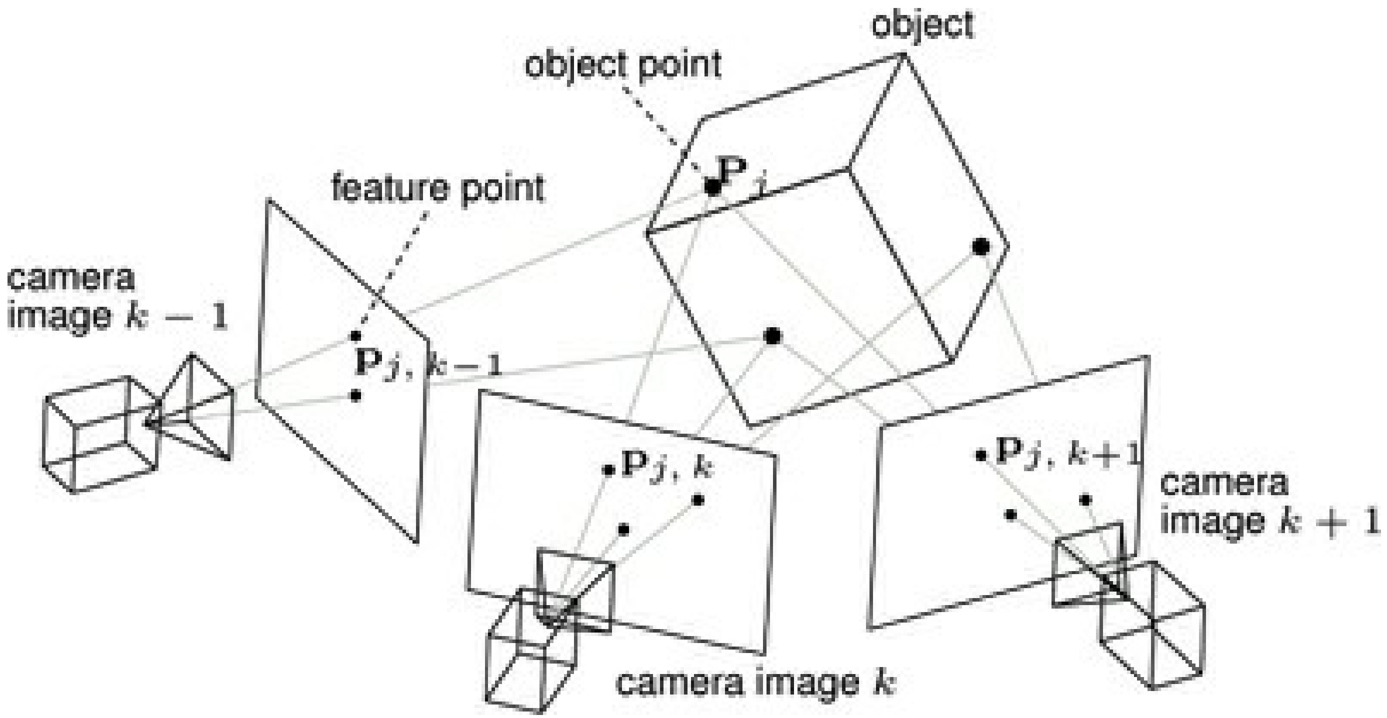
Sparse Bundle Adjustment

Vortrag von Matej Svejda

Computer Vision & Visual Tracking for Robotic Applications

Was ist Bundle Adjustment?

- Auf deutsch: Bündelblockausgleichung ☺
- „Lösung“ eines Henne-Ei-Problems:
- Gegeben:
  - Bilder einer (unbewegten) Szene aus verschiedenen Blickwinkeln (views)
  - Menge von korrespondierenden feature points
- Gesucht:
  - Welche Position hatte die Kamera bei der Aufnahme der einzelnen Bilder? => 9 Parameter pro Bild
  - Was sind die 3D-Koordinaten der einzelnen feature points? => 3 Parameter pro feature point


object
object point
$\mathbf{P}_j$
feature point
camera
image $k-1$
$\mathbf{P}_{j,\,k-1}$
$\mathbf{P}_{j,\,k}$
$\mathbf{P}_{j,\,k+1}$
camera
image $k+1$
camera image k

Mathematische Formulierung

- *n* feature points werden in *m* views gesehen mit den Korrdinaten $P_{i,j}, i \in \{1,\ldots,n\}, j \in \{1,\ldots,m\}$
- Ist feature point *i* auf view j zu sehen, dann sei $v_{i,j}=1$ sonst $v_{i,j}=0$
- Seien a_j die Kameraparameter von view *j* und b_i die Korrdinaten von feature point *i*
- *q* sei die Projektionsfunktion und *d* die euklidische Distanz
- Bundle adjustment minimiert für *a* und *b* die Kostenfunktion

$$c(a,b)=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m} v_{ij}\, d\left(q\left(a_i, b_j\right), P_{i,j}\right)^2$$

Wie kommt man zur Lösung?

- Gesucht ist das Minimum Kostenfunktion
- Die Kostenfunktion ist immer positiv und für die exakte Lösung gleich Null
- Exakte Lösung ist aber nicht möglich, da kein LGS
  - Kameraparameter werden mit den feature point Koordinaten bei der Projektion nichtlinear kombiniert
  - Euklidische Distanz quadriert die Koordinatendifferenz
- Ansatz:
  - *Iterative Methoden*

Iterative Methoden

- Alternative Schreibweise der Kostenfunktion:

$$p = \begin{pmatrix} a \\ b \end{pmatrix}, \quad \varepsilon(p) = q(p) - P \quad \Rightarrow \quad c(p) = \frac{1}{2} \varepsilon(p)^T \varepsilon(p)$$

- Man beginnt mit einer Initialen Schätzung der gesuchten Parameter p_0
- Bei jedem Schritt in der Iteration wird die Schätzung der Parameter aktualisiert $p_{i+1} = p_i + u$
- Die Parameter werden (im Idealfall) so abgeändert, dass die Kostenfunktion sukzessive gegen Null geht
- Verschiedene iterative Methoden unterscheiden sich darin, wie das Update *u* berechnet wird

Newton Methode

- Die Kostenfunktion wird quadratisch approximiert (Taylor)

$$c(p_i+u) \approx c(p_i)+c_p(p_i)u+u^T c_{pp}(p_i)u$$

- c_p ist der transponierte Gradient, c_{pp} die Hesse-Matrix
- *u* soll den Wert annehmen, bei dem $c(p_i+u)$ minimal wird

$$c(p_i+u) \rightarrow min \;\Leftrightarrow\; c_u(p_i+u)=0$$

- Also wird die quadratische Approx. nach *u* abgeleitet

$$\boxed{c_{pp}(p_i)u+c_p(p_i)=0}$$

- Aus dieser Gleichung kann *u* bestimmt werden

Abwandlungen der Newton Methode

- Für eine vektorwertige Fkt. *f* gilt: $\partial_x\left(f(x)^T f(x)\right)=2 f_x(x)^T f(x)$

$$c_p(p) = \frac{1}{2}\partial_x\left(\varepsilon(p)^T \varepsilon(p)\right) = \varepsilon_p(p)^T \varepsilon(p)$$

$$c_{pp}(p) = \varepsilon_p(p)^T \varepsilon_p(p)+\varepsilon_{pp}(p)^T \varepsilon(p)$$

- **Gauss-Newton:** $\varepsilon(p)$ wird als linear angenommen

$$\Rightarrow\ c_{pp}(p)=\varepsilon_p(p)^T \varepsilon_p(p) \qquad \boxed{\varepsilon_p(p)^T \varepsilon_p(p)u=-\varepsilon_p(p)^T \varepsilon(p)}$$

- **Gradientenabstieg:** $c_{pp}(p)$ wird mit $\lambda\, Id$ approximiert

$$\boxed{\lambda u=-\varepsilon_p(p)^T \varepsilon(p)}$$

Levenberg-Marquardt

- Kombination von Gauss-Newton und Gradientenabstieg

$$(\lambda\ Id+\varepsilon_p(p)^T\varepsilon_p(p))u=-\varepsilon_p(p)^T\varepsilon(p)$$

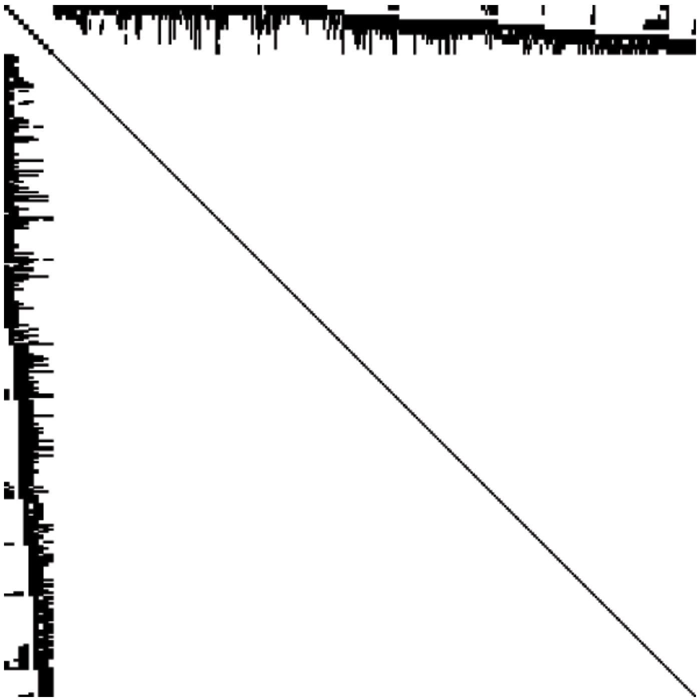
- Der Wert für *λ* wird so gewählt, dass die Kostenfunktion nach dem Update einen niedrigeren Wert hat
- Dadurch bleibt die Iteration nicht so leicht in lokalen Minima stecken
- Levenberg-Marquardt wechselt also je nach Bedarf zwischen Gradientenabstieg und Gauss-Newton

Sparse (dünnbesetzte) Matrizen bei LM

- Bei jedem Iterationsschritt muss mehrmals die „normal equation" gelöst werden:

  $$(\lambda\ Id + J^T J)u = J^T \varepsilon$$

- Dabei ist $J := \varepsilon_p(p)$ die Jacobi-Matrix
- Bei sparse bundle adjustment wird die blockbasierte, sparse Struktur der Matrix ausgenutzt



Struktur der sparsen Matrix bei LM (1)

- Die Jacobi-Matrix besteht aus zwei Blöcken $J=[\varepsilon_a \quad \varepsilon_b]=[A \quad B]$
- Dadurch erhält man für die Iterationsgleichung

$$(\lambda\, Id + J^T[A|B])\begin{pmatrix} u_a \\ u_b \end{pmatrix} = J^T\varepsilon \iff (\lambda\, Id + \begin{bmatrix} A^T A & A^T B \\ B^T A & B^T B \end{bmatrix})\begin{pmatrix} u_a \\ u_b \end{pmatrix} = \begin{pmatrix} A^T \varepsilon \\ B^T \varepsilon \end{pmatrix}$$

$$\iff \begin{bmatrix} \tilde{A} & W \\ W^T & \tilde{B} \end{bmatrix}\begin{pmatrix} u_a \\ u_b \end{pmatrix} = \begin{pmatrix} \varepsilon_A \\ \varepsilon_B \end{pmatrix} \iff M\begin{pmatrix} u_a \\ u_b \end{pmatrix} = \begin{pmatrix} \varepsilon_A \\ \varepsilon_B \end{pmatrix}$$

- Die Tatsache, dass sich die Matrix in dieser Form in Blöcke aufteilen lässt, vereinfacht die Invertierung von M
  - Man muss nur die Matrix $\tilde{A}$ und das s.g. Schurkomplement $\tilde{B} - W^T \tilde{A}^{-1} W$ invertieren

Struktur der sparsen Matrix bei LM (2)

- Zusätzlich zur Blockstruktur von M wird die Struktur der Jacobi-Matrix J ausgenutzt
- Diese ergibt sich daher, dass feature-points nur mit den Kameraparametern kombiniert werden, auf denen sie zu sehen sind

$$J=\begin{bmatrix} A & B \end{bmatrix} \quad , \quad A=[A_1 A_2 \ldots A_n]^T \quad , \quad B=diag(B_1, B_2, \ldots, B_n)$$

- Durch diese Struktur kann $\tilde{A}$ und das Schurrkomplement effizienter gespeichert werden
- Auch das Invertieren kann effizienter implementiert werden
  - Die Invertierung einer Matrix hat eine Komplexität von $O(N^3)$
  - Wird die sparse Strutkur ausgenützt kann das auf bis zu $O(N)$ reduziert werden

Anwendung von Sparse Bundle Adjustment

- In allen Bereichen, bei denen man die Position von Kamera und feature points bestimmen will
- Dense Multi-View Reconstruction:

  Feature points extrahieren → feature points matchen → Kamerabewegung mit SBA bestimmen → Dense recosuntruction

  http://www.youtube.com/watch?v=vdt66tsPjCM
- Schleifenerkennung bei SLAM:

  Aufnahmen mit distinktiven feature points speichern (key-frame) → mit SBA Schleifen in den key-frames erkennen

  http://www.youtube.com/watch?v=_-p08o_oTO4&feature=player_detailpage#t=41s

Das wars...

Danke für die Aufmerksamkeit ☺