10973	0.081328	-0.610558	-0.5	-3.1	-0.5						
4	-0.264773	0.055156	-0.894553	0.00001	0.4	0.00001						
5	-0.0601	0.054513	-0.725221	0.00001	0.2	0.00001						
6	0.100197	0.042822	-0.593114	-0.2	-1.2	-0.2						
7	-0.151294	-0.004943	-0.773228	0.1	0.2	0.1						
8	0.087079	-0.011657	-0.579469	0.00001	1	0.00001						
9	-0.180456	-0.083628	-0.770431	0.00001	-0.3	0.00001						
10	0.068379	-0.10747	-0.564724	0.00001	0.1	0.00001						

Data Foto

Foto 1

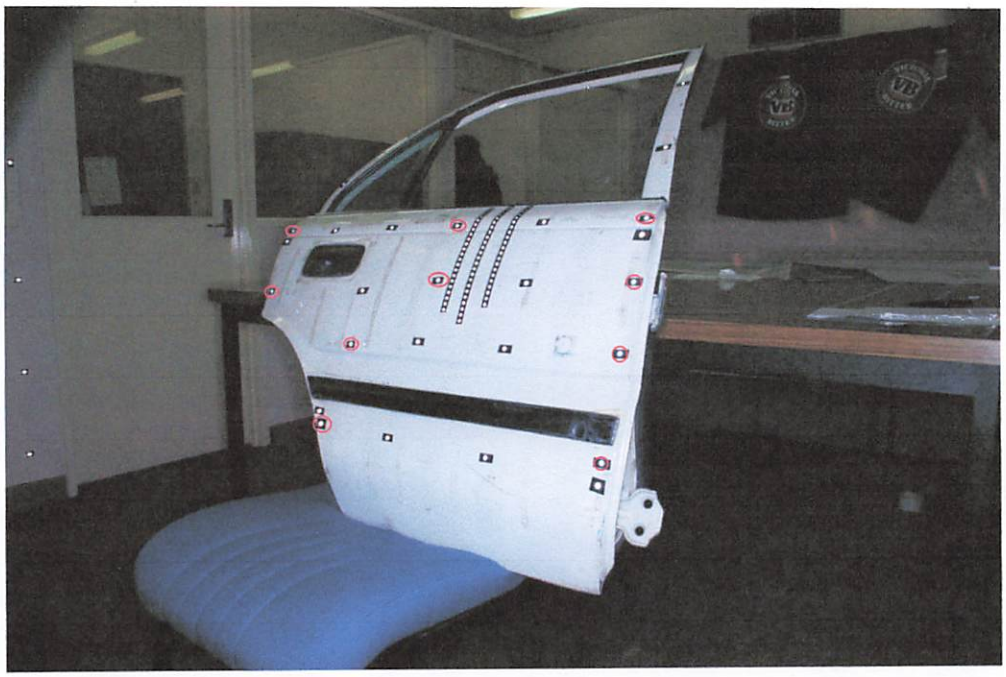
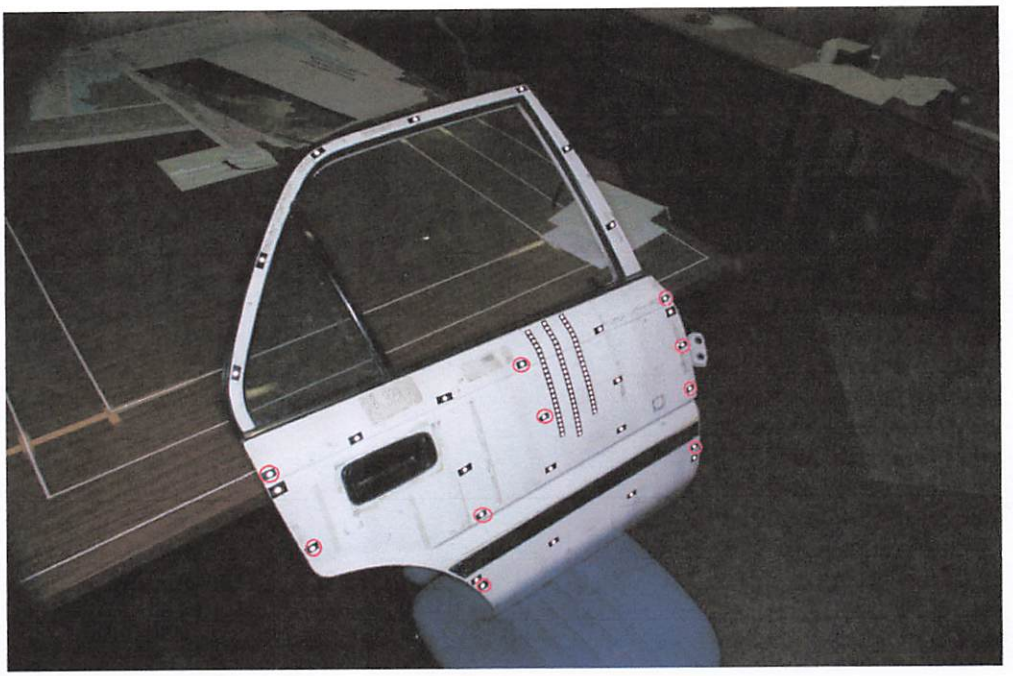


Foto 2

Foto 3

Foto 4



LAMPIRAN B
Data Hasil Perhitungan

HASIL PERHITUNGAN SELF-CALIBRATING BUNDLE ADJUSTMENT

Hasil Parameter EO

omega	Phi	Kappa	Xl	Yl	Zl
0.000105	-0.02461	0.000966	-0.01938	-0.0066	-0.00258
-0.60307	0.091865	0.130176	0.157952	0.553239	-0.05802
-0.90235	-1.37998	-1.02784	-0.67933	0.124651	-0.71815
-1.22456	-0.86115	-1.19141	-0.63401	0.610347	-0.5932

Hasil Koordinat Object Space Point 3D

X	Y	Z
-0.24883	0.128105	-0.92536
-0.0431	0.1084	-0.74961
0.109593	0.081502	-0.61046
-0.2652	0.054999	-0.89442
-0.05941	0.054345	-0.72507
0.099946	0.04294	-0.59328
-0.15055	-0.00502	-0.77287
0.086718	-0.01156	-0.57983
-0.18029	-0.08386	-0.77019
0.067917	-0.10753	-0.56494

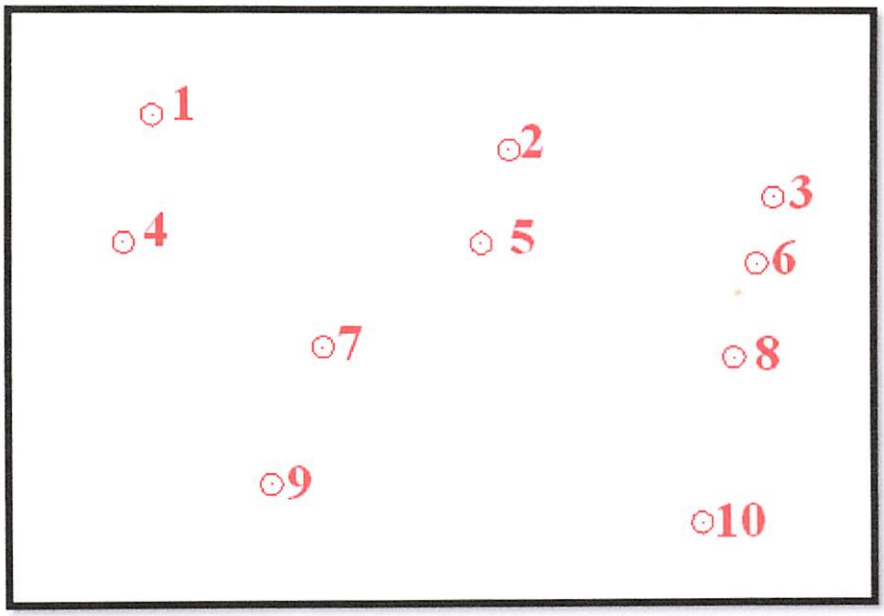
Hasil Parameter Kalibrasi

X0	=	0.067576
Y0	=	0.127609
fOKUS	=	18.55043
K1	=	-0.0002
K2	=	-4.13E-06
K3	=	5.88E-08
P1	=	0.000385
P2	=	-0.00037
B1	=	0.003022
B2	=	-0.00316

Dof : 176

Standart Deviasi : 0.000561

Data Hasil Plotting Di Autocad



Data Foto

LAMPIRAN C
Listing Code Program

Source Code dalam Class Program.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace listdata
{
    class Program
    {
        private const string FILE_NAME = "p4foto.txt";
        static void Main(string[] args)
        {
            bacadata baca = new bacadata(FILE_NAME);
            baca.Gate();
            Console.ReadLine();
        }
    }
}
```

Source Code dalam Class BacaData.cs

```
        paramKalibrasi kal = new paramKalibrasi();
        List<ListMatrix> listmatrix = new List<ListMatrix>();
        List<double[,]> B1 = new List<double[,]>();
        List<double[,]> B2 = new List<double[,]>();
        List<double[,]> L = new List<double[,]>();
        List<double[,]> B3 = new List<double[,]>();
        List<double[,]> bobot = new List<double[,]>();
        List<double[,]> bobotxyz = new List<double[,]>();
        public Photo p = new Photo();
        ParameterEO peo = new ParameterEO();
        List<Point3D> pt3D = new List<Point3D>();
        List<point2D> listpoint2D = new List<point2D>();
        List<ParameterEO> parametereo = new List<ParameterEO>();
        List<Photo> ph = new List<Photo>();
        double focal;

        string fileName;

        #region properties
        public List<ListMatrix> ListM
        {
            get { return listmatrix; }
        }
        public List<double[,]> MatrixL
        {
            get { return L; }
        }
        public List<double[,]> MatrixB1
        {
            get { return B1; }
        }
        public List<double[,]> MatrixB3
```

```
        {
            get { return B3; }
        }
        public List<double[,]> MatrixB2
        {
            get { return B2; }
        }
        public List<double[,]> Matrixbobot
        {
            get { return bobot; }
        }
        public List<double[,]> Matrixbobotxyz
        {
            get { return bobotxyz; }
        }

        public List<Point3D> ListPoint3D
        {
            get { return pt3D; }
        }
        public List<Photo> ListPhoto
        {
            get { return ph; }

        }
        public List<point2D> ListPoint2D
        {
            get { return listpoint2D; }
        }
        public List<ParameterEO> ListPhotoEO
        {
            get { return parametereo; }
        }

        #endregion

        public bacadata(string fileName)
        {
            this.fileName = fileName;

        }
        public bool BacaData()
        {
            p.Parametereo = new ParameterEO();
            if (!File.Exists(this.fileName))
            {
                Console.WriteLine(String.Format("{} File Tidak Ada",
this.fileName));
            }
            string[] bacaText = File.ReadAllLines(this.fileName);
            int rows = bacaText.Count();
            string[] pC;

            for (int i = 0; i < rows; i++)
            {
```

```
            String[] data = bacaText[i].Split(new char[] { ' ', '\t' });
            String[] ptnbr;
            String[] eox;
            String[] optx;
            if (i == 0)
            {
                pC = bacaText[i].Split(new char[] { ' ', '\t' });
                kal.FOkus = double.Parse(pC[0]);
                focal = kal.FOkus;
            }
            else if (data.Length == 1)
            {
                ptnbr = bacaText[i].Split(new char[] { ' ', '\t' });
            }
            else if (data.Length == 13)
            {
                eox = bacaText[i].Split(new char[] { ' ', '\t' });
                inputEO(eox);
            }
            else if (data.Length == 5)
            {
                optx = bacaText[i].Split(new char[] { ' ', '\t' });
                baca_p2(optx);

            }
            else if (data.Length == 7)
            {
                String[] opt = bacaText[i].Split(new char[] { ' ', '\t' });
                baca_p3(opt);
            }
        }
        return true;
    }
    public void baca_p3(string[] photocoord3D)
    {
        Point3D p3d = new Point3D();
        List<string> lst = new List<string>();
        foreach (string a in photocoord3D)
        {
            lst.Add(a);
        }
        p3d.Label = lst[0];
        p3d.X = double.Parse(lst[1]);
        p3d.Y = double.Parse(lst[2]);
        p3d.Z = double.Parse(lst[3]);
        p3d.SX = double.Parse(lst[4]);
        p3d.SY = double.Parse(lst[5]);
        p3d.SZ = double.Parse(lst[6]);

        ListPoint3D.Add(p3d);
    }
    public void baca_p2(string[] photocoordinate)
    {
        point2D pt2D = new point2D();
        List<string> lst = new List<string>();
```

```
            foreach (string a in photocoordinate)
            {
                lst.Add(a);
            }
            pt2D.Label = lst[0];
            pt2D.X = double.Parse(lst[1]);
            pt2D.Y = double.Parse(lst[2]);
            pt2D.SX = double.Parse(lst[3]);
            pt2D.SY = double.Parse(lst[4]);
            ListPoint2D.Add(pt2D);
            p.ListPoint2D = listpoint2D;

        }

        public void inputEO(string[] eo)
        {
            ParameterEO deo = new ParameterEO();
            List<string> lst = new List<string>();
            foreach (string s in eo)
            {
                lst.Add(s);
            }
            deo.Omega = double.Parse(lst[1]);
            deo.Phi = double.Parse(lst[2]);
            deo.Kappa = double.Parse(lst[3]);
            deo.XL = double.Parse(lst[4]);
            deo.YL = double.Parse(lst[5]);
            deo.ZL = double.Parse(lst[6]);

            p.Parametereo.Omega=double.Parse(lst[1]);
            p.Parametereo.Phi = double.Parse(lst[2]);
            p.Parametereo.Kappa = double.Parse(lst[3]);
            p.Parametereo.XL = double.Parse(lst[4]);
            p.Parametereo.YL = double.Parse(lst[5]);
            p.Parametereo.ZL = double.Parse(lst[6]);
            ListPhotoEO.Add(deo);
        }

        public void Gate()
        {
            if(! BacaData())
                throw new Exception("file does not exist");
            SCBA(p.ListPoint2D, ListPoint3D,ListPhotoEO);
        }

public void SCBA(List<point2D> p2d, List<Point3D> p3d,List<ParameterEO> eo)
        {
            #region
            double[,] MNorm = new double[((6 * ListPhotoEO.Count) + 10 + (3 *
ListPoint3D.Count)+7), ((6 * ListPhotoEO.Count) + 10 + (3 *
ListPoint3D.Count)+7)];
            double[,] TNorm = new double[((6 * ListPhotoEO.Count) + 10 + (3 *
ListPoint3D.Count)+7), 1];
            for (int iter = 0; iter < 1; iter++)
            {
```

```
                List<double[,]> ml = new List<double[,]>();
                int d = iter + 1;
                int h = eo.Count;
                List<double[,]> bd1 = new List<double[,]>();
                List<double[,]> bh = new List<double[,]>();
                List<double[,]> bd2 = new List<double[,]>();

                double[,] B3gabung = new double[h * ListPoint3D.Count * 2, 10];
                double[,] B1gabung = new double[h * ListPoint3D.Count * 2, 6 *
h];
                double[,] B2gabung = new double[h * ListPoint3D.Count * 2, 3 *
ListPoint3D.Count];
                double[,] Lgabung = new double[h * ListPoint3D.Count * 2, 1];

                double[,] Helmert = new double[3 * ListPoint3D.Count, 7];

                Console.WriteLine("Iterasi : " + d);

                for (int j = 0; j < p3d.Count; j++)
                {

                    double[,] Matrik_bobot_XYZ = Matrik_Bobot_XYZ(p3d[j].SX,
p3d[j].SY, p3d[j].SZ);


                    #region NEW
                    double[,] dot1 = new double[2 * ListPhotoEO.Count, 6 *
ListPhotoEO.Count];
                    double[,] dot3 = new double[2 * ListPhotoEO.Count, 10];
                    double[,] dot2 = new double[2 * ListPhotoEO.Count, 3];
                    double[,] ldt = new double[2 * ListPhotoEO.Count, 1];

                    double[,] hlmrt = new double[3, 7];
                    #endregion

                    #region input
                    List<string> label = new List<string>();
                    List<double> x = new List<double>();
                    List<double> y = new List<double>();
                    List<double> sx = new List<double>();
                    List<double> sy = new List<double>();
                    for (int i = 0; i < p2d.Count; i++)
                    {
                        if (p3d[j].Label == p2d[i].Label)
                        {
                            sx.Add(p2d[i].SX);
                            sy.Add(p2d[i].SY);
                            x.Add(p2d[i].X);
                            y.Add(p2d[i].Y);
                        }
                    }
                    //ListMatrix lis = new ListMatrix();
                    List<double[,]> B1 = Bdot(ListPoint3D);
                    List<double[,]> B3 = Bdot3(x, y);
                    List<double[,]> B2 = Bdot2(ListPoint3D);
```

```
                List<double[,]> bobot = Matrik_Bobot_xy(sx, sy);

                ml = Epsilon(p3d[j].X, p3d[j].Y, p3d[j].Z, x, y);
                foreach (double[,] a in B1)
                {
                    MatrixB1.Add(a);
                }
                foreach (double[,] s in B3)
                {
                    MatrixB3.Add(s);
                }
                foreach (double[,] b in B2)
                {
                    MatrixB2.Add(b);
                }
                foreach (double[,] b in bobot)
                {
                    Matrixbobot.Add(b);
                }
                foreach (double[,] s in ml)
                {
                    MatrixL.Add(s);
                }
                foreach (double[,] s in bobotxyz)
                {
                    Matrixbobotxyz.Add(s);
                }

                #endregion

                #region print
                //printmatrixDebug(this.MatrixB1[j], "Matirx B1");
                //printmatrixDebug(this.MatrixB3[j], "Matirx B3");
                //printmatrixDebug(this.MatrixB2[j], "Matirx B2");
                //printmatrixDebug(this.MatrixL[j], "Matirx L");
                //printmatrixDebug(this.Matrixbobot[j], "Matirx Bobot");
                #endregion

                int r = j * h;
                for (int g = 0; g < ListPhotoEO.Count; g++)
                {

                    int n = j * ListPhotoEO.Count;
                    int f = g * 6;
                    int s = g * 2;


                    #region review
                    double[,] mBdot = MatrixB1[n + g];
                    dot1[s, f] = mBdot[0, 0]; dot1[s, f + 1] = mBdot[0, 1];
dot1[s, f + 2] = mBdot[0, 2]; dot1[s, f + 3] = mBdot[0, 3]; dot1[s, f + 4] =
mBdot[0, 4]; dot1[s, f + 5] = mBdot[0, 5];
                    dot1[s + 1, f] = mBdot[1, 0]; dot1[s + 1, f + 1] =
mBdot[1, 1]; dot1[s + 1, f + 2] = mBdot[1, 2]; dot1[s + 1, f + 3] = mBdot[1,
3]; dot1[s + 1, f + 4] = mBdot[1, 4]; dot1[s + 1, f + 5] = mBdot[1, 5];
```

```
                        double[,] mBhat = MatrixB3[n + g];
                        dot3[s, 0] = mBhat[0, 0]; dot3[s, 1] = mBhat[0, 1];
dot3[s, 2] = mBhat[0, 2]; dot3[s, 3] = mBhat[0, 3]; dot3[s, 4] = mBhat[0, 4];
dot3[s, 5] = mBhat[0, 5]; dot3[s, 6] = mBhat[0, 6]; dot3[s, 7] = mBhat[0, 7];
dot3[s, 8] = mBhat[0, 8]; dot3[s, 9] = mBhat[0, 9];
                        dot3[s + 1, 0] = mBhat[1, 0]; dot3[s + 1, 1] = mBhat[1,
1]; dot3[s + 1, 2] = mBhat[1, 2]; dot3[s + 1, 3] = mBhat[1, 3]; dot3[s + 1, 4]
= mBhat[1, 4]; dot3[s + 1, 5] = mBhat[1, 5]; dot3[s + 1, 6] = mBhat[1, 6];
dot3[s + 1, 7] = mBhat[1, 7]; dot3[s + 1, 8] = mBhat[1, 8]; dot3[s + 1, 9] =
mBhat[1, 9];

                        double[,] mBdot2 = MatrixB2[n + g];
                        dot2[s, 0] = mBdot2[0, 0]; dot2[s, 1] = mBdot2[0, 1];
dot2[s, 2] = mBdot2[0, 2];
                        dot2[s + 1, 0] = mBdot2[1, 0]; dot2[s + 1, 1] =
mBdot2[1, 1]; dot2[s + 1, 2] = mBdot2[1, 2];

                        double[,] mldot = MatrixL[n + g];
                        ldt[s, 0] = mldot[0, 0];
                        ldt[s + 1, 0] = mldot[1, 0];
                        #endregion

                    }
                    bd1.Add(dot1);
                    bh.Add(dot3);
                    bd2.Add(dot2);

                    int ad = ListPoint3D.Count;
                    int af = j * 2;

                    //printmatrixDebug(dot1, "dattaatta");
                    //printmatrixDebug(dot3, "dotttt3333");
                    //printmatrixDebug(dot2, "dot22222");
                    //printmatrixDebug(ldt, "matrix ldot1");

                    double[,] sb = bd1[j];
                    double[,] sh = bh[j];
                    double[,] sb2 = bd2[j];
                    for (int as1 = 0; as1 < ListPhotoEO.Count; as1++)
                    {
                        int ast = j * (h * 2);
                        int m = j * 3;
                        for (int as11 = 0; as11 < ListPhotoEO.Count * 2;
as11++)
                        {
                            for (int wq = 0; wq < ListPhotoEO.Count * 6; wq++)
                            {
                                B1gabung[ast + as11, wq] = sb[as11, wq];
                            }
                            for (int wq = 0; wq < 10; wq++)
                            {
                                B3gabung[ast + as11, wq] = sh[as11, wq];
                            }
```

```
                            B2gabung[ast + as11, m] = sb2[as11, 0];
B2gabung[ast + as11, m + 1] = sb2[as11, 1]; B2gabung[ast + as11, m + 2] =
sb2[as11, 2];
                            Lgabung[ast + as11, 0] = ldt[as11, 0];
                        }
                    }

                    #region helmert
                    hlmrt[0, 0] = 1; hlmrt[0, 4] = ListPoint3D[j].Z; hlmrt[0,
5] = -ListPoint3D[j].Y; hlmrt[0, 6] = ListPoint3D[j].X;
                    hlmrt[1, 1] = 1; hlmrt[1, 3] = -ListPoint3D[j].Z; hlmrt[1,
5] = ListPoint3D[j].X; hlmrt[1, 6] = ListPoint3D[j].Y;
                    hlmrt[2, 2] = 1; hlmrt[2, 3] = ListPoint3D[j].Y; hlmrt[2,
4] = -ListPoint3D[j].X; hlmrt[2, 6] = ListPoint3D[j].Z;
                    //printmatrixDebug(hlmrt, "matrix helmert");


                    int qw = 3 * j;
                    Helmert[qw, 0] = hlmrt[0, 0]; Helmert[qw, 1] = hlmrt[0, 1];
Helmert[qw, 2] = hlmrt[0, 2]; Helmert[qw, 3] = hlmrt[0, 3]; Helmert[qw, 4] =
hlmrt[0, 4]; Helmert[qw, 5] = hlmrt[0, 5]; Helmert[qw, 6] = hlmrt[0, 6];
                    Helmert[qw + 1, 0] = hlmrt[1, 0]; Helmert[qw + 1, 1] =
hlmrt[1, 1]; Helmert[qw + 1, 2] = hlmrt[1, 2]; Helmert[qw + 1, 3] = hlmrt[1,
3]; Helmert[qw + 1, 4] = hlmrt[1, 4]; Helmert[qw + 1, 5] = hlmrt[1, 5];
Helmert[qw + 1, 6] = hlmrt[1, 6];
                    Helmert[qw + 2, 0] = hlmrt[2, 0]; Helmert[qw + 2, 1] =
hlmrt[2, 1]; Helmert[qw + 2, 2] = hlmrt[2, 2]; Helmert[qw + 2, 3] = hlmrt[2,
3]; Helmert[qw + 2, 4] = hlmrt[2, 4]; Helmert[qw + 2, 5] = hlmrt[2, 5];
Helmert[qw + 2, 6] = hlmrt[2, 6];
                    #endregion
                }
                //printmatrixDebug(Helmert, "total Helmert");
                double[,] HelmeTrans = Matrix.Transpose<double>(Helmert);
                //printmatrixDebug(B2gabung, "b2 gabung");
                //printmatrixDebug(B3gabung, "b3Gabung");
                //printmatrixDebug(B1gabung, "B1gabung");
                //printmatrixDebug(Lgabung, "L Gabung");
                #region Proses Hitung matrix N_DOT N_HAT N_DOTDOT N_U N_STRIP
N_TILD T_DOT T_HAT T_DOTDOT
                double[,] N_DOT = a_Transpose_b(B1gabung, B1gabung);
                double[,] N_HAT = a_Transpose_b(B3gabung, B3gabung);
                double[,] N_DOTDOT = a_Transpose_b(B2gabung, B2gabung);

                double[,] N_U = a_Transpose_b(B1gabung, B3gabung);
                double[,] N_STRIP = a_Transpose_b(B1gabung, B2gabung);
                double[,] N_TILD = a_Transpose_b(B3gabung, B2gabung);

                double[,] N_UTrans = Matrix.Transpose<double>(N_U);
                double[,] N_STrans = Matrix.Transpose<double>(N_STRIP);
                double[,] N_TildTrans = Matrix.Transpose<double>(N_TILD);

                double[,] T_DOT = a_Transpose_b(B1gabung, Lgabung);
                double[,] T_HAT = a_Transpose_b(B3gabung, Lgabung);
                double[,] T_DOTDOT = a_Transpose_b(B2gabung, Lgabung);
                #endregion
```

```
//printmatrixDebug(N_HAT, " matrix N Hat");
//printmatrixDebug(N_TildTrans, "N_TILD transpose");
//printmatrixDebug(N_DOTDOT, "matrix NDOT");
//printmatrixDebug(T_DOT, "matrix T DOT");
//printmatrixDebug(T_HAT, " matrix T Hat");
//printmatrixDebug(T_DOTDOT, "Matrix T_DOTDOT");
//printmatrixDebug(N_U, "matrix N U");
//printmatrixDebug(N_STRIP, "matrix N Strip");
//printmatrixDebug(N_TILD, "Matrix N Tild");
#region Normal Equation
#region For Normal Matrix
for (int j = 0; j < ListPhotoEO.Count * 6; j++)
{
    for (int ad = 0; ad < ListPhotoEO.Count * 6; ad++)
    {
        MNorm[j, ad] = N_DOT[j, ad];//ndot
    }
    for (int ad = 0; ad < 10; ad++)
    {
        for (int sd = 0; sd < ListPhotoEO.Count * 6; sd++)
        {
            int qw = ad + ListPhotoEO.Count * 6;
            MNorm[qw, sd] = N_UTrans[ad, sd];//NU transpose
        }
    }
    for (int ad = 0; ad < ListPoint3D.Count*3; ad++)
    {
        for (int sd = 0; sd < ListPhotoEO.Count * 6; sd++)
        {
            int qw = ad + ((ListPhotoEO.Count * 6) + 10);
            MNorm[qw, sd] = N_STrans[ad, sd];//NStrip Transpose
        }
    }
    for (int ad = 0; ad < 10; ad++)
    {
        int qw = ad + ListPhotoEO.Count * 6;
        MNorm[j, qw] = N_U[j, ad];//NU
    }

    for (int ad = 0; ad < 10; ad++)
    {
        for (int ad1 = 0; ad1 < 10; ad1++)
        {
            int qw = ad1 + ListPhotoEO.Count * 6;
            int qw1 = ad + ListPhotoEO.Count * 6;
            MNorm[qw1, qw] = N_HAT[ad, ad1];//NHAT
        }
    }
    for (int ad = 0; ad < 3*ListPoint3D.Count; ad++)
    {
        for (int ad1 = 0; ad1 < 10; ad1++)
        {
            int qw = ad1 + ListPhotoEO.Count * 6;
            int qw1 = ad + ((ListPhotoEO.Count * 6) + 10);
```

```
                MNorm[qw1, qw] = N_TildTrans[ad, ad1];
            }
        }
        for (int ad = 0; ad < 3 * ListPoint3D.Count; ad++)
        {
            int qw = ad + ((ListPhotoEO.Count * 6) + 10);
            MNorm[j, qw] = N_STRIP[j, ad];
        }
        for (int ad1 = 0; ad1 < 10; ad1++)
        {
            for (int ad = 0; ad < 3 * ListPoint3D.Count; ad++)
            {
                int qw = ad + ((ListPhotoEO.Count * 6)+10) ;
                int qw1 = ad1 + (ListPhotoEO.Count * 6) ;
                MNorm[qw1, qw] = N_TILD[ad1, ad];
            }
        }
        for (int ad1 = 0; ad1 < 3 * ListPoint3D.Count; ad1++)
        {
            for (int ad = 0; ad < 3 * ListPoint3D.Count; ad++)
            {
                int qw = ad + ((ListPhotoEO.Count * 6) + 10);
                int qw1 = ad1 + ((ListPhotoEO.Count * 6) + 10);
                MNorm[qw1, qw] = N_DOTDOT[ad1, ad];
            }
        }
        for (int ad1 = 0; ad1 < 7; ad1++)
        {
            for (int ad = 0; ad < 3 * ListPoint3D.Count; ad++)
            {
                int qw = ad + ((ListPhotoEO.Count * 6) + 10);
                int qw1 = ad1 + ((ListPhotoEO.Count * 6) +
                10+(ListPoint3D.Count*3));
                MNorm[qw1, qw] = HelmeTrans[ad1, ad];
            }
        }
        for (int ad = 0; ad < 3 * ListPoint3D.Count; ad++)
        {
            for (int ad1 = 0; ad1 < 7; ad1++)
            {
                int qw = ad1 + ((ListPhotoEO.Count * 6) + 10 +
                (ListPoint3D.Count * 3));
                int qw1 = ad + ((ListPhotoEO.Count * 6) + 10 );
                MNorm[qw1, qw] = Helmert[ad, ad1];
            }
        }

    }
    #endregion

    #region For T Matrix
    for (int j = 0; j < ListPhotoEO.Count * 6; j++)
    {
        TNorm[j, 0] = T_DOT[j, 0];
    }
```

```
                for (int j = 0; j < 10; j++)
                {
                    int qw = j + (ListPhotoEO.Count * 6);
                    TNorm[qw, 0] = T_HAT[j, 0];
                }
                for (int j = 0; j < 3 * ListPoint3D.Count; j++)
                {
                    int qw = j + ((ListPhotoEO.Count * 6) + 10);
                    TNorm[qw, 0] = T_DOTDOT[j, 0];
                }
                #endregion
                #endregion


                double[,] saaaa = a_inverse_b(MNorm, TNorm);

                printmatrixDebug(saaaa, "Hasik Koreksi");
                double[,] vTv = a_Transpose_b(Lgabung, Lgabung);
                double std = vTv[0, 0] / ((2 * h * ListPoint3D.Count + 10) - (6
                * h + 3 * ListPoint3D.Count + 10));
                Console.WriteLine("Standart Deviasi : " + std);
Console.WriteLine(" ");
                for (int i = 0; i < ListPhotoEO.Count; i++)
                {
                    int rt = i * 6;
                    ListPhotoEO[i].Omega = ListPhotoEO[i].Omega + saaaa[rt, 0];
                    ListPhotoEO[i].Phi = ListPhotoEO[i].Phi + saaaa[rt + 1, 0];
                    ListPhotoEO[i].Kappa = ListPhotoEO[i].Kappa + saaaa[rt + 2,
0];

                    ListPhotoEO[i].XL = ListPhotoEO[i].XL + saaaa[rt + 3, 0];
                    ListPhotoEO[i].YL = ListPhotoEO[i].YL + saaaa[rt + 4, 0];
                    ListPhotoEO[i].ZL = ListPhotoEO[i].ZL + saaaa[rt + 5, 0];
                    Console.WriteLine("Omega ={0},Phi ={1},Kappa ={2},XL
                    ={3},YL ={4},ZL ={5}", ListPhotoEO[i].Omega,
                    ListPhotoEO[i].Phi, ListPhotoEO[i].Kappa,
                    ListPhotoEO[i].XL, ListPhotoEO[i].YL, ListPhotoEO[i].ZL);
                }
                for (int l = 0; l < ListPoint3D.Count; l++)
                {
                    int b = 6 * ListPhotoEO.Count + 10;
                    int a = (l * 3)+(b);
                    ListPoint3D[l].X = ListPoint3D[l].X + saaaa[a, 0];
                    ListPoint3D[l].Y = ListPoint3D[l].Y + saaaa[a + 1 , 0];
                    ListPoint3D[l].Z = ListPoint3D[l].Z + saaaa[a + 2 , 0];
                    Console.WriteLine("X ={0},Y ={1},Z ={2}", ListPoint3D[l].X,
                    ListPoint3D[l].Y, ListPoint3D[l].Z);
                }

                int c = 6 * ListPhotoEO.Count;
                kal.X0 = kal.X0 + saaaa[c, 0];
                kal.Y0 = kal.Y0 + saaaa[c + 1, 0];
                kal.FOkus = kal.FOkus + saaaa[c + 2, 0];
                kal.K1 = kal.K1 + saaaa[c + 3, 0];
                kal.K2 = kal.K2 + saaaa[c + 4, 0];
                kal.K3 = kal.K3 + saaaa[c + 5, 0];
```

```
                kal.P1 = kal.P1 + saaaa[c + 6, 0];
                kal.P2 = kal.P2 + saaaa[c + 7, 0];
                kal.B1 = kal.B1 + saaaa[c + 8, 0];
                kal.B2 = kal.B2 + saaaa[c + 9, 0];

                Console.WriteLine("parameter kalibrasi : "+kal.X0 + "   " +
                kal.X0 + "   " + kal.FOkus + "   " + kal.K1 + "   " + kal.K2 + "
                " + kal.K3 + "   " + kal.P1 + "   " + kal.P2 + "   " + kal.B1 + "
                " + kal.B2);
            }

            #endregion
        }

        #region Fungsi perkalian

        double[,] a_Transpose_b(double[,] a, double[,] b)
        {
            double[,] result = Matrix.Multiply(Matrix.Transpose<double>(a), b);
            return result;
        }

        double[,] a_inverse_b(double[,] a, double[,] b)
        {
            double[,] hasil = Matrix.Multiply((Matrix.Inverse(a)), b);
            return hasil;
        }
        #endregion
        #region MATRIX B
        public List<double[,]> Bdot(List<Point3D> p3d)
        {
            List<double[,]> b1 = new List<double[,]>();
            for (int k = 0; k < p3d.Count; k++)
            {
                for (int i = 0; i < ListPhotoEO.Count; i++)
                {
                    double[,] bdot = new double[2, 6];
                    double[,] rot = new double[3, 3];
                    double so = Math.Sin(ListPhotoEO[i].Omega);
                    double co = Math.Cos(ListPhotoEO[i].Omega);
                    double sp = Math.Sin(ListPhotoEO[i].Phi);
                    double cp = Math.Cos(ListPhotoEO[i].Phi);
                    double sk = Math.Sin(ListPhotoEO[i].Kappa);
                    double ck = Math.Cos(ListPhotoEO[i].Kappa);

                    //menghitung elements matriks rotasi dengan mengunakan data
parameter EO
                    double m11 = cp * ck; double m12 = so * sp * ck + co * sk;
                    double m13 = -co * sp * ck + so * sk;
                    double m21 = -cp * sk; double m22 = -so * sp * sk + co *
                    ck; double m23 = co * sp * sk + so * ck;
                    double m31 = sp; double m32 = -so * cp; double m33 = co *
cp;

                    rot[0, 0] = m11; rot[0, 1] = m12; rot[0, 2] = m13;
```

```
                    rot[1, 0] = m21; rot[1, 1] = m22; rot[1, 2] = m23;
                    rot[2, 0] = m31; rot[2, 1] = m32; rot[2, 2] = m33;
                    //printmatrixDebug(rot, "matrix rotasi");

                    double DX = p3d[k].X - ListPhotoEO[i].XL, DY = p3d[k].Y -
                    ListPhotoEO[i].YL, DZ = p3d[i].Z - ListPhotoEO[i].ZL;
                    //r,s,q
                    double r = (rot[0, 0] * DX )+ (rot[0, 1] * DY) + (rot[0, 2]
* DZ);
                    double s = (rot[1, 0] * DX )+ (rot[1, 1] * DY )+ (rot[1, 2]
* DZ);
                    double q = (rot[2, 0] * DX )+ (rot[2, 1] * DY )+ (rot[2, 2]
* DZ);

                    //Console.WriteLine("R : " + r + "S : " + s + "Q : " + q);
                    //form b-dot(terhadap parameter unknown)
                    bdot[0, 0] = (focal / (q * q)) * (r * (-m33 * DY + m23 *
                    DZ) - q * (-m13 * DY + m12 * DZ));
                    bdot[0, 1] = (focal / (q * q)) * (r * (cp * DX + so * sp *
                    DY - co * sp * DZ) - q * ((-sp * ck * DX) + (so * cp * ck *
                    DY) - (co * cp * ck * DZ)));
                    bdot[0, 2] = ((-focal / q) * (m21 * DX + m22 * DY + m23 *
DZ));
                    bdot[0, 3] = -((focal / (q * q)) * (r * m31 - q * m11));
                    bdot[0, 4] = -((focal / (q * q) )* (r * m32 - q * m12));
                    bdot[0, 5] = -((focal / (q * q) )* (r * m33 - q * m13));

                    bdot[1, 0] = (focal / (q * q)) * (s * (-m33 * DY + m32 *
                    DZ) -( q * (-m23 * DY + m22 * DZ)));
                    bdot[1, 1] = (focal / (q * q)) * (s * (cp * DX + so * sp *
                    DY - co * sp * DZ) - q * (sp * sk * DX - so * cp * sk * DY
                    + co * cp * sk * DZ));
                    bdot[1, 2] = (focal / q) * (m11 * DX + m12 * DY + m13 *
DZ);
                    bdot[1, 3] = -((focal / (q * q)) * (s * m31 - q * m21));
                    bdot[1, 4] = -((focal / (q * q)) * (s * m32 - q * m22));
                    bdot[1, 5] = -((focal / (q * q)) * (s * m33 - q * m23));
                    b1.Add(bdot);
                }
            }
            return b1;
        }
        public List<double[,]> Bdot2(List<Point3D> p3d)
        {
            List<double[,]> b2 = new List<double[,]>();
            for (int k = 0; k < p3d.Count; k++)
            {
                for (int i = 0; i < ListPhotoEO.Count; i++)
                {
                    double[,] B2 = new double[2, 3];
                    double[,] rot = new double[3, 3];
                    double so = Math.Sin(ListPhotoEO[i].Omega);
                    double co = Math.Cos(ListPhotoEO[i].Omega);
                    double sp = Math.Sin(ListPhotoEO[i].Phi);
                    double cp = Math.Cos(ListPhotoEO[i].Phi);
```

```
                    double sk = Math.Sin(ListPhotoEO[i].Kappa);
                    double ck = Math.Cos(ListPhotoEO[i].Kappa);

                //menghitung elements matriks rotasi dengan mengunakan data
parameter EO
                    double m11 = cp * ck; double m12 = so * sp * ck + co * sk;
                     double m13 = -co * sp * ck + so * sk;
                    double m21 = -cp * sk; double m22 = -so * sp * sk + co *
                    ck; double m23 = co * sp * sk + so * ck;
                    double m31 = sp; double m32 = -so * cp; double m33 = co *
cp;

                    rot[0, 0] = m11; rot[0, 1] = m12; rot[0, 2] = m13;
                    rot[1, 0] = m21; rot[1, 1] = m22; rot[1, 2] = m23;
                    rot[2, 0] = m31; rot[2, 1] = m32; rot[2, 2] = m33;

                    double DX = p3d[k].X - ListPhotoEO[i].XL, DY = p3d[k].Y -
                    ListPhotoEO[i].YL, DZ = p3d[k].Z - ListPhotoEO[i].ZL;
                    //r,s,q
                    double r = rot[0, 0] * DX + rot[0, 1] * DY + rot[0, 2] *
DZ;
                    double s = rot[1, 0] * DX + rot[1, 1] * DY + rot[1, 2] *
DZ;
                    double q = rot[2, 0] * DX + rot[2, 1] * DY + rot[2, 2] *
DZ;

                    B2[0, 0] = ((focal / (q * q)) * (r * m31 - q * m11));
                    B2[0, 1] = ((focal / (q * q) )* (r * m32 - q * m12));
                    B2[0, 2] = ((focal /(q * q) )* (r * m33 - q * m13));

                    B2[1, 0] = ((focal / (q * q)) * (s * m31 - q * m21));
                    B2[1, 1] = (focal / (q * q) * (s * m32 - q * m22));
                    B2[1, 2] = (focal / (q * q) * (s * m33 - q * m23));
                    b2.Add(B2);
                }
            }
            return b2;
        }
        private List<double[,]> Matrik_Bobot_xy(List<double> sx, List<double>
sy)
        {
            List<double[,]> bobot = new List<double[,]>();
            for (int i = 0; i < sx.Count; i++)
            {
                double[,] Bobotxy = new double[2, 2];
                Bobotxy[0, 0] = 1 / Math.Pow((sx[i]), 2);
                Bobotxy[1, 1] = 1 / Math.Pow((sy[i]), 2);
                bobot.Add(Bobotxy);
            }
            return bobot;
        }
        public double[,] Matrik_Bobot_XYZ(double sx, double sy, double sz)
        {
            double[,] Bobotxyz = new double[3, 3];
            for (int i = 0; i < ListPhotoEO.Count; i++)
```

```
    {
        Bobotxyz[0, 0] = 1 / Math.Pow(sx, 2);
        Bobotxyz[1, 1] = 1 / Math.Pow(sy, 2);
        Bobotxyz[2, 2] = 1 / Math.Pow(sz, 2);
    }
    //printmatrixDebug(Bobotxyz, "bbot");
    return Bobotxyz;
}
public List<double[,]> Bdot3(List<double> x, List<double> y)
{
    List<double[,]> b3 = new List<double[,]>();
    for (int i = 0; i < ListPhotoEO.Count; i++)
    {
        double[,] b3dot = new double[2, 10];

        double x0 = kal.X0, y0 =kal.Y0; //parameter IO

        double x1, y1, r;

        x1 = x[i] - x0; y1 = y[i] - y0;
        r = Math.Sqrt((x1 * x1) + (y1 * y1));
        b3dot[0, 0] = -1; b3dot[0, 1] = 0; b3dot[0, 2] = -x1 / focal;
        b3dot[0, 3] = x1 * r * r; b3dot[0, 4] = x[i] * Math.Pow(r, 4);
        b3dot[0, 5] = x1 * Math.Pow(r, 6); b3dot[0, 6] = 2 * x1 * x1 +
        r; b3dot[0, 7] = 2 * x1 * y1; b3dot[0, 8] = x1; b3dot[0, 9] =
        y1;

        b3dot[1, 0] = 0; b3dot[1, 1] = -1; b3dot[1, 2] = -y1 / focal;
        b3dot[1, 3] = y1 * Math.Pow(r, 2); b3dot[1, 4] = y[i] *
        Math.Pow(r, 4);
        b3dot[1, 5] = y1 * Math.Pow(r, 6); b3dot[1, 6] = 2 * x1 * y1;
        b3dot[1, 7] = 2 * y1 * y1 + r; b3dot[1, 8] = 0; b3dot[1, 9] =
        0;
        b3.Add(b3dot);
    }

    return b3;
}
#endregion
#region MATRIX B
public List<double[,]> Bdot(List<Point3D> p3d)
{
    List<double[,]> b1 = new List<double[,]>();
    for (int k = 0; k < p3d.Count; k++)
    {
        for (int i = 0; i < ListPhotoEO.Count; i++)
        {
            double[,] bdot = new double[2, 6];
            double[,] rot = new double[3, 3];
            double so = Math.Sin(ListPhotoEO[i].Omega);
            double co = Math.Cos(ListPhotoEO[i].Omega);
            double sp = Math.Sin(ListPhotoEO[i].Phi);
            double cp = Math.Cos(ListPhotoEO[i].Phi);
            double sk = Math.Sin(ListPhotoEO[i].Kappa);
            double ck = Math.Cos(ListPhotoEO[i].Kappa);
```

```
                //menghitung elements matriks rotasi dengan mengunakan data
parameter EO
                    double m11 = cp * ck; double m12 = so * sp * ck + co * sk;
                    double m13 = -co * sp * ck + so * sk;
                    double m21 = -cp * sk; double m22 = -so * sp * sk + co *
                    ck; double m23 = co * sp * sk + so * ck;
                    double m31 = sp; double m32 = -so * cp; double m33 = co *
cp;

                    rot[0, 0] = m11; rot[0, 1] = m12; rot[0, 2] = m13;
                    rot[1, 0] = m21; rot[1, 1] = m22; rot[1, 2] = m23;
                    rot[2, 0] = m31; rot[2, 1] = m32; rot[2, 2] = m33;
                    //printmatrixDebug(rot, "matrix rotasi");

                    double DX = p3d[k].X - ListPhotoEO[i].XL, DY = p3d[k].Y -
                    ListPhotoEO[i].YL, DZ = p3d[i].Z - ListPhotoEO[i].ZL;
                    //r,s,q
                    double r = (rot[0, 0] * DX )+ (rot[0, 1] * DY) + (rot[0, 2]
* DZ);
                    double s = (rot[1, 0] * DX )+ (rot[1, 1] * DY )+ (rot[1, 2]
* DZ);
                    double q = (rot[2, 0] * DX )+ (rot[2, 1] * DY )+ (rot[2, 2]
* DZ);

                    //Console.WriteLine("R : " + r + "S : " + s + "Q : " + q);
                    //form b-dot(terhadap parameter unknown)
                    bdot[0, 0] = (focal / (q * q)) * (r * (-m33 * DY + m23 *
                    DZ) - q * (-m13 * DY + m12 * DZ));
                    bdot[0, 1] = (focal / (q * q)) * (r * (cp * DX + so * sp *
                    DY - co * sp * DZ) - q * ((-sp * ck * DX) + (so * cp * ck *
                    DY) - (co * cp * ck * DZ)));
                    bdot[0, 2] = ((-focal / q) * (m21 * DX + m22 * DY + m23 *
DZ));
                    bdot[0, 3] = -((focal / (q * q)) * (r * m31 - q * m11));
                    bdot[0, 4] = -((focal / (q * q) )* (r * m32 - q * m12));
                    bdot[0, 5] = -((focal / (q * q) )* (r * m33 - q * m13));

                    bdot[1, 0] = (focal / (q * q)) * (s * (-m33 * DY + m32 *
                    DZ) -( q * (-m23 * DY + m22 * DZ)));
                    bdot[1, 1] = (focal / (q * q)) * (s * (cp * DX + so * sp *
                    DY - co * sp * DZ) - q * (sp * sk * DX - so * cp * sk * DY
                    + co * cp * sk * DZ));
                    bdot[1, 2] = (focal / q) * (m11 * DX + m12 * DY + m13 *
DZ);
                    bdot[1, 3] = -((focal / (q * q)) * (s * m31 - q * m21));
                    bdot[1, 4] = -((focal / (q * q)) * (s * m32 - q * m22));
                    bdot[1, 5] = -((focal / (q * q)) * (s * m33 - q * m23));
                    b1.Add(bdot);
                }
            }
            return b1;
        }
        public List<double[,]> Bdot2(List<Point3D> p3d)
        {
```

```
            List<double[,]> b2 = new List<double[,]>();
            for (int k = 0; k < p3d.Count; k++)
            {
                for (int i = 0; i < ListPhotoEO.Count; i++)
                {
                    double[,] B2 = new double[2, 3];
                    double[,] rot = new double[3, 3];
                    double so = Math.Sin(ListPhotoEO[i].Omega);
                    double co = Math.Cos(ListPhotoEO[i].Omega);
                    double sp = Math.Sin(ListPhotoEO[i].Phi);
                    double cp = Math.Cos(ListPhotoEO[i].Phi);
                    double sk = Math.Sin(ListPhotoEO[i].Kappa);
                    double ck = Math.Cos(ListPhotoEO[i].Kappa);

                    //menghitung elements matriks rotasi dengan mengunakan data
parameter EO
                    double m11 = cp * ck; double m12 = so * sp * ck + co * sk;
                    double m13 = -co * sp * ck + so * sk;
                    double m21 = -cp * sk; double m22 = -so * sp * sk + co *
                    ck; double m23 = co * sp * sk + so * ck;
                    double m31 = sp; double m32 = -so * cp; double m33 = co *
cp;

                    rot[0, 0] = m11; rot[0, 1] = m12; rot[0, 2] = m13;
                    rot[1, 0] = m21; rot[1, 1] = m22; rot[1, 2] = m23;
                    rot[2, 0] = m31; rot[2, 1] = m32; rot[2, 2] = m33;

                    double DX = p3d[k].X - ListPhotoEO[i].XL, DY = p3d[k].Y -
                    ListPhotoEO[i].YL, DZ = p3d[k].Z - ListPhotoEO[i].ZL;
                    //r,s,q
                    double r = rot[0, 0] * DX + rot[0, 1] * DY + rot[0, 2] *
DZ;
                    double s = rot[1, 0] * DX + rot[1, 1] * DY + rot[1, 2] *
DZ;
                    double q = rot[2, 0] * DX + rot[2, 1] * DY + rot[2, 2] *
DZ;

                    B2[0, 0] = ((focal / (q * q)) * (r * m31 - q * m11));
                    B2[0, 1] = ((focal / (q * q) )* (r * m32 - q * m12));
                    B2[0, 2] = ((focal /(q * q) )* (r * m33 - q * m13));

                    B2[1, 0] = ((focal / (q * q)) * (s * m31 - q * m21));
                    B2[1, 1] = (focal / (q * q) * (s * m32 - q * m22));
                    B2[1, 2] = (focal / (q * q) * (s * m33 - q * m23));
                    b2.Add(B2);
                }
            }
            return b2;
        }
        private List<double[,]> Matrik_Bobot_xy(List<double> sx, List<double>
sy)
        {
            List<double[,]> bobot = new List<double[,]>();
            for (int i = 0; i < sx.Count; i++)
            {
```

```
            double[,] Bobotxy = new double[2, 2];
            Bobotxy[0, 0] = 1 / Math.Pow((sx[i]), 2);
            Bobotxy[1, 1] = 1 / Math.Pow((sy[i]), 2);
            bobot.Add(Bobotxy);
        }
        return bobot;
    }
    public double[,] Matrik_Bobot_XYZ(double sx, double sy, double sz)
    {
        double[,] Bobotxyz = new double[3, 3];
        for (int i = 0; i < ListPhotoEO.Count; i++)
        {
            Bobotxyz[0, 0] = 1 / Math.Pow(sx, 2);
            Bobotxyz[1, 1] = 1 / Math.Pow(sy, 2);
            Bobotxyz[2, 2] = 1 / Math.Pow(sz, 2);
        }
        //printmatrixDebug(Bobotxyz, "bbot");
        return Bobotxyz;
    }
    public List<double[,]> Bdot3(List<double> x, List<double> y)
    {
        List<double[,]> b3 = new List<double[,]>();
        for (int i = 0; i < ListPhotoEO.Count; i++)
        {
            double[,] b3dot = new double[2, 10];

            double x0 = kal.X0, y0 =kal.Y0; //parameter IO

            double x1, y1, r;

            x1 = x[i] - x0; y1 = y[i] - y0;
            r = Math.Sqrt((x1 * x1) + (y1 * y1));
            b3dot[0, 0] = -1; b3dot[0, 1] = 0; b3dot[0, 2] = -x1 / focal;
            b3dot[0, 3] = x1 * r * r; b3dot[0, 4] = x[i] * Math.Pow(r, 4);
            b3dot[0, 5] = x1 * Math.Pow(r, 6); b3dot[0, 6] = 2 * x1 * x1 +
            r; b3dot[0, 7] = 2 * x1 * y1; b3dot[0, 8] = x1; b3dot[0, 9] =
            y1;

            b3dot[1, 0] = 0; b3dot[1, 1] = -1; b3dot[1, 2] = -y1 / focal;
            b3dot[1, 3] = y1 * Math.Pow(r, 2); b3dot[1, 4] = y[i] *
            Math.Pow(r, 4);
            b3dot[1, 5] = y1 * Math.Pow(r, 6); b3dot[1, 6] = 2 * x1 * y1;
            b3dot[1, 7] = 2 * y1 * y1 + r; b3dot[1, 8] = 0; b3dot[1, 9] =
            0;
            b3.Add(b3dot);
        }

        return b3;
    }
    #endregion


    #region MATRIX l
    public List<double[,]> Epsilon( double xa, double ya, double za,
    List<double> x, List<double> y)
```

```
        {
            List<double[,]> l = new List<double[,]>();
            for (int i = 0; i < ListPhotoEO.Count; i++)
            {
                double[,] ldot = new double[2, 1];
                double[,] M = new double[3, 3];
                double xo = kal.X0, yo = kal.Y0;
                double DX = xa - ListPhotoEO[i].XL, DY = ya -
                ListPhotoEO[i].YL, DZ = za - ListPhotoEO[i].ZL;

                M[0, 0] = Math.Cos(ListPhotoEO[i].Phi) *
                Math.Cos(ListPhotoEO[i].Kappa);
                M[0, 1] = Math.Sin(ListPhotoEO[i].Omega) *
                Math.Sin(ListPhotoEO[i].Phi)*Math.Cos(ListPhotoEO[i].Kappa) +
                Math.Cos(ListPhotoEO[i].Omega) *
                Math.Sin(ListPhotoEO[i].Kappa);
                M[0, 2] = (-Math.Cos(ListPhotoEO[i].Omega) *
                Math.Sin(ListPhotoEO[i].Phi) * Math.Cos(ListPhotoEO[i].Kappa))
                + (Math.Sin(ListPhotoEO[i].Omega) *
                Math.Sin(ListPhotoEO[i].Kappa));

                M[1, 0] = -Math.Cos(ListPhotoEO[i].Phi) *
                Math.Sin(ListPhotoEO[i].Kappa);
                M[1, 1] = -Math.Sin(ListPhotoEO[i].Omega) *
                Math.Sin(ListPhotoEO[i].Phi) * Math.Sin(ListPhotoEO[i].Kappa)
                + Math.Cos(ListPhotoEO[i].Omega) *
                Math.Cos(ListPhotoEO[i].Kappa);
                M[1, 2] = Math.Cos(ListPhotoEO[i].Omega) *
                Math.Sin(ListPhotoEO[i].Phi) * Math.Sin(ListPhotoEO[i].Kappa)
                + Math.Sin(ListPhotoEO[i].Omega) *
                Math.Cos(ListPhotoEO[i].Kappa);

                M[2, 0] = Math.Sin(ListPhotoEO[i].Phi);
                M[2, 1] = -Math.Sin(ListPhotoEO[i].Omega) *
Math.Cos(ListPhotoEO[i].Phi);
                M[2, 2] = Math.Cos(ListPhotoEO[i].Omega) *
Math.Cos(ListPhotoEO[i].Phi);

                //r,s,q
                double r = M[0, 0] * DX + M[0, 1] * DY + M[0, 2] * DZ;
                double s = M[1, 0] * DX + M[1, 1] * DY + M[1, 2] * DZ;
                double q = M[2, 0] * DX + M[2, 1] * DY + M[2, 2] * DZ;

                ldot[0, 0] = (x[i] - xo) + (focal * r / q);
                ldot[1, 0] = (y[i] - yo) + (focal * s / q);
                l.Add(ldot);
            }
            return l;
        }
        #endregion


          #region print out
        private void printmatrixDebug(double[,] a, string name)
        {
```

```
            int lower1 = a.GetLowerBound(0);
            int upper1 = a.GetUpperBound(0);
            int lower2 = a.GetLowerBound(1);
            int upper2 = a.GetUpperBound(1);
            Console.WriteLine("\n" + name + " : ");
            for (int i = lower1; i <= upper1; i++)
            {
                for (int j = lower2; j <= upper2; j++)
                {
                    Console.Write(a[i, j].ToString() + "\t");
                }
                Console.WriteLine("");
            }
        }

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

            Console.WriteLine("");
        }
        #endregion
    }
}
```

Source Code dalam Class ParameterEO.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace listdata
{
    class ParameterEO
    {
        private double Xc;
        private double Yc;
        private double Zc;
        private double Omg;
        private double Pi;
        private double Kp;


        public double XL
        {
            get { return Xc; }
            set { Xc = value; }
        }
        public double YL
        {
            get { return Yc; }
            set { Yc = value; }
        }
        public double ZL
        {
            get { return Zc; }
            set { Zc = value; }
        }
        public double Omega
        {
            get { return Omg; }
            set { Omg = value; }
        }
        public double Phi
        {
            get { return Pi; }
            set { Pi = value; }
        }
        public double Kappa
        {
            get { return Kp; }
            set { Kp = value; }
        }
    }
}
```

Source Code dalam Class paramKalibrasi.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace listdata
{
    class paramKalibrasi
    {
        double x0;
        double y0;
        double fokus;
        double k1;
        double k2;
        double k3;
        double p1;
        double p2;
        double b1;
        double b2;

        public double X0
        {
            get { return x0; }
            set { x0 = value; }
        }
        public double Y0
        {
            get { return y0; }
            set { y0 = value; }
        }
        public double FOkus
        {
            get { return fokus; }
            set { fokus = value; }
        }
        public double K1
        {
            get { return k1; }
            set { k1 = value; }
        }
        public double K2
        {
            get { return k2; }
            set { k2 = value; }
        }
        public double K3
        {
            get { return k3; }
            set { k3 = value; }
        }
        public double P1
```

```
        {
            get { return p1; }
            set { p1 = value; }
        }
        public double P2
        {
            get { return p2; }
            set { p2 = value; }
        }
        public double B1
        {
            get { return b1; }
            set { b1 = value; }
        }
        public double B2
        {
            get { return b2; }
            set { b2 = value; }
        }
    }
}
```

Source Code dalam Class Photo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace listdata
{
    class Photo
    {
        private string photoname;
        private ParameterEO parametereo;
        private List<point2D> ptList;

        public void InputData(string photName, ParameterEO paramEO,
        List<point2D> p2Dlist)
        {
            this.photoname = photName;
            this.parametereo = paramEO;
            this.ptList = p2Dlist;
        }
        public string Photoname
        {
            get { return photoname; }
            set { photoname = value; }
        }
        public ParameterEO Parametereo
        {
            get { return parametereo; }
            set { parametereo = value; }
        }
        public List<point2D> ListPoint2D
```

```
        {
            get { return ptList; }
            set { ptList = value; }
        }
    }
}
```

Source Code dalam Class Point2D.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace listdata
{
    class point2D
    {
        private string label;
        private double x;
        private double y;
        private double sx;
        private double sy;

        public string Label
        {
            get { return label; }
            set { label = value; }
        }
        public double X
        {
            get { return x; }
            set { x = value; }
        }
        public double Y
        {
            get { return y; }
            set { y = value; }
        }
        public double SX
        {
            get { return sx; }
            set { sx = value; }
        }
        public double SY
        {
            get { return sy; }
            set { sy = value; }

        }
    }
}
```

Source Code dalam Class Point3D.cs

```
using System;
using System.Linq;
using System.Text;

namespace listdata
{
    class Point3D
    {
        private string label;
        private double x;
        private double y;
        private double z;
        private double sx;
        private double sy;
        private double sz;

        public string Label
        {
            get { return label; }
            set { label = value; }
        }
        public double X
        {
            get { return x; }
            set { x = value; }
        }

        public double Y
        {
            get { return y; }
            set { y = value; }
        }
        public double Z
        {
            get { return z; }
            set { z = value; }
        }
        public double SX
        {
            get { return sx; }
            set { sx = value; }
        }
        public double SY
        {
            get { return sy; }
            set { sy = value; }
        }
        public double SZ
        {
            get { return sz; }
            set { sz = value; }
        }
    }
```

LAMPIRAN D
Bentuk Matrix

Contoh susunan matrix B1 pada persamaan 2.53 yang memiliki ukuran sub-matrix 2X6 block diagonal

	Foto ke 1												Foto ke i					
Titik 1	0.6871	19.893	2.56	-20.07	-1E-04	5.387	0	0	0	0	0	0	0	0	0	0	0	0
	-18.90	-0.687	4.981	0.0002	-20.0	-2.76	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	....	....	....	....	....	....	0	0	0	0	0	0
	0	0	0	0	0	0	....	....	....	....	....	....	0	0	0	0	0	0
Titik j	0	0	0	0	0	0	0	0	0	0	0	0	-8.166	21.051	1.6813	-20.68	3.45593	9.8904
	0	0	0	0	0	0	0	0	0	0	0	0	-17.74	-2.983	6.92568	2.5417	-19.03	10.352
Titik 1	0.103	18.598	2.18	-20.07	-2E-04	0.949	0	0	0	0	0	0	0	0	0	0	0	0
	-18.81	-0.103	0.878	0.0002	-20.0	-2.35	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	....	....	....	....	....	....	0	0	0	0	0	0
	0	0	0	0	0	0	....	....	....	....	....	....	0	0	0	0	0	0
Titik j	0	0	0	0	0	0	0	0	0	0	0	0	-4.565	18.779	0.79151	-21.38	0.60358	5.7781
	0	0	0	0	0	0	0	0	0	0	0	0	-18.1	-2.455	2.58691	2.6675	-18.633	11.322

Contoh susunan matrix B3 pada persamaan 2.59 yang memiliki ukuran matrix 2ij X 10 untuk *self-calibrating bundle adjustment block invariant*

Dimana :

i : jumlah foto

j : jumlah titik

Foto ke 1	titik 1	-1	0	0.6871	19.89	2.56	-20.07	-1E-04	5.387	0.687	19.89
		0	-1	-18.91	-0.687	4.981	2E-04	-20.07	-2.769	-18.91	-0.687
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	titik ke j	-1	0	0.6871	19.89	2.56	-20.07	-1E-04	5.387	0.687	19.89
		0	-1	-18.91	-0.687	4.981	2E-04	-20.07	-2.769	-18.91	-0.687
Foto ke i	titik 1	-1	0	0.6871	19.89	2.56	-20.07	-1E-04	5.387	0.687	19.89
		0	-1	-18.91	-0.687	4.981	2E-04	-20.07	-2.769	-18.91	-0.687
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	titik ke j	-1	0	0.6871	19.89	2.56	-20.07	-1E-04	5.387	0.687	19.89
		0	-1	-18.91	-0.687	4.981	2E-04	-20.07	-2.769	-18.91	-0.687

Contoh susunan matrix B2 pada persamaan 2.55 yang memiliki ukuran sub-matrix 2i X 3 block diagonal

Dimana :

i : jumlah foto

	Titik ke 1						Titik ke j		
Foto ke 1	20.068	0.0001	-5.387	0	0	0	0	0	0
	-0.0002	20.068	2.769	0	0	0	0	0	0
	⋮	⋮	⋮	0	0	0	0	0	0
	⋮	⋮	⋮	0	0	0	0	0	0
Foto ke i	13.087	-6.595	25.05	0	0	0	0	0	0
	20.31	19.176	-3.962	0	0	0	0	0	0
	0	0	0	...	...	...	0	0	0
	0	0	0	...	...	...	0	0	0
Foto ke 1	0	0	0	0	0	0	24.74	2E-04	-1.443
	0	0	0	0	0	0	-2E-04	24.74	3.582
	0	0	0	0	0	0	⋮	⋮	⋮
	0	0	0	0	0	0	⋮	⋮	⋮
Foto ke i	0	0	0	0	0	0	1.207	-3.59	29.17
	0	0	0	0	0	0	0.933	29.34	3.789

Matrix $\dot{N}$ pada persamaan 2.63 yang merupakan hasil pekalian matrix antara $B1^{T}$. B1 maka akan membentuk matrix dengan ukuran 6i x 6i yang meliki sub-matrix 6 x 6 block diagonal. Seperti pada contoh matrix $\dot{N}$ dibawah ini.

3493.7	28.443	-217	-14.64	3750	111.6	0	0	0	0	0	0	0	0	0	0	0	0
28.443	3631.9	97.69	-3823	14.64	256.2	0	0	0	0	0	0	0	0	0	0	0	0
-217.18	97.694	116.1	-105.7	-235	5E-04	0	0	0	0	0	0	0	0	0	0	0	0
-14.637	-3823	-106	4027.1	-0.7	-254	0	0	0	0	0	0	0	0	0	0	0	0
3750.4	14.641	-235	-0.003	4027	114.2	0	0	0	0	0	0	0	0	0	0	0	0
111.56	256.17	5E-04	-254	114.2	135.8	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	⋮	⋮	⋮	⋮	⋮	⋮	0	0	0	0	0	0
0	0	0	0	0	0	⋮	⋮	⋮	⋮	⋮	⋮	0	0	0	0	0	0
0	0	0	0	0	0	⋮	⋮	⋮	⋮	⋮	⋮	0	0	0	0	0	0
0	0	0	0	0	0	⋮	⋮	⋮	⋮	⋮	⋮	0	0	0	0	0	0
0	0	0	0	0	0	⋮	⋮	⋮	⋮	⋮	⋮	0	0	0	0	0	0
0	0	0	0	0	0	⋮	⋮	⋮	⋮	⋮	⋮	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	3607	-543	-488.6	500.4	2978	-2518
0	0	0	0	0	0	0	0	0	0	0	0	-543	3778	-171.7	-3963	554.4	837.89
0	0	0	0	0	0	0	0	0	0	0	0	-489	-172	181	181.2	-475.8	305.19
0	0	0	0	0	0	0	0	0	0	0	0	500.4	-3963	181.17	4186	-585.8	-822.5
0	0	0	0	0	0	0	0	0	0	0	0	2978	554.4	-475.8	-586	2897	-1916
0	0	0	0	0	0	0	0	0	0	0	0	-2518	837.9	305.19	-822	-1916	1826.5

Matrix $\ddot{N}$ pada persamaan 2.63 yang merupakan hasil pekalian matrix antara $B3^T$. B3 maka akan membentuk matrix dengan ukuran 10 x 10. Seperti pada contoh matrix $\ddot{N}$ dibawah ini.

40	0	-0.8	841.77	37458	2E+06	-1322	-0.404	14.88	32.24
0	40	-1.74	897.95	29491	1E+06	-0.404	-726	0	0
-0.8021	-1.737	2.47	-1352	65656	-2E+06	97.17	105.7	-30.95	-0.011
841.77	897.95	-1352	857453	3E+07	1E+09	-80588	-64056	17387	-187.9
37458	29491	53532	3E+07	1E+09	5E+10	-3E+06	-2E+06	6E+05	-13615
2E+06	1E+06	33332	1E+09	5E+10	2E+12	-1E+08	-9E+07	2E+07	-7E+05
-1322.1	-0.404	97.17	-80588	12121	-1E+08	82767	-802.8	-1330	-279.5
-0.4038	-726	105.7	-64056	4323	-9E+07	-802.8	37261	-114.7	-473.5
14.884	0	-31	17387	6E+05	2E+07	-1330	-114.7	574.3	0.202
32.235	0	-0.01	-187.9	3421	-7E+05	-279.5	-473.5	0.202	276.3

Matrix $\ddot{N}$ pada persamaan 2.63 yang merupakan hasil pekalian matrix antara $B2^T$. B2 maka akan membentuk matrix dengan ukuran 3i x 3i yang memiliki sub matrix 3 x 3 block diagonal. Seperti pada contoh matrix $\ddot{N}$ dibawah ini.

1653.1	184.35	757.3	0	0	0	0	0	0
184.35	2867.8	-378	0	0	0	0	0	0
757.33	-377.9	2350	0	0	0	0	0	0
0	0	0	.....	.....	....	0	0	0
0	0	0	.....	.....	.....	0	0	0
0	0	0	.....	.....	.....	0	0	0
0	0	0	0	0	0	1721	202.9	61.71
0	0	0	0	0	0	202.9	2167	-180.1
0	0	0	0	0	0	61.71	-180.1	1378

Matrix $\tilde{N}$ pada persamaan 2.63 yang merupakan hasil pekalian matrix antara $B1^T$. B3 maka akan membentuk matrix dengan ukuran 6i x 10.

Matrix $\overline{N}$ pada persamaan 2.63 yang merupakan hasil pekalian matrix antara $B1^T$. B2 maka akan membentuk matrix dengan ukuran 6i x 3j.

Matrix $\hat{N}$ pada persamaan 2.63 yang merupakan hasil pekalian matrix antara $B3^T$. B2 maka akan membentuk matrix dengan ukuran 10 x 3j.

Dimana :

i : jumlah foto

j : jumlah titik.