SKRIPSI

EVALUASI METODE PENENTUAN PENDEKATAN PARAMETER *EXTERIOR ORIENTATION* (EO) DENGAN MENGGUNAKAN METODE *CLOSED FORM SOLUTION* UNTUK PEMOTRETAN PADA MULTI FOTO KONVERGEN

(Studi Kasus : Jembatan Rel Kereta Api Lawang dan Fly Over Arjosari Malang)

Diajukan untuk memenuhi persyaratan
dalam mencapai gelar sarjana S1 Teknik Geodesi

Disusun Oleh :

SRI MULYATI

05.25.009

JURUSAN TEKNIK GEODESI

FAKULTAS TEKNIK SIPIL DAN PERENCANAAN

INSTITUT TEKNOLOGI NASIONAL

MALANG

2010

LEMBAR PENGESAHAN

Evaluasi Metode Penentuan Pendekatan Parameter *Exterior Orientation* (EO) dengan Menggunakan Metode *Closed Form Solution* untuk Pemotretan pada Multi Foto Konvergen

SKRIPSI

Dipertahankan dihadapan Majelis Penguji Sidang Skripsi

Jenjang Starata Satu (S-1)

Pada hari : Jumat
Tanggal : 20 Agustus 2010
Dan diterima untuk memenuhi salah satu persyaratan guna memperoleh gelar Sarjana Teknik.

Disusun oleh :

Sri Mulyati 05.25.009

Panitia Ujian Tugas Akhir

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

Anggota Penguji

Penguji I	Penguji II	Penguji III
Ir. Leo Patimena, M.Sc	**Silvester Sari Sai, ST, MT**	**Dr. Edwin Tjahjadi, ST. M.Geom.Sc**

JURUSAN TEKNIK GEODESI

FAKULTAS TEKNIK SIPIL DAN PERENCANAAN

INSTITUT TEKNOLOGI NASIONAL

MALANG

2010

LEMBAR PERSETUJUAN

SKRIPSI

Evaluasi Metode Penentuan Pendekatan Parameter *Exterior Orientation* (EO) dengan Menggunakan Metode *Closed Form Solution* untuk Pemotretan pada Multi Foto Konvergen

Diajukan Sebagai Salah Satu Syarat Memperoleh Gelar Sarjana Teknik Geodesi S-1

Institut Teknologi Nasional Malang

Disusun Oleh :

Sri Mulyati

05.25.009

Meyetujui,

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

Mengetahui

Ketua Jurusan Teknik Geodesi S-1

Hery Purwanto, ST, M.Sc

Abstraksi

Kata Kunci : *Closed Form Solution, Exterior Orientation (EO)*, metode *Fischler-Bolles*, metode Church, metode *Zeng*

Menentukan parameter orientasi luar (*exterior orientation*) yang meliputi tiga koordinat posisi kamera (X_L, Y_L, Z_L) dan tiga parameter sudut rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ) mempunyai nilai yang tidak terbatas dan perlu ditentukan nilai pendekatan awal. Permasalahan dasar ini dapat diperoleh dengan menerapkan metode *closed form solution* yang meliputi metode *Fischler-Bolles*, metode *Church* dan metode *Zeng*.

Pada metode *Fischler-Bolles*, parameter EO dapat ditentukan dengan penentuan lokasi posisi (LDP) minimal terdapat tiga titik antara bidang foto dan bidang objek (P3P), sehingga akan diperoleh dua solusi untuk parameter koordinat posisi kamera dan matriks rotasi. Solusi unik dari PnP akan diperoleh jika dilakukan prinsip *fitting line* pada kombinsi titik yang disajikan.

Pada metode *church* bahwa parameter EO dapat ditentukan dengan prinsip dasar bahwa nilai sudut untuk dua piramid mempunyai nilai yang sama yang dibentuk oleh piramid bidang foto dan bidang objek yang berpusat pada posisi kamera. Dengan adanya posisi kamera terkoreksi maka dapat disusun matriks rotasi dari foto yang miring.

Zeng dan *Wang* (1992) melakukan penelitian dengan mengembangkan metode yang telah dijelaskan dan diuji oleh *(Fischler* dan *Bolles*, 1981*)* yaitu penentuan lokasi atau *Location determination Problem* (LDP) untuk tiga titik pada satu foto, sehingga dapat diperoleh posisi koordinat kamera dan parameter rotasi dengan menggunakan prinsip perkalian murni

PERNYATAAN KEASLIAN SKRIPSI

Saya yang bertanda tangan dibawah ini :

Nama : **Sri Mulyati**

NIM : **05.25.009**

Program Studi : **Teknik Geodesi S-1**

Fakultas : **Fakultas Teknik Sipil Dan Perencanaan**

Menyatakan dengan sesungguhnya bahwa Skripsi saya dengan judul :

"Evaluasi Metode Penentuan Pendekatan Parameter *Exterior Orientation* (EO) dengan Menggunakan Metode *Closed Form Solution* untuk Pemotretan pada Multi Foto Konvergen (*Studi Kasus : Jembatan Rel Kereta Api Lawang dan Fly Over Arjosari Malang*)" adalah hasil karya saya sendiri, bukan merupakan duplikat serta tidak mengutip atau menyadur dari hasil karya orang lain kecuali disebutkan sumbernya.

Malang, 29 September 2010

Yang membuat pernyataan

Sri Mulyati

NIM : 05.25.009

KATA PENGANTAR

Puji syukur penulis panjatkan kepada ALLAH SWT, karena berkat rahmat-Nya, penulis dapat menyelesaikan skripsi yang berjudul **"Evaluasi Metode Penentuan Pendekatan Parameter *Exterior Orientation* (EO) dengan Menggunakan Metode *Closed Form Solution* untuk Pemotretan pada Multi Foto Konvergen (*Studi Kasus : Jembatan Rel Kereta Api Lawang dan Fly Over Arjosari Malang*)"**, dimana penulisan skripsi ini disusun sebagai salah satu syarat untuk meraih gelar Sarjana Teknik pada Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.

Penulisan ini tidak akan dapat terselesaikan tanpa bantuan dan dukungan berbagai pihak. Oleh karena itu, peneliti ingin mengucapkan terima kasih yang sebesar-besarnya kepada :

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

7. Segenap dosen, staff pengajar dan *recording* Jurusan Teknik Geodesi Fakultas Teknik Sipil dan Perencanaan Institut Teknologi Nasional Malang.
8. Bapak, Ibu, dan adek-adekku, yang selalu memberikan dukungan, semangat dan doa.
9. Team Deformasi dan *Rapid mapping* yang selalu memberika kerja sama dan dukungannya.
10. Semua pihak yang telah membantu peneliti yang tidak dapat disebutkan satu persatu.

Penulis menyadari bahwa penulisan laporan ini masih belum sempurna, baik dari segi materi, sistematika pembahasan, maupun susunan bahasa. Oleh karena itu, kritik dan saran yang membangun sangat penulis harapkan. Hasil penelitian ini dan dengan segala keterbatasannya dipersembahkan kepada dunia pendidikan, semoga ada manfaatnya untuk pengembangan sumber daya manusia di negara tercinta ini.

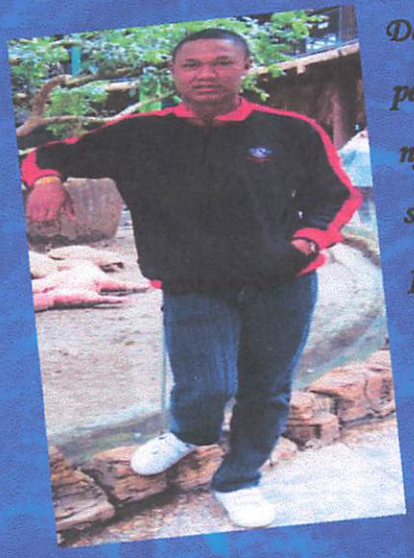
Malang, 29 September 2010

Penulis

بسم الله الرحمن الرحيم

Akhirnya waktu ini tiba juga, saya bisa selesaikan studi dan mendapat gelar S1 Jurusan Teknik Geodesi di Institut Teknologi Nasional Malang. Semua ini tidak akan tercapai kalau tidak ada dukungan dan doa dari berbagai pihak, oleh karena itu saya mau mengucapkan terima kasih kepada :

Bapak dan mama.....terima kasih buanyak ya atas motivasi, dukungan, perhatian dan kasih sayang yang tak terhingga dan sokongan dananya,,,hehehehee,,,,. Ahirnya aku bisa buktiin bahwa aku bisa selesaikan kuliah ini. Aku persembahkan karya dan prestasiku semua ini untuk bapak dan mama..... "You are the best of parents, nothing else in my life"



Dek BUDI,,,,makasih ya untuk support, perhatian dan remonnya,,,semua sangat ngedukung mbak untuk selesein semuannya,,,,,selesein kuliahnya pasti kamu bisa,,,,

Dek Aisyah,,,,yang centil n ngangenin,,,,makasih banyak untuk semuanya,,,,

Untuk adek-adekq,,,,jadilah seseorang yang bisa dibanggakan bagi orang tua n masyarakat meski tidak banyak,,,,buktikan kepada semua orang bahwa kalian bisa n menjadi yang terbaik.....kalian adalah "PelitaQ dalam mengarungi kehidupan ini"

Pak Edwin (Pak Bos)....trima kasih untuk bimbingannya slama ini. Dari bapak saya tau pa arti fotogrametri, belajar untuk mencoba sesuatu yang dimulai dari nol sampai akhirnya saya mengerti. Semoga saya bisa tepati janji saya,,,,mohon doa restunya,,,,

Pak Hery (Big Bos)... Matur nuwun sanget pak...untuk semua ilmu perkuliahan dan ilmu kehidupan serta wejangan2 yang bapak berikan kepada saya,,,,saya akan bawa kemana pun saya pergi,,,,,

Pak Silvester,,,,,makasih banyak pak untuk kepercayaan yang bapak berikan kepada saya untuk menjadi asisten, Akhirnya saya sedikit paham tentang Kartografi dan GPS, serta ilmu yang lain semoga dapat bermanfaat untuk selanjutnya,,,

Bu sulis,,,,,makasih ya bu untuk bantuan, masukannya. Bu adalah BUNDAq d Kampus.....pokoknya banyak dah,,,,ndak bisa diungkapin dengan kata-kata dah,,,

Cayoo Geo 05,,,,,Bro n Sis....akhirnya kita bisa lulus sama-sama (sesuai dengan janji qt) wah senangnya,,,,,Tanzil, Ona, Agus, Weni, Riri, Eno, Via, ALben, Chandra, Dody, Gede.....Kalian adalah kado terindah yang pernah aku dapatkan dalam hidupQ...trima kasih untuk persahabatan dan persaudaraan yang kalian berikan,,,,, Meski jarak memisahkan qt tpi kalian tetap didalam hatiQ,,,,canda tawa, bete, susah senang yang qt lalui akan menjadi "Kisah Klasik di Masa Depan". Untuk Gede tetap semangat ya,,,,pasti kamu bisa. September 2011 kamu harus wisuda,,,,,meski qt tidak bersama tapi doa qt selalu untuk kamu....

Untuk keluarga lab SIG : Pace Roger, Pace Yusak, kak Akbar, Kak Desy..thanks A Lot n Miss U ALL....

Untuk kak Indra,,,,perhatian, dukungan, semangat yang kakak berikan,,,amat sangat berarti untukq, sehingga aku bisa selesaikan Skripsi ini, Terima kasih selalu ada dalam suka dan dukaq,,,,meski aku banyak ngeluh tapi kakak selalu tetap berikan motivasi. Karena dirimu, sekarang aku lebih mengerti mana yg terpenting dan penting, semua pasti bisa tercapai jika selalu berusaha untuk maju dan pantang menyerah.

Untuk Kak Gandi, Kak Beno, Kak Dony,,,tetep semangat ya,,,
Kak Edi,Kak Doly, Kak Megy, kak Danik, Kak Grace, Kak Rosa, Kak Fauzan, kak Andi, trima kasih untuk bantuan dan persaudaraannya,,,
Untuk kak Elwin, Kak Ari, kak Gigih, Kak Arif, Kak Justin, Kak Desiana, tetap semangat ya,,,,,kak nanang, kak titin, kak triana, kak nophay,kak mawar,
Trima ksih bantuannya dan persaudaraannya,,,,,
Untuk temen-temen 2006 semuanya,,,,,,tetap semangat dan cepetan lulus ,,,jangan pantang menyerah,,,tunjukin bahwa kalian bisa,,,
Untuk adek2q angkatan 2007,2008,2009.2010....trima kasih untuk semuanya n menjadi bagian dari keluarga besar Teknik geodesi ITN Malang....berikan yang terbaek dari kalian demi almamater bersama,,,,,semoga kalian semua bisa menjadi laskar2 Geodesi selanjutnya yang memberikan perubahan di bidang Geodesi.....

Buat Anak-anak POHARIN D 176 : Salma, Meirsa, Tita, Tifa, Cicil, Anita, Deka, k' Ambu, Inge, Nurul, Mbak Desi makasih untuk persaudaraan dan bantuannya

Anak-anak rumah : kak Dadi, kak Thalib, Kak Adin, kak jali, kak Wahyu, wahyu,Alfan,Opick, Ovin, Zaky, Ayu, Yedi, Pras, Indra kecil, dan semua keluarga besar,,,,,,makasih banyak untuk rumah kedua yang kalian ciptakan,,,,kalian akan tetap dihatiQ untuk selamanya...

Buat semuanya yang tidak tertulis ataupun terlewati namanya,,,mohon maklum dan mohon maaf,,,,,tima kasih bnyak untuk apapun yang telah diberikan kepadaku.....

Akhir kata Wabilahi Taufik Wal Hidayah, Wassalammu 'alaikum.....

"Tak ada Gading yang tak retak, kesempurnaan hanya milik ALLAH SWT, kesalahan dan ketidaksempurnaan datangnya dari saya pribadi"

Daftar Isi

BAB II DASAR TEORI

BAB III PELAKSANAAN PENELITIAN

Daftar Gambar

Daftar Tabel

BAB I

PENDAHULUAN

1.1 Latar Belakang

Menentukan letak, posisi dan karakteristik geometri intrinsik dari kamera, umumnya dikenal dengan masalah dasar dalam fotogrametri yang meliputi penentuan parameter dalam kamera (*interior orientation*) dan parameter orientasi luar (*exterior orientation*), serta penentuan titik objek dalam bentuk 3D (*Grussenmeyer* dan *Al-Khalil*, 2002). Parameter dalam (*interior orientation*) meliputi koordinat foto (x_o,y_o) dan panjang fokus (f), sedangkan parameter orientasi luar (*exterior orientation*) terdiri dari tiga koordinat posisi kamera (X_L,Y_L,Z_L) dan tiga parameter sudut rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ) *(Mikhail,et al).* Akan tetapi, nilai yang mungkin untuk parameter orientasi luar (*exterior orientation*) dalam jumlah yang tidak terbatas dengan nilai residu yang bermacam-macam, sehingga perlu ditentukan terlebih dahulu batasan nilai pendekatan awal dari parameter orientasi luar (*exterior orientation*) agar lebih efektif dalam proses perhitungan selanjutnya. Dimana nilai pendekatan parameter orientasi luar (*exterior orientation*) dapat diperoleh jika terdapat nilai residu paling kecil (*Grussenmeyer* dan *Al-Khalil*, 2002).

Pendekatan parameter luar (*exterior orientation*) dapat diperoleh dengan metode *Closed form Solution* yang didefinisikan sebagai kumpulan metode untuk menyelesaikan bentuk persamaan non linier menjadi persamaan linier dalam bidang fotogrametri yang sesuai dan sedekat mungkin terhadap nilai parameter sebenarnya dengan nilai residu sekecil mungkin *(Shih* dan *Wang,* 1987*)*. Didalam *closed form*

solution terdapat beberapa metode yaitu meliputi : metode *Fischler-Bolles*, metode *Rampal*, metode *Church*, dan metode *Zeng*.

Satu metode yang dikembangkan oleh *Prof. Earl Church* dikenal dengan nama metode *Church*. Dengan metode *Church* parameter posisi kamera yang terkoreksi dapat diperoleh jika nilai awal posisi kamera (X_L, Y_L, Z_L) diketahui, dengan prinsip dasar bahwa nilai sudut untuk dua piramid mempunyai nilai yang sama yang dibentuk oleh piramid bidang foto dan bidang objek yang berpusat pada posisi kamera. Dengan adanya posisi kamera terkoreksi maka dapat disusun matriks rotasi dari foto yang miring (*American Society of Photogrammetry*, 1980).

Rampal (1979) menyatakan bahwa untuk memperoleh nilai parameter pendekatan orientasi luar (*exterior orientation*) dapat diperoleh dengan syarat :

1. Tidak memerlukan nilai pendekatan parameter.
2. Tidak memerlukan nilai pendekatan sudut rotasi.
3. Dapat digunakan tipe foto untuk di udara dan di darat .
4. Hanya digunakan satu foto dengan informasi koordinat foto dan koordinat objek.

Metode yang dikembangkan oleh *(Fischler* dan *Bolles*, 1981*)* menyebutkan bahwa untuk mendapatkan 6 parameter orientasi luar (*exterior orientation*) yang terdiri dari posisi kamera dan parameter rotasi, dapat diperoleh dengan menentukan posisi yang disebut dengan metode *The Location Determination Problem (LDP)* pada satu foto. Dimana penentuan posisi akan diselesaikan dengan solusi perkalian dari jumlah "*n*" titik, yang dikenal dengan permasalahan *PnP* yaitu jumlah titik yang

saling berhubungan antara bidang objek dan bidang foto menjadi (3,4,5) atau masalah *P3P*, *P4P*, dan *P4P*. Hal ini dikarenakan jika $n < 3$ maka solusi unik untuk permasalahan posisi tidak akan terpecahkan dan jika $n \geq 3$ maka akan diperoleh solusi unik secara linier.

Zeng dan *Wang* (1992) melakukan penelitian dengan mengembangkan metode yang telah dijelaskan dan diuji oleh *(Fischler* dan *Bolles*, 1981*)*. Metode itu dikenal dengan metode permasalahan penentuan lokasi atau *Location determination Problem* (LDP) untuk tiga titik pada satu foto, sehingga dapat diperoleh posisi koordinat kamera dan parameter rotasi dengan menggunakan prinsip perkalian murni (*Fischler* dan *Bolles*, 1981; dan *Zeng* dan *Wang*, 1992).

1.2 Maksud dan Tujuan Penelitian

1.2.1 Maksud Penelitian

Penelitian ini dilakukan untuk menentukan nilai pendekatan parameter luar (*exterior orientation*) yang terdiri dari tiga koordinat posisi kamera (X_L, Y_L, Z_L) dan tiga parameter sudut rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ), dengan menggunakan metode *closed form solution* yang meliputi : metode *Fischler-Bolles*, metode *Church*, metode *Rampal* dan metode *Zeng* sehingga diperoleh batasan untuk parameter koordinat kamera dan parameter sudut rotasi.

1.2.2 Tujuan Penelitian

Menyusun algoritma dari proses perhitungan untuk menentukan nilai pendekatan parameter luar (*exterior orientation*) yang terdiri dari tiga koordinat

posisi kamera (X_L, Y_L, Z_L) dan tiga parameter sudut rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ), dengan menggunakan metode *closed form solution*.

1.3 Perumusan Penelitian

Didalam proses evaluasi metode penentuan pendekatan parameter luar (*exterior orientation*), digunakan beberapa metode *closed form solution* dengan data awal berupa koordinat foto (x,y), panjang fokus (f) dan koordinat objek (X,Y,Z) untuk minimal 4 titik yang sama pada koordinat foto dan koordinat objek. Sehingga perlu dianalisa metode *closed form solution* yang akurat dan cocok untuk mendapatkan nilai pendekatan parameter luar (*exterior orientation*) dengan nilai residu sekecil mungkin.

1.4 Batasan Masalah

Adapun batasan-batasan masalah yang akan dibahas yaitu antara lain :

1. Menentukan nilai pendekatan parameter luar (*exterior orientation*) untuk satu foto dengan menggunakan metode *closed form solution*.
2. Menganalisa hasil akhir dari masing-masing metode *closed form solution* yang meliputi metode *Fischler-Bolles*, metode *Church*, dan metode *Zeng*
3. Menyusun algoritma dari masing-masing metode *closed form solution* yang meliputi metode *Fischler-Bolles*, metode *Church*, dan metode *Zeng*

1.5 Manfaat Penelitian

Penelitian yang dilakukan ini akan memberikan manfaat yaitu dihasilkannya sebuah algoritma dari hasil evaluasi penentuan pendekatan parameter luar (*exterior orientation*) yang akan digunakan untuk proses perhitungan selanjutnya.

1.6 Tinjauan Pustaka

Fotogrametri merupakan suatu teknik untuk memperoleh informasi mengenai posisi, ukuran dan bentuk dari objek dengan mengidentifikasi beberapa foto secara langsung (*Cooper* dan *Robson*, 2001). Menentukan letak, posisi, dan jenis geometri secara intrinsik dari kamera dikenal sebagai permasalahan dasar dalam fotogrametri yang diringkas dalam beberapa tahapan yaitu penentuan parameter orientasi dalam (*interior orientation*) dan orientasi luar (*exterior orientation*) dari kamera, dan juga penentuan koordinat 3D titik objek *(Grussenmeyer* dan *Al Khalil,* 2000*)*. Didalam fotogrametri terdapat tiga dasar kondisi yang sering digunakan untuk perhitungan parameter orientasi luar (*interior orientation*). Kondisi ini dikenal dengan kondisi *collinearity* (kondisi kesegarisan), *coplanarity* (kondisi kesebidangan) dan *coangularity* (kondisi kesudutan).

Parameter orientasi luar (*Exterior Orientation* atau *EO)* dari satu foto, digunakan untuk menstabilkan hubungan antara sistem koordinat foto (x,y,z) dan sistem koordinat objek (X,Y,Z), sehingga dapat ditentukan enam parameter *unknown* dari EO yaitu terdiri dari tiga koordinat posisi kamera (X_L, Y_L, Z_L) dan tiga parameter sudut rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ) *(Mikhail,et al).* Untuk mendapatkan batasan nilai pendekatan dan nilai residu sekecil mungkin untuk parameter luar

(*Exterior Orientation*) maka diperlukan metode yang akurat yaitu dengan menggunakan metode *closed form solution*.

Metode *Closed form Solution* merupakan kumpulan metode untuk menyelesaikan bentuk persamaan non linier menjadi persamaan linier dalam bidang fotogrametri untuk memperoleh parameter pendekatan yang sesuai dan sedekat mungkin terhadap nilai parameter sebenarnya dengan nilai residu sekecil mungkin (*Shih* dan *Faig*, 1987).

Didalam bab II akan dijelaskan tentang macam-macam metode closed *form solution* yang meliputi metode *Fischler-Bolles*, metode *Church*, metode *Rampal* dan metode *Zeng*, sehingga diperoleh batasan nilai untuk parameter koordinat kamera (X_L,Y_L,Z_L) dan parameter sudut rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ). Pada bab III akan dijelaskan tentang proses perhitungan dari masing-masing metode *closed form solution*. Dan pada bab IV meliputi proses analisa terhadap hasil perhitungan dari masing-masing metode *closed form solution* untuk mendapatkan nilai pendekatan awal parameter orientasi luar (*exterior orientation*) yang akan disimpulkan pada bab V.

BAB II

DASAR TEORI

Penentuan enam parameter orientasi luar merupakan dasar yang terpenting dalam bidang fotogrametri yaitu terdiri dari tiga koordinat posisi kamera (X_L, Y_L, Z_L) dan tiga parameter sudut rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ) *(Mikhail,et al)*. Enam parameter EO dapat diperoleh dengan metode *Closed form Solution* yang didefinisikan sebagai kumpulan metode untuk menyelesaikan bentuk persamaan non linier dalam bidang fotogrametri untuk memperoleh parameter pendekatan yang sesuai dan sedekat mungkin terhadap nilai parameter sebenarnya dengan nilai residu sekecil mungkin *(Shih* dan *Wang, 1987)*.

Didalam *closed form solution*, terdapat 4 metode tetapi pembahasan didalam dasar teori ini hanya menjelaskan empat metode, yaitu : metode *Fischler-Bolles*, metode *Rampal*, metode *Church*, dan metode *Zeng*. Keempat metode diatas akan lebih detail lagi dijelaskan dalam sub bab 2.3, 2.4, 2.5, dan 2.6.

2.1 Orientasi Luar (*Exterior Orientation*)

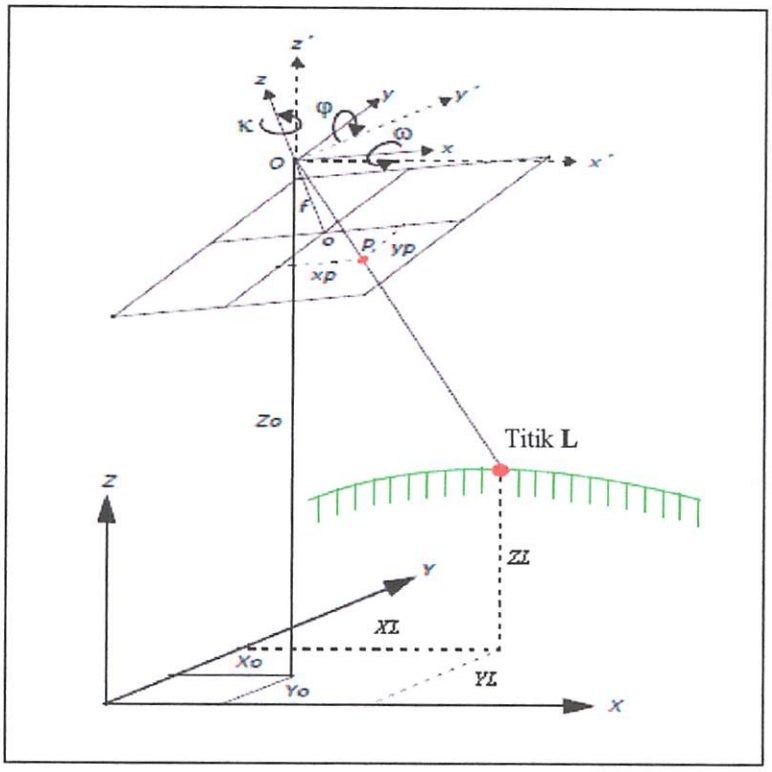
Menentukan letak, posisi, dan jenis geometri secara intrinsik dari kamera umumnya dikenal dengan masalah dasar dalam fotogrametri yang meliputi penentuan parameter orientasi dalam dan orientasi luar dari kamera, serta penentuan koordinat 3D titik objek *(Grussenmeyer* dan *Al Khalil, 2000)*. Didalam fotogrametri terdapat tiga dasar kondisi yang sering digunakan untuk perhitungan parameter orientasi luar.

Kondisi ini dikenal dengan kondisi *collinearity* (kondisi kesegarisan), *coplanarity* (kondisi kesebidangan) dan *coangularity* (kondisi kesudutan).

Grussenmeyer dan *Al Khalil* (2002) berpendapat bahwa permasalahan penentuan parameter orientasi luar di *computer vision* dikenal sebagai estimasi posisi, prinsip dan cara perhitungannya berbeda dengan prinsip fotogrametri. Tujuan dasar penelitian adalah untuk memperoleh solusi sebenarnya dari permasalahan estimasi posisi dengan menggunakan sejumlah informasi objek yang berdasarkan atas konsep dari proyeksi geometri secara aljabar yang digunakan untuk menurunkan parameter kamera sebagai ekuivalen untuk persamaan *Direct Linier Transformation (DLT)*.

2.1.1 Parameter Orientasi Luar (*Exterior Orientation* atau EO)

Parameter orientasi luar (*Exterior Orientation* atau *EO*) dari satu foto digunakan untuk menstabilkan hubungan antara sistem koordinat foto (x,y,f) dan sistem koordinat objek (X,Y,Z), sehingga dapat ditentukan enam parameter *unknown* dari EO yang terdiri dari tiga koordinat posisi kamera (X_L,Y_L,Z_L) dan tiga parameter sudut rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ) *(Mikhail,et al)*. Hubungan dari parameter EO dapat diilustrasikan seperti Gambar (2.1).



Gambar 2.1 Elemen dari Orientasi Luar, Leica (2006)

Dimana :

x_p, y_p : koordinat *principal point*

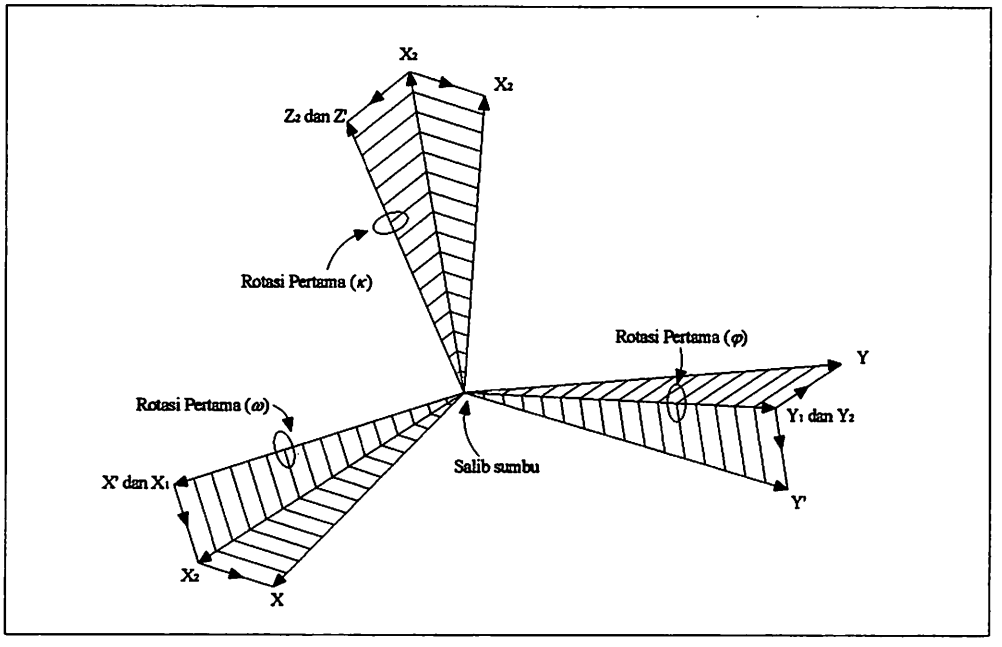
f : panjang fokus

ω, φ, κ : parameter rotasi yang berputar pada sumbu koordinat x,y,z

X_o, Y_o, Z_o : koordinat 3-dimensi titik objek

X_L, Y_L, Z_L : koordinat 3-dimensi kamera saat pemotretan

Dengan arah rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ) pada masing-masing sumbu koordinat x,y,z, sebagai berikut (*Wolf* dan *Dewitt, 2000)* :



Gambar 2.2 perputaran sumbu rotasi

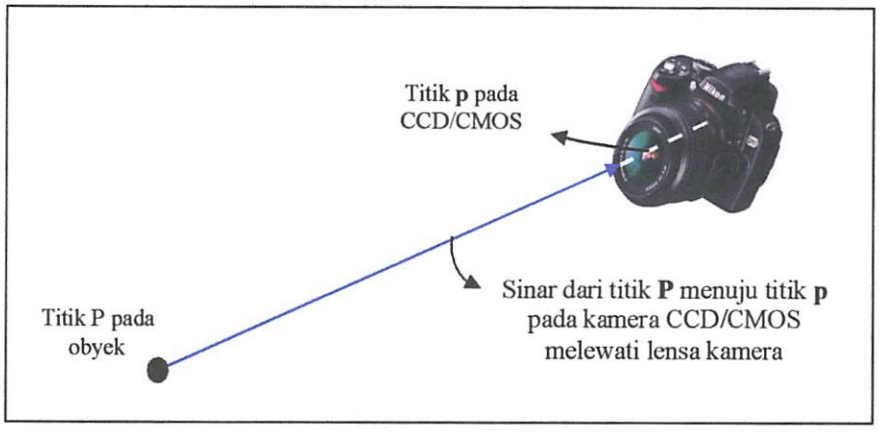
dimana :

ω,φ,κ : parameter rotasi yang berputar pada sumbu koordinat x,y,z

x,y,z : sistem koordinat foto

Dapat diketahui dari Gambar (2.2) bahwa *omega* (ω) diputar pada sumbu x sistem koordinat foto x,y,z, *phi* (φ) diputar pada sumbu y sistem koordinat foto x,y,z, dan *kappa* (κ) diputar pada sumbu z sistem koordinat foto x,y,z. Perputaran sudut ini didefinisikan sebagai hal yang positif jika ketiga parameter rotasi diputar berlawanan dengan arah jarum jam pada masing-masing sumbu perputaran sudut rotasi (*Wolf* dan *dewitt*, 2000).

Dari Gambar (2.1) dapat dilihat terbentuknya kondisi yang disebut kondisi kesegarisan (*collinearity*). Dimana berkas sinar pantulan dari obyek **P** yang menuju titik tengah **p** (*perspective cen*ter) pada bidang sensor kamera berupa garis lurus (*Cooper* dan *Robson*, 2001) Gambar (2.3). Tetapi pada kenyataannya, berkas sinar pada bidang sensor kamera mengalami pembelokkan (distorsi) baik karena kecacatan dalam proses perakitan dan penyusunan komponen lensa maupun karena ketidakstabilan posisi sensor CCD/CMOS didalam cangkang kamera.



Gambar 2.3 Kondisi ideal berkas sinar lensa kamera, Cooper dan Robson (2001)

Sehingga dapat dituliskan persamaan dari kondisi kolinear pada Gambar (2.3), sebagai berikut *(Mikhail,et al)* :

$$x - x_0 = -f \frac{m_{11}(X - X_L) + m_{12}(Y - Y_L) + m_{13}(Z - Z_L)}{m_{31}(X - X_L) + m_{32}(Y - Y_L) + m_{33}(Z - Z_L)}$$
$$y - y_0 = -f \frac{m_{21}(X - X_L) + m_{22}(Y - Y_L) + m_{23}(Z - Z_L)}{m_{31}(X - X_L) + m_{32}(Y - Y_L) + m_{33}(Z - Z_L)} \quad (2.1)$$

dimana :

x_0, y_0	: koordinat *principal point*
f	: panjang fokus
m_{ij}	: elemen dari matriks rotasi
X_L, Y_L, Z_L	: koordinat 3-dimensi posisi kamera
x,y	: koordinat foto
X,Y,Z	: koordinat 3-dimensi titik objek

Dari data awal yang tersedia berupa koordinat foto (x,y) dan koordinat tititk obyek (X,Y,Z), maka persamaan (2.1) dapat diselesaikan melalui metode Hitung Kuadrat Terkecil. Namun karena dalam prosesnya dibutuhkan suatu nilai pendekatan awal untuk parameter orientasi luar, maka didalam penyelesaiannya dilakukan dengan menggunakan teknik seperti yang dijelaskan oleh *(Rampal et.al)* untuk mendapatkan nilai pendekatan awal agar pada proses iterasi diperoleh posisi konvergen secara cepat (*Faig*, 1973).

2.1.2 Matrik Rotasi

Didalam penyusunan matrik rotasi diperlukan tiga parameter rotasi yaitu *omega* (ω), *phi* (φ), dan *kappa* (κ). Dengan menggunakan tiga sudut rotasi yang mempunyai hubungan antara sistem koordinat foto (x,y,z) dan sistem koordinat objek (X,Y,Z atau x',y',z') yang dapat ditentukan. prinsip dari perputaran parameter rotasi menggunakan prinsip tangan kanan, dengan sumbu rotasi positif searah dengan arah

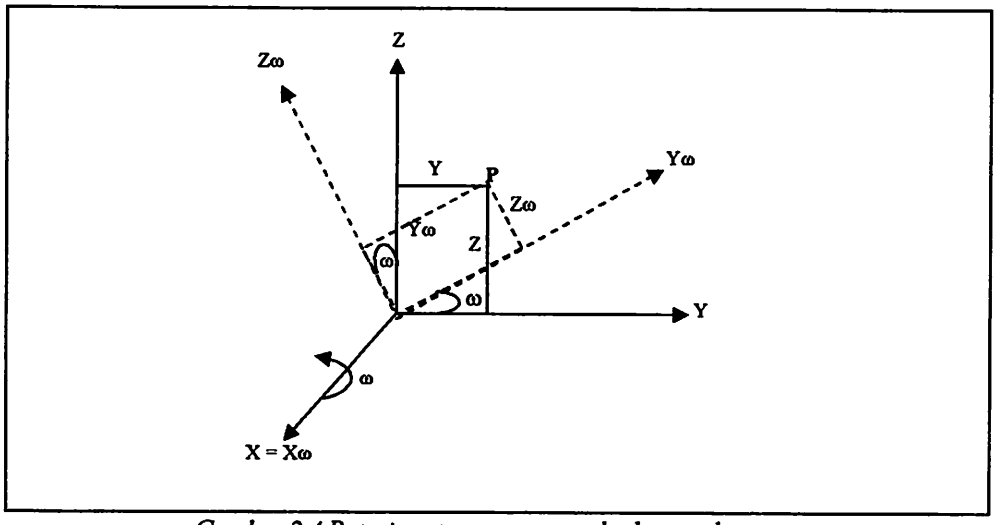
jarum jam. Sebuah matrik dengan dimensi 3x3 dapat mendifinisikan hubungan antara dua sistem yang digunakan. Adapun matrik rotasi dapat didefinisikan sebagai berikut *(Mikhail,et al)* :

$$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \qquad (2.2)$$

dimana :

m_{ij} : elemen dari matriks rotasi

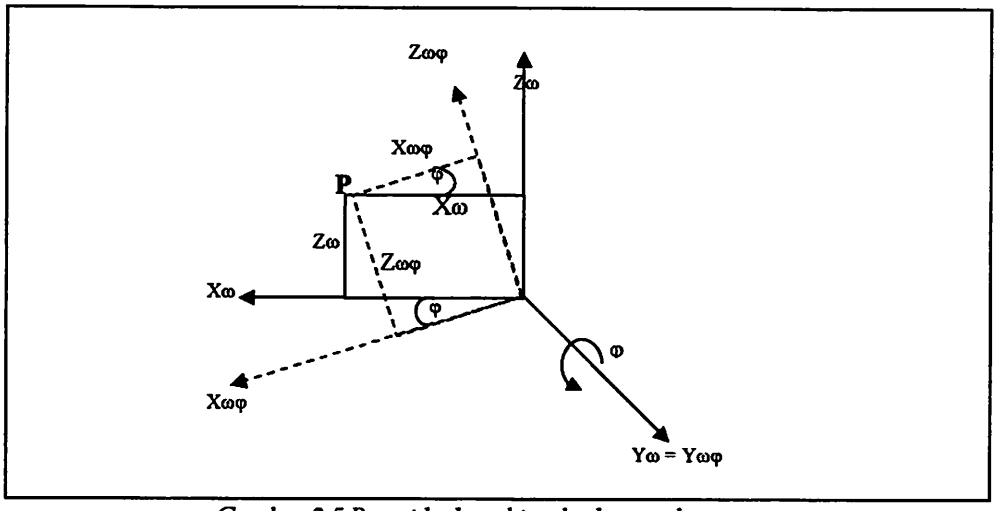
Untuk menyusun matrik diatas, tiga sudut rotasi harus didefinisikan secara bergantian. Rotasi pertama dilakukan terhadap sumbu x (*omega*), rotasi yang kedua dilakukan terhadap sumbu y (*phi*) dan rotasi yang ketiga dilakukan terhadap sumbu z (*kappa*). Sehingga akan diperoleh suatu persamaan rotasi, yang digunakan untuk menyusun matrik rotasi, seperti pada persamaan (2.2). Dan dapat dijabarkan sebagai berikut (*Mikhail, et. al*) :



Gambar 2.4 Rotasi pertama omega terhadap sumbu x

Dari sistem rotasi pertama (*omega*) terhadap sumbu x pada gambar diatas diperoleh persamaan sebagai berikut :

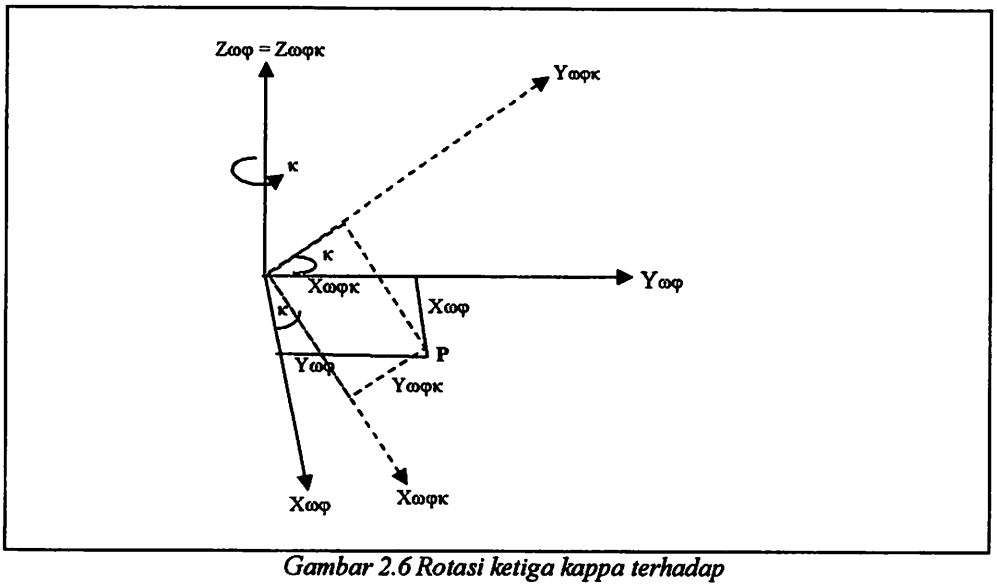
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\omega & \sin\omega \\ 0 & -\sin\omega & \cos\omega \end{bmatrix} \begin{bmatrix} X_\omega \\ Y_\omega \\ Z_\omega \end{bmatrix} = M_\omega \begin{bmatrix} X_\omega \\ Y_\omega \\ Z_\omega \end{bmatrix} \qquad (2.3)$$



Gambar 2.5 Rotasi kedua phi terhadap sumbu y

Dari sistem rotasi kedua (*phi*) terhadap sumbu y pada gambar di atas diperoleh persamaan berikut :

$$\begin{bmatrix} X_\omega \\ Y_\omega \\ Z_\omega \end{bmatrix} = \begin{bmatrix} \cos\varphi & 0 & -\sin\varphi \\ 0 & 1 & 0 \\ \sin\varphi & 0 & \cos\varphi \end{bmatrix} \begin{bmatrix} X_{\omega\varphi} \\ Y_{\omega\varphi} \\ Z_{\omega\varphi} \end{bmatrix} = M_\varphi \begin{bmatrix} X_{\omega\varphi} \\ Y_{\omega\varphi} \\ Z_{\omega\varphi} \end{bmatrix} \qquad (2.4)$$



Gambar 2.6 Rotasi ketiga kappa terhadap

Dari sistem rotasi ketiga (*kappa*) terhadap sumbu z pada Gambar 2.6 diperoleh persamaan berikut :

$$\begin{bmatrix} X_{\omega\varphi} \\ Y_{\omega\varphi} \\ Z_{\omega\varphi} \end{bmatrix} = \begin{bmatrix} \cos\kappa & \sin\kappa & 0 \\ -\sin\kappa & \cos\kappa & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_{\omega\varphi\kappa} \\ Y_{\omega\varphi\kappa} \\ Z_{\omega\varphi\kappa} \end{bmatrix} = M_{\kappa} \begin{bmatrix} X_{\omega\varphi} \\ Y_{\omega\varphi} \\ Z_{\omega\varphi} \end{bmatrix} \qquad (2.5)$$

Dengan mengkombinasi persamaan (2.3),(2.4), dan (2.5), akan didapat hubungan antara koordinat titik objek (*P*) relatif terhadap sistem koordinat (XYZ) dan ($X_{\omega\varphi\kappa}$,$Y_{\omega\varphi\kappa}$,$Z_{\omega\varphi\kappa}$), yaitu :

$$P = M_{\omega} x M_{\varphi} x M_{\kappa} x P_{\omega\varphi\kappa} \qquad (2.6)$$

Persamaan diatas diganti menjadi :

$$M_{\omega} x M_{\varphi} x M_{\kappa} \qquad (2.7)$$

Dimana matrik M mempunyai dimensi 3x3, sehingga diperoleh persamaan (2.2), dengan masing-masing elemen matrik koefisien sebagai berikut *(Mikhail,et al)* :

$$
\begin{aligned}
m_{11} &= \cos_{\varphi} x \cos \kappa \\
m_{12} &= \cos_{\omega} x \sin_{\kappa} + \sin_{\omega} x \sin_{\varphi} x \cos_{\kappa} \\
m_{13} &= \sin_{\omega} x \cos_{\kappa} - \cos_{\omega} x \sin_{\varphi} x \sin_{\kappa} \\
\\
m_{21} &= -\cos_{\varphi} x \sin_{\kappa} \\
m_{22} &= \cos_{\omega} x \cos_{\kappa} - \sin_{\omega} x \sin_{\varphi} x \sin_{\kappa} \\
m_{23} &= \sin_{\omega} x \cos_{\kappa} + \cos_{\omega} x \sin_{\varphi} x \sin_{\kappa} \\
\\
m_{31} &= \sin_{\varphi} \\
m_{32} &= -\sin_{\omega} x \cos_{\varphi} \\
m_{33} &= \cos_{\omega} x \cos_{\varphi}
\end{aligned}
\qquad (2.8)
$$

dimana :

m_{ij} : elemen dari matriks rotasi

2.2 *Closed Form Solution*

Nilai pendekatan parameter luar (*Exterior Orientation*) dapat diperoleh dengan menerapkan konsep perhitungan dari metode *Closed form Solution*. Dimana konsep perhitungannya merubah bentuk persamaan non linier dalam bidang fotogrametri untuk memperoleh parameter pendekatan yang sesuai dan sedekat mungkin terhadap nilai parameter sebenarnya dengan nilai residu sekecil mungkin (*Shih* dan *Faig*, 1987). Model persamaan *collinearity* (kesegarisan) memberikan penyelesaian yang lebih lazim dan biasa digunakan, sehingga dengan menggunakan model persamaan tersebut dapat ditentukan enam parameter secara tepat. Akan tetapi, pendekatan ini memerlukan proses linierisasi, yang berdasarkan pada proses penentuan nilai yang benar dari nilai pendekatan awal (*Shih* dan *Faig*, 1987).

Didalam *Closed form Solution*, terdapat beberapa solusi untuk model persamaan tersebut antara lain : *Church*, memberikan penyelesaian berdasarkan model piramid foto, yang dikembangkan 50 tahun yang lalu dan dikenal dengan metode *Church* (*American Society of Photogrammetry,1980*). *Church* menggunakan model persamaan yang hampir sama dengan model persamaan *collinearity* (kesegarisan) dengan menurunkan satu set parameter yang diketahui parameter posisi yang dicakup. Akan tetapi bentuk persamaan metode *Church* merupakan persamaaan yang non-linier, sehingga perlu dilakukan proses linierisasi.

Metode Church mengabaikan persyaratan untuk penentuan nilai pendekatan awal dan diasumsikan bahwa : bidang objek mendekati sejajar dengan bidang foto yang membentuk model piramid, sehingga diperoleh nilai sudut yang sama antara

sudut koordinat kamera-koordinat objek dan koordinat kamera-koordinat objek pada hukum *cosinus*.

Berbeda dengan 3 parameter dan 6 parameter reseksi, terdapat 11 parameter reseksi yang dikembangkan oleh (*Azis* dan *Karara*, 1971). Model ini dikenal dengan DLT (*Direct Linier Transformation*) yang mencakup 11 parameter aljabar dan tidak membutuhkan kalibrasi kamera serta nilai pendekatan awal. Prinsip yang mendasar adalah perbandingan antara koordinat foto dan koordinat objek secara langsung yang menyatukan persamaan *collinearity* (kesegarisan) untuk mendapatkan koreksi untuk distorsi lensa (*Aziz* dan *Karara*, 1971). (*Hadem*, 1981) dan (*Okamoto*, 1981) menunjukkan bahwa 11 parameter DLT adalah setara dengan 6 parameter orientasi luar dan 5 parameter orientasi dalam.

Metode yang dikembangkan oleh *(Fischler* dan *Bolles*, 1981*)* menyebutkan bahwa untuk mendapatkan 6 parameter orientasi luar yang terdiri dari posisi kamera dan parameter rotasi, dilakukan dengan menentukan posisi yang disebut dengan metode *The Location Determination Problem (LDP)* pada satu foto. Dimana penentuan posisi akan diselesaikan dengan solusi perkalian dari jumlah "*n*" titik, yang dikenal dengan permasalahan *PnP* yaitu jumlah titik yang saling berhubungan antara bidang objek dan bidang foto menjadi (3,4,5) atau masalah *P3P*, *P4P*, dan *P4P*. Hal ini dikarenakan jika $n < 3$ maka solusi unik untuk permasalahan posisi tidak akan terpecahkan dan jika $n \geq 3$ maka akan diperoleh solusi unik secara linier *(Fischer* dan *Bolles*, 1981*)*.

Inti dari masalah penentuan lokasi pada analisa foto adalah untuk menstabilkan hubungan antara perwakilan dua parameter yang diberikan oleh lokasi tertentu. Untuk menentukan lokasi bidang dari foto diperoleh dengan menentukan satu set titik kontrol objek yang muncul pada foto atau disebut dengan masalah penentuan parameter orientasi luar dari kamera. Dengan adanya lokasi spasial yang relatif dari titik kontrol dan adanya nilai untuk setiap pasang sudut titik kontrol dari tambahan titik yang disebut titik tengah kamera (*Center of perspective* atau CP), ditemukan panjang dari kaki (*leg*) yang digabung oleh CP ke titik kontrol yang lain. Proses ini dinamakan "*perspective-n-problem* (PnP)" *(Fischer* dan *Bolles,* 1981*)*.

(Zeng dan *Wang*, 1992) melakukan penelitian dengan menggunakan metode yang telah dijelaskan dan diuji oleh *(Fischler* dan *Bolles,* 1981*)*. Metode itu dikenal dengan metode permasalan penentuan lokasi atau *Location determination Problem* (LDP) untuk analisa foto dan memperoleh posisi koordinat objek dengan menggunakan prinsip perkalian murni (*Fischler* dan *Bolles*, 1981; dan *Zeng* dan *Wang*, 1992).

Metode yang dikembangkan oleh (*Zeng* dan *Wang*, 1992) mencakup tiga tahapan inti yaitu :

1. Penyelesaian untuk memperoleh parameter pendekatan posisi koordinat kamera (X_L,Y_L,Z_L).
2. Penyelesaian untuk memperoleh parameter rotasi *omega* (ω), *phi* (φ), dan *kappa* (κ).

3. Mendiskusikan penyelesaian untuk memperoleh parameter reseksi dengan menggunakan prinsip kurva kritis (*danger cylinder*).

2.2.1 Macam-macam *Closed Form Solution*

Didalam bidang fotogrametri terdapat beberapa metode *closed from solution*, tetapi pembahasan didalam dasar teori ini hanya menjelaskan empat metode, yaitu :

1. Metode *Fischler-Bolles*
2. Metode *Rampal*
3. Metode *Church*
4. Metode *Zhang*

Keempat metode diatas akan lebih detail lagi dijelaskan dalam sub bab 2.3, 2.4., 2.5 dan 2.6.

2.3 Metode *Fischler-Bolles*

Didalam Metode *Fischler-Bolles* akan lebih banyak dijelaskan tentang *The Location Determination Problem* (LDP) yaitu memberi ilustrasi sebuah foto dari objek yang telah diketahui lokasinya, dengan menentukan titik-titik pada bidang foto yang telah diketahui. Dimana penentuan posisi akan diselesaikan dengan solusi perkalian dari jumlah "*n*" titik, yang dikenal dengan permasalahan *PnP* yaitu jumlah titik yang saling berhubungan antara bidang objek dan bidang foto menjadi (3,4,5) atau masalah *P3P*, *P4P*, dan *P4P*. Hal ini dikarenakan jika $n < 3$ maka solusi unik

untuk permasalahan posisi tidak akan terpecahkan dan jika $n \geq 3$ maka akan diperoleh solusi unik secara linier. (*Fischer* dan *Bolles*, 1981).

2.3.1. Paradigma *RANSAC*

Secara konsep, interpretasi akan melibatkan dua aktifitas yang berbeda yaitu : pertama, menemukan kecocokan yang terbaik antara data dan satu model yang tersedia; kedua, menghitung nilai yang terbaik untuk parameter bebas dari model yang dipilih (nilai parameter pendekatan) (*Fischler* dan *Bolles*, 1981). Konsep yang ada membentuk paradigma baru didalam metode *RANSAC*, yang dikenal paradigma *RANSAC*. Paradigma ini meliputi tiga parameter yaitu :

1. Menentukan batas toleransi untuk menstabilkan nilai kesesuaian model.
2. Menentukan nilai maksimum dari percobaan untuk menentukan jumlah ketetapan.
3. Menurunkan batasan ukuran ketetapan yang dapat diterima.

Dengan adanya paradigma diatas, maka dapat ditentukan suatu ketetapan untuk mendapatkan batasan jumlah titik (n) yang mempunyai standart deviasi kurang dari nol (SD(k) < 0) (persamaan 2.9), yang diperlukan untuk memperoleh solusi yang unik dalam metode PnP.

$$SD(k) = \sqrt{(1 - w^n) * (1 / w^n)} \qquad (2.9)$$

dimana :

SD (k) : standart deviasi dari jumlah percobaan *(k)*

w : nilai kemungkinan pada tiap titik data yang dipilih

n : data titik

dari (persamaan 2.9), diperoleh hasil tabulasi seperti di bawah ini (*Fischler* dan *Bolles*, 1981) :

Tabel 2.1 Tabulasi nilai E(k) untuk hubungan nilai *w* dan *n*

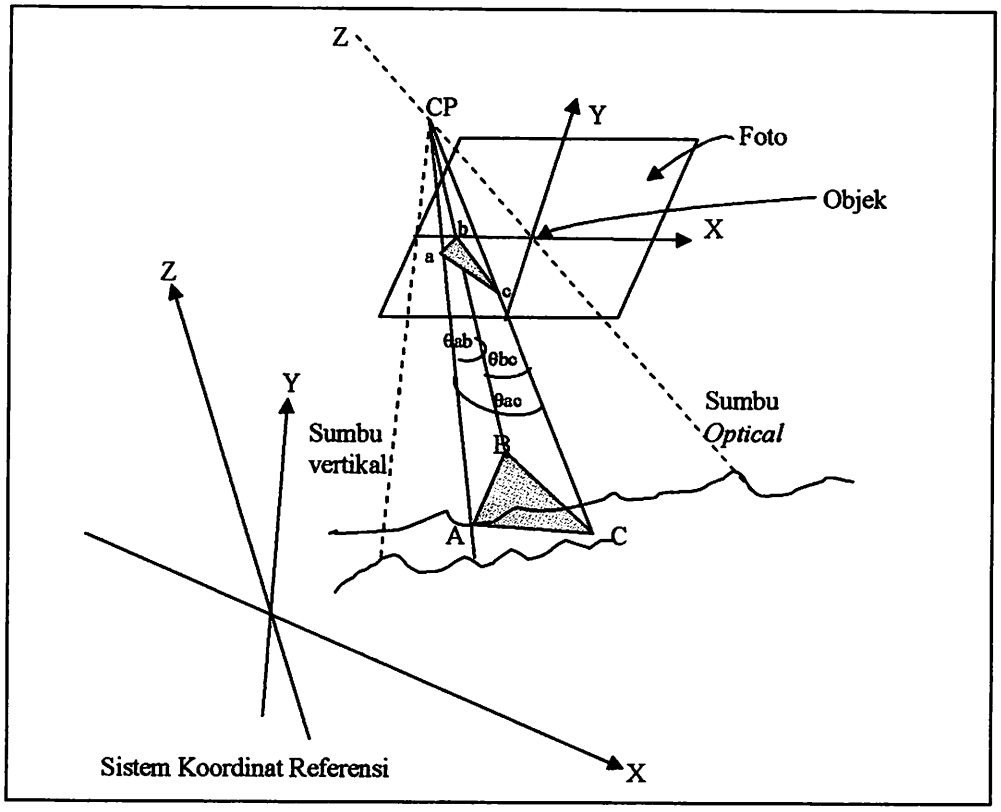
w	n =1	2	3	4	5	6
0.9	1.1	1.2	1.4	1.5	1.7	1.9
0.8	1.3	1.6	2	2.4	3	3.8
0.7	1.4	2	2.9	4.2	5.9	8.5
0.6	1.7	2.8	4.6	7.7	13	21
0.5	2.0	4.0	8.0	16	32	64
0.4	2.5	6.3	16	39	98	244
0.3	3.3	11	37	123	412	
0.2	5	25	125	625		

Sehingga dari tabel 2.1 dapat diketahui bahwa diketahui bahwa n =1 dan 2 dengan persentasi kemungkinan (w) 90 % mempunyai jumlah solusi yang tidak terbatas, sedangkan $n \geq 3$ dengan persentasi kemungkinan (w) 90 % mempunyai jumlah solusi yang unik.

2.3.2. Penentuan Lokasi (LDP)

Inti masalah pada analisa foto adalah menstabilkan hubungan antara perwakilan dua elemen yang diberikan oleh lokasi tertentu. Untuk menentukan lokasi bidang dari foto diperoleh dengan menentukan satu set titik kontrol objek yang muncul pada foto atau disebut dengan masalah penentuan elemen orientasi luar dari kamera atau permasalahan kalibrasi kamera atau permasalahan hubungan bidang foto foto dan bidang objek (*Fischler dan Bolles*, 1981). Permasalahan penentuan lokasi secara rutin diselesaikan dengan teknik kuadrat terkecil (*least squares*) misalnya pada penjelasan (*Wolf*, 1983).

Dapat dinyatakan bahwa definisi LDP yaitu memberikan satu set objek *m*, dimana diketahui koordinat 3D dari sistem koordinat dan memberi satu set koordinat foto dari objek *m* yang terlihat sehingga dapat ditentukan lokasi yang dicari (relatif untuk sistem koordinat objek) dari koordinat foto yang ditentukan (*Fischler* dan *Bolles*, 1981). (*Fischler* dan *Bolles*, 1981) mengasumsikan bahwa terdapat dua persamaan titik tengah foto (*principal point*) pada sistem koordinat foto, dimana sumbu *optical* dari lubang kamera terletak segaris dengan titik tengah bidang foto dan fokus dari sistem yang telah diketahui, Gambar (2.7). Dengan mudah maka dapat dihitung beberapa pasang sudut titik objek dari titik tengah kamera (*Center of perspective* atau CP).

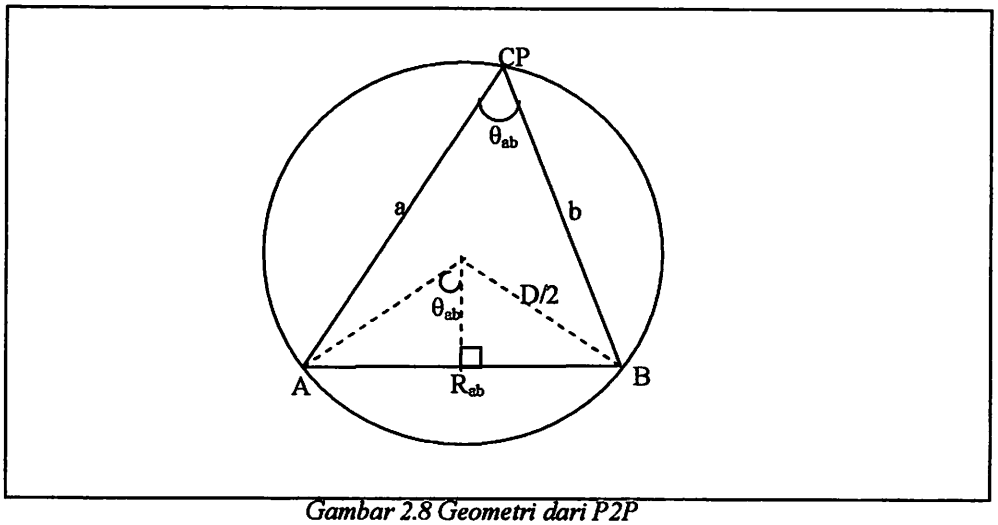


Gambar 2.7 Geometri dari Location Determination Problem

Dengan adanya lokasi spasial yang relatif dari titik kontrol dan nilai untuk setiap pasang sudut titik kontrol dari tambahan titik yang disebut titik tengah kamera (*Center of perspective* atau CP), ditemukan panjang dari kaki (*leg*) yang digabung oleh CP ke titik kontrol yang lain. Proses ini dinamakan "*perspective-n-problem* (PnP)" (*Fischler* dan *Bolles*, 1981).

2.3.3. Metode *Perspective-N-Point Problem* (PnP)

Dalam permasalahan penentuan titik *n* pada metode (PnP) perlu dilakukan analisa, dengan menguji untuk beberapa titik objek. Permasalahan P1P (n = 1) tidak memberikan informasi yang *constrain*, sehingga jumlah solusi yang mungkin jumlahnya tidak terbatas. Untuk permasalahan P2P (n = 2), sesuai dengan ilustrasi pada Gambar 2.8. Juga diakui bahwa solusi yang mungkin jumlanya tidak terbatas, dimana CP dapat terletak dimana saja pada posisi lingkaran atau diameter *Rab/sin (θab)*, dan perputaran pada gabungan dua titik kontrol A dan B (*Fischler* dan *Bolles*, 1981).



Gambar 2.8 Geometri dari P2P

Untuk permasalahan P3P (n = 3) diperlukan panjang dari tiga *leg* (kaki) *tetrahedron* yang memberikan dimensi kaki dan permukaan sudut yang berlawanan dengan tiga bidang sudut (lihat Gambar 2.9), sehingga diperoleh solusi dari permasalahan n = 3, terdiri dari tiga persamaan yaitu (*Fischler* dan *Bolles*, 1981) :

$$(Rab)^2 = a^2 + b^2 - 2 * a * b * \cos(\theta ab)$$
$$(Rac)^2 = a^2 + c^2 - 2 * a * c * \cos(\theta ac) \qquad (2.10)$$
$$(Rbc)^2 = b^2 + c^2 - 2 * b * c * \cos(\theta bc)$$

dimana :

R_{ab}, R_{ac}, R_{bc} : jarak antara titik kontrol

a,b,c : panjang kaki (*legs*) antara titik tengah kamera dan titik kontrol

$\theta_{ab}, \theta_{ac}, \theta_{bc}$: sudut yang terbentuk pada titik kontrol

Dimana hal itu dikenal dengan parameter *n* yang bebas dalam persamaan *polynomial*. Dengan diketahui nilai *n* maka dapat diperoleh solusi yang tidak lebih banyak daripada derajat produk masing-masing (*Fischler* dan *Bolles*, 1981).

Pada P4P (n = 4), keempat titik kontrol terletak pada satu garis lurus (tidak meliputi CP dan tidak lebih dari dua titik kontrol yang terletak pada garis tunggal). Untuk 5 titik kontrol, disarankan posisi secara umum terdapat dua solusi untuk permasalahan P5P, 4 atau lebih titik ini harus terletak pada garis lurus yang sama dan tidak lebih panjang dari posisi secara umum.

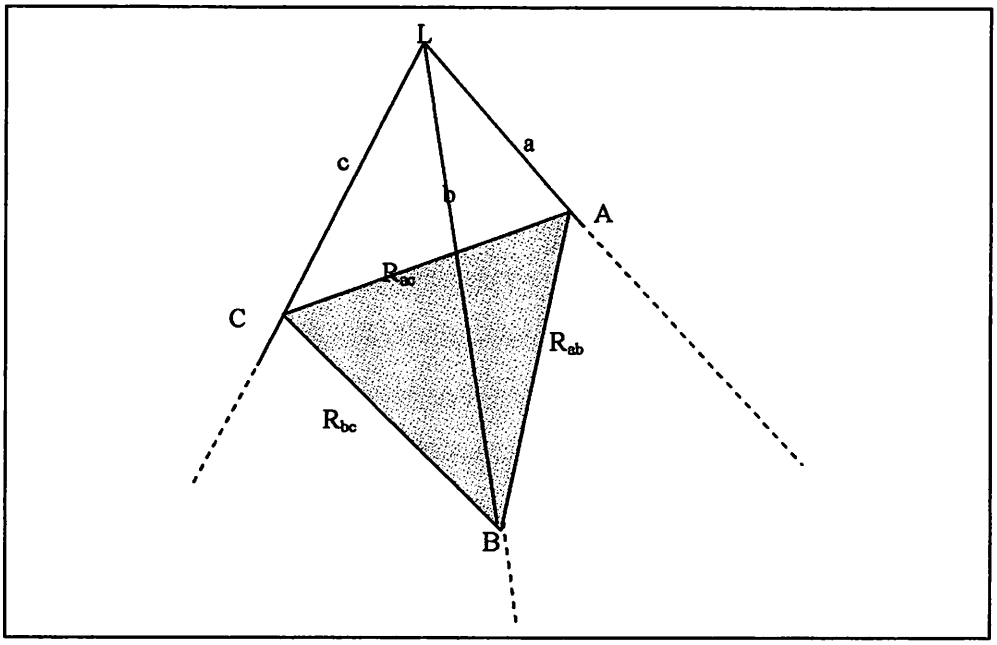
Untuk membuktikan 6 atau lebih titik kontrol secara umum akan menghasilkan solusi yang unik untuk permasalahan P6P, dengan catatan terdapat 12 koefisien dari matrik T 3x4 yang dikhususkan untuk pemetaan (pada koordinat *homogeneous*) dari tiga atau dua bidang yaitu tiap enam parameter memberikan tiga persamaan baru dan terdapat satu tambahan parameter *unknown* (koordinat *homogeneous* faktor skala) (*Fischler* dan *Bolles*, 1981).

2.3.4. Metode *Perspective-3-Point Problem* (P3P)

Pada sub bab ini akan dijelaskan lebih detail lagi proses perhitungannnya, sehingga diperoleh solusi dengan menggunakan 3 titik koordinat objek (*Fischler* dan *Bolles*, 1981). Didalam menstabilkan permasalahan P3P (n = 3) diperoleh 2 solusi, dengan menurunkan bentuk *closed form solution*. Proses perhitungan dapat dilakukan dalam tiga tahap yaitu (*Fischler* dan *Bolles*, 1981) :

1. Menentukan panjang kaki dari CP dengan menggunakan 3 titik kontrol objek dan sudut muka yaitu sudut yang terbentuk dari 3 pasang titik kontrol yang menuju CP.
2. Menentukan lokasi CP dalam bentuk 3D dalam sistem koordinat foto.
3. Menghitung orientasi sistem koordinat foto ke bentuk 3D.

Solusi untuk P3P dapat dilakukan dengan menentukan panjang kaki dari CP dengan menggunakan 3 titik kontrol (lihat pada Gambar 2.9) dan diberikan R_{ab}, R_{ac}, dan R_{bc} yang merupakan panjang basis dari *tetrahedron*, *a,b* dan *c* merupakan panjang kaki dari tetrahedron, cos*(θab)*, *cos(θac)*, dan *cos(θbc)* adalah hubungan sudut muka dari 3 bidang sudut.



Gambar 2.9 Geometri dari P3P

Solusi diatas dapat diperoleh dengan penyelesaian secara simultan dari persamaan (*Fischer* dan *Bolles*, 1981) :

$$\begin{aligned}(Rab)^2 &= a^2 + b^2 - 2 * a * b * \cos(\theta ab)\\ (Rac)^2 &= a^2 + c^2 - 2 * a * c * \cos(\theta ac)\\ (Rbc)^2 &= b^2 + c^2 - 2 * b * c * \cos(\theta bc)\end{aligned} \qquad (2.11)$$

dimana :

R_{ab},R_{ac},R_{bc} : jarak antara titik kontrol

a,b,c : panjang kaki (*legs*) antara titik tengah kamera dan titik kontrol

θ_{ab},θ_{ac},θ_{bc} : sudut yang terbentuk pada 3 bidang sudut

Dimana dapat ditentukan untuk nilai sudut dari cos (θ_{ab}), $cos\ (\theta_{ac})$, dan $cos\ (\theta_{bc})$ menggunakan persamaan (2.12).

$$\cos\theta_{ab} = \frac{x_a x_b + y_a y_b + f^2}{(L_a)\ (L_b)}$$

$$\cos\theta_{bc} = \frac{x_b x_c + y_b y_c + f^2}{(L_b)\ (L_c)} \qquad (2.12)$$

$$\cos\theta_{ac} = \frac{x_a x_c + y_a y_c + f^2}{(L_c)\ (L_a)}$$

dimana :

$\theta_{ab}, \theta_{ac}, \theta_{bc}$: sudut yang terbentuk pada 3 bidang sudut

x_i, y_i : koordinat foto

f : panjang fokus

L_a, L_b, L_c : jarak antara koordinat foto

Nilai untuk parameter *a,b,* dan *c* diperoleh dari bentuk persamaan kuadrat *polynomial* sebagai berikut (*Fischler* dan *Bolles,* 1981) :

$$G_4 x(x^4) + G_3 x(x^3) + G_2 x(x^2) + G_1 x(x^1) + G_0 x(x0) = 0 \qquad (2.13)$$

Nilai koefisien dari persamaan (2.13) dapat dituliskan sebagai berikut (*Fischler* dan *Bolles,* 1981).

$$G_4 = (K_1K_2 - K_1 - K_2)^2 - 4K_1K_2(\cos(\theta_{bc})^2)$$

$$G_3 = 4(K_1K_2 - K_1 - K_2)K_2x(1-K_1)\cos(\theta_{ab}) + 4K_1\cos(\theta_{bc})x((K_1K_2 + K_2 - K_1)x\cos(\theta_{ac}) + 2K_2x\cos(\theta_{ab})x\cos(\theta_{bc}))$$

$$G_2 = (2K_2x(1-K_1)\cos(\theta_{ab}))^2 + 2x(K_1K_2 + K_1 - K_2)x(K_1K_2 - K_1 - K_2) + 4 \quad (2.14)$$
$$xK_1((K_1 - K_2)x(\cos(\theta_{bc})^2) + (1-K_2)xK_1x(\cos(\theta_{ac})^2) - 2xK_2x(1+K_1)$$
$$x\cos(\theta_{ab})x\cos(\theta_{ac})x\cos(\theta_{bc}))$$

$$G_1 = 4x(K_1K_2 + K_1 - K_2)xK_2x(1-K_1)x\cos(\theta_{ab}) + 4K_1x(K_1K_2 - K_1 + K_2)x$$
$$\cos(\theta_{ac})x\cos(\theta_{bc}) + 2K_1K_2x\cos(\theta_{ab})x(\cos(\theta_{ac})^2))$$

$$G_0 = (K_1K_2 + K_1 - K_2)^2 - 4x(K_1^2)xK_2x(\cos(\theta_{ac})^2)$$

dimana

G_i : nilai koefisien dari bentuk *biqudratic polynomial*

$\theta_{ab}, \theta_{ac}, \theta_{bo}$: sudut yang terbentuk pada 3 bidang sudut

K_i : nilai koefisien dari perbandingan jarak ketiga titik kontrol

Dan untuk nilai koefisien *Ki* yaitu (*Fischler* dan *Bolles,* 1981) :

$$K_1 = \frac{(R_{bc})^2}{(R_{ac})^2} \quad dan \quad K_2 = \frac{(R_{bc})^2}{(R_{ab})^2} \qquad (2.15)$$

Dari tiap real positif pada bentuk persamaan kuadrat *polynomial* dari persamaan (2.13), dapat ditentukan real posisi tunggal untuk tiap panjang kaki *a* dan *b*. Sehingga diperoleh persamaan seperti dibawah ini (*Fischler* dan *Bolles,* 1981) :

$$a = \frac{R_{ab}}{\sqrt{(x^2) - 2 * x\cos(\theta_{ab}) + 1}}$$

$$b = a * x \qquad (2.16)$$

dimana :

R_{ab} : jarak antara titik kontrol A dan B

θ_{ab} : sudut yang terbentuk pada titik kontrol A dan B

a,b : panjang kaki (*legs*) antara titik tengah kamera dan titik kontrol

x : nilai akar real positif

Dan untuk mencari nilai y maka *(Fischer dan Bolles, 1981)*:

$$y = \cos(\theta_{ac}) + \sqrt{(\cos(\theta_{ac}))^2 + \frac{(R_{ac})^2 - (a^2)}{(a^2)}} \qquad (2.17)$$

Dimana :

R_{ac} : jarak antara titik kontrol A dan C

θ_{ac} : sudut yang terbentuk pada titik kontrol A dan C

a : panjang kaki (*legs*) antara titik tengah kamera dan titik kontrol A

Untuk tiap nilai positif dari y dapat diperoleh nilai c yang menjadi (*Fischler* dan *Bolles*, 1981) :

$$c = y * a \qquad (2.18)$$

dimana :

a,c : panjang kaki (*legs*) antara titik tengah kamera dan titik kontrol

Ketika nilai y diperoleh maka solusi yang dihasilkan menjadi *invalid*. Dengan mengecek kembali hasil dari parameter *a,b* dan *c* terhadap persamaan (2.11).

2.3.5. Metode *Perspective-4-Point Problem* (P4P)

Pada sub bab ini menjabarkan tentang teknik analisa untuk memperoleh solusi untuk permasalahan P4P, jika keempat titik kontol yang terletak pada satu garis lurus (*Fischler* dan *Bolles*, 1981). Dengan menentukan hubungan antara 4 titik yang terletak pada satu garis lurus (yang disebut sistem koordinat foto), dan 4 titik yang berbeda antara titik tengah kamera dan sistem koordinat foto (seperti : fokus), dan juga diketahui titik tengah foto (*Fischler* dan *Bolles*, 1981).

Dalam proses perhitungannya, dilakukan terlebih dahulu mencari 3D dari posisi kamera yang relatif untuk sistem koordinat objek. Dengan memberikan notasi sebelumnya, agar mempermudah pada saat proses perhitungan, yaitu sebagai berikut (*Fischler* dan *Bolles*, 1981) :

1. Memberikan 4 titik koordinat foto dengan label (P_i), dan 4 titik koordinat objek dengan label (Q_i) (pada rumus 2.20).
2. Dengan asumsi sistem koordinat adalah 2D sehingga mempunyai koordinat *principal point* atau titik tengak foto (PPI).

3. Dengan asumsi lain bahwa sistem koordinat objek mempunyai nilai Z = 0 pada sistem koordinat referensi. Dengan teknik standart yang tersedia untuk merubah dari sistem koordinat foto menjadi sistem koordinat objek di permukaan tanah.
4. Koordinat *homogeneous* dapat diasumsikan.
5. Mewakili simbol piramid yang megubah urutan dari struktur tersebut.

Sehingga dapat diperoleh solusi dari permasalahan P4P yaitu dengan proses perhitungan yang diawali dengan menghitung matrik T dengan dimensi 3x3, yang dipetakan dari koordinat objek dan koordinat foto. Dengan susunan matrik T, sebagai berikut (*Fischler* dan *Bolles*, 1981) :

$$[P_i] = [T] * [Q_i] \tag{2.19}$$

dimana,

$$[P_i] = [k_i * x_i, k_i * y_i, k_i]^T$$
$$[Q_i] = [X_i, Y_i, 1]^T \tag{2.20}$$

dimana :

P_i : label untuk koordinat foto

Q_i : label untuk koordinat objek

k_i : nilai koefisien

x_i, y_i : koordinat piksel

X_i, Y_i : koordinat objek

Dengan bentuk ideal garis pada objek, dengan koordinat $[0,0,1]^T$ dengan memetakan kedalam garis yang dihilangkan pada foto [VLI] dengan transformasi :

$$[VLI] = [inv[T]]^T * [0,0,1]^T \qquad (2.21)$$

Kemudian menentukan jarak *DI* dari posisi sebenarnya *PPI* untuk menghilangkan garis $[VLI] = [a_1,a_2,a_3]^T$:

$$DI = \left| \frac{a_3}{\sqrt{[(a_1)^2 + (a_2)^2]}} \right| \qquad (2.22)$$

Pada tahap selanjutnya dilakukan perhitungan untuk nilai sudut dua bidang (*tilt*) dengan sudut θ antara garis foto dan objek.

$$\theta = \arctan\left(\frac{f}{DI}\right) \qquad (2.23)$$

dimana :

f : panjang fokus

DI : nilai jarak

Dengan bentuk ideal garis pada foto, dengan koordinat $[0,0,1]^T$ dengan memetakan kedalam garis yang dihilangkan pada objek [VLO] dengan transformasi :

$$[VLO] = [T]^T * [0,0,1]^T \qquad (2.24)$$

Sehingga dapat ditentukan lokasi dari titik [PPO] pada objek ([PPO] yang terletak pada sumbu *optical* dari titik tengah lensa, yaitu :

$$[PPO] = [inv[T]]^T * [0,0,1]^T \qquad (2.25)$$

Dari persamaan (2.25) dapat digunakan untuk menghitung jarak *DO* dari [PPO] $=[c_1,c_2,c_3]^T$ untuk menghilangkan garis [VLO] = $[b_1,b_2,b_3]^T$ pada objek, yaitu :

$$DO = \left| \frac{b_1 * c_1 + b_2 * c_2 + b_3 * c_3}{c_3 * \sqrt{[(b_1)^2 + (b_2)^2]}} \right| \qquad (2.26)$$

Kemudian menghitung bentuk datar dari nilai sudut $ sebagai sudut antara bentuk normal [VLO] = $[b_1,b_2,b_3]^T$ dan sumbu X pada objek, yaitu :

$$\$ = \arctan\left(\frac{-b_2}{b_1}\right) \qquad (2.27)$$

Menentukan X_{SGN} dan Y_{SGN}, yaitu jika garis (sejajar dengan sumbu X pada objek) melalui [PPO] yang bersilangan dengan [VLO] menuju disebelah kanan dari [PPO], menjadi $X_{SGN} = 1$, selain itu $X_{SGN} = -1$. Jadi :

$$jika \quad \frac{b_1 * c_1 + b_2 * c_2 + b_3 * c_3}{b_1 * c_3} < 0 \qquad (2.28)$$

Maka nilai $X_{SGN} = 1$, selain itu $X_{SGN} = -1$.

Dengan cara yang sama untuk :

$$jika \quad \frac{b_1 * c_1 + b_2 * c_2 + b_3 * c_3}{b_2 * c_3} < 0 \qquad (2.29)$$

Maka nilai $X_{SGN} = 1$, selain itu $X_{SGN} = -1$.

Penyelesaian untuk koodinat lokasi dari CP atau koordinat kamera pada sistem koordinat objek diperoleh dengan persamaan (2.30).

$$
\begin{aligned}
DCP &= DO * \sin(\theta) \\
XCP &= XSGN * abs[DCP * \sin(\theta) * \cos(\$)] + \frac{c_1}{c_3} \\
YCP &= YSGN * abs[DCP * \sin(\theta) * \sin(\$)] + \frac{c_2}{c_3} \\
ZCP &= DCP * \cos(\theta)
\end{aligned}
\qquad (2.30)
$$

dimana :

DCP : jarak koordinat titik tengah kamera

XCP, YCP, ZCP : koordinat posisi kamera

Dengan catatan jika [VLI] yang ditentukan pada (*b*), maka mempunyai koordinat [0,0,*k*], kemudian foto dan objek posisinya sejajar ($\theta = 0$). Kemudian melanjutkan seperti prosedur diatas, maka akan diperoleh penyelesaian untuk informasi yang diinginkan dengan menggunakan rumus segitiga dan geometri *euclidean*.

2.4 Metode *Rampal*

Prinsip dari metode yang dikembangkan oleh (*Rampal*, 1979) yaitu dengan merubah persamaan non linier menjadi persamaan linier dari kondisi kesegarisan yang dilakukan dengan cara proses iterasi dengan asumsi adanya nilai pendekatan. Maka metode *Rampal* hanya dapat menghasilkan parameter sudut rotasi yaitu *omega*

(ω), *phi* (φ), dan *kappa* (κ) (*Rampal*, 1979). *Rampal* (1979) menyatakakan bahwa tanpa adanya nilai pendekatan parameter tetap akan dihasilkan koordinat kamera. Maka *Rampal* menerapkan metode *Church's* untuk mendapatkan nilai koordinat sebenarnya pada posisi yang konvergen dan kebenaranya, agar diperoleh solusi yang unik (*Rampal*, 1979).

Persamaan dasar dari metode *Rampal* yaitu menggunakan persamaan kesegarisan (persamaan 2.1), dari persamaan tersebut maka dilakukan proses linierisasi dengan asumsi $\omega = \varphi = \kappa = 0$ (persamaan 2.31) (*Rampal*, 1979).

$$x - x_0 = c\frac{(X - X_L)}{(Z - Z_L)}$$
$$y - y_0 = c\frac{(Y - Y_L)}{(Z - Z_L)} \qquad (2.31)$$

dimana :

x,y : koordinat foto

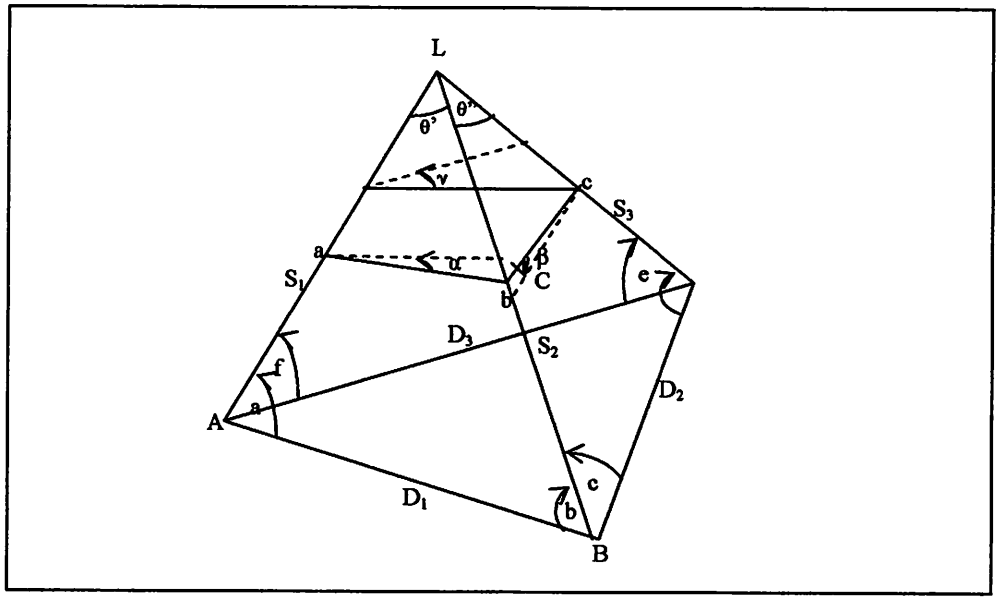
x_o,y_o : koordinat titik tengah foto (*principal point*)

X,Y,Z : koordinat objek

X_L,Y_L,Z_L : koordinat kamera.

Dari persamaan (2.31) maka akan diperoleh koordinat kamera (X_L,Y_L,Z_L). Secara normal proses iterasi akan dilakukan sebanyak 3 atau 4 kali iterasi untuk mendapatkan solusi yang unik dan konvergen (*Rampal*, 1979).

Pada proses perhitungannya (*Rampal*, 1979) menggunakan prinsip rumus segitiga dengan model piramid (lihat pada Gambar 2.11).



Gambar2.10 Geometri sudut foto

Dimana :

S_1,S_2,S_3 : jarak kaki (*leg*) antara koordinat kamera dan koordinat objek

D_1,D_2 dan D_3 : jarak koordinat objek

a,b,c,d,e,f : parameter sudut dalam geometri piramid sudut foto

α,β,γ : sudut kesendengan dari foto

θ', θ'' : sudut yang dibentuk antara koordinat kamera dan koordinat objek

Dari Gambar (2.10) dapat terbentuk bidang segitiga, sehingga dapat ditulis kondisi persamaan (*Rampal*, 1979):

$$
\begin{aligned}
S_1 &= \frac{D_1}{\sin\theta'}\sin(b'+\alpha) = \frac{D_3}{\sin\theta'''}\sin(e'-\gamma) \\
S_2 &= \frac{D_1}{\sin\theta'}\sin(a'-\alpha) = \frac{D_2}{\sin\theta''}\sin(d'+\beta) \\
S_3 &= \frac{D_2}{\sin\theta''}\sin(c'-\beta) = \frac{D_3}{\sin\theta'''}\sin(f'+\gamma)
\end{aligned}
\qquad (2.32)
$$

dimana α, β, dan γ adalah nilai kecil. Persamaan (2.31) merupakan persamaan non linier, maka harus dirubah menjadi persamaan linier menjadi (*Rampal*, 1979):

$$\begin{aligned} a_1\alpha + b_1\beta &= C_1 \\ a_2\alpha + b_2\gamma &= C_2 \\ a_3\beta + b_3\gamma &= C_3 \end{aligned} \qquad (2.33)$$

dimana diberikan persamaan :

$$\alpha = \frac{a_3 b_2 C_1 + b_3 C_2 b_1 - b_1 b_2 C_3}{a_1 a_3 b_2 + a_2 b_1 b_3} \qquad (2.34)$$

Diketahui bahwa nilai koefisien yaitu :

$$\begin{aligned} a_1 &= D_1 \cos a' \sin \theta'' \\ b_1 &= D_2 \cos d' \sin \theta' \\ C_1 &= D_1 \sin a' \sin \theta'' - D_2 \sin d' \sin \theta' \\ a_2 &= D_1 \cos b' \sin \theta'' \\ b_2 &= D_3 \cos e' \sin \theta' \\ C_2 &= D_3 \sin c' \theta' - D_1 \sin b' \sin \theta'' \\ a_3 &= D_2 \cos c' \sin \theta''' \\ b_3 &= D_3 \cos f' \sin \theta'' \\ C_3 &= D_2 \sin c' \theta''' - D_3 \sin f' \sin \theta'' \end{aligned} \qquad (2.35)$$

dimana :

D_1,D_2 dan D_3 : jarak koordinat objek

a',b',c',d',e',f': parameter sudut dalam geometri piramid sudut foto

θ', θ'' , θ''' : sudut yang dibentuk antara koordinat kamera dan koordinat objek

Dengan diketahui persamaan untuk θ' yaitu :

$$\cos\theta' = \frac{s_1^2 + s_2^2 - d^2}{2 s_1 s_2} \qquad (2.36)$$

dimana,

$$s_1^2 = (x_1 - x_0)^2 + (y_1 - y_0)^2 + C^2$$
$$s_2^2 = (x_2 - x_0)^2 + (y_2 - y_0)^2 + C^2 \qquad (2.37)$$
$$d^2 = (x_2 - x_1)^2 + (y_2 - y_1)^2$$

dimana:

s_i : jarak koordinat foto dan nilai kuadrat dari koefisien C

d : jarak koordinat foto

Dengan cara yang sama, maka diperoleh :

$$\cos a' = \frac{s_1^2 + d^2 - s_2^2}{2 s_1 d}$$
$$\cos b' = \frac{s_2^2 + d^2 - s_1^2}{2 s 2 d} \qquad (2.38)$$

Dari persamaan (2.32) dan (2.35) diperoleh persamaan baru untuk menentukan α, β, dan γ yang mungkin dengan menggunakan data koordinat foto dan objek. Sehingga hasil sudut sebelumnya menjadi mungkin untuk menyelesaikan S_1, S_2 dan S_3 dengan persamaan sederhana (persamaan 2.39).

$$\frac{S_1}{\sin(a' - \alpha)} = \frac{D_1}{\sin \theta'} \qquad (2.39)$$

Dengan menggunakan cara yang sama, dapat ditentukan nilai S_2 dan S_3. Jika S_1 dan S_2 maka persamaan (2.38) dapat dirubah kedalam bentuk persamaan linier.

$$\frac{1 + D_3 + D_5}{D_6} = \frac{\Delta X \Delta X_2 + \Delta Y \Delta Y_2 + \Delta Z \Delta Z_2 + D S_2}{\Delta X \Delta X_1 + \Delta Y \Delta Y_1 + \Delta Z \Delta Z_1 + D S_1} = \cot\left(\frac{a' - \alpha}{2}\right) \cot\left(\frac{b' + \alpha}{2}\right) = k \qquad (2.40)$$

dengan menggunakan persamaan (2.40) maka :

$$\Delta X(X_2 - X_0) + \Delta Y(Y_2 - Y_0) + \Delta Z(Z_2 - Z_0) = D(kS_1 - S_2) \qquad (2.41)$$

dimana,

$$X_2 = X_1 + \Delta X\,; Y_2 = Y_1 + \Delta Y\,; Z_2 = Z_1 + \Delta Z \qquad (2.42)$$

Sehingga dapat diperleh persamaan baru untuk memperoleh koordinat posisi kamera, yaitu :

$$\Delta XX_0 + \Delta YY_0 + \Delta ZZ_0 = \Delta XX_1 + \Delta YY_1 + \Delta ZZ_1 - \frac{D_1(D_1 + D_2 - kS_1)}{k-1} \qquad (2.43)$$

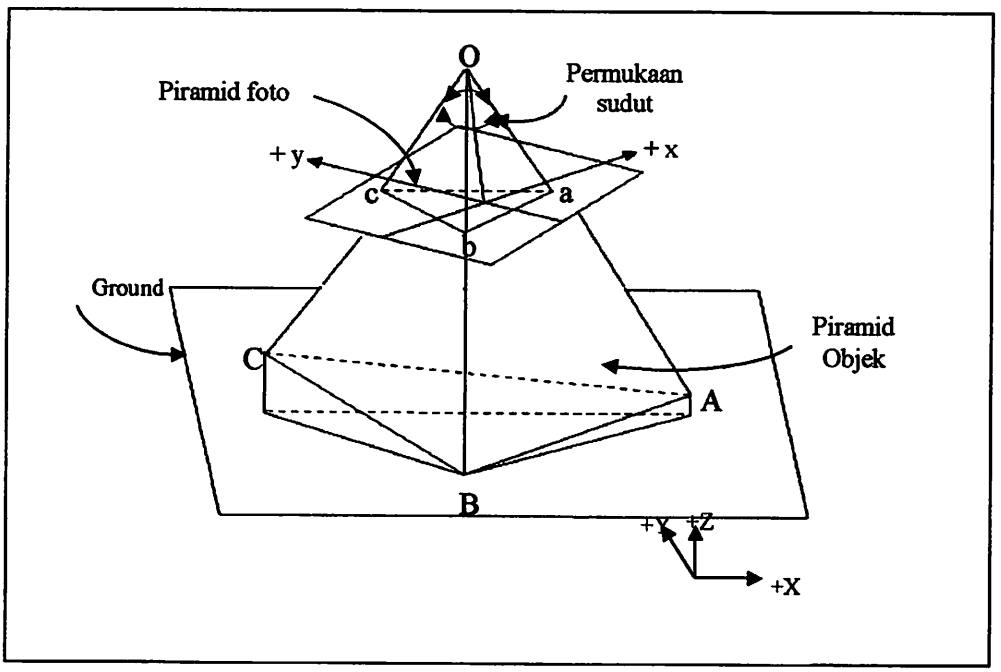
persamaan (2.42) merupakan persamaan linier untuk *direct (closed) form* dari koordinat posisi kamera. Dapat diambil kesimpulan bahwa metode R*rampal* :

1. Tidak memerlukan nilai pendekatan parameter.
2. Tidak memerlukan nilai pendekatan sudut rotasi.
3. Dapat digunakan tipe foto untuk di udara dan di darat .
4. Hanya digunakan satu foto dengan informasi koordinat foto dan koordinat objek.

2.5 Metode *Church*

Merupakan salah satu metode *closed form solution* yang berkaitan dengan *Tilted Photography* yaitu menentukan *tilt*, *swing* dan *azimut* dari foto sendeng. Metode ini telah dikembangkan oleh *Profesor Earl Church* dari Universitas *Syracuse*, dan dikenal dengan metode *Church*. Dengan metode *Church* parameter posisi kamera

yang terkoreksi dan susunan matriks rotasi dapat diperoleh jika nilai awal posisi kamera (X_L, Y_L, Z_L) diketahui, dengan prinsip dasar bahwa nilai sudut untuk dua piramid mempunyai nilai yang sama yang dibentuk oleh piramid bidang foto dan bidang objek yang berpusat pada posisi kamera (*Church*, 1980 dan *Wolf*, 1983).



Gambar 2.11 Hubungan geometri untuk metode Church

Dari Gambar (2.11) diperoleh prinsip dasar dari metode *Church* bahwa permukaan sudut untuk dua piramid yang definisikan oleh titik OABC dan Oabc mempunyai nilai sudut yang sama. Sehingga dapat ditulis :

$$\begin{aligned} \angle AOB &= \angle aOb \\ \angle BOC &= \angle bOc \\ \angle AOC &= \angle aOc \end{aligned} \qquad (2.44)$$

Sudut diatas digunakan untuk mendefiniskan :

$$\begin{aligned} K_1 &= \cos \angle_{AOB} \qquad k_1 = \cos \angle_{aOb} \\ K_2 &= \cos \angle_{BOC} \qquad k_2 = \cos \angle_{bOc} \\ K_3 &= \cos \angle_{AOC} \qquad k_3 = \cos \angle_{aOc} \end{aligned} \tag{2.45}$$

Kemudian persamaan kondisi diatas digunakan didalam metode *Church* sehingga dapat didefinisikan dengan mengikuti :

$$K_1 = k_1 \qquad K_2 = k_2 \qquad K_3 = k_3 \tag{2.46}$$

dimana :

K_i : nilai konstanta yang diperoleh dari hubungan sudut antara dua koordinat foto

k_i : nilai konstanta yang diperoleh dari hubungan sudut antara dua koordinat objek

2.5.1. Penentuan Koordinat Kamera

Metode *Church* memberikan beberapa langkah untuk mendapatkan enam parameter *unknown* dari orientasi luar (*Church*, 1980). Jika diketahui data awal yaitu berupa tiga titik kontrol objek (X_i,Y_i,dan Z_i) dan fokus kamera (f). Kemudian dibuat asumsi bahwa koordinat foto (x_i dan y_i) dan nilai koordinat z_i = -f. Hal yang pertama dilakukan dalam proses perhitungan adalah menghitung nilai jarak dari titik tengah kamera (*perspective center*) (d_i) menuju ke titik foto (jarak Oa, Ob dan Oc).

$$di = \sqrt{x_i^2 + y_i^2 + z_i^2} \tag{2.47}$$

dimana, i dari 1 sampai 3.

dengan diketahui nilai jarak (d_i), maka dapat dihitung tiga arah *cosinus* jarak (Oa,Ob,Oc) pada koordinat foto dengan tanda seperti *l*,*m* dan *n*.

$$l_i = \frac{x_i}{d_i} \qquad m_i = \frac{y_i}{d_i} \qquad n_i = \frac{z_i}{d_i} \tag{2.48}$$

Nilai parameter pada persamaan (2.48) digunakan untuk menghitung sudut *cosinus* (*k*) antara dua garis didalam piramid (koordinat foto) :

$$k_i = l_i l_j + m_i m_j + n_i n_j \tag{2.49}$$

dimana, j ϵ (2,3,1). Tahap selanjutnya yaitu menghitung nilai jarak (D_i) dari titik tengah kamera menuju titik kontrol objek (jarak OA,OB dan OC).

$$D_i = \sqrt{(X_i - X_L)^2 + (Y_i - Y_L)^2 + (Z_i - Z_L)^2} \tag{2.50}$$

Dengan menggunakan parameter D_i dapat dihitung tiga arah *cosinus* jarak dengan tanda *L,M* dan *N*.

$$L_i = \frac{X_i - X_L}{D_i}; \qquad M_i = \frac{Y_i - Y_L}{D_i}; \qquad N_i = \frac{Z_i - Z_L}{D_i}; \tag{2.51}$$

Hasil dari persamaan (2.51) yang digunakan untuk menghitung nilai sudut *cosinus* (*K*) antara dua garis terhadap objek yaitu :

$$K_i = L_i L_j + M_i M_j + N_i N_j \tag{2.52}$$

Nilai untuk K_i nilainya akan sama dengan masing-masing nilai untuk *ki* yang telah dihitung pada persamaan (2.49). Jika nilai perkiraan posisi kamera sama maka nilainya benar. Jika nilainya tidak sama maka koordinat kamera perlu disesuaikan.

Dari persamaan (2.50) dan (2.52) dapat ditentukan nilai parameter pelengkap pada persamaan (2.53) untuk menentukan nilai koefisien pada persamaan (2.54).

$$I_i = \frac{k_i}{D_i} - \frac{1}{D_j}; \qquad J_i = \frac{k_i}{D_j} - \frac{1}{D_i}; \tag{2.53}$$

$$\begin{aligned} A_i &= L_i I_j + L_j J_i \\ B_i &= M_i I_j + M_j J_i \\ C_i &= N_i I_j + N_j J_i \end{aligned} \tag{2.54}$$

dimana :

A,B,C : nilai koefisien parameter koreksi

L,M,N : nilai koefisien arah *cosinus* jarak

dengan menentukan nilai selisih dari sudut *cosinus* (k) koordinat foto dan sudut *cosinus* (K) koordinat objek, maka dapat diperoleh persamaan (2.56).

$$\Delta K_i = k_i - K_i \tag{2.55}$$

Hasil dari persamaan (2.54) digunakan untuk menghitung parameter untuk parameter perataan posisi kamera pada persamaan (2.58), dengan menentukan parameter pelengkap untuk persamaan perataan secara serentak dengan jumlah perkalian antara parameter E dan C pada persamaan (2.57).

$$\begin{aligned} E_i &= A_i B_j - A_j B_i \\ F_i &= A_i C_j - A_j C_i \\ G_i &= B_i C_j - B_j C_i \end{aligned} \tag{2.56}$$

$$\Delta = E_1C_3 + E_3C_2 + E_2C_1 \qquad (2.57)$$

Dari persamaan (2.56) dan (2.57), maka dapat dihitung nilai perataan posisi kamera,yaitu menjadi :

$$\begin{aligned} \Delta X &= \frac{G_2\Delta K_1 + G_3\Delta K_2 + G_1\Delta K_3}{\Delta} \\ \Delta Y &= \frac{F_2\Delta K_1 + F_3\Delta K_2 + F_1\Delta K_3}{(-\Delta)} \\ \Delta Z &= \frac{E_2\Delta K_1 + E_3\Delta K_2 + E_1\Delta K_3}{\Delta} \end{aligned} \qquad (2.58)$$

Sehingga diperoleh nilai parameter pendekatan awal koordinat kamera terkoreksi (*update*),yaitu :

$$\begin{aligned} X_L &= X_{L_{OLD}} + \Delta X \\ Y_L &= Y_{L_{OLD}} + \Delta Y \\ Z_L &= Z_{L_{OLD}} + \Delta Z \end{aligned} \qquad (2.59)$$

Jika dalam proses perhitungan hasil dari parameter koreksi dan residu masih bernilai besar ($\Delta X,\Delta Y,\Delta Z > 0.1$), maka dilakukan proses iterasi dengan menghitung kembali parameter pada persamaan (2.47) sampai dengan persamaan (2.59) hingga didapat nilai koreksi yang sekecil mungkin. Sehingga hasil akhir yang diperoleh adalah koordinat kamera terkoreksi (X_L,Y_L,Z_L).

2.5.2. Matriks Rotasi

Dalam menyusun matriks rotasi dapat diperoleh dengan menggunakan hasil parameter pada persamaan (2.48) dari data koordinat foto dan menjadi persamaan (2.60).

$$\begin{aligned} e_i &= l_i m_j - l_j m_i \\ f_i &= l_i n_j - l_j n_i \\ g_i &= m_i n_j - m_j n_i \end{aligned} \qquad (2.60)$$

Sehingga dari persamaan (2.60) dapat diperoleh jumlah perkalian dari parameter e dan n, yaitu :

$$\Delta = e_1 n_3 + e_2 n_1 + e_3 n_2 \qquad (2.61)$$

Dengan menggabungkan nilai parameter pada persamaan (2.51), (2.60) dan (2.61), maka diperoleh unsur parameter rotasi pada persamaan (2.62).

$$u_1 = \frac{L_1 g_2 + L_2 g_3 + L_3 g_1}{\Delta}$$
$$v_1 = \frac{L_1 f_2 + L_2 f_3 + L_3 f_1}{(-\Delta)}$$
$$w_1 = \frac{L_1 e_2 + L_2 e_3 + L_3 e_1}{\Delta}$$

$$u_2 = \frac{M_1 g_2 + M_2 g_3 + M_3 g_1}{\Delta}$$
$$v_2 = \frac{M_1 f_2 + M_2 f_3 + M_3 f_1}{(-\Delta)}$$
$$w_2 = \frac{M_1 e_2 + M_2 e_3 + M_3 e_1}{\Delta}$$

$$u_3 = \frac{N_1 g_2 + N_2 g_3 + N_3 g_1}{\Delta}$$
$$v_3 = \frac{N_1 f_2 + N_2 f_3 + N_3 f_1}{(-\Delta)} \qquad (2.62)$$
$$w_3 = \frac{N_1 e_2 + N_2 e_3 + N_3 e_1}{\Delta}$$

Dari unsur persamaan (2.62) dapat disusun matriks rotasi yaitu :

$$R = \begin{bmatrix} u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{bmatrix} \qquad (2.63)$$

dimana :

u_i, v_i, w_i : unsur parameter rotasi

2.6 Metode *Zeng*

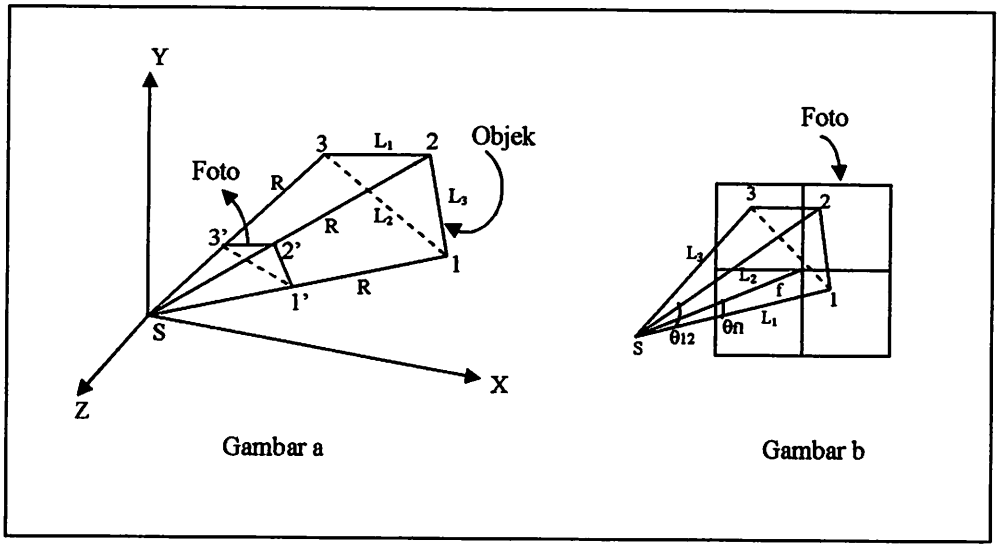
Pada permasalahan sebelum-sebelumnya, telah dijelaskan bahwa untuk memperoleh enam parameter orientasi luar diperlukan nilai pendekatan dengan menggunakan prinsip *closed form solution. Zeng* dan *Wang* (1992) melakukan penelitian dengan menggunakan metode yang telah dijelaskan dan diuji oleh (*Fischler* dan *Bolles*, 1980). Metode itu dikenal dengan metode permasalahan penentuan lokasi atau *Location determination Problem* (LDP) untuk analisa foto sehingga diperoleh posisi koordinat objek dan matriks rotasi dengan menggunakan prinsip perkalian murni (*Fischler* dan *Bolles*, 1980; dan *Zeng* dan *Wang*, 1992).

Metode yang dikembangkan oleh (*Zeng* dan *Wang*, 1992) mencakup tiga tahapan inti yaitu :

1. Penyelesaian untuk memperoleh posisi koordinat kamera (X_L,Y_L,Z_L).
2. Penyelesaian untuk memperoleh susunan matriks rotasi (R), berdasarkan atas algoritma *Pope-Hinsken* yang telah diuji oleh (*Hinsken*, 1988).
3. Mendiskusikan penyelesaian untuk memperoleh parameter reseksi dengan menggunakan prinsip kurva kritis (*danger cylinder*).

2.6.1 Penentuan Koordinat Kamera

Untuk memperoleh enam parameter orientasi luar maka (*Zeng* dan *Wang*, 1992) terlebih dahulu menentukan posisi koordinat kamera dengan diketahui koordinat foto dan koordinat objek dari satu foto (Gambar 2.12).



Gambar 2.12 Hubungan antara gambar a
(pyramid koordinat foto) dan gambar b(koordinat objek)

Dari Gambar 2.12 diatas dapat diketahui bahwa S merupakan posisi koordinat kamera, 1,2, dan 3 merupakan koordinat titik kontrol objek, 1',2', dan 3' merupakan koordinat titik foto (dan tidak segaris), L_i dan R_i merupakan jarak antara jarak posisi koordinat objek dan koordinat titik foto pada piramid (*Zeng* dan *Wang*, 1992). Dengan catatan urutan posisi titik 1',2', dan 3' searah dengan arah jarum jam. Dan sistem koordinat foto sebenarnya di S yang digunakan pada perhitungan. Misalnya x_i,y_i, adalah koordinat foto, f merupakan jarak titik tengah foto dan arahnya berlawanan dengan arah sumbu z sehingga z = -f pada sistem koordinat foto, dan X_i, Y_i, dan Z_i merupakan koordinat objek pada sistem koordinat foto (*Zeng* dan *Wang*, 1992). Dari Gambar 2.12 dapat disusun suatu formula, sebagai berikut (*Zeng* dan *Wang*, 1992) :

$$
\begin{aligned}
x_1 &= -fx\frac{X_1}{Z_1}; \qquad y_1 = -fx\frac{Y_1}{Z_1} \\
x_2 &= -fx\frac{X_2}{Z_2}; \qquad y_1 = -fx\frac{Y_2}{Z_2} \\
x_3 &= -fx\frac{X_3}{Z_3}; \qquad y_1 = -fx\frac{Y_3}{Z_3}
\end{aligned} \tag{2.64}
$$

$$
\begin{aligned}
(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2 &= L_3^2 \\
(X_2 - X_3)^2 + (Y_2 - Y_3)^2 + (Z_2 - Z_3)^2 &= L_1^2 \\
(X_3 - X_1)^2 + (Y_3 - Y_1)^2 + (Z_3 - Z_1)^2 &= L_2^2
\end{aligned} \tag{2.65}
$$

dimana :

xi,yi : koordinat foto

Xi,Yi,Zi : koordinat objek

L_1,L_2,L_3 : nilai jarak koordinat objek

f : panjang fokus

Dimana L_1,L_2,L_3 dihitung dengan menggunakan koordinat titik objek pada sistem koordinat foto(*Zeng* dan *Wang*, 1992).

$$
\begin{aligned}
AxZ_1^2 - DxZ_1xZ_2 + BxZ_2^2 &= L_3^2 \\
BxZ_2^2 - ExZ_2xZ_3 + CxZ_3^2 &= L_1^2 \\
CxZ_3^2 - FxZ_3xZ_1 + AxZ_1^2 &= L_2^2
\end{aligned} \tag{2.66}
$$

dimana,

$$
\begin{aligned}
A &= (x_1^2 + y_1^2 + f^2)/f^2 \\
B &= (x_2^2 + y_2^2 + f^2)/f^2 \\
C &= (x_3^2 + y_3^2 + f^2)/f^2 \\
D &= (2x_1x_2 + 2y_1y_2 + 2f^2)/f^2 \\
E &= (2x_2x_3 + 2y_2y_3 + 2f^2)/f^2 \\
F &= (2x_3x_1 + 2y_3y_1 + 2f^2)/f^2
\end{aligned}
\qquad (2.67)
$$

dimana :

A,B,C,D,E : nilai konstanta yang dihitung dari koordinat foto dan fokus

xi,yi : koordinat foto

f : panjang fokus

Maka dapat disusun persamaan baru untuk mencari nilai konstanta dari D_1,E_1, dan F_1 dari persamaan (2.67) menjadi (*Zeng* dan *Wang*, 1992):

$$
\begin{aligned}
D_1 &= D/\sqrt{AxB} \\
E_1 &= E/\sqrt{BxC} \\
F_1 &= F/\sqrt{CxA}
\end{aligned}
\qquad (2.68)
$$

Setelah ditemukan persamaan (2.65) dengan asumsi itu maka diperoleh hubungan antara jarak posisi koordinat objek dan koordinat titik foto pada piramid, dapat ditulis suatu persamaan (*Zeng* dan *Wang*, 1992) :

$$
\begin{aligned}
R_1^2 - D_1xR_1xR_2 + R_2^2 &= L_3^2 \\
R_2^2 - E_1xR_2xR_3 + R_3^2 &= L_1^2 \\
R_3^2 - F_1xR_3xR_1 + R_1^2 &= L_2^2
\end{aligned}
\qquad (2.69)
$$

Nilai konstanta dari perbandingan jarak pada koordinat titik objek yaitu K_{12} dan K_{32}, dapat ditulis sebagai berikut (*Zeng* dan *Wang*, 1992):

$$K_{12} = \left(\frac{L_1}{L_2}\right)^2 \qquad K_{32} = \left(\frac{L_3}{L_2}\right)^2 \tag{2.70}$$

Dari persamaan (2.68) dan (2.70) diperoleh suatu bentuk persamaan baru yang mengandung operasi akar empat, menjadi (*Zeng* dan *Wang*, 1992):

$$N_1 n^4 + N_2 n^3 + N_3 n^2 + N_4 n^1 + N_5 n^0 = 0 \tag{2.71}$$

dimana,

$$\begin{aligned}
N_1 &= (1-K_{12})^2 + K_{32}^2 - 2x(1-K_{12})xK_{32} - K_{32}xE_1^2 + 4xK_{32}x(1-K_{12}) \\
N_2 &= K_{32}xF_1x(-2-2xK_{32}+4xK_{12}+E_1^2) + K_{12}x(E_1xD_1 + 2xF_1 - 2xK_{12}xF_1) \\
&\quad + E_1xD_1x(K_{32}-1) \\
N_3 &= K_{32}x(K_{32}xF_1^2 - F_1xD_1xE_1 - 4xK_{12} + 2xK_{32} - E_1^2 - 2xK_{12}xF_1^2) + K_{12}x \\
&\quad (K_{12}xF_1^2 + 2xK_{12} - F_1xE_1xD_1 - D_1^2) + D_1^2 - 2 + E_1^2 \\
N_4 &= K_{32}x(2xF_1 - 2xK_{32}xF_1 + E_1xD_1 + 4xK_{12}xF_1) + K_{12}x(E_1xD_1 - 2xK_{12}x \\
&\quad F_1 + F_1xD_1^2 - 2xF_1) - D_1xE_1 \\
N_5 &= K_{32}x(K_{32} - 2 - 2xK_{12}) + K_{12}x(K_{12} - D_1^2 + 2) + 1
\end{aligned} \tag{2.72}$$

dimana :

N_i : nilai koefisien dari bentuk *biqudratic polynomial*

D_1, E_1, F_1, : nilai konstanta

K_i : nilai koefisien dari perbandingan jarak ketiga titik kontrol

Setelah diperoleh nilai N_i maka persamaan (2.71) menjadi (*Zeng* dan *Wang*, 1992):

$$M_1 n^4 + M_2 n^3 + M_3 n^2 + M_4 n^1 + M_5 n^0 = 0 \qquad (2.73)$$

M_i merupakan nilai koefisien dari bentuk *biqudratic polynomial*, dengan asumsi nilai M_i diperoleh dari :

$$M_1 = \frac{N_1}{N_1}; \quad M_2 = \frac{N_2}{N_1}; \quad M_3 = \frac{N_3}{N_1}; \quad M_4 = \frac{N_4}{N_1}; \quad M_5 = \frac{N_5}{N_1}; \qquad (2.74)$$

Persamaan (2.74) merupakan tipe persamaan aljabar akar pangkat empat dan mempunyai 4 akar real positif dan 4 akar *imaginer*. Meskipun metode solusi ini dapat diketahui, tetapi harus dilakukan dengan cara yang tepat sehingga memperoleh solusi yang pasti. Sehingga dapat diperoleh hasil akar empat dari n_1, n_2, n_3, dan n_4 pada persamaan (2.73). Akan tetapi, hanya nilai real positif yang diperlukan dalam metode *zeng's* dan setiap syarat *unknowns* dari derajat kedua. Dengan memecahkan persamaan aljabar akar empat tersebut satu dari empat nilai real positif yang dapat digunakan untuk menetukan koordinat posisi kamera dapat ditentukan (*Zeng* dan *Wang*, 1992).

Dari tiap nilai positif *n*, dapat diketahui jarak antara koordinat posisi kamera dan koordinat objek (R_1,R_2, dan R_3) yaitu diperoleh dengan melakukan proses *intersection* (persilangan) titik dengan radius R_1, R_2, R_3 dan berpusat pada tiga titik kontrol (*Zeng* dan *Wang*, 1992). Persamaan untuk menghitung nilai R_1 dapat dituliskan sebagai berikut :

$$R_1 = p = \frac{L_2}{\sqrt{n^2 - F_1 xn + 1}} \tag{2.75}$$

$$R_3 = nxp \tag{2.76}$$

dimana :

F_1 : nilai konstanta

R_i : jarak antara koordinat posisi kamera dan koordinat objek

Dengan menggunakan persamaan (2.69) dapat dihitung nilai R_2 yang mempunyai dua solusi, yaitu (*Zeng* dan *Wang*, 1992):

$$R_2 = (D_1 xR_1 \pm \sqrt{D_1^2 xR_1^2 - 4x(R_1^2 - L_3^2)})/2 \tag{2.77}$$

$$R_2 = (E_1 xR_3 \pm \sqrt{E_1^2 xR_3^2 - 4x(R_3^2 - L_1^2)})/2 \tag{2.78}$$

dimana :

D_1, E_1 : nilai konstanta

R_i : jarak antara koordinat posisi kamera dan koordinat objek

Sehingga dapat disusun persamaan untuk hubungan antara jarak R_1, R_2, dan R_3 dan koordinat titik objek dan koordinat kamera, yaitu (*Zeng* dan *Wang*, 1992):

$$\begin{aligned}
(X_1 - X_L)^2 + (Y_1 - Y_L)^2 + (Z_1 - Z_L)^2 &= R_1^2 \\
(X_2 - X_L)^2 + (Y_2 - Y_L)^2 + (Z_2 - Z_L)^2 &= R_2^2 \\
(X_3 - X_L)^2 + (Y_3 - Y_L)^2 + (Z_3 - Z_L)^2 &= R_3^2
\end{aligned} \tag{2.79}$$

dimana :

X_i,Yi,Zi : koordinat titik kontrol

X_L,Y_L,Z_L : koordinat kamera

R_i : jarak antara koordinat posisi kamera dan koordinat objek

Maka persamaan (2.79) dapat dijabarkan menjadi (*Awange* dan *Grafarend*, 2004) :

$$\begin{aligned} R_1^2 - R_2^2 &= X_1^2 - X_2^2 + Y_1^2 - Y_2^2 + Z_1^2 - Z_2^2 + a \\ R_2^2 - R_3^2 &= X_2^2 - X_3^2 + Y_2^2 - Y_3^2 + Z_2^2 - Z_3^2 + b \end{aligned} \qquad (2.80)$$

dimana,

$$\begin{aligned} a &= 2X(X_2 - X_1) + 2Y(Y_2 - Y_1) + 2Z(Z_2 - Z_1) \\ b &= 2X(X_3 - X_2) + 2Y(Y_3 - Y_2) + 2Z(Z_3 - Z_2) \end{aligned} \qquad (2.81)$$

Notasi a dan b dimisalkan sebagai hubungan antara parameter yang dicari pada formula disisi kanan dan sisi kiri pada persamaan (2.80) setetah menghitung nilai koreksi arah vektor (T) dari solusi koordinat kamera yang telah ditentukan. arah vektornya (T) yang diperoleh dari hasil vektor ***12*** (selisih dari koordinat objek titik 1 dan 2), ***13***(selisih dari koordinat objek titik 1 dan 3) dan ***1S*** (selisih dari koordinat kamera dan koordinat objek) seperti persamaan (2.82).

$$T = 1S \bullet (12 * 13) \qquad (2.82)$$

Jika T mempunyai nilai positif (+) maka nilai dari solusi koordinat posisi kamera adalah benar dan jika T mempunyai nilai negatif (-) maka nilai dari solusi koordinat

posisi kamera adalah salah, dengan menggunakan minimal tiga koordinat foto dan koordinat objek serta diberikan titik koordinat keempat sebagai koreksi akhir dari selisih koreksi vektor. Solusi unik dari koordinat kamera akan diperoleh dengan melakukan proses perhitungan selisih koreksi vektor, dengan nilai selisih vektor paling kecil dengan arah sumbu x yang didefinisikan sebagai 1.

2.6.2 Matriks Rotasi

Didalam perhitungan fungsi trigonometri, terdapat permasalahan singular. Para ahli fotogrametri telah menyusun matrik rotasi dengan menggunakan parameter aljabar (*Thompson*, 1959) tetapi metode ini tidak membuktikan secara praktis (*Zeng* dan *Wang*, 1992).

(*Pope*, 1970) mengusulkan menggunakan empat parameter aljabar dengan notasi yaitu d,a,b dan c, yang digunakan untuk menyusun matrik rotasi (R). (*Hinsken*, 1988) menurunkan satu set formula yang kompleks untuk membuat parameter (*Pope*, 1970) yang dapat digunakan untuk perhitungan fotogrametri yang praktis. Secara detail algoritma tentang penyusunan matrik rotasi (R) dapat ditemukan pada (*Hinsken*, 1988) dan (*Zeng* dan *Wang*, 1992). Secara praktis matrik rotasi dapat disusun menjadi (*Mikhail, et al*) :

$$R=\begin{bmatrix} d^2+a^2-b^2-c^2 & 2(ab+cd) & 2(ac-bd) \\ 2(ab-cd) & d^2-a^2+b^2-c^2 & 2(bc+ad) \\ 2(ac+bd) & 2(bc-ad) & d^2-a^2-b^2+c^2 \end{bmatrix} \qquad (2.83)$$

dimana,

$$d^2 + a^2 - b^2 + c^2 = 1 \qquad (2.84)$$

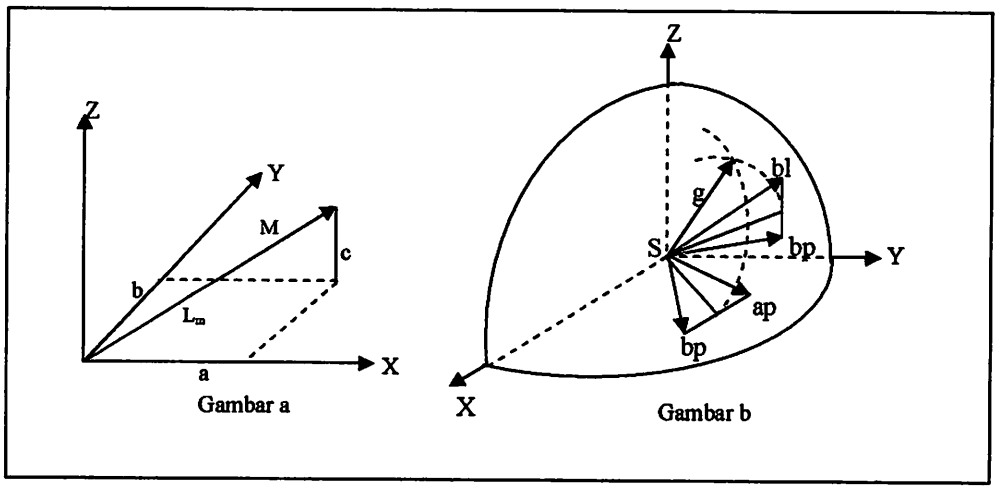
Hal itu telah dibuktikan secara praktis, algoritma *Pop-Hinsken* tidak bermasalah untuk bentuk singular dan menuju ke sistem yang konvergen (*Zeng* dan *Wang*, 1992). Untuk memberikan interpretasi fisik untuk parameter ini dapat diasumsikan menjadi (*Zeng* dan *Wang*, 1992):

$$M = [a \quad b \quad c]^T \qquad (2.85)$$

Dengan adanya matr*ik* parameter dan rotasi, maka penyususnan matrik menjadi (*Zeng* dan *Wang*, 1992):

$$RM = M \qquad (2.86)$$

Sekarang, setelah mentransformasikan M dengan R, maka M tetap tidak berubah sesuai dengan persamaan (2.86), dan dapat diilustrasikan seperti gambar dibawah ini (*Zeng* dan *Wang*, 1992):



Gambar 2.13 Hubungan antara gambar a (Geometri a,b,c) dan gambar b (penetuan sumbu rotasi pada koordinat objek dan foto)

$|M|$ = merupakan nilai magnitude dari M. Dengan memisalkan sudut antara vektor t dan R ditransformasikan terhadap t yang dinotasikan dengan simbol γ. Kemudian menjadi (*Zeng* dan *Wang*, 1992):

$$\frac{t \circ Rt}{|t||x||t|} = \cos\gamma \qquad (2.87)$$

Dengan mengembangkan dan memanipulasi persamaan (2.87) dapat diperoleh (*Zeng* dan *Wang*, 1992):

$$L_m = \sqrt{(1-\cos\gamma)/2} = \sqrt{1-d^2} \qquad (2.88)$$

Jadi, *sudut* rotasi γ dari sistem koordinat telah dinyatakan pada syarat-syarat dari parameter rotasi d pada persamaan (2.88). Sehingga akan diperoleh parameter d.

Tahapan selanjutnya adalah menentukan sumbu rotasi, dengan asumsi bahwa (*Zeng* dan *Wang*, 1992):

al,bl : merupakkan dua unit vektor titik kontrol dari koordinat posisi kamera.

ap,bp : merupakan dua unit vektor titik foto yang berhubungan dengan koordinat posisi kamera.

g(XYZ) : sumbu rotasi sistem koordinat dari objek ke foto.

Garis vektor dari ap dan al akan membentuk garis tegak lurus dan membagi dua bagian sudut antara ap dan al pada garis *sm*1*g*. Dengan menggunakan bp dan bl, dengan cara yang sama dapat dibentuk, sehingga terdapat perpotongan garis pada

sm2g yang memberikan sumbu rotasi *g* pada gambar (2.13) sehingga diperoleh bahwa (*Zeng* dan *Wang*, 1992):

$$ap = \begin{pmatrix} ap_x & ap_y & ap_z \end{pmatrix}^T \qquad bp = \begin{pmatrix} bp_x & bp_y & bp_z \end{pmatrix}^T \tag{2.89}$$

Dengan arah parameter sumbu rotasi *g* adalah (*Zeng* dan *Wang*, 1992):

$$\begin{aligned} p &= (ap_y - al_y)(bp_z - bl_z) - (ap_z - al_z)(bp_y - bl_y) \\ q &= (ap_z - al_z)(bp_x - bl_x) - (ap_x - al_x)(bp_z - bl_z) \\ r &= (ap_x - al_x)(bp_y - bl_y) - (ap_y - al_y)(bp_x - bl_x) \end{aligned} \tag{2.90}$$

Dari persamaan (2.89) dan (2.90), maka dapat diketahui sudut rotasi dari sistem koordinat objek ke foto yaitu γ, menjadi (*Zeng* dan *Wang*, 1992):

$$\cos\gamma = \frac{(gxap) \circ (gxal)}{|gxap| x |gxal|} \tag{2.91}$$

Dari persamaan (2.91) dapat diketahui nilai dari parameter *d*, yaitu (*Zeng* dan *Wang*, 1992) :

$$d = \sqrt{(1 + \cos\gamma)/2} \tag{2.92}$$

Oleh karena nilai arah dan komponen koordinat sumbu rotasi *g* sepadan, menjadi (*Zeng* dan *Wang*, 1992):

$$\begin{aligned} a &= p\sqrt{(1-d^2)/(p^2+q^2+r^2} \\ b &= q\sqrt{(1-d^2)/(p^2+q^2+r^2} \\ c &= r\sqrt{(1-d^2)/(p^2+q^2+r^2} \end{aligned} \tag{2.93}$$

dimana *a,b* dan *c* merupakan parameter matrik rotasi.

Untuk dapat diperoleh susunan matrik seperti pada persamaan (2.83) digunakan 3 koordinat titik kontrol objek, cukup untuk menemukan nilai dari *d,a,b,*dan *c*, sehingga koordinat titik kontrol objek yang keempat digunakan untuk menguji kecocokan hasil dari 2 solusi akhir dari koordinat posisi kamera. Jika terdapat kecocokan antara vektor koordinat foto dan koordinat objek dengan petunjuk arah pada koordinat x pada foto dan koordinat X pada objek didefinisikan sebagai satu, maka solusi itulah yang paling tepat dan unik dari dua solusi akhir yang ada (*Zeng* dan *Wang*, 1992).

BAB III

PELAKSANAAN PENELITIAN

3.1 Persiapan Penelitian

Untuk dapat tercapai tujuan penelitian yang diinginkan, hendaknya perlu dirancang suatu alur dalam pelaksaanaan penelitian. Mulai dari persiapan penelitian sampai dengan pengolahan data. Jadi sebelum melaksanakan proses penelitian, tahapan pertama yang harus dilakukan adalah menyiapkan segala unsur-unsur yang digunakan untuk mendukung kelancaran proses penelitian.

3.1.1 Lokasi Penelitian

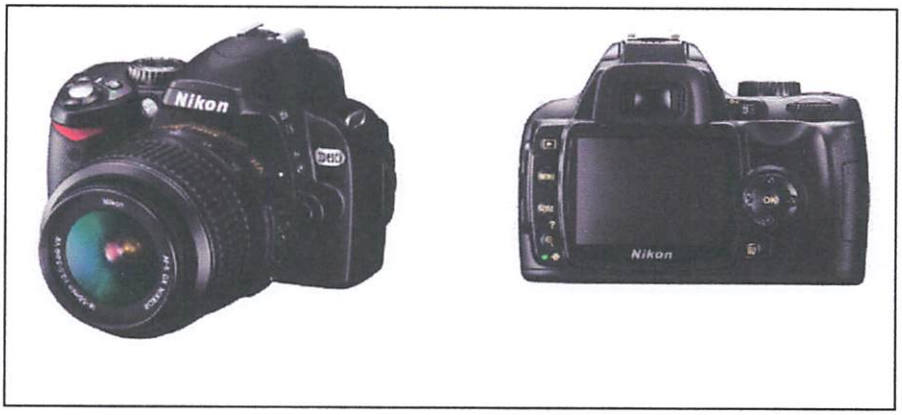
Penelitian ini dilaksanakan di dua tempat yaitu F*ly over* Arjosari dan Jembatan Rel Kereta Api Lawang.

3.1.2 Alat dan Bahan Penelitian

Adapun alat dan bahan yang digunakan untuk mendukung jalannya pelaksanaan penelitian ini meliputi perangkat keras (*hardware*) dan perangkat lunak (*software*), yaitu :

3.1.2.1 Perangkat Keras (*Hardware*)

1. Kamera SLR Nikon D 60.
    - Tipe Lensa : *Single-lens reflex digital camera.*
    - LCD *screen* : 2.5 *inch.*
    - *Image sensor* : 23.6 x 15.8 mm CCD *sensor.*

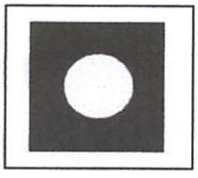


Gambar 3.1 Kamera SRL Nikon D 60 tampak depan dan belakang

Gambar 3.2 Aksaesoris dari Kamera
(Tipe Lensa, Baterai, Memory)

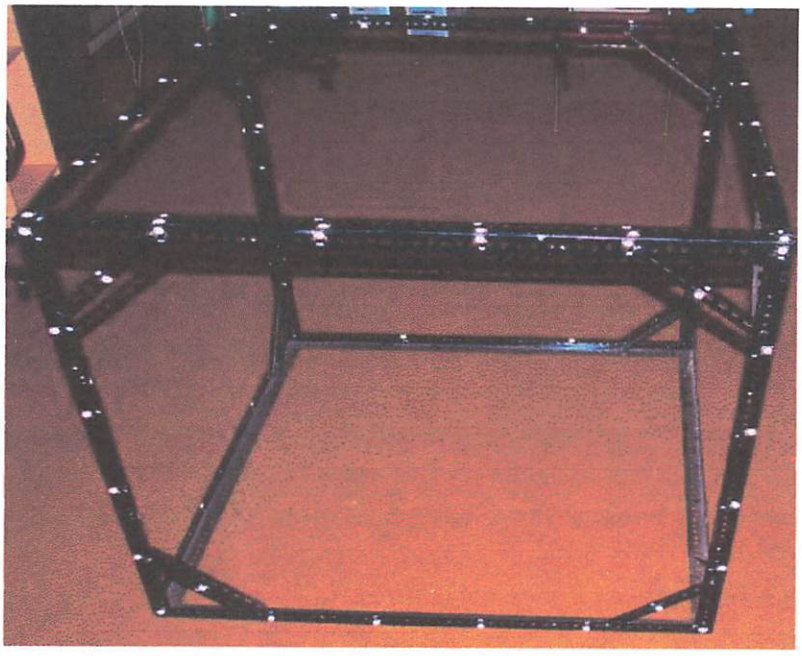
2. Laptop *Intel Pentium Core 2 Duo*.

3. Stiker Retro Target.



Gambar 3.3 Stiker Retro Target

4. Balok Kalibrasi.



Gambar 3.4. Balok kalibrasi

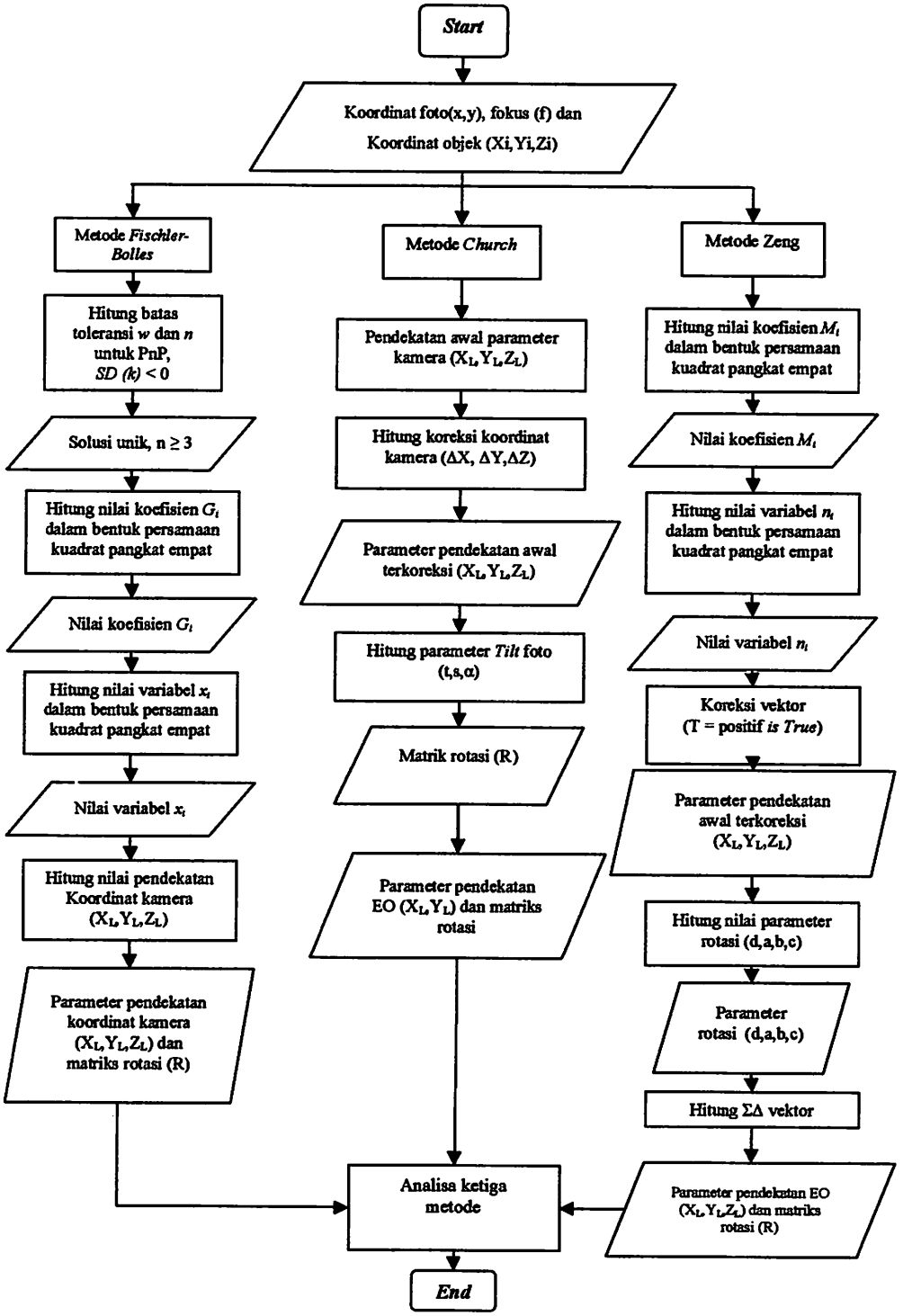
3.1.2.2 Perangkat Lunak (*Software*)

1. *Microsoft office 2003.*
2. *Microsoft visual studio 2010*

3.2 Pelaksanaan Penelitian

Alur pelaksanaan penelitian secara keseluruhan meliputi metode dan data yang dibutuhkan pada penelitian ini dilakukan melalui beberapa tahapan seperti yang ditunjukkan pada diagram alir penelitian pada gambar 3.5.

3.2.1 Diagram Alir Penelitian



Gambar 3.5 Diagram alir penelitian

Penjelasan Diagram Alir Penelitian :

Dibawah ini akan dijelaskan secara detail tentang pelaksanaan penelitian, yang meliputi :

1. Tahap Persiapan

   Tahapan pertama yang harus dilakukan adalah menyiapkan segala unsur-unsur yang dapat digunakan untuk mendukung kelancaran proses penelitian, baik berupa lokasi penelitian, materi, alat dan bahan penelitian, metode penelitian dan metode perhitungan yang akan digunakan.

2. Tahap Perhitungan Data

   Didalam tahap ini dilakukan proses perhitungan dengan menentukan data awal koordinat foto (x,y), fokus (f), dan koordinat objek (X_i,Y_i,Z_i), pada satu foto. Dari data awal yang telah diketahui diproses dengan menggunakan metode *closed form solution* yang meliputi metode *Fischler-Bolles*, metode *Church* dan metode *Zeng* untuk memperoleh nilai parameter pendekatan *exterior orientation* (EO) yang terdiri dari parameter (X_L,Y_L,Z_L) dan matriks rotasi (R).

3. Tahap Analisa

   Pada tahap ini merupakan tahap menganalisa parameter yang telah dihasilkan dari masing-masing metode dengan menggunakan data awal yang sama untuk ketiga metode, yang berkaitan dengan nilai parameter pendekatan *exterior orientation* (EO) yaitu parameter ($X_L,Y_L,Z_L,\omega,\varphi,\kappa$), setelah dilakukan proses perhitungan data.

3.3 Proses Perhitungan Data

Pada proses perhitungan data ini digunakan data awal yang sama pada ketiga metode *closed form solution* yang meliputi metode *Fischler-Bolles*, metode *Church* dan metode *Zeng*.

3.3.1 Metode *Fischler-Bolles*

Didalam metode *Fischler-Bolles* terdapat beberapa metode LDP (*Location Determination Problem*) atau umumnya dikenal dengan sebutan PnP (*Perspective-n-problem*), untuk mengetahui koordinat posisi kamera pada satu foto. Maka perlu ditentukan batasan toleransi (*Threshold t*).

3.3.1.1 Batas Toleransi (*Threshold t*)

Untuk mendapatkan ketetapan untuk *n* yang mempunyai solusi unik, dengan standart deviasi k, maka perlu ditentukan batas toleransi untuk *n* titik seperti ditampilkan pada tabel 3.1.

Tabel 3.1 Tabulasi nilai E(k) untuk hubungan nilai *w* dan *n*

w	n =1	2	3	4	5	6
0.9	1.1	1.2	1.4	1.5	1.7	1.9
0.8	1.3	1.6	2	2.4	3	3.8
0.7	1.4	2	2.9	4.2	5.9	8.5
0.6	1.7	2.8	4.6	7.7	13	21
0.5	2.0	4.0	8.0	16	32	64
0.4	2.5	6.3	16	39	98	244
0.3	3.3	11	37	123	412	
0.2	5	25	125	625		

dimana *w* merupakan nilai kemungkinan yang dipilih tiap data titik pada foto, *n* merupakan jumlah titik dan *E(k)* merupakan nilai yang diinginkan dari *k*. Untuk menghitung nilai *E(k)* dan *SD(k)* dapat dilihat pada persamaan (3.1) dan (3.2).

$$E(k) = \frac{1}{b} = w^{-n} \qquad (3.1)$$

$$SD(k) = \sqrt{1 - w^n} * (1/w^n) \qquad (3.2)$$

Dari tabel 3.1 diketahui bahwa *n* =1 dan 2 dengan persentasi kemungkinan (*w*) 90 % mempunyai jumlah solusi yang tidak terbatas, sedangkan n $\geq$ 3 dengan persentasi kemungkinan (*w*) 90 % mempunyai jumlah solusi yang unik. Selanjutnya setelah diperoleh batasan toleransi, maka dapat ditentukan metode LDP dengan menggunakan *n* titik.

3.3.1.2 Analisa Solusi untuk *Perspective-3-Point Problem* (P3P)

3.3.1.2.1. Penentuan Koordinat Kamera

Untuk menentukan koordinat posisi kamera dapat diperoleh melalui beberapa tahapan dalam bentuk algoritma yang disusun dalam bahasa pemrograman c# (didalam *console*),antara lain :

1) Menghitung jarak pada koordinat objek (R_{ab},R_{bc} dan R_{ac}) pada persamaan (2.10) yaitu :

```
Rab = Accord.Math.Distance.Euclidean(new double[3] { X1, Y1, Z1 },
new double[3] { X2, Y2, Z2 });
Rac = Accord.Math.Distance.Euclidean(new double[3] { X1, Y1, Z1 },
new double[3] { X3, Y3, Z3 });
Rbc = Accord.Math.Distance.Euclidean(new double[3] { X2, Y2, Z2 },
new double[3] { X3, Y3, Z3 });
```

2) Menghitung unit vektor pada koordinat foto, yaitu :

```
uvA = Tools.CalculateUnitVector(new double[3] { x1, y1, z1 });
uvB = Tools.CalculateUnitVector(new double[3] { x2, y2, z2 });
uvC = Tools.CalculateUnitVector(new double[3] { x3, y3, z3 });
uvD = Tools.CalculateUnitVector(new double[3] { x4, y4, z4 });
```

3) Menghitung nilai *cosinus* dari panjang kaki (*legs*) pada persamaan (2.12) yaitu :

```
cosTab = Tools.DotProduct(uvA, uvB);
cosTac = Tools.DotProduct(uvA, uvC);
cosTbc = Tools.DotProduct(uvB, uvC);
```

4) Menterjemahkan nilai persamaan dari bentuk *polynomial* pangkat empat pada persamaan (2.14), yaitu :

```
double Estimate_G4(double k1, double k2, double cbc)
 {
 return (Math.Pow(k1 * k2 - k1 - k2, 2) - (4 * k1 * k2 * cbc * cbc));
 }

double Estimate_G3(double k1, double k2, double cab, double cac, double
cbc)
{
double t1 = 4 * (k1 * k2 - k1 - k2) * k2 * (1 - k1) * cab;
double t2 = 4 * k1 * cbc * ((k1 * k2 + k2 - k1) * cac + 2 * k2 * cab *
cbc);

return (t1 + t2);
}

double Estimate_G2(double k1, double k2, double cab, double cac, double
cbc)
{
double t1 = Math.Pow(2 * k2 * (1 - k1) * cab, 2);
double t2 = 2 * (k1 * k2 + k1 - k2) * (k1 * k2 - k1 - k2);
double t3 = 4 * k1 * ((k1 - k2) * (cbc * cbc) + (1 - k2) * k1 * (cac *
cac) - 2 * k2 * (1 + k1) * cab * cac * cbc);

return (t1 + t2 + t3);
}

double Estimate_G1(double k1, double k2, double cab, double cac, double
cbc)
{
return (4 * (k1 * k2 + k1 - k2) * k2 * (1 - k1) * cab
                + 4 * k1 * ((k1 * k2 - k1 + k2) * cac * cbc + 2 * k1 *
k2 * cab * (cac * cac)));
}

double Estimate_G0(double k1, double k2, double cac)
{
return (Math.Pow(k1 * k2 + k1 - k2, 2) - 4 * (k1 * k1) * k2 * (cac *
cac));
```

```
}
```

5) Mencari nilai akar real positif dari bentuk persamaan *polynomial* pangkat empat, yaitu :

```
    double[] coefs = new double[5] { G0, G1, G2, G3, G4 };
    PolyLib.Polynomial p = new PolyLib.Polynomial(coefs);
    PolyLib.Complex[] roots = p.Roots();
       for (int i = 0; i < roots.Length; i++)
    Trace.WriteLine("real: " + roots[i].Re.ToString() + "  Imaginary: " +
roots[i].Im.ToString());
    double[] posRoots = Tools.FindPositiveRoots(roots);
    Tools.PrintVectorDebug(ref posRoots, "Roots in FischlerBolles");
```

6) Hasil dari solusi nilai dua akar yang didapat digunakan untuk menghitung panjang kaki (*legs*) dari CP pada tiga titik kontrol.

7) Setelah diperoleh dua solusi untuk nilai panjang kaki (*legs*), maka dilakukan proses perhitungan untuk penentuan koordinat posisi kamera (X_L,Y_L,Z_L).

3.3.1.2.2. Matriks Rotasi

Setelah diperoleh koordinat posisi kamera, maka matriks rotasi dapat disusun dengan tahapan sebagai berikut :

1) Menghitung panjang kaki (*legs*) koordinat foto di sistem kamera, yaitu :

```
//Length of Op1 Op2 Op3 camera system, unit in mm then convert to meter
double Op1 = Math.Sqrt(x1 * x1 + y1 * y1 + z1 * z1); double Op1m = Op1 /
1000.0;
double Op2 = Math.Sqrt(x2 * x2 + y2 * y2 + z2 * z2); double Op2m = Op2 /
1000.0;
double Op3 = Math.Sqrt(x3 * x3 + y3 * y3 + z3 * z3); double Op3m = Op3 /
1000.0;
Trace.WriteLine("Op1 Op2 Op3: " + Op1.ToString() + "  " + Op2.ToString()
+ "  " + Op3.ToString());
Trace.WriteLine("Op1m Op2m Op3m: " + Op1m.ToString() + "  " +
Op2m.ToString() + "  " + Op3m.ToString());
```

2) Menghitung nilai unit vektor koordinat foto pada sistem kamera, yaitu :

```
#region Compute i, j, k in camera system
//Compute i, j, k in camera system
```

```
//Compute a direction of i of the camera system
double[] p1p2 = new double[3] { x2 - x1, y2 - y1, z2 - z1 };
double[] p1p3 = new double[3] { x3 - x1, y3 - y1, z3 - z1 };
double length_p1p2 = Accord.Math.Norm.Euclidean(p1p2);
Trace.WriteLine("length_p1p2: " + length_p1p2.ToString() + " Check: " +
Math.Sqrt(p1p2[0] * p1p2[0] + p1p2[1] * p1p2[1] + p1p2[2] *
p1p2[2]).ToString());
double[] i_camera = Accord.Math.Matrix.Divide(p1p2, length_p1p2);
Tools.PrintVectorDebug(ref i_camera, "i camera");

//Compute a direction of k of the camera system
double[] p1p2crossp1p3 = Accord.Math.Matrix.VectorProduct(p1p2, p1p3);
double length_p1p2crossp1p3 = Accord.Math.Norm.Euclidean(p1p2crossp1p3);
Trace.WriteLine("p1p2crossp1p3 length: " +
length_p1p2crossp1p3.ToString());
double[] k_camera = Accord.Math.Matrix.Divide(p1p2crossp1p3,
length_p1p2crossp1p3);
Tools.PrintVectorDebug(ref k_camera, "k_camera");

//Compute a direction j of the camera system
double[] j_camera = Accord.Math.Matrix.VectorProduct(k_camera,
i_camera);
Tools.PrintVectorDebug(ref j_camera, "j_camera");
```

3) Menyusun matriks rotasi untuk koordinat foto, yaitu :

```
//Construct a rotation matrix of camera system
double[,] rot_camera = new double[3,
3]{{i_camera[0],j_camera[0],k_camera[0]},

{i_camera[1],j_camera[1],k_camera[1]},

{i_camera[2],j_camera[2],k_camera[2]}};
Tools.PrintMatrixDebug(ref rot_camera, "rot_camera");
```

4) Menghitung nilai unit vektor koordinat objek pada sistem kamera, yaitu :

```
#region Compute i, j, k in object space system
//Compute vector OP1, OP2, OP3 , but now are based on object space
system of P1,P2,P3; units are in meter
double[] OP1 = new double[3] { X1 - X, Y1 - Y, Z1 - Z };
double[] OP2 = new double[3] { X2 - X, Y2 - Y, Z2 - Z };
double[] OP3 = new double[3] { X3 - X, Y3 - Y, Z3 - Z };
Tools.PrintVectorDebug(ref OP1, "Vector OP1");
Tools.PrintVectorDebug(ref OP2, "Vector OP2");
Tools.PrintVectorDebug(ref OP3, "Vector OP3");

//Construct unit vector P1P2P3 from O in object space
double[] uOP1 = CalculateUnitVector(OP1); //unit vector OP1 in object
space
double[] uOP2 = CalculateUnitVector(OP2); //unit vector OP2 in object
space
```

```
double[] uOP3 = CalculateUnitVector(OP3); //unit vector OP3 in object
space
Tools.PrintVectorDebug(ref uOP1, "unit vector uOP1");
Tools.PrintVectorDebug(ref uOP2, "unit vector uOP2");
Tools.PrintVectorDebug(ref uOP3, "unit vector uOP3");

//Construct position of P1 P2 P3 from O in object space
double[] oP1 = Accord.Math.Matrix.Add(camPosInObject,
Accord.Math.Matrix.Multiply(Op1m, uOP1));
double[] oP2 = Accord.Math.Matrix.Add(camPosInObject,
Accord.Math.Matrix.Multiply(Op2m, uOP2));
double[] oP3 = Accord.Math.Matrix.Add(camPosInObject,
Accord.Math.Matrix.Multiply(Op3m, uOP3));
Tools.PrintVectorDebug(ref oP1, "Coords. of P1");
Tools.PrintVectorDebug(ref oP2, "Coords. of P2");
Tools.PrintVectorDebug(ref oP3, "Coords. of P3");

//Compute a direction of i of the camera system
double[] P1P2 = new double[3] { oP2[0] - oP1[0], oP2[1] - oP1[1], oP2[2]
- oP1[2] };
double[] P1P3 = new double[3] { oP3[0] - oP1[0], oP3[1] - oP1[1], oP3[2]
- oP1[2] };
double length_P1P2 = Accord.Math.Norm.Euclidean(P1P2);
double[] i_object = Accord.Math.Matrix.Divide(P1P2, length_P1P2);
Tools.PrintVectorDebug(ref i_object, "i_object");

//Compute a direction of k of the object system
double[] P1P2crossP1P3 = Accord.Math.Matrix.VectorProduct(P1P2, P1P3);
double length_P1P2crossP1P3 = Accord.Math.Norm.Euclidean(P1P2crossP1P3);
Trace.WriteLine("P1P2crossP1P3 length: " +
length_P1P2crossP1P3.ToString());
double[] k_object = Accord.Math.Matrix.Divide(P1P2crossP1P3,
length_P1P2crossP1P3);
Tools.PrintVectorDebug(ref k_object, "k_object");
```

5) Menyusun matriks rotasi untuk koordinat foto pada persamaan (2.2) yaitu :

```
//Compute a direction j of the object system
double[] j_object = Accord.Math.Matrix.VectorProduct(k_object,
i_object);
Tools.PrintVectorDebug(ref j_object, "j_object");


//Construct a rotation matrix of camera system
double[,] rot_object = new double[3,
3]{{i_object[0],j_object[0],k_object[0]},

{i_object[1],j_object[1],k_object[1]},

{i_object[2],j_object[2],k_object[2]}};
Tools.PrintMatrixDebug(ref rot_object, "rot_object");

#endregion
```

6) Menghitung rotasi total dari perkalian antara matriks rotasi pada koordinat foto dan koordinat objek pada sistem kamera, yaitu :

```
//Compute total rotation
double[,] rotation = Accord.Math.Matrix.Multiply(rot_object,
Accord.Math.Matrix.Transpose(rot_camera));
Tools.PrintMatrixDebug(ref rotation, "Total Rotation");
//Test
double[,] test = Accord.Math.Matrix.Multiply(rotation,
Accord.Math.Matrix.Transpose(rotation));
Tools.PrintMatrixDebug(ref test, "Test Rotation");

return rotation;
}
#endregion
```

3.3.2 Metode *Church*

Metode *Church* merupakan metode kedua dalam pembahasan tentang *closed form solution* pada bab III, metode ini sama halnya dengan metode sebelumnya dengan menggunakan data awal yaitu : koordinat foto (x,y), fokus (f), koordinat objek (X,Y,Z), dan koordinat pendekatan posisi kamera sehingga untuk mendapatkan solusi unik yang dicari maka perlu dilakukan proses perhitungan.

3.3.2.1 Penentuan Koordinat Kamera

Untuk menentukan koordinat kamera dapat diperoleh dengan beberapa tahapan dan algoritma yang diperoleh disusun dalam bahasa pemrograman c# (didalam *console*),antara lain :

1) Menentukan nilai jarak (2.47) dan nilai arah *cosinus* sesuai dengan persamaan (2.48) pada koordinat foto, yaitu:

```
public double[] ChurchAdjustment(double[] Initpose, double[] x, double[]
y, double[] X, double[] Y, double[] Z, double f, ref double err)
        {
            int ndata = x.Length;
            double[] pose = new double[3];
```

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

2) Menentukan nilai konstanta K dan nilai koefisien (A,B,C,E,F,G) pada koordinat objek sesuai dengan persamaan (2.49-2.56), yaitu :

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

3) Menentukan nilai koordinat posisi kamera terkoreksi sesuai dengan persamaan (2.59), yaitu :

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

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

```
        if (Matrix.Max(Matrix.Abs(cor)) < 1.0e-20)
            break;
    }
    pose[0] = Initpose[0];
    pose[1] = Initpose[1];
    pose[2] = Initpose[2];

    return pose;
```

3.3.2.2 Matriks Rotasi

1) Dalam menyusun matriks rotasi dapat diperoleh dengan menggunakan hasil parameter pada persamaan (2.60) dari data koordinat foto, yaitu :

$$e_i = l_i m_j - l_j m_i$$
$$f_i = l_i n_j - l_j n_i$$
$$g_i = m_i n_j - m_j n_i$$

2) Menentukan nilai Δ yaitu jumlah perkalian dari parameter e dan n, yaitu :

$$\Delta = e_1 n_3 + e_2 n_1 + e_3 n_2$$

3) Menyusun matriks rotasi sesuai dengan persamaan (2.62 dan 2.63), yaitu :

$$u_1 = \frac{L_1 g_2 + L_2 g_3 + L_3 g_1}{\Delta}$$
$$v_1 = \frac{L_1 f_2 + L_2 f_3 + L_3 f_1}{(-\Delta)}$$
$$w_1 = \frac{L_1 e_2 + L_2 e_3 + L_3 e_1}{\Delta}$$

$$u_2 = \frac{M_1 g_2 + M_2 g_3 + M_3 g_1}{\Delta}$$
$$v_2 = \frac{M_1 f_2 + M_2 f_3 + M_3 f_1}{(-\Delta)}$$
$$w_2 = \frac{M_1 e_2 + M_2 e_3 + M_3 e_1}{\Delta}$$

$$u_3 = \frac{N_1 g_2 + N_2 g_3 + N_3 g_1}{\Delta}$$
$$v_3 = \frac{N_1 f_2 + N_2 f_3 + N_3 f_1}{(-\Delta)}$$
$$w_3 = \frac{N_1 e_2 + N_2 e_3 + N_3 e_1}{\Delta}$$

$$R = \begin{bmatrix} u_1 & v_1 & w_1 \\ u_2 & v_2 & w_2 \\ u_3 & v_3 & w_3 \end{bmatrix}$$

Setelah diperoleh susunan matriks rotasi, maka perlu dilakukan pengujian kembali dengan menggunakan titik kontrol yang lain dalam foto yang sama terhadap *azimuth* garis utama. Dengan memperhatikan tanda aljabar yang benar bagi semua koordinat foto dan koordinat objek yang digunakan, sehingga dengan menggunakan metode *Church* dapat diperoleh parameter posisi kamera terkoreksi (X_L,Y_L,Z_L) dan matriks rotasi (R) dari satu foto.

3.3.3 Metode *Zeng*

Metode *Zeng* merupakan metode ketiga dalam pembahasan tentang *closed form solution* pada bab III, metode ini sama halnya dengan metode sebelumnya dengan menggunakan data awal yaitu : koordinat foto (x,y), fokus (f) dan koordinat objek (X,Y,Z), untuk mendapatkan solusi unik yang dicari maka perlu dilakukan proses perhitungan.

3.3.3.1 Penentuan Koordinat Kamera

Untuk menentukan koordinat kamera dapat diperoleh dengan beberapa tahapan dalam proses perhitungan yang disusun dalam sebuah algoritma pada bahasa pemrograman c# (didalam *console*), antara lain :

1) Menghitung nilai jarak koordinat objek (L) pada persamaan (2.65), yaitu :

$$(X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2 = R_3^2$$
$$(X_2 - X_3)^2 + (Y_2 - Y_3)^2 + (Z_2 - Z_3)^2 = R_1^2$$
$$(X_3 - X_1)^2 + (Y_3 - Y_1)^2 + (Z_3 - Z_1)^2 = R_2^2$$

Pada console c#, yaitu :

```
{
double[] L = new double[3];      //L[0]=L1=a, L[1]=L2=b, L[2]=L3=c
//P1
double x1 = P1[0]; double y1 = P1[1]; double z1 = P1[2];
//P2
double x2 = P2[0]; double y2 = P2[1]; double z2 = P2[2];
//P3
double x3 = P3[0]; double y3 = P3[1]; double z3 = P3[2];

//P2P3 = a = L1 = Rbc
L[0] = Math.Pow(x2 - x3, 2.0) + Math.Pow(y2 - y3, 2.0) + Math.Pow(z2 -
z3, 2.0);

//P1P3 = b = L2 = Rac
L[1] = Math.Pow(x1 - x3, 2.0) + Math.Pow(y1 - y3, 2.0) + Math.Pow(z1 -
z3, 2.0);
```

```
//P1P2 = c = L3 = Rab
L[2] = Math.Pow(x1 - x2, 2.0) + Math.Pow(y1 - y2, 2.0) + Math.Pow(z1 -
z2, 2.0);

return L;
}
```

2) Menghitung nilai koefisien N pada bentuk persamaan *polynomial* pangkat empat, pada persamaan (2.71 dan 2.72) yaitu :

$$N_1 n^4 + N_2 n^3 + N_3 n^2 + N_4 n^1 + N_5 n^0 = 0$$

dimana diketahui :

$$N_1 = (1 - K_{12})^2 + K_{32}^2 - 2x(1 - K_{12})xK_{32} - K_{32}xE_1^2 + 4xK_{32}x(1 - K_{12})$$

$$N_2 = K_{32}xF_1x(-2 - 2xK_{32} + 4xK_{12} + E_1^2) + K_{12}x(E_1xD_1 + 2xF_1 - 2xK_{12}xF_1) + E_1xD_1x(K_{32} - 1)$$

$$N_3 = K_{32}x(K_{32}xF_1^2 - F_1xD_1xE_1 - 4xK_{12} + 2xK_{32} - E_1^2 - 2xK_{12}xF_1^2) + K_{12}x(K_{12}xF_1^2 + 2xK_{12} - F_1xE_1xD_1 - D_1^2) + D_1^2 - 2 + E_1^2$$

$$N_4 = K_{32}x(2xF_1 - 2xK_{32}xF_1 + E_1xD_1 + 4xK_{12}xF_1) + K_{12}x(E_1xD_1 - 2xK_{12}x F_1 + F_1xD_1^2 - 2xF_1) - D_1xE_1$$

$$N_5 = K_{32}x(K_{32} - 2 - 2xK_{12}) + K_{12}x(K_{12} - D_1^2 + 2) + 1$$

Pada console c#, yaitu :

```
double N1 = (1 - K12) * (1 - K12) + K32 * K32 - 2 * (1 - K12) * K32 -
K32 * E1 * E1 + 4 * K32 * (1 - K12);
double N2 = K32 * F1 * (-2 - 2 * K32 + 4 * K12 + E1 * E1) + K12 * (E1 *
D1 + 2 * F1 - 2 * K12 * F1) + E1 * D1 * (K32 - 1);
double N3 = K32 * (K32 * F1 * F1 - F1 * D1 * E1 - 4 * K12 + 2 * K32 - E1
* E1 - 2 * K12 * F1 * F1) +
K12 * (K12 * F1 * F1 + 2 * K12 - F1 * E1 * D1 - D1 * D1) + D1 * D1 - 2 +
E1 * E1;
 double N4 = K32 * (2 * F1 - 2 * K32 * F1 + E1 * D1 + 4 * K12 * F1) +
K12 * (E1 * D1 - 2 * K12 * F1 + F1 * D1 * D1 - 2 * F1) - D1 * E1;
```

```
double N5 = K32 * (K32 - 2 - 2 * K12) + K12 * (K12 - D1 * D1 + 2) + 1;

double[] N1_5 = new double[5] { N1, N2, N3, N4, N5 };
return N1_5;
```

3) Menghitung nilai real positif dari akar *polynomial*, yaitu :

```
private double[] ExtractRealPositivePolynomialRoots(double[] M1_5)
{
int count = M1_5.Length;
double[] vals = new double[count];
for (int i = 0; i < count; i++)
{
vals[count - 1 - i] = M1_5[i];
}
//Tools.PrintVectorDebug(ref vals, "vals");

PolyLib.Polynomial p = new PolyLib.Polynomial(vals);
PolyLib.Complex[] roots = p.Roots();
//for (int i = 0; i < roots.Length; i++)
//    Trace.WriteLine("real: " + roots[i].Re.ToString() + "  Imaginary:
" + roots[i].Im.ToString());
double[] posRoots = Tools.FindPositiveRoots(roots);
return posRoots;
}
```

4) Mencari nilai posisi kamera pada koordinat X_s dari bentuk persamaan pangkat dua (dengan menggunakan metode *Awange* dan *Grafarend)* untuk menentukan Y_s dan Z_s, yaitu :

```
int countR2 = R2.Length;
for (int kk = 0; kk < countR2; kk++)
{
double r2 = R2[kk];
{
double r1 = R1R3s[0, col];
double r3 = R1R3s[1, col];
double a = r1 * r1 - r2 * r2 - X1 * X1 + X2 * X2 - Y1 * Y1 + Y2 * Y2 -
Z1 * Z1 + Z2 * Z2;
double b = r2 * r2 - r3 * r3 - X2 * X2 + X3 * X3 - Y2 * Y2 + Y3 * Y3 -
Z2 * Z2 + Z3 * Z3;

 double a11 = Y2 - Y1; double a12 = Z2 - Z1;
 double a21 = Y3 - Y2; double a22 = Z3 - Z2;
 double c1 = -(X2 - X1); double c2 = -(X3 - X2);
 double b1 = 0.5 * a; double b2 = 0.5 * b;

 double e = 1.0 / (a11 * a22 - a12 * a21); double f = a22 * b1 - a12 *
b2; double g = a22 * c1 - a12 * c2;
double h = a11 * b2 - a21 * b1;
double i = a11 * c2 - a21 * c1;
```

```
double k = X1 * X1 + Y1 * Y1 + Z1 * Z1;

double l = e * e * i * i + e * e * g * g + 1;
double m = 2 * (e * e * f * g + e * e * h * i - X1 - e * g * Y1 - e * i
* Z1);
double n = k - r1 * r1 - 2 * Y1 * e * f + e * e * f * f - 2 * Z1 * e * h
+ e * e * h * h;

///find the squared roots
double[] coef = new double[3] { n, m, l };
PolyLib.Polynomial p2 = new PolyLib.Polynomial(coef);
PolyLib.Complex[] roots = p2.Roots();
double[] posRoots = Tools.FindPositiveRoots(roots);   //two values of X
for each set of R1R3
//Tools.PrintVectorDebug(ref posRoots, "posRoots of Xs");
if (posRoots.Length != 0) //avoid an empty roots
{
//Trace.WriteLine("upperbound of posRoots: " +
posRoots.GetUpperBound(0).ToString());
double Xs1 = posRoots[0];
double Xs2 = posRoots[1];
//find Y & Z for each set of X
double Ys1 = e * (f + g * Xs1);
double Ys2 = e * (f + g * Xs2);
double Zs1 = e * (h + i * Xs1);
double Zs2 = e * (h + i * Xs2);
double[] XsYsZs1 = new double[3] { Xs1, Ys1, Zs1 };
double[] XsYsZs2 = new double[3] { Xs2, Ys2, Zs2 };
list.Add(XsYsZs1);
list.Add(XsYsZs2);
}
```

5) Setelah diperoleh koordinat posisi kamera (X_s,Y_s,Z_s) maka dilakukan proses koreksi arah vektor (*T*) nilai positif untuk mendapatkan solusi unik dari dua solusi koordinat posisi kamera yang dihasilkan. Proses koreksi arah vektor (T) yaitu dari hasil vektor ***12*** (selisih dari koordinat objek titik 1 dan 2), ***13***(selisih dari koordinat objek titik 1 dan 3) dan ***1S*** (selisih dari koordinat kamera dan koordinat objek) seperti pada persamaan (3.52).

$$T = 1S \bullet (12 * 13) \qquad (3.52)$$

3.3.3.2 Matriks Rotasi

1) Setelah diperoleh koordinat posisi kamera, maka dapat disusun matriks rotasi, seperti pada persamaan (2.89-2.93), yaitu :

```
private double[] Find_a_b_c_d(double[] al, double[] bl, double[] ap,
double[] bp)
        {
//Compute Eq.50 page 332
//enum coord {x=0,y=1,z=2};
double p = (ap[1] - al[1]) * (bp[2] - bl[2]) - (ap[2] - al[2]) * (bp[1]
- bl[1]);
double q = (ap[2] - al[2]) * (bp[0] - bl[0]) - (ap[0] - al[0]) * (bp[2]
- bl[2]);
double r = (ap[0] - al[0]) * (bp[1] - bl[1]) - (ap[1] - al[1]) * (bp[0]
- bl[0]);
//Trace.WriteLine(" p q r: " + p.ToString() + "  " + q.ToString() + "  "
+ r.ToString());

//direction numbers of the rotation axis g(XYZ) form object space to
image space
double[] g = new double[3] { p, q, r };
double cosGamma = FindcosGamma(ap, al, g);
```

2) Menyusun parameter rotasi (d,a,b,c) pada persamaan (2.83), yaitu :

$$R = \begin{bmatrix} d^2 + a^2 - b^2 - c^2 & 2(ab + cd) & 2(ac - bd) \\ 2(ab - cd) & d^2 - a^2 + b^2 - c^2 & 2(bc + ad) \\ 2(ac + bd) & 2(bc - ad) & d^2 - a^2 - b^2 + c^2 \end{bmatrix}$$

Pada console c#, yaitu :

```
//Find d, a, b, c
double d = Math.Sqrt((1 + cosGamma) / 2.0);
double d2 = d * d;
double pqr = p * p + q * q + r * r;

double a = p * Math.Sqrt((1 - d2) / pqr);
double b = q * Math.Sqrt((1 - d2) / pqr);
double c = r * Math.Sqrt((1 - d2) / pqr);

return new double[4] { a, b, c, d };
}

private double[,] ComposeRotMatrix(double[] abcd)
{
double a = abcd[0]; double b = abcd[1]; double c = abcd[2]; double d =
abcd[3];
```

```
double a2 = a * a; double b2 = b * b; double c2 = c * c; double d2 = d *
d;
double r11 = d2 + a2 - b2 - c2; double r12 = 2 * (a * b + c * d); double
r13 = 2 * (a * c - b * d);
double r21 = 2 * (a * b - c * d); double r22 = d2 - a2 + b2 - c2; double
r23 = 2 * (b * c + a * d);
double r31 = 2 * (a * c + b * d); double r32 = 2 * (b * c - a * d);
double r33 = d2 - a2 - b2 + c2;

double[,] rot = new double[3, 3]{{r11,r12,r13},
                                 {r21,r22,r23},
                                 {r31,r32,r33}};
return rot;
```

3.3.4 Analisa Metode *Closed Form Solution*

Dari proses perhitungan yang telah dilakukan dapat dianalisa bahwa dari ketiga metode *closed form solution* mempunyai solusi masing-masing dari tiap-tiap metode untuk memperoleh parameter *exterior orientation* (EO) yang tediri dari parameter pendekatan koordinat posisi kamera (X_L,Y_L,Z_L) dan matriks rotasi (R) sehingga diperoleh solusi unik dari ketiga metode *closed form solution* yang diwujudkan dalam sebuah algoritma dengan menggunakan bahasa pemrograman c# (dalam *console*).

BAB IV

HASIL DAN PEMBAHASAN

4.1 Hasil Perhitungan

Data yang akan dianalisa dalam bab IV diperoleh dengan melakukan perhitungan di *Microsoft Excel* dan *console c#* (*visual studio 2010*), dengan menggunakan metode *closed form solution* yang meliputi : metode *Fischler-Bolles*, metode *Church* dan metode *Zeng*.

4.1.1 Metode *Fischler-Bolles*

4.1.1.1 Batas Toleransi (*Threshold t*)

Dalam menentukan batasan toleransi dalam metode *RANSAC*, telah dilakukan percobaan secara acak terhadap titik-titik objek dalam satu foto. Dari hasil perhitungan diperoleh hasil tabulasi pada Tabel 4.1, dimana hasil tersebut sudah diverifikasi oleh (*Fischler* dan *Bolles*, 1981).

Tabel 4.1 Tabulasi nilai E(k) untuk hubungan nilai *w* dan *n*

w	n =1	2	3	4	5	6
0.9	1.1	1.2	1.4	1.5	1.7	1.9
0.8	1.3	1.6	2	2.4	3	3.8
0.7	1.4	2	2.9	4.2	5.9	8.5
0.6	1.7	2.8	4.6	7.7	13	21
0.5	2.0	4.0	8.0	16	32	64
0.4	2.5	6.3	16	39	98	244
0.3	3.3	11	37	123	412	
0.2	5	25	125	625		

Perhitungan selengkapnya, dapat dilihat pada lampiran A

dimana :

w : nilai kemungkinan (dalam satuan persen) yang dipilih tiap data titik pada foto.

n : merupakan jumlah titik dan $E(k)$ merupakan nilai yang diinginkan dari k.

Dengan menggunakan ketentuan (SD(k)) < 0, maka diperoleh hasil bahwa n = 1 dan 2 dengan persentasi kemungkinan (w) 90 % mempunyai jumlah solusi yang tidak terbatas, sedangkan n ≥ 3 dengan persentasi kemungkinan (w) 90 % mempunyai jumlah solusi yang unik. Selanjutnya setelah diperoleh batasan toleransi, maka dapat ditentukan metode LDP dengan menggunakan n titik.

4.1.1.2 Metode *Perspective-3-Point Problem* (P3P)

Dalam proses perhitungan koordinat kamera menggunakan metode *P3P*, diketahui data awal yaitu koordinat foto (x,y), koordinat objek (X,Y,Z), dan panjang fokus (f), seperti pada Tabel 4.2.

Tabel 4.2 Data awal

ID	Koordinat Objek (m)			Koordinat Foto (mm)		fokus (mm)
	X	Y	Z	x	y	
1	-0.101356	-0.068686	-0.661268	3.15447	2.41421	-24.00
2	0.089863	-0.071431	-0.667876	-2.84669	2.43875	
3	0.088922	0.085207	-0.778067	-3.36471	-1.50055	
4	-0.102298	0.087524	-0.771782	4.12840	-1.49685	

Data awal pada Tabel 4.2 digunakan untuk menghasilkan nilai koordinat kamera dan matriks rotasi, seperti yang ditunjukan pada Tabel 4.3 & Tabel 4.4.

Tabel 4.3 Solusi 1 untuk nilai pendekatan awal posisi kamera

ID	X_L	Y_L	Z_L
Solusi 1	0.03701	0.69539	-0.71768

Solusi 1 untuk matriks rotasi

$$R = \begin{bmatrix} -0.998033 & 0.034382 & 0.052427 \\ 0.051274 & -0.033600 & 0.998119 \\ 0.036079 & 0.998844 & 0.031771 \end{bmatrix}$$

Tabel 4.4 Solusi 2 untuk nilai pendekatan awal posisi kamera

ID	X_L	Y_L	Z_L
Solusi 2	0.21605	-0.17426	0.03567

Solusi 2 untuk matriks rotasi

$$R = \begin{bmatrix} -0.957238 & -0.044187 & 0.285908 \\ -0.026755 & -0.970511 & -0.239569 \\ 0.288063 & -0.236974 & 0.927827 \end{bmatrix}$$

Perhitungan selengkapnya, dapat dilihat pada lampiran A

Dengan metode *P3P* dapat dihasilkan multi solusi untuk nilai pendekatan parameter luar (*exterior orientation*) yang meliputi koordinat kamera (X_L,Y_L,Z_L) dan matriks rotasi pada Tabel 4.3 dan Tabel 4.4.

4.1.2 Metode *Church*

4.1.2.1 Koordinat Kamera

Dalam proses perhitungan koordinat kamera menggunakan metode *Church*, diketahui data awal yaitu koordinat foto (x,y), koordinat objek (X,Y,Z), panjang fokus (f), dan parameter pendekatan awal koordinat kamera (X_L,Y_L,Z_L), seperti pada Tabel 4.5.

Tabel 4.5 Data awal

ID	Koordinat Objek (m)			Koordinat Foto (mm)		fokus (mm)
	X	Y	Z	x	y	
1	-0.101356	-0.068686	-0.661268	3.15447	2.41421	-24.00
2	0.089863	-0.071431	-0.667876	-2.84669	2.43875	
3	0.088922	0.085207	-0.778067	-3.36471	-1.50055	
4	-0.102298	0.087524	-0.771782	4.12840	-1.49685	

Tabel 4.6 Parameter awal koordinat kamera

ID	X_L	Y_L	Z_L
Sol 1	0.03701	0.69539	-0.71768
Sol 2	0.21605	-0.17426	0.03567

Data awal pada Tabel 4.5 dan Tabel 4.6 digunakan untuk menghasilkan nilai koordinat kamera terkoreksi (*update*) dan matriks rotasi, seperti yang ditunjukan pada Tabel 4.7 & Tabel 4.8.

Tabel 4.7 Nilai parameter pendekatan awal koordinat kamera terkoreksi (*update*)

Parameter pendekatan awal *update* koordinat kamera (m)			
ID	X_L	Y_L	Z_L
Sol 1	0.21605203	-0.174263311	-0.174263311
Sol 2	0.03701003	0.695385149	-0.717683602

Perhitungan selengkapnya, dapat dilihat pada lampiran B

4.1.2.2 Matriks Rotasi

Untuk menentukan nilai parameter rotasi (*omega, phi* dan *kappa*) perlu disusun terlebih dahulu matriks rotasi pada multi solusi, seperti pada Tabel 4.8.

Tabel 4.8 Susunan matrik rotasi pada kedua solusi

ID	No	Parameter Koefisien		
		u	v	w
Solusi 1	1	-0.998033	0.034382	0.052427
	2	0.051274	-0.033600	0.998119
	3	0.036079	0.998844	0.031771
Solusi 2	1	-0.957238	-0.044187	0.285908
	2	-0.026755	-0.970511	-0.239569
	3	0.288063	-0.236974	0.927827

Perhitungan selengkapnya, dapat dilihat pada lampiran B

Keterangan tabel :

u,v,w : nilai unsur paramater rotasi

4.1.3 Metode *Zeng*

4.1.3.1 Koordinat Kamera

Dalam proses perhitungan koordinat kamera menggunakan metode *Zeng*, diketahui data awal yaitu koordinat foto (x,y), koordinat objek (X,Y,Z), dan panjang fokus (f), seperti pada Tabel 4.9.

Tabel 4.9 Data awal

ID	Koordinat Objek (m)			Koordinat Foto (mm)		fokus (mm)
	X	Y	Z	x	y	
1	-0.101356	-0.068686	-0.661268	3.15447	2.41421	-24.00
2	0.089863	-0.071431	-0.667876	-2.84669	2.43875	
3	0.088922	0.085207	-0.778067	-3.36471	-1.50055	
4	-0.102298	0.087524	-0.771782	4.12840	-1.49685	

Perhitungan selengkapnya, dapat dilihat pada lampiran C

Dengan menggunakan data awal pada Tabel 4.9 maka dapat dihasilkan nilai pendekatan parameter koordinat kamera (X_L,Y_L,Z_L), seperti pada tabel dibawah ini :

Tabel 4.10 Nilai parameter pendekatan awal koordinat kamera

ID	X_L	Y_L	Z_L
Sol 1	0.03701	0.69539	-0.71768

Perhitungan selengkapnya, dapat dilihat pada lampiran C

Dari proses perhitungan dapat diperoleh solusi unik untuk nilai parameter pendekatan awal koordinat kamera (X_L,Y_L,Z_L) pada solusi 1 dengan koreksi nilai arah T bernilai positif, seperti ditunjukan pada Tabel 4.10.

4.1.3.2 Matriks Rotasi

Setelah diketahui solusi untuk nilai pendekatan posisi kamera maka dapat disusun matriks rotasi, seperti dibawah ini :

$$R = \begin{bmatrix} -0.998033 & 0.034382 & 0.052427 \\ 0.051274 & -0.033600 & 0.998119 \\ 0.036079 & 0.998844 & 0.031771 \end{bmatrix}$$

4.2 Analisa dan Evaluasi Hasil Perhitungan Metode *Closed Form Solution*

Dari hasil perhitungan ketiga metode *closed form solution* yang meliputi : metode *Fischler-Bolles*, metode *Church* dan metode *Zeng*, dapat dihasilkan nilai pendekatan awal parameter luar (*exterior orientation*). Hasil analisa dan evaluasi ketiga metode *closed form solution*, yaitu :

Tabel 4.11 Hasil Tabulasi Metode *Closed Form Solution*

Metode *closed form solution*	ID Solusi	Posisi kamera			Matrik rotasi		
		X_L (m)	Y_L (m)	Z_L (m)	u	v	w
Metode *Fischler-Bolles*	1	0.03701	0.69539	-0.71768	-0.998	0.0513	0.0361
					0.0344	-0.0336	0.9988
					0.0524	0.9981	0.0318
	2	0.21602	-0.17426	0.03567	-0.9572	-0.0442	0.2859
					-0.0268	-0.9705	-0.2396
					0.2881	-0.237	0.9278
Metode *Church*	1	0.21602	-0.17426	0.03567	-0.9572	-0.0442	0.2859
					-0.0268	-0.9705	-0.2396
					0.2881	-0.237	0.9278
	2	0.03701	0.69539	-0.71768	-0.998	0.0513	0.0361
					0.0344	-0.0336	0.9988
					0.0524	0.9981	0.0318
Metode *Zeng*		0.03701	0.69539	-0.71768	-0.998	0.0513	0.0361
					0.0344	-0.0336	0.9988
					0.0524	0.9981	0.0318

Sehingga dapat diketahui bahwa masing-masing metode mempunyai proses koreksi yang berbeda untuk menghasilkan parameter luar (*exterior orientation*), seperti yang ditunjukkan pada Tabel 4.11.

4.2.1 Metode *Fischler-Bolles*

Pada proses perhitungan dengan menggunakan metode LDP pada tiga titik (*P3P*) dapat diperoleh dua solusi dari dua nilai akar positif untuk koordinat kamera dan matriks rotasi. Dari dua solusi yang diberikan dapat dihasilkan nilai pendekatan parameter luar (*exterior orientation*) yang unik dengan menggunakan metode *fitting line* pada minimal 4 titik yang diambil secara acak.

4.2.2 Metode *Church*

Dari proses perhitungan dengan metode *Church* dapat dihasilkan koordinat kamera *update* (X_L,Y_L,Z_L) dan matriks rotasi dengan parameter koreksi koordinat kamera (ΔX, ΔY, ΔZ) $\leq$ 0.0000 m.

4.2.3 Metode *Zeng*

Pada proses perhitungan dengan menggunakan metode *zeng* dapat diketahui bahwa terdapat 4 nilai akar *real* positif dan 4 nilai akar *imaginer*. Akan tetapi, untuk proses perhitungan selanjutnya hanya akan digunakan nilai akar *real* positif sehingga akan diperoleh dua solusi untuk nilai parameter pendekatan awal koordinat kamera (X_L,Y_L,Z_L) dengan koreksi nilai arah vektor (T) positif. Untuk memperoleh solusi unik untuk nilai parameter pendekatan awal koordinat kamera (X_L,Y_L,Z_L) dan

parameter rotasi yaitu dari selisih terkecil nilai vektor antara koordinat foto dan koordinat objek pada keempat titik..

BAB V

PENUTUP

5.1 Kesimpulan

Evaluasi metode untuk menentukan nilai pendekatan awal parameter luar (*exterior orientation*) yang terdiri dari tiga parameter koordinat kamera (X_L,Y_L,Z_L) dan matriks rotasi dengan menggunakan metode *closed form solution* dapat diperoleh hasil sebagai berikut :

1) Solusi yang diperoleh dengan metode *Fischler-Bolles*, metode *Church* dan metode *Zeng* mempunyai syarat koreksi yang berbeda-beda tetapi menghasilkan nilai akhir yang sama untuk nilai pendekatan awal parameter luar (*exterior orientation*).
2) Dengan menggunakan metode *Fischler-Bolles* pada tiga titik (P3P) dapat diperoleh posisi foto yang konvergen yaitu pada solusi 1 dengan posisi kamera (X : 0.03701; Y : 0.69539; Z : -0.71768) dari dua solusi yang ada. Dengan menggunakan metode *fitting line* pada minimal 4 titik yang diambil secara acak maka akan diperoleh solusi unik.
3) Dengan menggunakan metode *Church* dapat diperoleh dua solusi yang masing-masing menghasilkan koordinat posisi kamera terkoreksi dengan nilai koreksi (ΔX, ΔY, ΔZ) $\leq$ 0.0000 m dan susunan matriks rotasi dari kedua solusi.
4) Dengan menggunakan metode *Zeng* pada tiga titik dapat diperoleh 4 nilai akar real positif, sehingga perlu dilakukan eliminasi dengan melakukan proses koreksi terhadap arah vektor (*T*) positif maka akan dihasilkan 2 solusi koordinat

posisi kamera. Untuk mendapatkan solusi unik dari kedua solusi maka dapat diperoleh dari selisih terkecil nilai vektor antara koordinat objek dan koordinat foto dengan nilai x didefinisikan dengan nilai 1 yaitu pada posisi (X : 0.03701; Y : 0.69539; Z : -0.71768).

5.2 Saran

Saran yang dapat diberikan untuk penelitian tentang "**Evaluasi Metode Penentuan Pendekatan Parameter *Exterior Orientation* (EO) dengan menggunakan Metode *Closed Form Solution* untuk Pemotretan pada Multi Foto Konvergen**", antara lain :

1) Untuk mendapatkan solusi unik pada metode *Fischler-Bolles* perlu dilakukan *fitting line* pada minimal 4 titik yang diambil secara acak sehingga dapat diketahui kombinasi yang cocok dalam penentuan koordinat posisi kamera pada satu foto antara koordinat foto dan koordinat objek.

DAFTAR PUSTAKA

Atkinson, K.B., 2001. " Close Range Photogrammetry and Machine Vision". Whittles Publishing. Scotland, UK.

Abdel Aziz dan Karara dan Karara, M., 1971. "*The Direct linier Transformation*".

Awange, J.L., dan grafarend, E.W., 2004. "*Solving Algebraic Computational Problem in Geodesy and Geoinformatika*". Springer, New York.

Roberts, R., 2006. " *Using RANSAC to Recover Camera Location*".

Church, P. E., (1980). *"Tilted Photography"*. The Center for Photogrammetric Training University Syracuse.

Cooper, M.A.R, dan Robson, S., 2001. *"Teory of Close Range Photogrammetry"*.

Faig, W., 1973. *"Convergent Photos for Close Range"*. Photogrammetric Engineering 39 (6), 605-610.

Fischler, M.A., dan Bolles, R.C., 1980. *"Random Samples Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography"*.

Fischler, M.A., dan Bolles, R.C., 1981. "*Random Samples Consensus : A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography"*.

Grussenmeyer, P., dan Al-Khalil, O., 2002. *"Solutions for Exterior Orientation In Photogrammetry, A Review"*. The Photogrammetric Record an International Journal of Photogrammetry.

Hinsken, D. I. L., 1988. *"A Singularity Free Algorithm For Spatial Orientation of Bundles"*. Federal Republic of Germany, Institut fur Photogrammetrie und Cartographie.

Kraus, K., 1993. *"Photogrammetry"*, 4th Ed., Fred . Duemmlers Verlag, Bonn, Germany.

Leica, Geosystem, 2006. *"Tutorial Leica Photogrammetry Suite Project Manager"*.

Matas, J dan Chum, O., 2005. " Randomized RANSAC with Sequential Probability Ratio Test". Center for Machine Perception, Dept. of Cybernetics, CTU Prague, Korlovo nam 13, CZ 121 35.

Mikhail, E.M., Bethel, J., dan McGlone, J., 2001. *"Introduction To Modern Photogrammetry"* , John Wiley & Sons, Inc, New York.

Nister, D., "RANSAC". Center for Visualization & Virtual Environments.

Pope, A. J., (1970). *"An Advantages Alternative Parameterization of Rotations for Analytical Photogrammetry*". ESSA Technical Report C&GS 39.

Rampal, K.K., 1979. *"A Closed Solution for Space Resection"*. Departement of Civil Engineering. Indian Institute of techhnology Kanpur. Kanpur, India.

Schenk, T., "Automatic Exterior Orientation". Digital Photogrammetry, Vol : 1. terra Scince.

Schut, G.H., 1959. *"Contruction of Orthogonal Matrices and Their Application in Analitycal Photogrammetria"*, 1594) : 149-162.

Shih, T.Y., dan Faig, W., 1987. *"A Solution for Space Resection in Closed Form"*. Departement of Surveying Engineering. University of New Brunswick. Canada.

Shih, T.Y., 1990. *"The Duality and Critical Condition in the Formulation and Decomposition of a Rotation Matrix"*, *Photogrammetric Engineering & Remote Sensing*, 56(8):1173-1179.

Stewart, C.V. " Random Sampling Estimation Without Prior Scale". Department of Computer Science Rensselaer Polytechnic Institute, Troy, USA.

9

Svoboda, T., 2008. " RANSAC (RANdom SAmple Consensus)". Czech Technical University in Prague, Center for Machine Perception.

Thompson, E.H., 1959. *"A Method for the Construction of Orthogonal Matrices"*.

Thompson, E.H., 1969. *"Introduction to the Algebra of Matrices with Some Applications"*. The University of Toronto Press

Torr, P. " Twenty Five Years of RANSAC".

Wolf, P. R., (1983). *"Elements Of Photogrammetry With Air Photo Interpretation and Remote Sensing"*. McGraw-Hill,Inc. All Rights Reserved.

Xu, M., dan Petron, M., 2008. " Distributed RANSAC for 3D Reconstruction". Imperial College London, exhibition Road, London,UK.

Zeng, Z.dan Wang, X., (1992). *"A General Solution of a Closed Form Space Resection"*. Departement of Mining Engineering, The Central-South University of Technology.

Lampiran proses perhitungan

dan algoritma dalam *console aplication C#*

Lampiran A

Metode *Fischler-Bolles*

A.1 Perhitungan Batas Toleransi (*Threshold t*)

Untuk contoh, nilai kemungkinan pada tiap titik (w) adalah 90 % (w = 0.9), data titik yang baik (n) yaitu 3, sehingga dapat dihitung nilai kemungkinan yang diinginkan E(k) dan standart deviasi $SD(k)$ dari jumlah percobaan (k) yaitu :

$$E(k) = w^{-n}$$
$$= 0.9^{-3}$$
$$= 1.37$$

$$SD(k) = \sqrt{(1 - w^n)} * (\frac{1}{w^n})$$
$$= \sqrt{(1 - 0.9^3)} * (\frac{1}{0.9^3})$$
$$= 0.714$$

Tabel Tabulasi nilai E(k) untuk hubungan nilai w dan n

w	n =1	2	3	4	5	6
0.9	1.1	1.2	1.4	1.5	1.7	1.9
0.8	1.3	1.6	2	2.4	3	3.8
0.7	1.4	2	2.9	4.2	5.9	8.5
0.6	1.7	2.8	4.6	7.7	13	21
0.5	2.0	4.0	8.0	16	32	64
0.4	2.5	6.3	16	39	98	244
0.3	3.3	11	37	123	412	
0.2	5	25	125	625		

A.2 Penentuan Koordinat Kamera

1. Input Data Koordinat Foto (x,y), Koordinat Objek (X_i,Y_i,Z_i) dan panjang fokus.

ID	Koordinat Objek (m)			Koordinat Foto (mm)		fokus (mm)
	X	Y	Z	x	y	
1	-0.101356	-0.068686	-0.661268	3.15447	2.41421	-24.00
2	0.089863	-0.071431	-0.667876	-2.84669	2.43875	
3	0.088922	0.085207	-0.778067	-3.36471	-1.50055	
4	-0.102298	0.087524	-0.771782	4.12840	-1.49685	

2. Dari data yang diketahui maka dapat dihasilkan nilai jarak antara tiga titik koordinat objek (A,B,C) dan titik tengah koordinat kamera (lihat persamaan 2.11).

ID	JARAK
Rab	0.1914
Rac	0.2712
Rbc	0.1915

3. Kemudian dapat dihasilkan nilai sudut antara koordinat foto dan titik tengah kamera (lihat persamaan 2.12).

ID	NILAI
Cos θ_{12}	0.9695275
Cos θ_{23}	0.9866175
Cos θ_{13}	0.9510534

4. Nilai Konstanta untuk perbandingan jarak koordinat titik objek (lihat persamaan 2.15)

ID	Koefisien
K1	0.498820
K2	1.001709

5. Nilai koefisien G_i untuk mencari nilai akar dari bentuk persamaan *biquadratic polynomial* (lihat persamaan 2.14).

Koefisien	NILAI
G4	-0.9438
G3	3.7009
G2	-5.4747
G1	3.6194
G0	-0.9018

6. Hasil nilai akar dari bentuk persamaan *biquadratic polynomial*

ID	ROOTS
X_1	complex
X_2	complex
X_3	1.00000
X_4	0.9273

7. Dari beberapa nilai parameter maka dapat diperoleh dua solusi untuk koordinat kamera.

ID	X_L	Y_L	Z_L
Sol 1	0.03701	0.69539	-0.71768
Sol 2	0.21605	-0.17426	0.03567

A.3 Matriks Rotasi

Sehingga dapat disusun matriks rotasi dari kedua solusi, yaitu sebagai berikut:

ID	Parameter		
	u	v	w
Sol 1	-0.99803	0.03438	0.05242
	0.05127	-0.03360	0.99812
	0.03608	0.99884	0.03177
Sol 2	-0.95724	-0.04418	0.28591
	-0.02676	-0.97052	-0.23957
	0.28807	-0.23697	0.92783

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

namespace CFResec
{
    class FB
    {
        #region Fields
        //Focal Length
        double f;

        //Object space points
        double X1; double Y1; double Z1;    //Point P1 or A
        double X2; double Y2; double Z2;    //Point P2 or B
        double X3; double Y3; double Z3;    //Point P3 or C
        double X4; double Y4; double Z4;    //Point P4 or D

        //image space points
        double x1; double y1; double z1;    //Point p1 or A
        double x2; double y2; double z2;    //Point p2 or B
        double x3; double y3; double z3;    //Point p3 or C
        double x4; double y4; double z4;    //Point p4 or D

        //Object Space Vectors
        double[] P1;    //point A
        double[] P2;    //point B
        double[] P3;    //point C
        double[] P4;    //point D

        //camera space vectors
        double[] p1;    //point A
        double[] p2;    //point B
        double[] p3;    //point C
        double[] p4;    //point D

        //unit vector from PC to image points: uvA = a; uvB = b; uvC = c;
uvD = d
        double[] uvA; double[] uvB; double[] uvC; double[] uvD;

        //cosine of an angle from 1st Triangle: cos(Tab)=a.b; cos(Tac)=a.c;
cos(Tbc)=b.c
        double cosTab = 0; double cosTac = 0; double cosTbc = 0;

        //In Triangle ABC: a = distance LC ; b = distance LB; c = distance
LC (LEGS)
        //all coords are in object space
        double a = 0; double b = 0; double c = 0;

        //Distances b/w control points: Rab = AB, Rac = AC, Rbc = BC
        double Rab = 0, Rac = 0, Rbc = 0;
        #endregion

        #region Constructors
        public FB(double focal, double[,] objPts, double[,] imgPts)
        {
            f = focal;
```

```
            #region Object Space Points of P1,P2,P3 (selected control
points) & P4 (additional)
            //Point A
            this.X1 = objPts[0, 0];
            this.Y1 = objPts[0, 1];
            this.Z1 = objPts[0, 2];
            Trace.WriteLine("X1 Y1 Z1: " + X1.ToString() + "  " +
Y1.ToString() + "  " + Z1.ToString());
            //Point B
            this.X2 = objPts[1, 0];
            this.Y2 = objPts[1, 1];
            this.Z2 = objPts[1, 2];
            Trace.WriteLine("X2 Y2 Z2: " + X2.ToString() + "  " +
Y2.ToString() + "  " + Z2.ToString());
            //Point C
            this.X3 = objPts[2, 0];
            this.Y3 = objPts[2, 1];
            this.Z3 = objPts[2, 2];
            Trace.WriteLine("X3 Y3 Z3: " + X3.ToString() + "  " +
Y3.ToString() + "  " + Z3.ToString());
            //Point D
            this.X4 = objPts[3, 0];
            this.Y4 = objPts[3, 1];
            this.Z4 = objPts[3, 2];
            Trace.WriteLine("X4 Y4 Z4: " + X4.ToString() + "  " +
Y4.ToString() + "  " + Z4.ToString());
            #endregion

            #region Image/CAMERA space Points of p1, p2, p3 (selected image
points) & p4 (additional)
            //Point 1 ==> A
            this.x1 = imgPts[0, 0];
            this.y1 = imgPts[0, 1];
            this.z1 = -f;
            Trace.WriteLine("x1 y1 z1: " + x1.ToString() + "  " +
y1.ToString() + "  " + z1.ToString());
            //Point 2 ==> B
            this.x2 = imgPts[1, 0];
            this.y2 = imgPts[1, 1];
            this.z2 = -f;
            Trace.WriteLine("x2 y2 z2: " + x2.ToString() + "  " +
y2.ToString() + "  " + z2.ToString());
            //Point 3 ==> C
            this.x3 = imgPts[2, 0];
            this.y3 = imgPts[2, 1];
            this.z3 = -f;
            Trace.WriteLine("x3 y3 z3: " + x3.ToString() + "  " +
y3.ToString() + "  " + z3.ToString());
            //Point 4 ==> D
            this.x4 = imgPts[3, 0];
            this.y4 = imgPts[3, 1];
            this.z4 = -f;
            Trace.WriteLine("x4 y4 z4: " + x4.ToString() + "  " +
y4.ToString() + "  " + z4.ToString());
            #endregion

            #region Compute Rab, Rac, Rbc
            Rab = Accord.Math.Distance.Euclidean(new double[3] { X1, Y1, Z1
}, new double[3] { X2, Y2, Z2 });
```

```
            Rac = Accord.Math.Distance.Euclidean(new double[3] { X1, Y1, Z1
}, new double[3] { X3, Y3, Z3 });
            Rbc = Accord.Math.Distance.Euclidean(new double[3] { X2, Y2, Z2
}, new double[3] { X3, Y3, Z3 });
            #endregion

            #region Unit vectors
            uvA = Tools.CalculateUnitVector(new double[3] { x1, y1, z1 });
            uvB = Tools.CalculateUnitVector(new double[3] { x2, y2, z2 });
            uvC = Tools.CalculateUnitVector(new double[3] { x3, y3, z3 });
            uvD = Tools.CalculateUnitVector(new double[3] { x4, y4, z4 });
            #endregion

            #region Cosines Theta(ab) = cosTab, Cosines Theta(ac) = cosTac;
Cosines Theta(bc) = cosTbc
            cosTab = Tools.DotProduct(uvA, uvB);
            cosTac = Tools.DotProduct(uvA, uvC);
            cosTbc = Tools.DotProduct(uvB, uvC);
            #endregion


            #region OBJECT Space Vectors
            this.P1 = new double[3] { X1, Y1, Z1 };
            this.P2 = new double[3] { X2, Y2, Z2 };
            this.P3 = new double[3] { X3, Y3, Z3 };
            this.P4 = new double[3] { X4, Y4, Z4 };
            #endregion

            #region CAMERA Space Vectors
            this.p1 = new double[3] { x1, y1, z1 };
            this.p2 = new double[3] { x2, y2, z2 };
            this.p3 = new double[3] { x3, y3, z3 };
            this.p4 = new double[3] { x4, y4, z4 };
            #endregion

        }
        #endregion

        #region Public Methods
        public void FischlerBollesSolution()
        {
            double K1 = (Rbc * Rbc) / (Rac * Rac);
            double K2 = (Rbc * Rbc) / (Rab * Rab);
            double G4 = Estimate_G4(K1, K2, cosTbc);
            double G3 = Estimate_G3(K1, K2, cosTab, cosTac, cosTbc);
            double G2 = Estimate_G2(K1, K2, cosTab, cosTac, cosTbc);
            double G1 = Estimate_G1(K1, K2, cosTab, cosTac, cosTbc);
            double G0 = Estimate_G0(K1, K2, cosTac);

            double[] coefs = new double[5] { G0, G1, G2, G3, G4 };
            PolyLib.Polynomial p = new PolyLib.Polynomial(coefs);
            PolyLib.Complex[] roots = p.Roots();
            for (int i = 0; i < roots.Length; i++)
                Trace.WriteLine("real: " + roots[i].Re.ToString() + "
Imaginary: " + roots[i].Im.ToString());
            double[] posRoots = Tools.FindPositiveRoots(roots);
            Tools.PrintVectorDebug(ref posRoots, "Roots in
FischlerBolles");

            //Compute all possibilities of Legs of a b c
```

```
            List<double[]> abcList = new List<double[]>();
            foreach (double rts in posRoots)
            {
                double lega = Leg_a(rts);
                double legb = Leg_b(rts, lega);
                double[] yArray = Est_y(lega);
                double[] legcArray = Leg_c(yArray, lega);
                foreach (double legc in legcArray)
                {
                    abcList.Add(new double[3] { lega, legb, legc });
                }
            }

            Trace.WriteLine("List of all possibilities of abc legs:");
            List<double[]> validatedList_abc = new List<double[]>();
            foreach (double[] abc in abcList)
            {
                if (Validate_abc(abc))
                    validatedList_abc.Add(abc);
            }

            foreach (double[] a in validatedList_abc)
            {
                Trace.WriteLine(String.Format("{0} \t {1} \t {2}", a[0],
a[1], a[2]));

            }

            //Find all posible solution for camera positions
            List<double[]> cameraPositionList = new List<double[]>();
            for (int i = 0; i < validatedList_abc.Count; i++)
            {
                Trace.WriteLine("s1s2s3: " +
validatedList_abc.ElementAt(i)[0].ToString() + "  " +
validatedList_abc.ElementAt(i)[1].ToString() + "  " +
validatedList_abc.ElementAt(i)[2].ToString());
                double[] cameraPos =
Tools.ComputeCameraPositionUsingMullerKillian(validatedList_abc.ElementAt(i
), this.P1, this.P2, this.P3);
                Tools.PrintVectorDebug(ref cameraPos, "camera Position in
object space");
                bool sss = Tools.InDegenerateCase(cameraPos, this.P1,
this.P2, this.P3, validatedList_abc.ElementAt(i),
                                                                this.cosTbc,
this.cosTac, this.cosTab);
                if (!sss)
                    cameraPositionList.Add(cameraPos);
                double[,] Rot = Tools.ComputeRotationMatrix(cameraPos,
this.P1, this.P2, this.P3, this.p1, this.p2, this.p3);
            }

        }

        #endregion

        #region Private Methods for Computing G4, G3, G2, G1
        double Estimate_G4(double k1, double k2, double cbc)
        {
            return (Math.Pow(k1 * k2 - k1 - k2, 2) - (4 * k1 * k2 * cbc *
cbc));
```

```
        }

        double Estimate_G3(double k1, double k2, double cab, double cac,
double cbc)
        {
            double t1 = 4 * (k1 * k2 - k1 - k2) * k2 * (1 - k1) * cab;
            double t2 = 4 * k1 * cbc * ((k1 * k2 + k2 - k1) * cac + 2 * k2
* cab * cbc);

            return (t1 + t2);
        }

        double Estimate_G2(double k1, double k2, double cab, double cac,
double cbc)
        {
            double t1 = Math.Pow(2 * k2 * (1 - k1) * cab, 2);
            double t2 = 2 * (k1 * k2 + k1 - k2) * (k1 * k2 - k1 - k2);
            double t3 = 4 * k1 * ((k1 - k2) * (cbc * cbc) + (1 - k2) * k1 *
(cac * cac) - 2 * k2 * (1 + k1) * cab * cac * cbc);

            return (t1 + t2 + t3);
        }

        double Estimate_G1(double k1, double k2, double cab, double cac,
double cbc)
        {
            return (4 * (k1 * k2 + k1 - k2) * k2 * (1 - k1) * cab
                + 4 * k1 * ((k1 * k2 - k1 + k2) * cac * cbc + 2 * k1 * k2 *
cab * (cac * cac)));
        }

        double Estimate_G0(double k1, double k2, double cac)
        {
            return (Math.Pow(k1 * k2 + k1 - k2, 2) - 4 * (k1 * k1) * k2 *
(cac * cac));
        }
        #endregion

        #region Private Methods for computing legs a b c
        double Leg_a(double x)
        {
            return (Rab / (Math.Sqrt(x * x - 2 * x * cosTab + 1)));
        }

        double Leg_b(double x, double a)
        {
            return a * x;
        }

        double[] Est_y(double a)
        {
            double root = Math.Sqrt(cosTac * cosTac + ((Rac * Rac - a * a)
/ (a * a)));
            List<double> lst = new List<double>();
            double yplus = cosTac + root;
            if (yplus > 0.0)
                lst.Add(yplus);
            double yminus = cosTac - root;
            if (yminus > 0.0)
                lst.Add(yminus);
```

```
            double[] t = lst.ToArray();
            return t;
        }

        double[] Leg_c(double[] y, double a)
        {
            List<double> lst = new List<double>();
            foreach (double yy in y)
            {
                lst.Add(yy * a);
            }

            return lst.ToArray();
        }

        bool Validate_abc(double[] abc)
        {
            double legb = abc[1];
            double legc = abc[2];
            double rhs = legb * legb + legc * legc - 2 * legb * legc *
cosTbc;
            double delta = Math.Abs(Rbc * Rbc - rhs);
            Trace.WriteLine(String.Format("{0} \t {1} \t {2} \t {3}",
abc[0], legb, legc, delta));
            if (delta < 0.001)
                return true;
            else
                return false;
        }

        #endregion


    }
}
```

Lampiran B

Metode *Church*

B.1 Penentuan Koordinat Kamera

1. Data input yaitu meliputi data koordinat foto (x,y), koordinat objek (X_i,Y_i,Z_i), panjang fokus (f) dan nilai pendekatan parameter koordinat kamera (X_L,Y_L,Z_L).

ID	Koordinat Objek (m)			Koordinat Foto (mm)		fokus (mm)
	X	Y	Z	x	y	
1	-0.101356	-0.068686	-0.661268	3.15447	2.41421	-24.00
2	0.089863	-0.071431	-0.667876	-2.84669	2.43875	
3	0.088922	0.085207	-0.778067	-3.36471	-1.50055	
4	-0.102298	0.087524	-0.771782	4.12840	-1.49685	

ID	X_L	Y_L	Z_L
Sol 1	0.03701	0.69539	-0.71768
Sol 2	0.21605	-0.17426	0.03567

2. Dari data input koordinat foto yang diketahui maka dapat dihasilkan nilai jarak (d_i) (lihat persamaan 2.47) dan nilai parameter dari arah *cosinus* (l,m,n) (lihat persamaan 2.48) serta nilai sudut *cosinus* (k) (lihat persamaan 2.49), sehingga dapat diperoleh :

ID	Parameter				
	d	l	m	n	k
Sol 1	24.32651	0.12967	0.09924	-0.98658	0.96953
	24.29097	-0.11719	0.10040	-0.98802	0.98662
	24.28112	-0.13857	-0.06180	-0.98842	0.95105
Sol 2	24.3265	0.1297	0.0992	-0.9866	0.9695
	24.2910	-0.1172	0.1004	-0.9880	0.9866
	24.2811	-0.1386	-0.0618	-0.9884	0.9511

3. Dari data input koordinat objek yang diketahui maka dapat dihasilkan nilai jarak (D_i) (lihat persamaan 2.50) dan nilai parameter dari arah *cosinus* (L,M,N) (lihat persamaan 2.51) serta nilai sudut *cosinus* (K) (lihat persamaan 2.52) yaitu :

ID	Parameter				
	D	L	M	N	K
Sol 1	0.77855	-0.17772	-0.98141	0.07246	0.96953
	0.77025	0.06862	-0.99555	0.06466	0.98662
	0.61536	0.08436	-0.99159	-0.09813	0.95105
Sol 2	0.77306	-0.41059	0.13657	-0.90154	0.96953
	0.72213	-0.17474	0.14240	-0.97426	0.98662
	0.86351	-0.14722	0.30048	-0.94236	0.95105

4. Kemudian mencari nilai parameter pelengkap (I dan J) (lihat persamaan 2.53), nilai koefisien (A,B,C) (lihat persamaan 2.54) dan nilai selisih dari sudut *cosinus* (ΔK) (lihat persamaan 2.55).

ID	Parameter					
	I	J	A	B	C	ΔK
Sol 1	-0.05298	-0.02573	0.00765	0.07760	-0.00550	0.00000
	-0.34417	0.30505	0.00212	0.04015	-0.05219	0.00000
	0.26109	-0.40350	0.09374	0.13710	-0.05486	0.00000
Sol 2	-0.13064	0.04902	0.04507	-0.01086	0.07001	0.00000
	0.20820	-0.24223	-0.00072	-0.04314	0.02543	0.00000
	-0.19219	0.07219	-0.00135	-0.04789	0.11603	0.00000

5. Untuk memperoleh nilai koreksi dari koordinat kamera diperlukan nilai parameter pelengkap untuk persamaan secara serentak, sehingga dapat dihasilkan nilai parameter (E,F,G) (lihat persamaan 2.56) dan nilai dari jumlah perkalian antara parameter E dan C (lihat persamaan 2.57).

ID	Parameter			
	E	F	G	Δ
Sol 1	0.00014	-0.00039	-0.00383	-0.00031
	-0.00347	0.00478	0.00495	
	0.00623	-0.00010	0.00350	
Sol 2	-0.00195	0.00120	0.00274	-0.00017
	-0.00002	-0.00005	-0.00379	
	0.00217	-0.00532	-0.00209	

6. Dari beberapa parameter pada proses perhitungan sebelumnya maka dapat dicari nilai koreksi untuk parameter koordinat kamera (lihat persamaan 2.58), sehingga dapat diperloeh hasil sebagai berikut :

Parameter Koreksi Koordinat Kamera	
$\Delta X_1 =$	0.000000
$\Delta Y_1 =$	-0.000005
$\Delta Z_1 =$	-0.000004
$\Delta X_2 =$	0.000002
$\Delta Y_2 =$	-0.000003
$\Delta Z_2 =$	0.000000

7. Setelah diperoleh nilai koreksi parameter koordinat kamera maka dapat diperoleh nilai pendekatan *update* parameter koordinat kamera seperti dibawah ini :

Parameter Pendekatan Awal *Update* Koordinat Kamera	
$X_{L1} =$	0.037010
$Y_{L1} =$	0.695385
$Z_{L1} =$	-0.717684
$X_{L2} =$	0.21605
$Y_{L2} =$	-0.17426
$Z_{L2} =$	0.03567

B.2 Matriks Rotasi

1. Dengan menggunakan nilai parameter dari arah *cosinus* (*l,m,n*) yang telah diketahui, maka dapat ditentukan nilai parameter (*e,f,g*) (lihat persamaan 2.60) dan nilai jumlah perkalian dari parameter e dan n (lihat persamaan 2.61), yaitu :

ID	Parameter			
	e	f	g	Δ
Sol 1	0.02465	-0.24374	0.00100	-0.03956
&	0.02115	-0.02108	-0.16029	
Sol 2	-0.00574	0.26488	0.15906	

2. Setelah itu dapat disusun matriks rotasi (lihat persamaan 2.62), yaitu :

ID	Parameter		
	u	v	w
Sol 1	-0.99803	0.03438	0.05242
	0.05127	-0.03360	0.99812
	0.03608	0.99884	0.03177
Sol 2	-0.95724	-0.04418	0.28591
	-0.02676	-0.97052	-0.23957
	0.28807	-0.23697	0.92783

```
public double[] ChurchAdjustment(double[] Initpose, double[] x, double[] y,
double[] X, double[] Y, double[] Z, double f, ref double err)
        {
            int ndata = x.Length;
            double[] pose = new double[3];

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

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


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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

                if (Matrix.Max(Matrix.Abs(cor)) < 1.0e-20)
                    break;
            }
            pose[0] = Initpose[0];
            pose[1] = Initpose[1];
            pose[2] = Initpose[2];

            return pose;
```

Lampiran C

Metode *Zeng*

C.1 Penentuan Koordinat Kamera

1. Data input yaitu meliputi data koordinat foto (x,y), koordinat objek (X_i,Y_i,Z_i), dan panjang fokus (f).

ID	Koordinat Objek (m)			Koordinat Foto (mm)		fokus (mm)
	X	Y	Z	x	y	
1	-0.101356	-0.068686	-0.661268	3.15447	2.41421	-24.00
2	0.089863	-0.071431	-0.667876	-2.84669	2.43875	
3	0.088922	0.085207	-0.778067	-3.36471	-1.50055	
4	-0.102298	0.087524	-0.771782	4.12840	-1.49685	

2. Dengan menggunakan data koordinat titik objek, maka dapat diahsiilkan nilai jarak (R_i) (lihar persamaan 2.65), sehingga dapat diperoleh hasil sebagai berikut :

Parameter R	
R_1 =	0.191516331
R_2 =	0.271165049
R_3 =	0.191352877

3. Untuk proses selanjutnya digunakan data koordinat foto dan panjang fokus untuk menghasilkan nilai konstanta (A,B,C,D,E,F) (lihat persamaan 2.67), sehingga dapat diperoleh :

Nilai Konstanta			
A =	1.0273943	D =	1.989263
B =	1.0243944	E =	2.020552
C =	1.0235641	F =	1.950568

4. Dengan menggunakan nilai nilai konstanta (A,B,C,D,E,F), dapat ditentukan nilai dari parameter (D_1,E_1,F_1) (lihat persamaan 2.68), yaitu :

Parameter D_1,E_1,F_1	
D_1 =	1.93905
E_1 =	1.97323
F_1 =	1.90211

5. Dengan asumsi dari hasil nilai jarak koordinat objek maka dapat diperoleh nilai konstanta (K) (lihat persamaan 2.70), sehingga dapat diperoleh :

Nilai Konstanta K	
K_{12} =	0.49882
K_{32} =	0.49797

6. Dengan diketahui nilai konstanta (K) dan nilai dari parameter (D_1,E_1,F_1), maka dapat diperoleh nilai konstanta N_i pada bentuk kuadrat pangkat empat (lihat persamaan 2.72), sehingga dapat diperoleh :

Nilai Konstanta N	
N_1 =	-0.94062278
N_2 =	3.678987223
N_3 =	-5.415356781
N_4 =	3.551938057
N_5 =	-0.873828864

7. Setelah nilai konstanta N_i diketahui maka nilai koefisien M_i dapat diperoleh dengan menggunakan persamaan (2.74), sehingga hasilnya dapat dilihat dibawah ini :

Nilai Koefisien M	
M_1 =	1.0000000
M_2 =	-3.9112249
M_3 =	5.7572035
M_4 =	-3.7761557
M_5 =	0.9289897

8. Dibawah ini merupakan nilai akar yang diberikan dari nilai koefisien M_i yang telah diketahui nilainya.

Root			
1	2	3	4
1.117	complex	complex	0.7904

9. Proses perhitungan selanjutnya adalah menjabarkan bentuk persamaan pangkat empat dengan menggunkan teknik aljabar seperti dibawah ini untuk mendapatkan nilai dari koordinat kamera ($X_L Y_L, Z_L$) (lihat persamaan 2.75-2.81) sehingga hasil keseluruhan dapat dilihat dibawah ini :

root 1.117	
R_1 =	0.7731
R_2 =	0.7769
R_3 =	0.8635

root complex	
R_1 =	#VALUE!
R_2 =	#VALUE!
R_3 =	#VALUE!

root complex	
R_1 =	#VALUE!
R_2 =	#VALUE!
R_3 =	#VALUE!

root 0.7904	
R_1 =	0.7786
R_2 =	0.7703
R_3 =	0.6154

Parameter Aljabar Akar Empat	
a =	0.001
b =	0.019
a_{11} =	-0.003
a_{12} =	-0.007
a_{21} =	0.157
a_{22} =	-0.110
c_1 =	-0.191
c_2 =	0.001
b_1 =	0.001
b_2 =	0.010

Parameter Aljabar Akar Empat	
a =	#VALUE!
b =	#VALUE!
a_{11} =	-0.003
a_{12} =	-0.007
a_{21} =	0.157
a_{22} =	-0.110
c_1 =	-0.191
c_2 =	0.001
b_1 =	#VALUE!
b_2 =	#VALUE!

Parameter Aljabar Akar Empat	
a =	#VALUE!
b =	#VALUE!
a_{11} =	-0.003
a_{12} =	-0.007
a_{21} =	0.157
a_{22} =	-0.110
c_1 =	-0.191
c_2 =	0.001
b_1 =	#VALUE!
b_2 =	#VALUE!

Parameter Aljabar Akar Empat	
a =	0.020
b =	0.376
a_{11} =	-0.003
a_{12} =	-0.007
a_{21} =	0.157
a_{22} =	-0.110
c_1 =	-0.191
c_2 =	0.001
b_1 =	0.010
b_2 =	0.188

e =	747.657
f =	0.000
g =	0.021
h =	0.000
i =	0.030
k =	0.452

e =	747.657
f =	#VALUE!
g =	0.021
h =	#VALUE!
i =	0.030
k =	0.452

e =	747.657
f =	#VALUE!
g =	0.021
h =	#VALUE!
i =	0.030
k =	0.452

e =	747.657
f =	0.000
g =	0.021
h =	-0.002
i =	0.030
k =	0.452

l =	750.727
m =	28.304
n =	-0.249

l =	750.727
m =	#VALUE!
n =	#VALUE!

l =	750.727
m =	#VALUE!
n =	#VALUE!

l =	750.727
m =	-33.710
n =	0.220

10. Dari proses penjabaran tersebut, maka diperoleh dua solusi untuk parameter koordinat kamera, seperti dibawah ini :

Solusi 1		Solusi 2	
$X_{cam\ 1}$ =	-0.0451	$X_{cam\ 1}$ =	0.037
$Y_{cam\ 1}$ =	0.0000	$Y_{cam\ 1}$ =	0.6955
$Z_{cam\ 1}$ =	0.0000	$Z_{cam\ 1}$ =	-0.7175
$X_{cam\ 2}$ =	0.0074	$X_{cam\ 2}$ =	0.0079
$Y_{cam\ 2}$ =	0.0000	$Y_{cam\ 2}$ =	0.2370
$Z_{cam\ 2}$ =	0.0000	$Z_{cam\ 2}$ =	-1.3691

11. Dengan koreksi arah vektor (T) diperoleh dua nilai positif yaitu pada solusi 2 dan untuk koreksi kemungkinan koordinat posisi kamera diperoleh :

$X_{cam\ 2}$ =	0.0370
$Y_{cam\ 2}$ =	0.6955
$Z_{cam\ 2}$ =	-0.7175

	-0.1777	
al =	-0.9814	1
	0.0722	0.7787

	0.0686	
bl =	-0.9956	1
	0.0644	0.7704

	0.1297	
ap =	0.0992	1
	-0.9866	24.3265

	-0.1172	
bp =	0.1004	1
	-0.9880	24.2910

p =	0.0230
q =	0.5202
r =	0.5376

g x ap =	-0.5666
	-0.0925
	-0.0652

g x al =	0.5652
	0.0972
	0.0698

gap.gal =	-0.3338
lg*apl =	0.5778
lg*all =	0.5778

cos γ	-0.999931

d =	0.005856
a =	0.030795
b =	0.695007
c =	0.718320

C.2 Matriks Rotasi

$$R = \begin{vmatrix} -0.9980 & 0.0512 & 0.0361 \\ 0.0344 & -0.0339 & 0.9988 \\ 0.0524 & 0.9981 & 0.0320 \end{vmatrix}$$

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

namespace CFResec.Zeng
{
    class Zeng
    {
        double f;
        //double[,] objectSpacePoints = new double[4, 3];
        //double[,] imageSpacePoints = new double[4, 2];
        //Object space points
        double X1; double Y1; double Z1;
        double X2; double Y2; double Z2;
        double X3; double Y3; double Z3;
        double X4; double Y4; double Z4;
        //image space points
        double x1; double y1; double z1;
        double x2; double y2; double z2;
        double x3; double y3; double z3;
        double x4; double y4; double z4;

        public Zeng(ref double focalLength, ref double[,] objectpoints, ref
double[,] imagepoints)
        {
            f = focalLength;
            //Trace.WriteLine("f: " + f.ToString());
            //objectSpacePoints = objectpoints;
            #region Object Space Points of P1,P2,P3 (selected control
points) & P4
            //Point 1
            this.X1 = objectpoints[0, 0];
            this.Y1 = objectpoints[0, 1];
            this.Z1 = objectpoints[0, 2];
            //Point 2
            this.X2 = objectpoints[1, 0];
            this.Y2 = objectpoints[1, 1];
            this.Z2 = objectpoints[1, 2];
            //Point 3
            this.X3 = objectpoints[2, 0];
            this.Y3 = objectpoints[2, 1];
            this.Z3 = objectpoints[2, 2];
            //Point 4
            this.X4 = objectpoints[3, 0];
            this.Y4 = objectpoints[3, 1];
            this.Z4 = objectpoints[3, 2];
            #endregion

            #region Image space Points of p1, p2, p3 (selected image
points) & p4
            //Point 1
            this.x1 = imagepoints[0, 0];
            this.y1 = imagepoints[0, 1];
            this.z1 = -f;
            //Point 2
            this.x2 = imagepoints[1, 0];
            this.y2 = imagepoints[1, 1];
```

```
            this.z2 = -f;
            //Point 3
            this.x3 = imagepoints[2, 0];
            this.y3 = imagepoints[2, 1];
            this.z3 = -f;
            //Point 4
            this.x4 = imagepoints[3, 0];
            this.y4 = imagepoints[3, 1];
            this.z4 = -f;
            #endregion

            //Tools.Multiply(ref objectSpacePoints, 1000);   //convert from
Metre to mm
            //Tools.PrintMatrixDebug(ref objectSpacePoints, "objectSpcePt
Zeng");
            //imageSpacePoints = imagepoints;

            //Tools.PrintMatrixDebug(ref imageSpacePoints,
"imageSpacePoints in Zeng");
        }

        public void ComputeClosedFormSolution()
        {
            #region OK
            ///Get squared of L1^2, L2^2, L3^2 from Eq.2 - Eq.4
            double[] L = Tools.SquaredDistances(new double[3] { X1, Y1, Z1
}, new double[3] { X2, Y2, Z2 }, new double[3] { X3, Y3, Z3 });
            //Tools.PrintVectorDebug(ref L, "Squared L1^2 L2^2 L3^2");

            ///Get ABCDEF from Eq.8
            double[] A_F = ComputeABCDEF();
            //Tools.PrintVectorDebug(ref A_F, "ABCDEF");

            ///Calculate Eq. 10, output: D1, E1, F1
            double[] Eq10 = ComputeEquation10(A_F);
            //Tools.PrintVectorDebug(ref Eq10, "D1 E1 F1");

            ///Get K12 K32, output: K12, K32
            double[] Ks = ComputeK12K32(L);
            //Tools.PrintVectorDebug(ref Ks, "K12 K32");

            ///Get N1, N2, N3, N4, N5 from Eq.21-26
            double[] N1_5 = ComputeN1_5(Eq10, Ks);
            //Tools.PrintVectorDebug(ref N1_5, "N1, N2, N3, N4, N5");

            ///Get M1, M2, M3, M4, M5 from Eq.27
            double[] M1_5 = ComputeM1_5(N1_5);
            //Tools.PrintVectorDebug(ref M1_5, "M1, M2, M3, M4, M5");

            ///Compute real positif polynomial roots of n (Eq.27)
            double[] roots = ExtractRealPositivePolynomialRoots(M1_5);
            Tools.PrintVectorDebug(ref roots, "roots of n");

            ///Compute Eq. 28 & 29: R1 from each pos root
            ///Output: [,]-->first row: R1; cols = number of R1
            ///--> second row: R3; cols = number of R3
            double[,] R1R3s = ComputeR1R3(roots, Math.Sqrt(L[1]), Eq10[2]);
            Tools.PrintMatrixDebug(ref R1R3s, "R1R3s");

            ///Compute Eq 30: R2s
```

```
            double[,] R2sEq30 = ComputeR2sEq30(R1R3s, Eq10[0], L[2]);

            ///Compute Eq 31: R2s
            double[,] R2sEq31 = ComputeR2sEq31(R1R3s, Eq10[1], L[0]);

            ///Compute the Final value of R2
            double[] R2 = GetFinalR2(R2sEq30, R2sEq31);
            Tools.PrintVectorDebug(ref R2, "Final R2s\n");

            ///Xs Ys Zs are Perspective centre coord in object space
            ///stored in XYZcamera
            List<double[]> XYZcamera =
EliminateEq33_37toFindSetOfXsYsZs(R1R3s, R2);
            int index = 0;
            foreach (double[] val in XYZcamera)
            {
                string txt = String.Format("XsYsZs in Object Space {0}: {1}
{2}  {3}", index++, val[0], val[1], val[2]);
                Trace.WriteLine(txt);
            }

            ///Compute T
            List<double[]> correctedXsYsZs =
RectifyXsYsZsBasedOnT(XYZcamera);  //page 330
            Trace.WriteLine("\nCorrected Xs Ys Zs in Object Space:");
            foreach (double[] d in correctedXsYsZs)
            {
                string txt = String.Format("XsYsZs : {0}  {1}  {2}", d[0],
d[1], d[2]);
                Trace.WriteLine(txt);
            }

            #endregion

            double[,] Rt = FindRotationAxis(correctedXsYsZs);

            Tools.PrintMatrixDebug(ref Rt, "-----------------Final Rt------
----------------");
        }



        #region Ok
        /// <summary>
        /// Compute T (page 330): if T>0, the exposure station is
determined correctly
        /// </summary>
        /// <param name="XYZcamera"></param>
        /// <returns>one to four solution of exposure stations are
determined</returns>
        private List<double[]> RectifyXsYsZsBasedOnT(List<double[]>
XYZcamera)
        {
            double[] P1 = new double[3] { X1, Y1, Z1 };
            double[] P2 = new double[3] { X2, Y2, Z2 };
            double[] P3 = new double[3] { X3, Y3, Z3 };
            List<double[]> list = new List<double[]>();

            foreach (double[] v in XYZcamera)
            {
```

```
                double Xs = v[0]; double Ys = v[1]; double Zs = v[2];
                double[] _1S = new double[3] { Xs - P1[0], Ys - P1[1], Zs -
P1[2] };
                double[] _12 = new double[3] { P2[0] - P1[0], P2[1] -
P1[1], P2[2] - P1[2] };
                double[] _13 = new double[3] { P3[0] - P1[0], P3[1] -
P1[1], P3[2] - P1[2] };

                double[] c = Tools.CrossProduct(_12, _13);
                double T = Tools.DotProduct(_1S, c);
                if (T > 0.0)
                {
                    double[] val = new double[3] { Xs, Ys, Zs };
                    list.Add(val);
                }
            }
            return list;
        }

        /// <summary>
        /// find all posible camera positions in terms of object space: Xs
Ys Zs
        /// </summary>
        /// <param name="R1R3s">a set of R1 & R3</param>
        /// <param name="R2">fix R2</param>
        /// <returns>a set of Xs Ys Zs control points in object space
System</returns>
        private List<double[]> EliminateEq33_37toFindSetOfXsYsZs(double[,]
R1R3s, double[] R2)
        {
            //the first root (^0) col 0-th
            int rows = R1R3s.GetUpperBound(0);
            Debug.Assert(rows + 1 == 2);
            int cols = R1R3s.GetUpperBound(1);
            List<double[]> list = new List<double[]>();

            //Iterate computation based on the numbers of R2
            int countR2 = R2.Length;
            for (int kk = 0; kk < countR2; kk++)
            {
                double r2 = R2[kk];
                for (int col = 0; col <= cols; col++)
                {
                    double r1 = R1R3s[0, col];
                    double r3 = R1R3s[1, col];

                    double a = r1 * r1 - r2 * r2 - X1 * X1 + X2 * X2 - Y1 *
Y1 + Y2 * Y2 - Z1 * Z1 + Z2 * Z2;
                    double b = r2 * r2 - r3 * r3 - X2 * X2 + X3 * X3 - Y2 *
Y2 + Y3 * Y3 - Z2 * Z2 + Z3 * Z3;

                    double a11 = Y2 - Y1; double a12 = Z2 - Z1;
                    double a21 = Y3 - Y2; double a22 = Z3 - Z2;
                    double c1 = -(X2 - X1); double c2 = -(X3 - X2);
                    double b1 = 0.5 * a; double b2 = 0.5 * b;

                    double e = 1.0 / (a11 * a22 - a12 * a21); double f =
a22 * b1 - a12 * b2; double g = a22 * c1 - a12 * c2;
                    double h = a11 * b2 - a21 * b1;
                    double i = a11 * c2 - a21 * c1;
```

```
                    double k = X1 * X1 + Y1 * Y1 + Z1 * Z1;

                    double l = e * e * i * i + e * e * g * g + 1;
                    double m = 2 * (e * e * f * g + e * e * h * i - X1 - e
* g * Y1 - e * i * Z1);
                    double n = k - r1 * r1 - 2 * Y1 * e * f + e * e * f * f
- 2 * Z1 * e * h + e * e * h * h;

                    ///find the squared roots
                    double[] coef = new double[3] { n, m, l };
                    PolyLib.Polynomial p2 = new PolyLib.Polynomial(coef);
                    PolyLib.Complex[] roots = p2.Roots();
                    double[] posRoots = Tools.FindPositiveRoots(roots);
//two values of X for each set of R1R3
                    //Tools.PrintVectorDebug(ref posRoots, "posRoots of
Xs");
                    if (posRoots.Length != 0) //avoid an empty roots
                    {
                        //Trace.WriteLine("upperbound of posRoots: " +
posRoots.GetUpperBound(0).ToString());
                        double Xs1 = posRoots[0];
                        double Xs2 = posRoots[1];
                        //find Y & Z for each set of X
                        double Ys1 = e * (f + g * Xs1);
                        double Ys2 = e * (f + g * Xs2);
                        double Zs1 = e * (h + i * Xs1);
                        double Zs2 = e * (h + i * Xs2);
                        double[] XsYsZs1 = new double[3] { Xs1, Ys1, Zs1 };
                        double[] XsYsZs2 = new double[3] { Xs2, Ys2, Zs2 };
                        list.Add(XsYsZs1);
                        list.Add(XsYsZs2);
                    }

                }
            }
            return list;
        }

        /// <summary>
        /// Compute A, B, C, D, E, F from Eq.8 page 328
        /// </summary>
        /// <returns>vecotr of ABCDEF</returns>
        private double[] ComputeABCDEF()
        {
            //double x1 = imageSpacePoints[0, 0]; double y1 =
imageSpacePoints[0, 1];
            //double x2 = imageSpacePoints[1, 0]; double y2 =
imageSpacePoints[1, 1];
            //double x3 = imageSpacePoints[2, 0]; double y3 =
imageSpacePoints[2, 1];

            double A = (x1 * x1 + y1 * y1 + f * f) / (f * f);
            double B = (x2 * x2 + y2 * y2 + f * f) / (f * f);
            double C = (x3 * x3 + y3 * y3 + f * f) / (f * f);

            double D = (2 * x1 * x2 + 2 * y1 * y2 + 2 * f * f) / (f * f);
            double E = (2 * x2 * x3 + 2 * y2 * y3 + 2 * f * f) / (f * f);
            double F = (2 * x3 * x1 + 2 * y3 * y1 + 2 * f * f) / (f * f);

            double[] A_F = new double[6] { A, B, C, D, E, F };
```

```
            return A_F;
        }

        /// <summary>
        /// Compute D1, E1, F1 from Eq.10
        /// </summary>
        /// <param name="A_F">ABCDEF fromEq.8</param>
        /// <returns>vector of D1,E1,E1</returns>
        private double[] ComputeEquation10(double[] A_F)
        {
            double D1 = A_F[3] / Math.Sqrt(A_F[0] * A_F[1]);
            double E1 = A_F[4] / Math.Sqrt(A_F[1] * A_F[2]);
            double F1 = A_F[5] / Math.Sqrt(A_F[2] * A_F[0]);

            double[] r = new double[3] { D1, E1, F1 };
            return r;
        }

        /// <summary>
        /// Compute K12 & K32
        /// </summary>
        /// <param name="L">vector of L1^2 L2^2 L3^2</param>
        /// <returns>vector of K12 K32</returns>
        private double[] ComputeK12K32(double[] L)
        {
            double L1 = Math.Sqrt(L[0]);
            double L2 = Math.Sqrt(L[1]);
            double L3 = Math.Sqrt(L[2]);
            double K12 = Math.Pow(L1 / L2, 2.0);
            double K32 = Math.Pow(L3 / L2, 2.0);
            double[] r = new double[2] { K12, K32 };
            return r;
        }

        /// <summary>
        /// Compute N1 N2 N3 N4 N5
        /// </summary>
        /// <param name="Eq10">D1 E1 F1</param>
        /// <param name="Ks">K12 K32</param>
        /// <returns>VECTOR OF N1, N2, N3, N4, N5</returns>
        private double[] ComputeN1_5(double[] Eq10, double[] Ks)
        {
            double K12 = Ks[0];
            double K32 = Ks[1];
            double D1 = Eq10[0];
            double E1 = Eq10[1];
            double F1 = Eq10[2];

            double N1 = (1 - K12) * (1 - K12) + K32 * K32 - 2 * (1 - K12) *
K32 - K32 * E1 * E1 + 4 * K32 * (1 - K12);
            double N2 = K32 * F1 * (-2 - 2 * K32 + 4 * K12 + E1 * E1) + K12
* (E1 * D1 + 2 * F1 - 2 * K12 * F1) + E1 * D1 * (K32 - 1);
            double N3 = K32 * (K32 * F1 * F1 - F1 * D1 * E1 - 4 * K12 + 2 *
K32 - E1 * E1 - 2 * K12 * F1 * F1) +
                        K12 * (K12 * F1 * F1 + 2 * K12 - F1 * E1 * D1 - D1
* D1) + D1 * D1 - 2 + E1 * E1;
            double N4 = K32 * (2 * F1 - 2 * K32 * F1 + E1 * D1 + 4 * K12 *
F1) +
                        K12 * (E1 * D1 - 2 * K12 * F1 + F1 * D1 * D1 - 2 *
F1) - D1 * E1;
```

```
            double N5 = K32 * (K32 - 2 - 2 * K12) + K12 * (K12 - D1 * D1 +
2) + 1;

            double[] N1_5 = new double[5] { N1, N2, N3, N4, N5 };
            return N1_5;
        }

        /// <summary>
        /// Compute M1 M2 M3 M4 M5
        /// </summary>
        /// <param name="N1_5">vector of N1 N2 N3 N4 N5</param>
        /// <returns>vector of M1(coef degree 4) M2 M3 M4 M5(coef degree
0)</returns>
        private double[] ComputeM1_5(double[] N1_5)
        {
            double N1 = N1_5[0];
            double N2 = N1_5[1];
            double N3 = N1_5[2];
            double N4 = N1_5[3];
            double N5 = N1_5[4];

            double M1 = 1.0;
            double M2 = N2 / N1;
            double M3 = N3 / N1;
            double M4 = N4 / N1;
            double M5 = N5 / N1;

            double[] M = new double[5] { M1, M2, M3, M4, M5 };
            return M;
        }

        /// <summary>
        /// Compute positive roots of 4 degrees polynomial
        /// </summary>
        /// <param name="M1_5">M1(^4) M2(^3) M3(^2) M4(^1) M5(^0)</param>
        /// <returns>vector of positive roots</returns>
        private double[] ExtractRealPositivePolynomialRoots(double[] M1_5)
        {
            int count = M1_5.Length;
            double[] vals = new double[count];
            for (int i = 0; i < count; i++)
            {
                vals[count - 1 - i] = M1_5[i];
            }
            //Tools.PrintVectorDebug(ref vals, "vals");

            PolyLib.Polynomial p = new PolyLib.Polynomial(vals);
            PolyLib.Complex[] roots = p.Roots();
            //for (int i = 0; i < roots.Length; i++)
            //    Trace.WriteLine("real: " + roots[i].Re.ToString() + "
Imaginary: " + roots[i].Im.ToString());
            double[] posRoots = Tools.FindPositiveRoots(roots);
            return posRoots;
        }

        /// <summary>
        /// compute a set of a pair R1s-R3s
        /// </summary>
        /// <param name="roots">pos val polynomial roots</param>
        /// <param name="L2">L2</param>
```

```
        /// <param name="F1">F1</param>
        /// <returns>R1s: first row;  R3s: 2nd row</returns>
        private double[,] ComputeR1R3(double[] roots, double L2, double F1)
        {
            int count = roots.Length;
            double[,] r1s = new double[2, count];
            for (int i = 0; i < count; i++)
            {
                r1s[0, i] = L2 / Math.Sqrt(roots[i] * roots[i] - F1 *
roots[i] + 1);    //R1s
                r1s[1, i] = roots[i] * r1s[0, i];   //R3s
            }

            return r1s;
        }

        /// <summary>
        /// R2 from Eq 30
        /// </summary>
        /// <param name="r1r3">a pair of R1-R3</param>
        /// <param name="D1">D1 from Eq.10</param>
        /// <param name="L3">squared Distance L3^2</param>
        /// <returns>R2 Accord. to Eq.30 page 329</returns>
        private double[,] ComputeR2sEq30(double[,] r1r3, double D1, double
L3)
        {
            int count = r1r3.GetUpperBound(1) + 1;
            double[,] r2s = new double[count, 2];
            Trace.WriteLine("\nInside ComputeR2sEq30:");
            for (int i = 0; i < count; i++) //rows = the number of column
of r1r3's pair
            {
                double DR = D1 * r1r3[0, i];
                double root = Math.Sqrt(D1 * D1 * r1r3[0, i] * r1r3[0, i] -
4 * (r1r3[0, i] * r1r3[0, i] - L3));
                double a = (DR + root) / 2;
                double b = (DR - root) / 2;
                Trace.WriteLine(a.ToString() + "    " + b.ToString());
                r2s[i, 0] = a; r2s[i, 1] = b;
            }
            return r2s;
        }

        /// <summary>
        /// R2 from Eq 31
        /// </summary>
        /// <param name="r1r3">a pair of R1-R3</param>
        /// <param name="E1">E1 from Eq.10</param>
        /// <param name="L1">squared Distance L1^2</param>
        /// <returns>R2 accord. to Eq.31 page 329</returns>
        private double[,] ComputeR2sEq31(double[,] r1r3, double E1, double
L1)
        {
            int count = r1r3.GetUpperBound(1) + 1;
            double[,] r2s = new double[count, 2];
            Trace.WriteLine("\nInside ComputeR2sEq31:");
            for (int i = 0; i < count; i++)
            {
                double ER = E1 * r1r3[1, i];
```

```
                double root = Math.Sqrt(E1 * E1 * r1r3[1, i] * r1r3[1, i] -
 4 * (r1r3[1, i] * r1r3[1, i] - L1));
                double a = (ER + root) / 2;
                double b = (ER - root) / 2;
                Trace.WriteLine(a.ToString() + "    " + b.ToString());
                r2s[i, 0] = a; r2s[i, 1] = b;
            }
            return r2s;
        }

        /// <summary>
        /// Select R2s from Eq.30 & Eq.31 which are equal
        /// </summary>
        /// <param name="R2sEq30">R2s from Eq.310</param>
        /// <param name="R2sEq31">R2s from Eq.31</param>
        /// <returns>a set of final r2s</returns>
        private double[] GetFinalR2(double[,] R2sEq30, double[,] R2sEq31)
        {
            Debug.Assert(R2sEq30.GetUpperBound(0) ==
R2sEq31.GetUpperBound(0));
            Debug.Assert(R2sEq30.GetUpperBound(1) ==
R2sEq31.GetUpperBound(1));
            List<double> list = new List<double>();
            for (int i = 0; i <= R2sEq30.GetUpperBound(1); i++)
            {
                for (int j = 0; j <= R2sEq30.GetUpperBound(0); j++)
                {
                    if (Math.Abs(R2sEq30[j, i] - R2sEq31[j, i]) <= 1e-5)
                    {
                        double val = (R2sEq30[j, i] + R2sEq31[j, i]) / 2.0;
                        list.Add(val);
                        Trace.WriteLine("val of R2: " + val.ToString());
                    }
                }
            }

            //list.Reverse(0, list.Count);
            double[] result = list.ToArray();
            //Tools.PrintVectorDebug(ref result, "result of R2");
            return result;
        }

        /// <summary>
        /// finding the final rotation axis (R) & XsYsZs (t)from the
probabilities of combination
        ///  two control points: P1P2 are used from Eq41-Eq45
        /// </summary>
        /// <param name="correctedXsYsZs">a set of camera centre
coordinates in object space system</param>
        /// <returns>a final set of rotation matrix (R) and position of the
camera (t)</returns>
        private double[,] FindRotationAxis(List<double[]> correctedXsYsZs)
        {
            Dictionary<double, double[,]> RotT = new Dictionary<double,
double[,]>();
            foreach (double[] xyz in correctedXsYsZs)
            {
                #region Image Space vectors of P1 P2: SP1 SP2
                ///P1 P2
                double Xs = xyz[0]; double Ys = xyz[1]; double Zs = xyz[2];
```

```
                double[] XsYsZs = xyz;
                double[] SP1 = new double[3] { X1 - Xs, Y1 - Ys, Z1 - Zs };
                double[] SP2 = new double[3] { X2 - Xs, Y2 - Ys, Z2 - Zs };
                #endregion

                //seek solution of combination of two points: P1P2
                #region Unit Vectors
                //Compute unit vectors of each object space point: P1, P2
                double[] uvP1 = Tools.CalculateUnitVector(SP1); //unit
vector of P1 in camera system
                double[] uvP2 = Tools.CalculateUnitVector(SP2); //unit
vector of P2 in camera system
                //double[] uvP3 = Tools.CalculateUnitVector(new double[3] {
X3, Y3, Z3 }); //unit vector of P3
                //double[] uvP4 = Tools.CalculateUnitVector(new double[3] {
X4, Y4, Z4 }); //unit vector of P3

                ///Calculate the unit vectors of p1, p2, p3 of image
coord/camera system
                ///z=-f
                double[] uvp1 = Tools.CalculateUnitVector(new double[3] {
x1, y1, z1 });
                double[] uvp2 = Tools.CalculateUnitVector(new double[3] {
x2, y2, z2 });
                //double[] uvp3 = Tools.CalculateUnitVector(new double[3] {
x3, y3, z3 });
                //double[] uvp4 = Tools.CalculateUnitVector(new double[3] {
x4, y4, z4 });
                #endregion

                #region Computations of a b c d
                //Find solution in three combination: solution 1: P1P2
                double[] al = uvP1; double[] bl = uvP2;     //P1 & P2
object space in camera system
                double[] ap = uvp1; double[] bp = uvp2;     //p1 & p2 image
space
                //Tools.PrintVectorDebug(ref al, "al");
                //Tools.PrintVectorDebug(ref bl, "bl");
                //Tools.PrintVectorDebug(ref ap, "ap");
                //Tools.PrintVectorDebug(ref bp, "bp");

                //Find d, a, b, c
                double[] abcd = Find_a_b_c_d(al, bl, ap, bp);
                //Trace.WriteLine("P1P2's   a b c d:");
                //Trace.WriteLine(abcd[0].ToString() + "  " +
abcd[1].ToString() + "  " + abcd[2].ToString() + "  " +
abcd[3].ToString());
                //double sum_abcd = abcd[0] * abcd[0] + abcd[1] * abcd[1] +
abcd[2] * abcd[2] + abcd[3] * abcd[3];
                //Trace.WriteLine("Check abcd sumSquared: " +
sum_abcd.ToString());
                double[,] rotP1P2 = ComposeRotMatrix(abcd);
                Tools.PrintMatrixDebug(ref rotP1P2, "Rot mat");

                ///Transform object space P1 P2 P3 P4 to camera system
                //double[] tP1 = Multiply(rotP1P2, new double[3] { X1, Y1,
Z1 });
                //tP1 = NormalisedX(tP1);
                #endregion
```

```
                #region Align direction of vectors in camera system to
object space
                ///Align direction of vector in camera system to object
space
                double[] rp1 = Multiply(Transpose(rotP1P2), new double[3] {
x1, y1, z1 });
                double[] rp2 = Multiply(Transpose(rotP1P2), new double[3] {
x2, y2, z2 });
                double[] rp3 = Multiply(Transpose(rotP1P2), new double[3] {
x3, y3, z3 });
                double[] rp4 = Multiply(Transpose(rotP1P2), new double[3] {
x4, y4, z4 });
                //Tools.PrintVectorDebug(ref rp1, "Rotated p1 into object
space");
                //Tools.PrintVectorDebug(ref rp2, "Rotated p2 into object
space");
                //Tools.PrintVectorDebug(ref rp3, "Rotated p3 into object
space");
                //Tools.PrintVectorDebug(ref rp4, "Rotated p4 into object
space");
                rp1 = NormalisedX(rp1);
                rp2 = NormalisedX(rp2);
                rp3 = NormalisedX(rp3);
                rp4 = NormalisedX(rp4);

                //Tools.PrintVectorDebug(ref rp1, "Normalised the Rotated
p1 into object space");
                //Tools.PrintVectorDebug(ref rp2, "Normalised the Rotated
p2 into object space");
                //Tools.PrintVectorDebug(ref rp3, "Normalised the Rotated
p3 into object space");
                //Tools.PrintVectorDebug(ref rp4, "Normalised the Rotated
p4 into object space");

                ///Shift the Origin of the object space system to camera
system
                double[] tP1 = Tools.Minus(new double[3] { X1, Y1, Z1 },
new double[3] { Xs, Ys, Zs });
                double[] tP2 = Tools.Minus(new double[3] { X2, Y2, Z2 },
new double[3] { Xs, Ys, Zs });
                double[] tP3 = Tools.Minus(new double[3] { X3, Y3, Z3 },
new double[3] { Xs, Ys, Zs });
                double[] tP4 = Tools.Minus(new double[3] { X4, Y4, Z4 },
new double[3] { Xs, Ys, Zs });
                tP1 = NormalisedX(tP1);
                tP2 = NormalisedX(tP2);
                tP3 = NormalisedX(tP3);
                tP4 = NormalisedX(tP4);
                //Tools.PrintVectorDebug(ref tP1, "Shifted to Camera System
of P1");
                //Tools.PrintVectorDebug(ref tP2, "Shifted to Camera System
of P2");
                //Tools.PrintVectorDebug(ref tP3, "Shifted to Camera System
of P3");
                //Tools.PrintVectorDebug(ref tP4, "Shifted to Camera System
of P4");

                ///Checking for the equality of unit vector direction
                double avgDistance = 0;
```

```
                bool isEqual = UnitVectorsOfPointsEqualityChecking(rp1,
rp2, rp3, rp4, tP1, tP2, tP3, tP4, ref avgDistance);
                Tools.PrintVectorDebug(ref XsYsZs, "Camera Position");
                Trace.WriteLine("Is The unit vector of this camera is
passed for equality checking: " + isEqual.ToString());
                Tools.PrintMatrixDebug(ref rotP1P2, "with rotationmatrix");
                Trace.WriteLine("With distance p4P4: " +
avgDistance.ToString());

                double[,] Rt = new double[3, 4];
                if (isEqual)
                {
                    Rt = AssemblyRt(rotP1P2, XsYsZs);
                    RotT.Add(avgDistance, Rt);
                }
                #region Obsolete
                ////Find solution in three combination: solution 1: P1P3
                //al = uvP1; bl = uvP3;     //P1 & P3 object space
                //ap = uvp1; bp = uvp3;     //p1 & p3 image space

                ////Find d, a, b, c
                //abcd = Find_a_b_c_d(al, bl, ap, bp);
                //Trace.WriteLine("P1P3's   a b c d:");
                //Trace.WriteLine(abcd[0].ToString() + "  " +
abcd[1].ToString() + "  " + abcd[2].ToString() + "  " +
abcd[3].ToString());
                //sum_abcd = abcd[0] * abcd[0] + abcd[1] * abcd[1] +
abcd[2] * abcd[2] + abcd[3] * abcd[3];
                //Trace.WriteLine("Check abcd sumSquared: " +
sum_abcd.ToString());

                ////Find solution in three combination: solution 1: P2P3
                //al = uvP2; bl = uvP3;     //P2 & P3 object space
                //ap = uvp2; bp = uvp3;     //p2 & p3 image space

                ////Find d, a, b, c
                //abcd = Find_a_b_c_d(al, bl, ap, bp);
                //Trace.WriteLine("P2P3's   a b c d:");
                //Trace.WriteLine(abcd[0].ToString() + "  " +
abcd[1].ToString() + "  " + abcd[2].ToString() + "  " +
abcd[3].ToString());
                //sum_abcd = abcd[0] * abcd[0] + abcd[1] * abcd[1] +
abcd[2] * abcd[2] + abcd[3] * abcd[3];
                //Trace.WriteLine("Check abcd sumSquared: " +
sum_abcd.ToString());

                #endregion
                #endregion
            }

            double[,] finalRt = new double[3, 4];
            Trace.WriteLine("Number of Rots: " + RotT.Count.ToString());
            if (RotT.Count == 1)
            {
                finalRt = RotT.ElementAt(0).Value;
            }
            else
            {
                finalRt = SelectFinalRt(RotT);
            }
```

```
            return finalRt;
        }

        private double FindcosGamma(double[] ap, double[] al, double[] g)
        {
            double[] gap = Tools.CrossProduct(g, ap);
            double[] gal = Tools.CrossProduct(g, al);
            double gapNorm = Tools.Norm2(gap);
            double galNorm = Tools.Norm2(gal);
            double gapDotgal = Tools.DotProduct(gap, gal);
            return (gapDotgal / (gapNorm * galNorm));
        }

        private double[] Find_a_b_c_d(double[] al, double[] bl, double[]
ap, double[] bp)
        {
            //Compute Eq.50 page 332
            //enum coord {x=0,y=1,z=2};
            double p = (ap[1] - al[1]) * (bp[2] - bl[2]) - (ap[2] - al[2])
* (bp[1] - bl[1]);
            double q = (ap[2] - al[2]) * (bp[0] - bl[0]) - (ap[0] - al[0])
* (bp[2] - bl[2]);
            double r = (ap[0] - al[0]) * (bp[1] - bl[1]) - (ap[1] - al[1])
* (bp[0] - bl[0]);
            //Trace.WriteLine(" p q r: " + p.ToString() + "  " +
q.ToString() + "  " + r.ToString());

            //direction numbers of the rotation axis g(XYZ) form object
space to image space
            double[] g = new double[3] { p, q, r };
            double cosGamma = FindcosGamma(ap, al, g);

            //Find d, a, b, c
            double d = Math.Sqrt((1 + cosGamma) / 2.0);
            double d2 = d * d;
            double pqr = p * p + q * q + r * r;

            double a = p * Math.Sqrt((1 - d2) / pqr);
            double b = q * Math.Sqrt((1 - d2) / pqr);
            double c = r * Math.Sqrt((1 - d2) / pqr);

            return new double[4] { a, b, c, d };
        }

        private double[,] ComposeRotMatrix(double[] abcd)
        {
            double a = abcd[0]; double b = abcd[1]; double c = abcd[2];
double d = abcd[3];
            double a2 = a * a; double b2 = b * b; double c2 = c * c; double
d2 = d * d;
            double r11 = d2 + a2 - b2 - c2; double r12 = 2 * (a * b + c *
d); double r13 = 2 * (a * c - b * d);
            double r21 = 2 * (a * b - c * d); double r22 = d2 - a2 + b2 -
c2; double r23 = 2 * (b * c + a * d);
            double r31 = 2 * (a * c + b * d); double r32 = 2 * (b * c - a *
d); double r33 = d2 - a2 - b2 + c2;

            double[,] rot = new double[3, 3]{{r11,r12,r13},
                                             {r21,r22,r23},
                                             {r31,r32,r33}};
```

```
            return rot;
        }

        /// <summary>
        /// Transform coord (xyz) to coord (x'y'z') using 3Rot3
        /// </summary>
        /// <param name="r"></param>
        /// <param name="v"></param>
        /// <returns></returns>
        private double[] TransformFrom(double[,] r, double[] v)
        {
            Debug.Assert(r.GetUpperBound(0) == r.GetUpperBound(1) ||
r.GetUpperBound(0) != 3);
            //Source coords
            double X = v[0]; double Y = v[1]; double Z = v[2];
            //Rotation matrix
            double r11 = r[0,0]; double r12 = r[0,1]; double r13 = r[0,2];
            double r21 = r[1,0]; double r22 = r[1,1]; double r23 = r[1,2];
            double r31 = r[2,0]; double r32 = r[2,1]; double r33 = r[2,2];

            double x = r11 * X + r12 * Y + r13 * Z;
            double y = r21 * X + r22 * Y + r23 * Z;
            double z = r31 * X + r32 * Y + r33 * Z;

            return new double[3] { x, y, z };
        }

        private double[,] Transpose(double[,] m)
        {
            int row = m.GetUpperBound(0);
            int col = m.GetUpperBound(1);
            double[,] r = new double[row + 1, col + 1];

            for (int i = 0; i <= row; i++)
            {
                for (int j = 0; j <= col; j++)
                {
                    r[i, j] = m[j, i];
                }
            }
            return r;

        }

        private double[] Multiply(double[,] m, double[] v)
        {
            Debug.Assert(m.GetUpperBound(1) == v.GetUpperBound(0));
            double x = m[0, 0] * v[0] + m[0, 1] * v[1] + m[0, 2] * v[2];
            double y = m[1, 0] * v[0] + m[1, 1] * v[1] + m[1, 2] * v[2];
            double z = m[2, 0] * v[0] + m[2, 1] * v[1] + m[2, 2] * v[2];

            return new double[3] { x, y, z };
        }

        private double[] NormalisedX(double[] v)
        {
            int count = v.Length;
            double x = v[0];
            double[] res = new double[count];
            for (int i = 0; i < count; i++)
```

```
                res[i] = v[i] / x;
            return res;
        }

        private bool EqualityChecking(double[] img, double[] obj, ref
double delta)
        {
            ///for all 4 homologeus points
            ///x ==1
            Debug.Assert(img.Length == obj.Length);
            double[] D = new double[img.Length];

            for (int i = 0; i < img.Length; i++)
            {
                D[i] = Math.Abs(img[i] - obj[i]);
            }

            /// compare DY and DZ only due to x=X=1
            if (D[1] < 1e-3 || D[2] < 1e-3)
            {
                delta = Math.Sqrt(D[1] * D[1] + D[2] * D[2]);
                Trace.WriteLine("delta in EqualityChecking(double[] img,
double[] obj, ref double delta): " + delta.ToString());
                return true;
            }

            return false;
        }

        private bool UnitVectorsOfPointsEqualityChecking(double[] rp1,
double[] rp2, double[] rp3, double[] rp4,
            double[] tP1, double[] tP2, double[] tP3, double[] tP4, ref
double avgDelta)
        {
            double deltap1P1 = 0; double deltap2P2 = 0; double deltap3P3 =
0; double deltap4P4 = 0;
            bool p1P1 = EqualityChecking(rp1, tP1, ref deltap1P1);
            bool p2P2 = EqualityChecking(rp2, tP2, ref deltap2P2);
            bool p3P3 = EqualityChecking(rp3, tP3, ref deltap3P3);
            bool p4P4 = EqualityChecking(rp4, tP4, ref deltap4P4);
            //avgDelta = (deltap1P1 + deltap2P2 + deltap3P3 + deltap4P4) /
4;
            avgDelta = deltap4P4;
            Trace.WriteLine("Equality checking p1P1: " + p1P1.ToString());
            Trace.WriteLine("Equality checking p2P2: " + p2P2.ToString());
            Trace.WriteLine("Equality checking p3P3: " + p3P3.ToString());
            Trace.WriteLine("Equality checking p4P4: " + p4P4.ToString());
            if (/*(p1P1 && p2P2 && p3P3) || */ p4P4)
                return true;
            else
                return false;
        }

        private double[,] AssemblyRt(double[,] R, double[] t)
        {
            double[,] m = new double[3, 4]{{R[0,0], R[0,1],R[0,2],t[0]},
                                           {R[1,0],R[1,1],R[1,2],t[1]},
                                           {R[2,0],R[2,1],R[2,2],t[2]}};
            return m;
        }
```

```
        private double[,] SelectFinalRt(Dictionary<double, double[,]> m)
        {
            double dist = 10000000;
            double[,] rt = new double[3, 4];
            foreach (KeyValuePair<double, double[,]> mat in m)
            {
                //dist = Math.Min(dist, mat.Key);
                if (mat.Key < dist)
                {
                    dist = mat.Key;
                    rt = mat.Value;
                }
            }

            return rt;

        }
        #endregion


    }
}
```