MMV2-Satellite

LaSTIG, Univ. Gustave Eiffel, IGN-ENSG

March 12-14, 2023

Pushbroom camera

Dataset

Joint aerial and satellite bundle adjustment - user's perspective

Pushbroom sensor - programmer's perspective

Physical model (rigorous)

- Image lines captured sequentially (along i coordinate)
- Each image line has its own orientation parameters
- No homogeneous parametrization across satellite

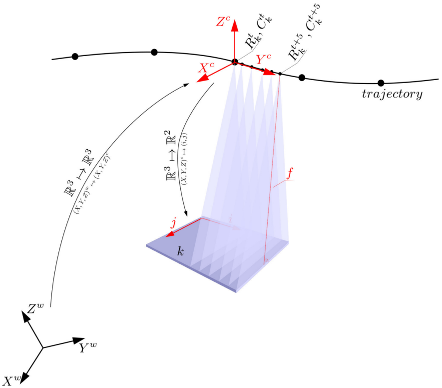


Figure: Pushbroom camera

$$\mathbf{p}_k = \mathcal{I}^t \left(\pi^t \left(\mathbf{R}_k^t \left(\mathbf{P} - \mathbf{C}_k^t\right)\right)\right)$$

Replacement models (empirical)

$$\mathbf{p}_k = \mathcal{I}^t\left(\pi^t\left(\mathbf{R}_k^t\left(\mathbf{P} - \mathbf{C}_k^t\right)\right)\right)$$

- Time-dependent collinearity replaced with a generic sensor-independent formulation,
- The physical meaning is lost

Replacement models (empirical)

$$\mathbf{p}_k = \mathcal{I}^t\left(\pi^t\left(\mathbf{R}_k^t\left(\mathbf{P} - \mathbf{C}_k^t\right)\right)\right)$$

- Time-dependent collinearity replaced with a generic sensor-independent formulation,
- The physical meaning is lost

- A few models exist:
  - grid interpolation
  - **RPC**
  - ...

Replacement models (empirical)

- Time-dependent collinearity replaced with a generic sensor-independent formulation,
- The physical meaning is lost

- A few models exist:
  - grid interpolation
  - **RPC**
  - ...
- Calculated from the physical sensor (and/or dense GCPs)

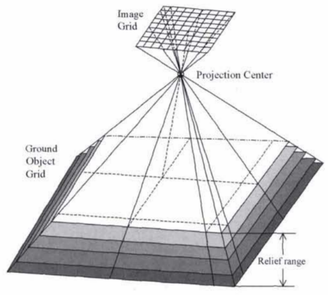


Figure: Replacement model learning data [TH01]

Rational Polynomial Coefficients RPC

Ground to image projection (inverse model)

- g, h rational polynomials

IN: φ, λ, h in geodetic coordinates

OUT: Y, X line and sample image coordinates (row,col)

$$\text{line:} \quad Y = \frac{Num_L(P, L, H)}{Den_L(P, L, H)} \sim g(\varphi, \lambda, h)$$

$$\text{sample:} \quad X = \frac{Num_S(P, L, H)}{Den_S(P, L, H)} \sim h(\varphi, \lambda, h)$$

Rational Polynomial Coefficients RPC

Ground to image projection (inverse model)

- g, h rational polynomials

IN: φ, λ, h in geodetic coordinates

OUT: Y, X line and sample image coordinates (row,col)

$$\text{line:} \quad Y = \frac{Num_L(P, L, H)}{Den_L(P, L, H)} \sim g(\varphi, \lambda, h)$$

$$\text{sample:} \quad X = \frac{Num_S(P, L, H)}{Den_S(P, L, H)} \sim h(\varphi, \lambda, h)$$

- RPC are applied on normalized coordinates:

$$\text{ground:} \quad P = \frac{\varphi - LAT_{OFF}}{LAT_{SCALE}}, \quad L = \frac{\lambda - LONG_{OFF}}{LONG_{SCALE}}, \quad H = \frac{h - HEIGHT_{OFF}}{HEIGHT_{SCALE}}$$

$$\text{image:} \quad y = Y \cdot LINE_{scale} + LINE_{OFF}, \quad x = X \cdot SAMPLE_{scale} + SAMPLE_{OFF}.$$

Rational Polynomial Coefficients RPC

Ground to image projection (inverse model)

- g, h rational polynomials

IN: φ, λ, h in geodetic coordinates

OUT: Y, X line and sample image coordinates (row,col)

$$\text{line:} \quad Y = \frac{Num_L(P, L, H)}{Den_L(P, L, H)} \sim g(\varphi, \lambda, h)$$

$$\text{sample:} \quad X = \frac{Num_S(P, L, H)}{Den_S(P, L, H)} \sim h(\varphi, \lambda, h)$$

Rational Polynomial Coefficients RPC

Ground to image projection (inverse model)

- g, h rational polynomials

IN: φ, λ, h in geodetic coordinates

OUT: Y, X line and sample image coordinates (row,col)

$$\text{line:} \quad Y = \frac{Num_L(P, L, H)}{Den_L(P, L, H)} \sim g(\varphi, \lambda, h)$$

$$\text{sample:} \quad X = \frac{Num_S(P, L, H)}{Den_S(P, L, H)} \sim h(\varphi, \lambda, h)$$

- Num, Den are 3^{rd} degree polynomials:

$$Num_L(P, L, H) = c_1 + c_2 L + c_3 P + c_4 H + c_5 LP + c_6 LH + c_7 PH + c_8 L^2 + c_9 P^2 + c_{10} H^2 + c_{11} PLH + c_{12} L^3 + c_{13} LP^2 + c_{14} LH^2 + c_{15} L^2 P + c_{16} P^3 + c_{17} PH^2 + c_{18} L^2 H + c_{19} P^2 H + c_{20} H^3 = \mathbf{c}^{\mathbf{T}} \mathbf{u}$$

$$Den_L(P, L, H) = d_1 + d_2 L + d_3 P + d_4 H + d_5 LP + d_6 LH + d_7 PH + d_8 L^2 + d_9 P^2 + d_{10} H^2 + d_{11} PLH + d_{12} L^3 + d_{13} LP^2 + d_{14} LH^2 + d_{15} L^2 P + d_{16} P^3 + d_{17} PH^2 + d_{18} L^2 H + d_{19} P^2 H + d_{20} H^3 = \mathbf{d}^{\mathbf{T}} \mathbf{u}$$

Rational Polynomial Coefficients RPC

Image to ground projection (direct model)

- k, l : rational polynomials of the inverse projection

IN: $\{y, x, h\}$ image coordinates, ellipsoidal height

OUT: $\{\varphi, \lambda\}$ geodetic coordinates

$$\varphi = k(y, x, \mathbf{h}) = \frac{Num_P(Y, X, H)}{Den_P(Y, X, H)}$$

$$\lambda = l(y, x, \mathbf{h}) = \frac{Num_{LA}(Y, X, H)}{Den_{LA}(Y, X, H)}$$

Rational Polynomial Coefficients RPC

Errors

- Inaccuracies of on-board georeferencing devices (GPS, star trackers, ...)

Rational Polynomial Coefficients RPC

Errors

- Inaccuracies of on-board georeferencing devices (GPS, star trackers, ...)
- Error modelling
  - narrow field of view $\rightarrow$ errors modelled as shifts in image space, non-linearities negligible

Rational Polynomial Coefficients RPC

Errors

- Inaccuracies of on-board georeferencing devices (GPS, star trackers, ...)
- Error modelling
  - narrow field of view → errors modelled as shifts in image space, non-linearities negligible
  - errors in translation and attitude are correlated → no separation between translation or attitude err.

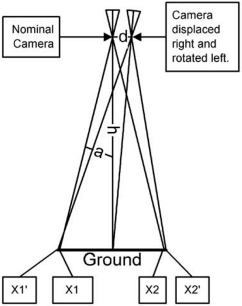


Figure: Position and attitude errors correlation. [GD03]

Rational Polynomial Coefficients RPC

Errors

- Inaccuracies of on-board georeferencing devices (GPS, star trackers, ...)
- Error modelling
  - narrow field of view → errors modelled as shifts in image space, non-linearities negligible
  - errors in translation and attitude are correlated → no separation between translation or attitude err.

$$y = g(\varphi, \lambda, h) + D_x(y, x)$$
$$x = h(\varphi, \lambda, h) + D_y(y, x)$$

where $D_x(y, x) = \sum \sum a_{ij} \cdot x^i y^j$

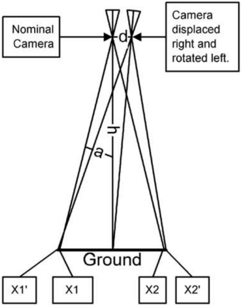


Figure: Position and attitude errors correlation. [GD03]

Pushbroom camera

Dataset

Joint aerial and satellite bundle adjustment - user's perspective

Pushbroom sensor - programmer's perspective

Dataset

- MMVII/MMVII-UseCaseDataSet/Argentique-Sat
- satellite and aerial images
- initial orientations [ZRPD21]
- tie points (intra- and inter-sensor)

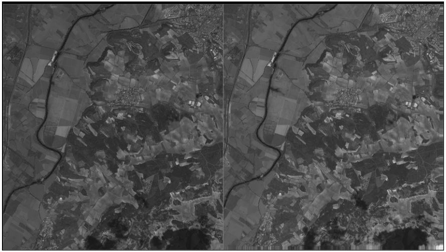


Figure: Pleiades stereo

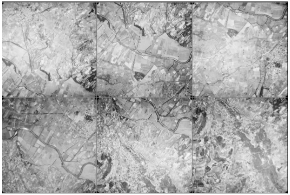


Figure: Aerial analogue x6 images

Pushbroom camera

Dataset

Joint aerial and satellite bundle adjustment - user's perspective

Pushbroom sensor - programmer's perspective

Joint aerial and satellite bundle adjustment - user's perspective

1. In terminal: `cd MMVII/MMVII-UseCaseDataSet/Argentique-Sat`
2. Import to MMV2 Photogrammetric Project
   - tie points (convert MMV1 to MMV2)
   - initial orientation of perspective camera (MMV1 $\rightarrow$ MMV2)
   - initial orientation of pushbroom camera
3. Define local coordinate frame
4. Associate your sensors' initial orientations to local coordinate frame
5. Define the pushbroom sensor as *"adjustable"*
   (perspective camera by def is adjustable)
6. Generate *"virtual"* ground control points
7. Bundle adjustment

Pushbroom camera

Dataset

Joint aerial and satellite bundle adjustment - user's perspective

Pushbroom sensor - programmer's perspective

Pushbroom sensor - programmer's perspective

Change branches

In terminal:

```
cd micmac
git checkout MMV2-Satellite-er
```

Pushbroom sensor - programmer's perspective

Sensor classes - current status, bound to evolve over time

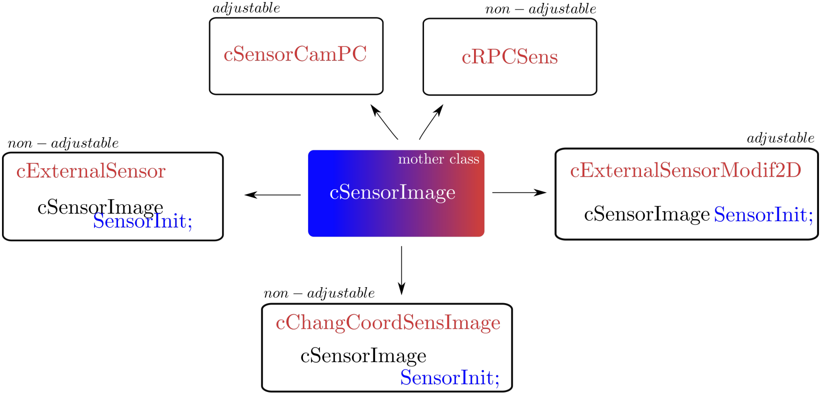
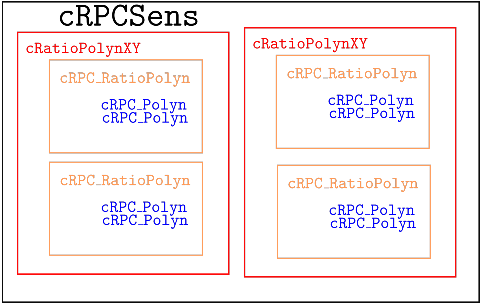


Figure: All sensors derive from the mother cSensorImage. *Real* sensors (`cSensorCamPC`,`cRPCSens`) directly implement the projection functions, while *virtual* sensors (`cExternalSensor,...`) rely on initial sensors. Not all sensor classes are *adjustable*.

Pushbroom sensor - programmer's perspective

RPC classes



$$Y = \frac{Num_L(P, L, H)}{Den_L(P, L, H)}$$

$$X = \frac{Num_S(P, L, H)}{Den_S(P, L, H)}$$

- `cRPC_Polyn` : *Num* (also *Den*), 3^{rd} deg polynomial
- `cRPC_RatioPolyn` $\frac{Num}{Den}$ rational polynomial composed of two `cRPC_Polyn`
- `cRatioPolynXY` : Y and X of type `cRPC_RatioPolyn`
- cRPCSens : two `cRatioPolynXY`, one for direct and one for inverse model

Pushbroom sensor - programmer's perspective

Three implementation tasks

1. Implement rational polynomials, `Sensors/cRPC.cpp`
   - init in `Dimap_ReadXMLModel`
   - fill in a polynomial, `Val` and `FillCubicCoeff` in `cRPC_Polyn`
   - compute the rational in `cRPC_RatioPolyn::Val`
   - apply the rational to a 3D point in `cRatioPolynXY::Val`
   - test with `MMVII TestRPC`

Pushbroom sensor - programmer's perspective

Three implementation tasks

1. Implement rational polynomials, `Sensors/cRPC.cpp`
   - init in `Dimap_ReadXMLModel`
   - fill in a polynomial, `Val` and `FillCubicCoeff` in `cRPC_Polyn`
   - compute the rational in `cRPC_RatioPolyn::Val`
   - apply the rational to a 3D point in `cRatioPolynXY::Val`
   - test with `MMVII TestRPC`
2. Implement the projection functions, `Sensors/cRPC.cpp`
   - write the normalisation code, `NormGround`, `NormIm` in `cRPCSens`
   - concatenate the normalisation and application of the rational polynomial in `cRPCSens::Ground2Image` & `cRPCSens::Image2Bundle`
   - test with `MMVII TestRPC`

Pushbroom sensor - programmer's perspective

Three implementation tasks

1. Implement rational polynomials, `Sensors/cRPC.cpp`
   - init in `Dimap_ReadXMLModel`
   - fill in a polynomial, `Val` and `FillCubicCoeff` in `cRPC_Polyn`
   - compute the rational in `cRPC_RatioPolyn::Val`
   - apply the rational to a 3D point in `cRatioPolynXY::Val`
   - test with `MMVII TestRPC`
2. Implement the projection functions, `Sensors/cRPC.cpp`
   - write the normalisation code, `NormGround`, `NormIm` in `cRPCSens`
   - concatenate the normalisation and application of the rational polynomial in `cRPCSens::Ground2Image` & `cRPCSens::Image2Bundle`
   - test with `MMVII TestRPC`
3. Prepare observations, unknowns and "context" for bundle adjustment

Pushbroom sensor - programmer's perspective

Third task – Preparing data for bundle adjustment

3a. Automated generation of the code to compute derivatives (and values)

- Create a class with your symbolic equation `formula` and `VNamesUnknowns`, `VNamesObs`, `FormulaName`, in `SymbDerGen/Formulas_RPC.h`
- add the class to `GenCodesFormula()` in `SymbDerGen/GenerateCodes.cpp`
- compile, then `MMVII GenCodeSymDer` then compile again
- play with `SymbComment,SymbCommentDer,SymbPrint...`

Pushbroom sensor - programmer's perspective

Third task – Preparing data for bundle adjustment

3a. Automated generation of the code to compute derivatives (and values)

- Create a class with your symbolic equation `formula` and `VNamesUnknowns`, `VNamesObs`, `FormulaName`, in `SymbDerGen/Formulas_RPC.h`
- add the class to `GenCodesFormula()` in `SymbDerGen/GenerateCodes.cpp`
- compile, then `MMVII GenCodeSymDer` then compile again
- play with `SymbComment,SymbCommentDer,SymbPrint...`

3b. Use the gen. code to compute derivatives of your function (and values)

- Create the Calculator, an interface class in SymbDerGen/GenerateCodes.cpp
- Declare the calc in `include/MMVII_PhgrDist.h`
- Implement `cRPCSens::DiffGround2Im` in `Sensors/cRPC.cpp`
  - will be called by the bundle adjustment
  - returns the values and derivatives at current value of unknown

test with `TestRPC` and/or `TestSensor`

Jacek Grodecki and Gene Dial.
Block adjustment of high-resolution satellite images described by rational polynomials.
Photogrammetric Engineering & Remote Sensing, 69(1):59–68, 2003.

C Vincent Tao and Yong Hu.
A comprehensive study of the rational function model for photogrammetric processing.
Photogrammetric engineering and remote sensing, 67(12):1347–1358, 2001.

Lulin Zhang, Ewelina Rupnik, and Marc Pierrot-Deseilligny.
Feature matching for multi-epoch historical aerial images.
ISPRS Journal of Photogrammetry and Remote Sensing, 182:176–189, 2021.