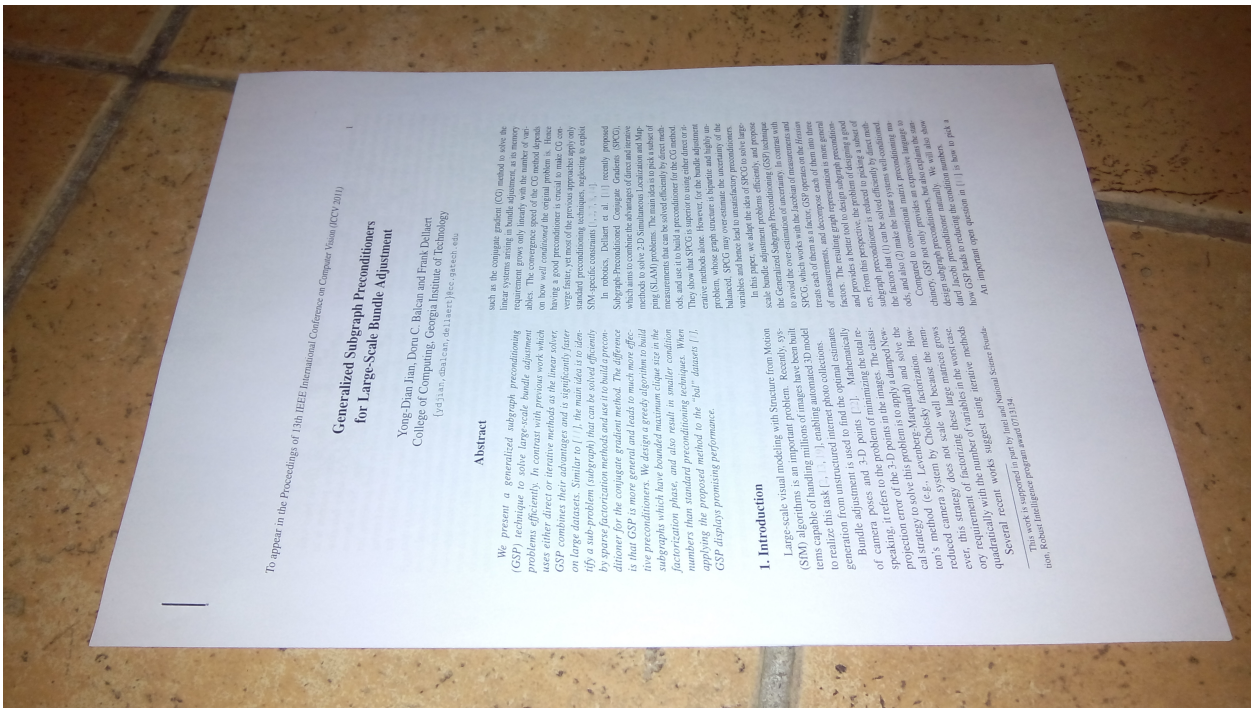
TP rectification homographique

Guillaume Bourmaud - guillaume.bourmaud@enseirb-matmeca.fr

L'objectif de ce TP est de mettre en oeuvre la notion d'homographie étudiée en cours.

Pour cela, nous allons considérer l'image suivante (lien de téléchargement) :



Cette image correspond à la projection d'une scène planaire (dans notre cas une feuille A4 posée sur le sol) dans le plan image d'une caméra idéale du point de vue du modèle sténopé. Dans cette image, le texte présent sur la feuille A4 est difficilement lisible en raison de la rotation et de la translation de la caméra vis-à-vis de la feuille lors de la prise de vue.

Nous avons vu que la transformation entre une scène planaire et le plan image est une homographie. Ainsi, le présent TP consiste à mettre en oeuvre une méthode de rectification homographique permettant d'estimer cette homographie pour l'appliquer à notre image et ainsi obtenir une image rectifiée (que l'on peut interpréter comme une **reconstruction de la scène planaire**) où le texte présent sur la feuille A4 est lisible.

Après rectification homographique, l'image rectifiée doit représenter la feuille A4 de telle sorte que cette dernière soit parfaitement lisible :

Generalized Subgraph Preconditioners for Large-Scale Bundle Adjustment

Yong-Dian Jian, Doru C. Balcan and Frank Dellaert
College of Computing, Georgia Institute of Technology
{ydjian,dbalcan,dellaert}@cc.gatech.edu

Abstract

We present a generalized subgraph preconditioning (GSP) technique to solve large-scale bundle adjustment problems efficiently. In contrast with previous work which uses either direct or iterative methods as the linear solver, GSP combines their advantages and is significantly faster on large datasets. Similar to [11], the main idea is to identify a sub-problem (subgraph) that can be solved efficiently by sparse factorization methods and use it to build a preconditioner for the conjugate gradient method. The difference is that GSP is more general and leads to much more effective preconditioners. We design a greedy algorithm to build subgraphs which have bounded maximum clique size in the factorization phase, and also result in smaller condition numbers than standard preconditioning techniques. When applying the proposed method to the "bal" datasets [1], GSP displays promising performance.

1. Introduction

Large-scale visual modeling with Structure from Motion (SfM) algorithms is an important problem. Recently, systems capable of handling millions of images have been built to realize this task [2, 13, 19], enabling automated 3D model generation from unstructured internet photo collections.

Bundle adjustment is used to find the optimal estimates of camera poses and 3-D points [22]. Mathematically speaking, it refers to the problem of minimizing the total reprojection error of the 3-D points in the images. The classical strategy to solve this problem is to apply a damped Newton's method (e.g., Levenberg-Marquardt) and solve the reduced camera system by Cholesky factorization. However, this strategy does not scale well because the memory requirement of factorizing these large matrices grows quadratically with the number of variables in the worst case.

Several recent works suggest using iterative methods such as the conjugate gradient (CG) method to solve the linear systems arising in bundle adjustment, as its memory requirement grows only linearly with the number of variables. The convergence speed of the CG method depends on how *well conditioned* the original problem is. Hence having a good preconditioner is crucial to make CG converge faster, yet most of the previous approaches apply only standard preconditioning techniques, neglecting to exploit SfM-specific constraints [1, 2, 7, 8, 14].

In robotics, Dellaert et al. [11] recently proposed Subgraph-Preconditioned Conjugate Gradients (SPCG), which aims to combine the advantages of direct and iterative methods to solve 2-D Simultaneous Localization and Mapping (SLAM) problems. The main idea is to pick a subset of measurements that can be solved efficiently by direct methods, and use it to build a preconditioner for the CG method. They show that SPCG is superior to using either direct or iterative methods alone. However, for the bundle adjustment problem, whose graph structure is bipartite and highly unbalanced, SPCG may over-estimate the uncertainty of the variables and hence lead to unsatisfactory preconditioners.

In this paper, we adapt the idea of SPCG to solve large-scale bundle adjustment problems efficiently, and propose the Generalized Subgraph Preconditioning (GSP) technique to avoid the over-estimation of uncertainty. In contrast with SPCG, which works with the Jacobian of measurements and treats each of them as a factor, GSP operates on the *Hessian* of measurements, and decompose each of them into three factors. The resulting graph representation is more general and provides a better tool to design subgraph preconditioners. From this perspective, the problem of designing a good subgraph preconditioner is reduced to picking a subset of the factors that (1) can be solved efficiently by direct methods, and also (2) make the linear systems well-conditioned.

Compared to conventional matrix preconditioning machinery, GSP not only provides an expressive language to design subgraph preconditioners, but also explains the standard Jacobi preconditioner naturally. We will also show how GSP leads to reducing the condition numbers.

An important open question in [11] is how to pick a

This work is supported in part by Intel and National Science Foundation, Robust Intelligence program award 0713134.

Travail sur feuille

1. Faire un schéma **en 3D**, en reprenant les notations du cours, où doivent figurer le référentiel de la caméra (O_c, x_c, y_c, z_c), le plan focal normalisé de la caméra, le plan P et son référentiel (O_p, x_p, y_p, z_p). Dessiner la feuille A4 dans le plan P (**en plaçant l'origine O_p sur le coin en haut à gauche de la feuille, en alignant l'axe x_p sur le bord haut de la feuille et l'axe y_p sur le bord gauche de la feuille**) et sa reprojection dans le plan focal normalisé. Dans le plan focal normalisé, faire apparaître les 4 coins de la feuille A4 : $\{\underline{m}_{c,i}\}_{i=1...4}$.
2. Faire deux schémas **en 2D**, en reprenant les notations du cours.
   A. Le premier schéma correspond au plan image de la caméra : représenter la feuille A4 et faire apparaître ses 4 coins $\{p_{c,i}\}_{i=1...4}$.
   B. Le second schéma correspond au plan P : représenter la feuille A4 et faire apparaître ses 4 coins $\{r_{p,i}\}_{i=1...4}$. Une feuille A4 étant de taille 21cm par 29,7cm, quelles sont les coordonnées des 4 coins $\{r_{p,i}\}_{i=1...4}$?
   C. Relier les 4 coins de la feuille A4 dans le plan P aux 4 coins de la feuille A4 dans le plan image afin d'obtenir 4 correspondances $\{(p_{c,i}, r_{p,i})\}_{i=1...4}$.

Créer un script MATLAB `define_corners.m`

Pour obtenir les coordonnées des 4 coins $\{p_{c,i}\}_{i=1...4}$, nous allons simplement utiliser une interface graphique (en MATLAB) permettant de cliquer sur ces 4 coins.

1. Charger l'image à rectifier (fonction `imread`)
2. Afficher l'image à rectifier (fonction `imshow`)
3. Cliquer sur les 4 coins de la feuille A4 (fonction `ginput`) pour obtenir $\{p_{c,i}\}_{i=1...4}$.
4. Sauvegarder $\{p_{c,i}\}_{i=1...4}$ dans un fichier `coins.mat` (fonction `save`)

Créer un script MATLAB `rectification_homographique.m`

1. Définir les 4 coins dans le plan P : $\{r_{p,i}\}_{i=1...4}$. Pour cela, vous pouvez considérer dans un premier temps que l'image rectifiée sera de taille 297×210.
2. Charger les 4 coins (fonction `load`) de la feuille A4 dans le plan P : $\{p_{c,i}\}_{i=1...4}$.
3. Dans une figure : afficher à gauche (fonction `subplot`) l'image à rectifier et les 4 coins $\{p_{c,i}\}_{i=1...4}$ (afficher chaque coin avec un cercle d'une couleur différente); afficher à droite, les 4 coins $\{r_{p,i}\}_{i=1...4}$ avec les mêmes couleurs que celles utilisées pour l'affichage de gauche. La commande `axis ij` permet de faire pointer l'axe y vers le bas. Vérifier visuellement que les correspondances sont correctement définies.
4. Coder la méthode d'estimation d'homographie vue en cours.
   A. Construire la matrice A de taille 8×8 et le vecteur p_c de taille 8×1
   B. Résoudre numériquement (fonction `mldivide`) le système linéaire $Ah = p_c$
   C. Construire la matrice d'homographie H_{cp} de taille 3×3 à partir du vecteur h
   D. Vérifier que $\underline{p}_{c,i} = \pi(H_{cp}\underline{r}_{p,i})\ \forall i \in [1...4]$ en traçant une croix pour chaque $\pi(H_{cp}\underline{r}_{p,i})$ dans le `subplot` de gauche.
5. Appliquer l'homographie estimée H_{cp} à l'image à rectifier (application d'une transformation à une image).
6. Sauvegarder l'image rectifiée au format pdf. Le document obtenu devrait être parfaitement lisible.