Dokumentation OpenGM Matching und Stereo

Autor: Walter Forkel walter.forkel@tu-dresden.de

Das Projekt ist in OpenGM via CMake eingebunden, also ein Unterprojekt von OpenGM. Es besteht aus zwei Teilen: *Point Cloud Matching* und *Stereo Reconstruktion with Occlusions*, die sich in den jeweiligen Unterordnern befinden. Als Editor für CMake Projekte kann ich *QtCreator* also IDE empfehlen, da dieser sehr gut an CMake angepasst ist.

Benötigte Bibliotheken:

- CMake http://www.cmake.org/
- OpenGM http://hci.iwr.uni-heidelberg.de/opengm2/
- OpenCV http://opencv.org/
- Ply http://people.cs.kuleuven.be/~ares.lagae/libply/
- HDF5 http://www.hdfgroup.org/
- Boost http://www.boost.org/

Das Installieren der Externen Bibliotheken in OpenGM ist in der Version 2.4 leider nur unter Linux einfach zu machen. Unter Windows ist das Script dazu nicht ausführbar, da wget fehlt. Um die externen Bibliotheken unter Windows zu installieren, kann man Cygwin installieren und damit dann auch wget. So sollte es funktionieren. Ab der OpenGM Version 3.0 sollen die Benutzung der Externen Bibliotheken wesentlich vereinfacht werden.

Git-Repository Location

- https://gitlab.azapps.de
- login: shk-computer-vision
- password: opengmshk

Der Branch finalVersion beinhaltet die aktuelle Version.

Point Cloud Matching / Feature Correspondence

Paper: *Lorenzo Torresani, Vladimir Kolmogorov, and Carsten Rother, Feature Correspondence via Graph Matching: Models and Global Optimization, Tech.Report, in ECCV, October 2008*

Bestehend aus drei Unterprogrammen:

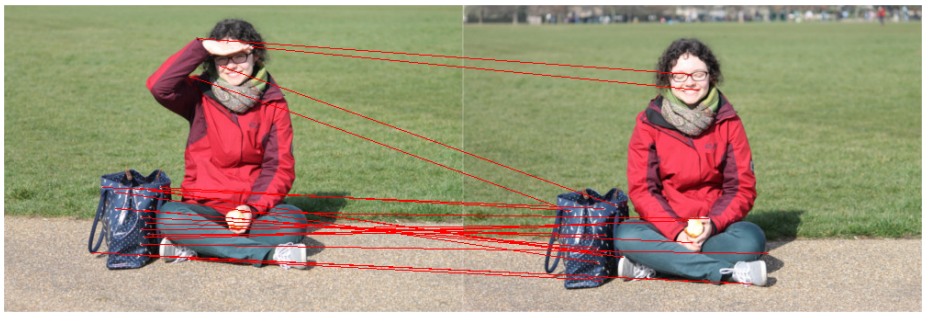

Figure 1: Example

- ImageMatching - Matching von zwei Bildern. Der Nutzer kann entscheiden, wie viele Punkte in jedem Bild benutzt werden sollen, um die Komplexität steuern zu können. Außerdem können alle Parameter interaktiv verändert werden, ebenso wie der Algorithmus.
- PointCloudMatching - Verbindet zwei Punktmengen, gegeben entweder im Ply Format oder kodiert als Bilder. In diesem Fall wird jedes nicht weiße Pixel ein Punkt, der gematcht werden soll. Generiert die gleichen Ausgaben wie ImageMatching, nur ohne Bild.
- PCImageConverter - Erstellt ein Bild aus gegebenen Punkten und Korrespondenzen. Akzeptiert als Eingabe Ply dateien, die von PCMatching erstellte matches.txt und generiert ein PNG Bild. Die Z-Koordinate wird dabei ignoriert. Oft ist es besser, wenn man sich die matches.ply Datei direkt in *Meshlab* ansieht.

Features:

- Verschiedene Algorithmen: TRWS, ICM, LazyFlipper, BruteForce, weitere können einfach hinzugefügt werdene
- Vielfältige Ausgaben: Model wird als hdf gespeichert, matches.ply kann in Meshlab geöffnet werden, in Bildern gefundene Keypoints werden als Bild gespeichert, das Matchingbild wird ausgegeben.
- Auswahl zwischen *Zweilabel Model* (binär, jedes mögliche Alignment von zwei Punkten ist eine Variable) oder *Mehrlabel Model* (Jeder Punkt im linken Bild ist eine Variable, die als Zustände 0 oder einen Punkt im rechten Bild haben kann.)
- Um Speicherplatz zu sparen, werden keine Marrays benutzt um die Unaries und Pairwise Terms zwischenzuspeichern, sondern durch Iteratoren werden diese Werte erst berechnet, wenn sie benötigt werden. Der Nachteil ist, dass das Programm dadurch sehr viel komplexer wird.

Probleme:

- Das finden der richtigen Parameter ist sehr zeitintensiv, bzw. sollte durch lernen gelöst werden??
- Binäres und Multilabel Model sollten äquivalent sein, es werden aber unterschiedliche Lösungen gefunden. Potentieller Fehler bei Multilabel Model. Zudem hat Dagmar eine Geschwindigkeitserhöhung um den Faktor 50x festgestellt, der sich hier aber nicht eingestellt hat. Meine Vermutung ist, dass es entweder mit einem Fehler in der Implementierung zusammenhängt, oder die Wahl der Parameter eine große Rolle spielt.

Mögliche Aufgaben für die Lehre:

- Gegeben nur das binäre Model, formulieren Sie die Multilabelversion und vergleichen Sie die Zeitkomplexität.
- Testen Sie weitere Inferenz-Algorithmen, z.B. Lazy Flipper und finden Sie jeweils geeignete Parameter.

Energy Funktion des Point Cloud Matchings

$$E(x) = \lambda^{\text{occl}} + \sum_{a \in A} \left(\lambda^{\text{app}} \theta_a^{\text{app}} - \frac{\lambda^{\text{occl}}}{\min(|P'|, |P''|)} + 2\lambda^{\text{coh}} \right) x_a \tag{1}$$

$$+ \sum_{(a,b) \in N} \left(\lambda^{\text{geom}} \theta_{ab}^{\text{geom}} - \lambda^{\text{coh}} \right) x_a x_b \tag{2}$$

$$\tag{3}$$

where

$$\theta_a^{\text{app}} = \text{distance between appearance discriptors (shape context)} \tag{4}$$

$$\theta_{ab}^{\text{geom}} = \eta(e^{\frac{\delta_{a,b}^2}{\sigma_l^2}} - 1) + (1 - \eta)(e^{\frac{\alpha_{a,b}^2}{\sigma_\alpha^2}} - 1) \tag{5}$$

$$\text{with } \delta_{(p',p''),(q',q'')} = \frac{|||p' - q'|| - ||p'' - q''|||}{||p' - q'|| + ||p'' - q''||} \tag{6}$$

$$\alpha_{(p',p''),(q',q'')} = \arccos\left(\frac{p' - q'}{||p' - q'||} \cdot \frac{p'' - q''}{||p'' - q''||} \right) \tag{7}$$

Densely connected Stereo matching

Paper: *Vladimir Kolmogorov and Carsten Rother, Comparison of energy minimization algorithms for highly connected graphs, in Proc. European Conference*

in Computer Vision (ECCV), Springer-Verlag, June 2006

Das Programm heißt *Stereo* und befindet sich in gleichnamingem Unterordner. Die Parameter Disparitätsstufen und die Coherence Kosten können mitgegeben werden. Zudem werden zwei recktifizierte Eingabebilder benötigt. Der einfache Aufruf von TRWS, so wie er in OpenGM implementiert ist, führt zu extrem langen Berechnungszeiten. Eine schnellere Alternative stellt Alpha-Expansion dar. Ausgegeben wird das Bild als Ply-Datei für Meshlab und als Depthmap-Bild.

Parameter

In dem Paper sind die Disparitäten zu den jeweiligen Bildern angegeben. Die Bilder können gedownloaded werden unter http://vision.middlebury.edu/stereo/data/

- Map 29
- Tsukuba 16
- Sawtooth 19

Mögliche Aufgaben für die Lehre

- Vergleichen Sie das Potts Model mit der Linear Truncated Label Difference und finden Sie jeweils geeignete Parameter für jede Funktion. Welche liefert das bessere Ergebnis?
- Implementieren Sie einen eigenen TRWS Algorithmus, der eine Zeitkomplexität von $O(NK)$ hat. Hinweis: Siehe Paper.
- Beschleunigen Sie die Berechnung, indem Sie die Lösung von einem verkleinerten Bild für die Lösung des Originalbildes benutzen.

Visual Studio OpenGM & OpenCV Setup

Das Projekt arbeitet mit CMake, daher ist dieser Teil nur noch teilweise notwendig. Wenn man ein Projekt mit OpenGM und OpenCV in VisualStudio erstellen möchte, darf man folgenden Schritte nicht vergessen:

In *Konfigurationseigenschaften -> Linker -> Zusätzliche Bibliotheksverzeichnisse* eintragen: openCV/build/x86/vc11/lib

In *Konfigurationseigenschaften -> C/C++ -> Zusätzliche Includeverzeichnisse* eintragen: openCV/build/include; openGM/include

In *Konfigurationseigenschaften -> Linker -> Eingabe -> Zusätzliche Abhängigkeiten* hinzufügen: opencv_calib3d246d.lib;opencv_contrib246d.lib;opencv_core246d.lib; opencv_features2d246d.lib;opencv_flann246d.lib;opencv_gpu246d.lib; opencv_haartraining_engined.lib;opencv_highgui246d.lib;opencv_imgproc246d.lib; opencv_legacy246d.lib;opencv_ml246d.lib;opencv_nonfree246d.lib;opencv_objdetect246d.lib; opencv_ocl246d.lib;opencv_photo246d.lib;opencv_stitching246d.lib;opencv_superres246d.lib; opencv_ts246d.lib;opencv_video246d.lib;opencv_videostab246d.lib;

Fazit

OpenGM ist ein gutes Framework, allerdings noch nicht ganz einfach zu benutzen, besonders unter Windows mit den externen Bibliotheken. Zudem ist gutes Wissen über CMake nötig. Die nächste Version von OpenGM soll dies wesentlich vereinfachen. Allerdings ist die Frage, ob es in der Lehre zu besserem Verständnis des eigentlichen Problems führt. An dieser Stelle bin ich mir nicht ganz sicher.