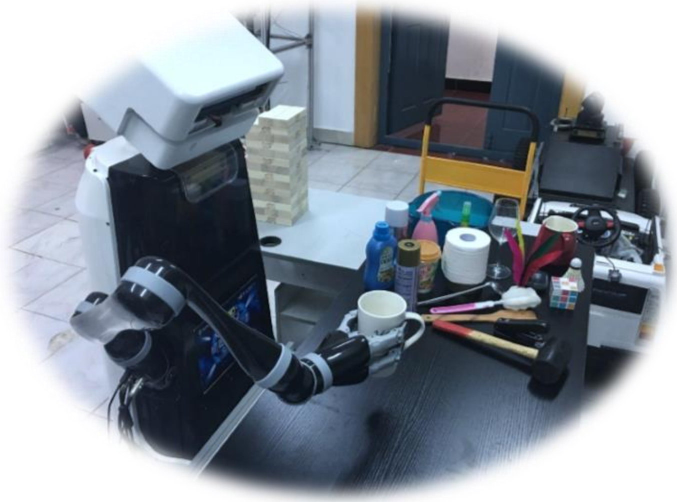
# Structure from motion



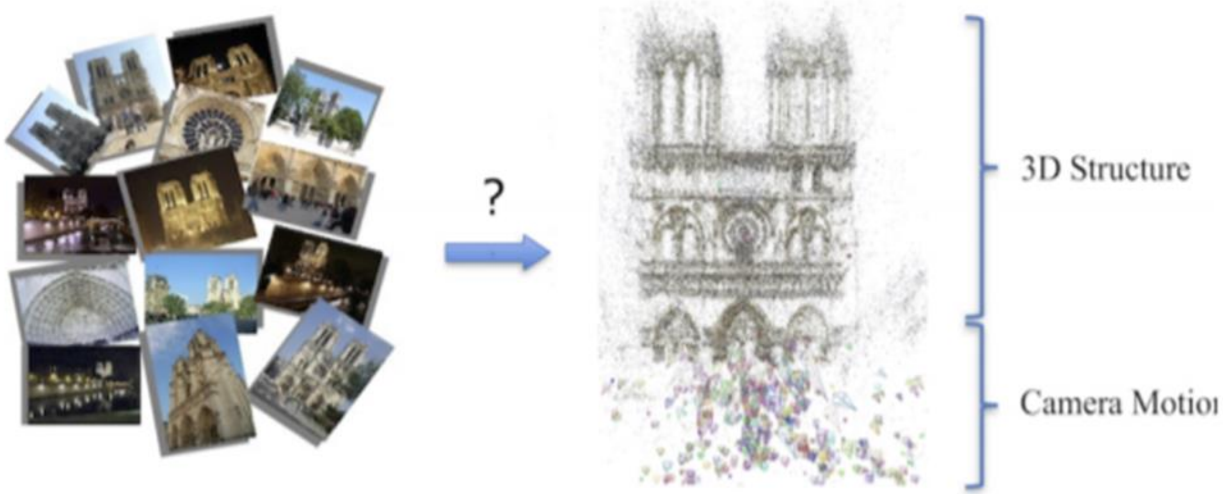
主　讲　人：李城

公　众　号：3D视觉工坊

# Contents

# What is Structure-from-Motion?



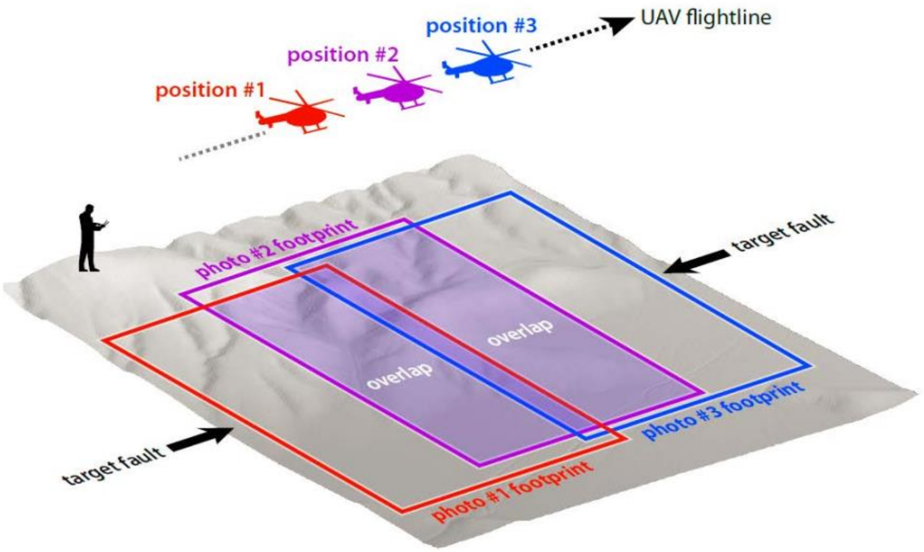


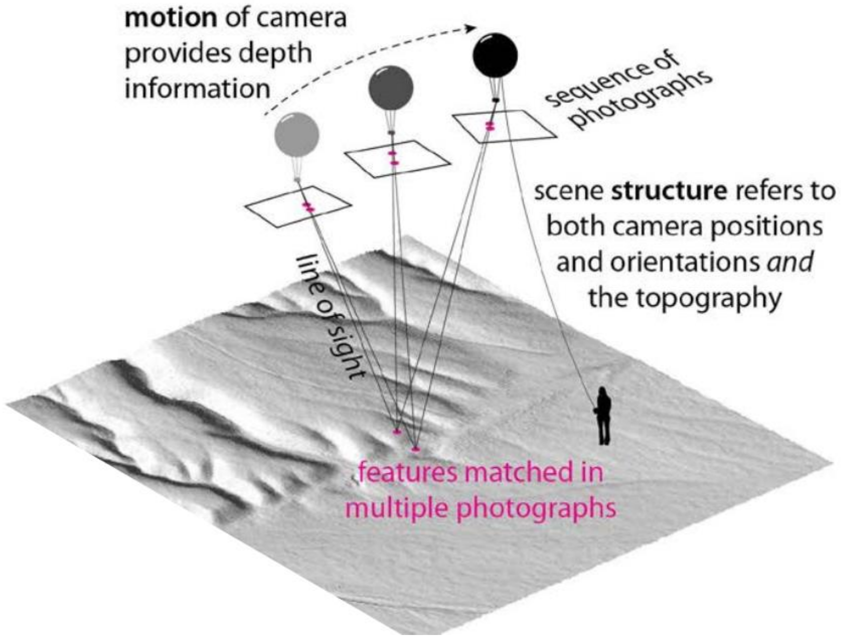

purpose :SfM is  to recover camera motion and scene geometry from images

# What is Structure-from-Motion?

Traditional stereo-photogrammetry
.Requires a stable platform(fix elevation)
.Known position and orientation

Structure from Motion
.Without prior pose
.Unstructured images

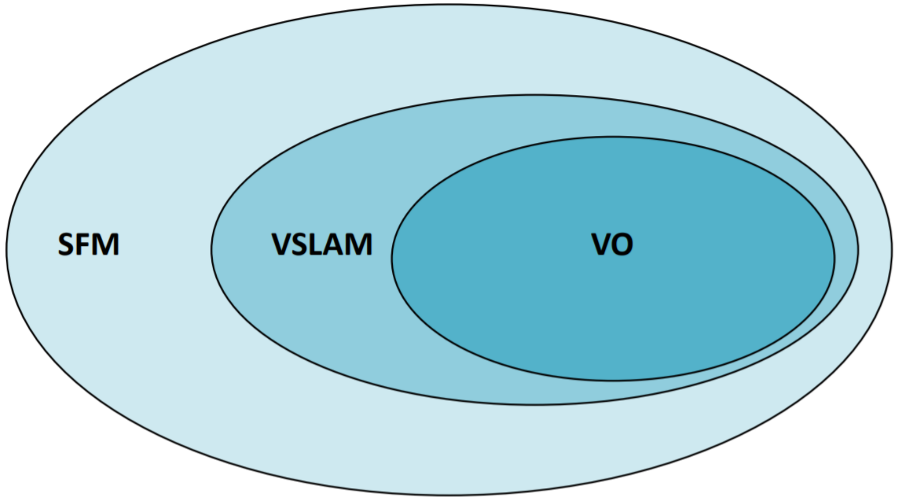


# Difference between SfM and SLAM



- SfM belongs to the category of photogrammetry,SLAM belongs to category of Robot vision




- SfM pays more attention to precision,SLAM cares more about speed.So SLAM is a real-time SfM

## Category and Frameworks

SfM category

| | |
|---|---|
| Global SfM | Incremental SfM |
| Hybrid SfM | Distributed SfM |

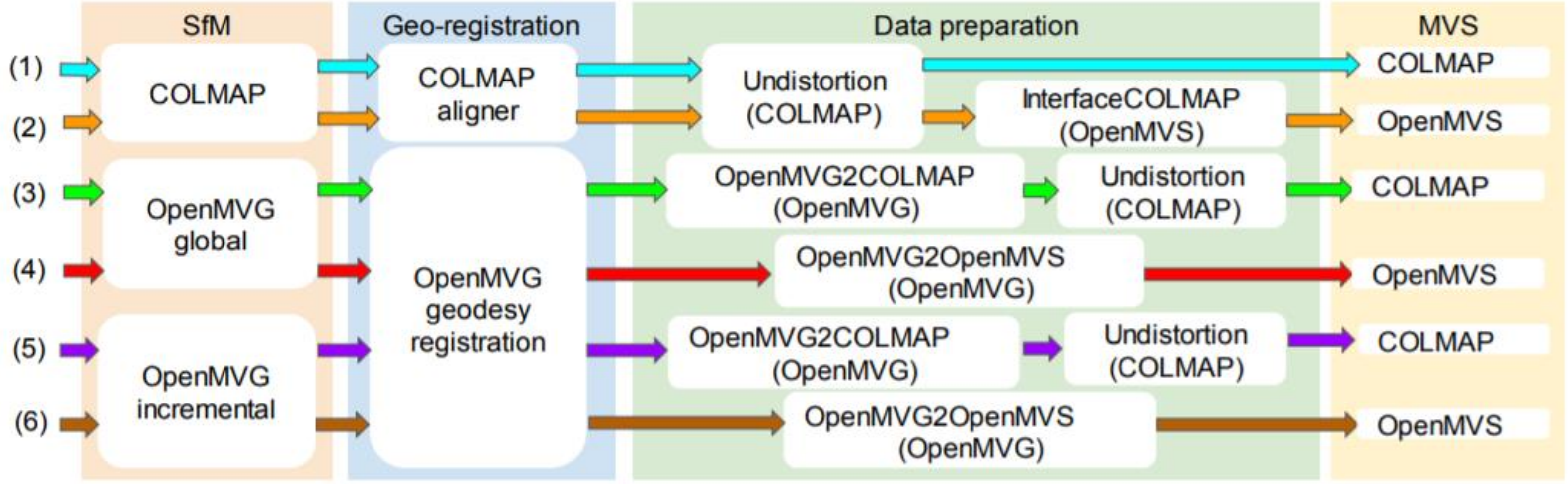
Well-known SfM open source frameworks are as follows:

# SfM algorithm introduction



| Open-source | Sfm pipeline | Robustness |
|---|---|---|
| Colmap | Incremental sfm | ★★★★☆ |
| openMVG | Incremental sfm and Global sfm | ★★★★☆ |
| Alicevision(no dense point cloud) | Similar openmvg(base Incremental openmvg) | ★★★☆☆ |
| MVE | Incremental sfm | ★★★☆☆ |

Different open source framework combinations are as follows:




***reference:A Benchmark for 3D Reconstruction from Aerial Imagery in an Urban Environment***

# SfM algorithm introduction



**Global Sfm**

Rotation averaging :

$$\min_{R_1,\ldots,R_n} \sum_{(i,j)\in E} d\left(R_i\tilde{R}_{ij}, R_j\right)^p \qquad \min_{R_1,\ldots,R_n} \sum_{(i,j)\in E} d\left(R_i\tilde{R}_{ij}, R_j\right)^p$$

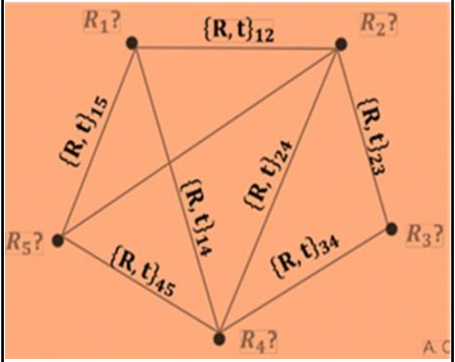
Translation averaging:

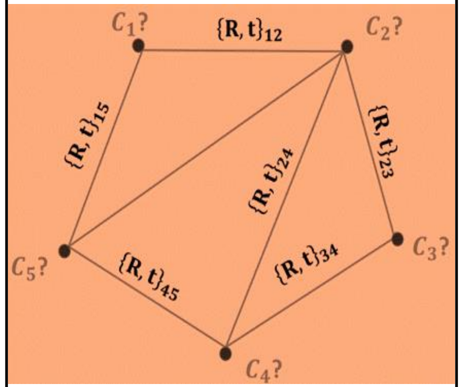
$$\boldsymbol{\lambda_{ij} t_{ij} = R_j(C_i - C_j)}$$
$$\boldsymbol{\mu_{ij} = R_j^T t_{ij}}$$

$$\min_{C} \left\| \boldsymbol{\mu_{ij}} - \frac{\boldsymbol{C_i - C_j}}{\|\boldsymbol{C_i - C_j}\|} \right\|$$

translation averaging in global SfM is difficult:
1) relative motion between two cameras doesn’t
encode the scale information(by essential matrices,5 dof),
global SfM methods don’t work for the data whose measurement
graph is not parallel rigid(degeneration：
all cameras on the same line )
2) sensitive to noisy data ,it’s difficult to filter bad EGs

Input Images → Two View Geometry → Rotation Averaging → Translation Averaging → Bundle Adjustment → 3D Model



# SfM algorithm introduction

公众号：3D视觉工坊

Global SfM bundle adjustment (openmvg source)

```
bool b_BA_Status = bundle_adjustment_obj.Adjust
  (
    sfm_data_,
    Optimize_Options(
      Intrinsic_Parameter_Type::NONE, // Intrinsics are held as constant
      Extrinsic_Parameter_Type::ADJUST_TRANSLATION, // Rotations are held as constant
      Structure_Parameter_Type::ADJUST_ALL,
      Control_Point_Parameter(),
      this->b_use_motion_prior_)
  );
if (b_BA_Status)
{
  if (!sLogging_file_.empty())
  {
    Save(sfm_data_,
      stlplus::create_filespec(stlplus::folder_part(sLogging_file_), "structure_00_refine_T_Xi", "ply"),
      ESfM_Data(EXTRINSICS | STRUCTURE));
  }

  // - refine only Structure and Rotations & translations
  b_BA_Status = bundle_adjustment_obj.Adjust
    (
      sfm_data_,
      Optimize_Options(
        Intrinsic_Parameter_Type::NONE, // Intrinsics are held as constant
        Extrinsic_Parameter_Type::ADJUST_ALL,
        Structure_Parameter_Type::ADJUST_ALL,
        Control_Point_Parameter(),
        this->b_use_motion_prior_)
    );
  if (b_BA_Status && !sLogging_file_.empty())
  {
    Save(sfm_data_,
      stlplus::create_filespec(stlplus::folder_part(sLogging_file_), "structure_01_refine_RT_Xi", "ply"),
      ESfM_Data(EXTRINSICS | STRUCTURE));
  }
}

if (b_BA_Status && ReconstructionEngine::intrinsic_refinement_options_ != Intrinsic_Parameter_Type::NONE) {
  // - refine all: Structure, motion:{rotations, translations} and optics:{intrinsics}
  b_BA_Status = bundle_adjustment_obj.Adjust
    (
      sfm_data_,
      Optimize_Options(
        ReconstructionEngine::intrinsic_refinement_options_,
        Extrinsic_Parameter_Type::ADJUST_ALL,
        Structure_Parameter_Type::ADJUST_ALL,
        Control_Point_Parameter(),
        this->b_use_motion_prior_)
    );
```

first BA

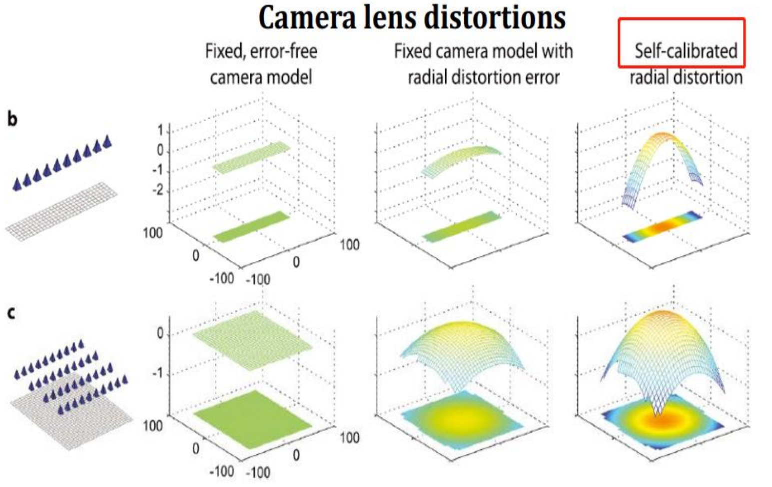
second BA

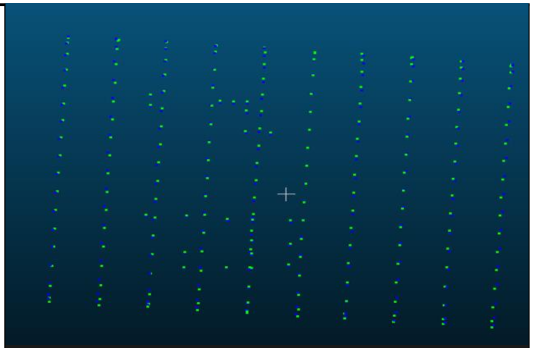
third BA

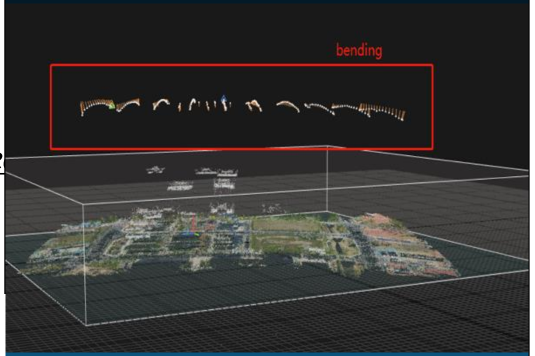
Why first BA fix oritation?

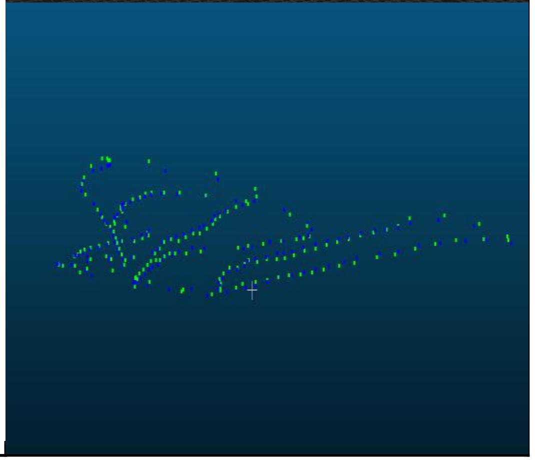
*first with a partial BA with fixed rotations, then with a global BA, is inspired by Olsson and Enqvist's approach .The idea is to prevent compensation of translation errors in the first step(relative pairwise rotation)by rotation adjustment, since rotations are more reliable.* (***reference***: ***<u>Global Fusion of Relative Motions for Robust, Accurate and Scalable Structure from Motion</u>***)

**No rigorous mathematical model**

<table>
<tr>
<td>How to deal the degeneration of global sfm</td>
<td>1) Calibrate the camera in advance<br><br>2) As for UAV samples, do not fly at a constant altitude on the flight path<br><br>3) Fusion of other sensors, such as GPS location constraints, etc(reference:<u>fusion of GPS and Structure-from-Motion using Constraine Bundle Adjustments</u>).<br>Good performance : large overlap<br><br><br></td>
</tr>
</table>



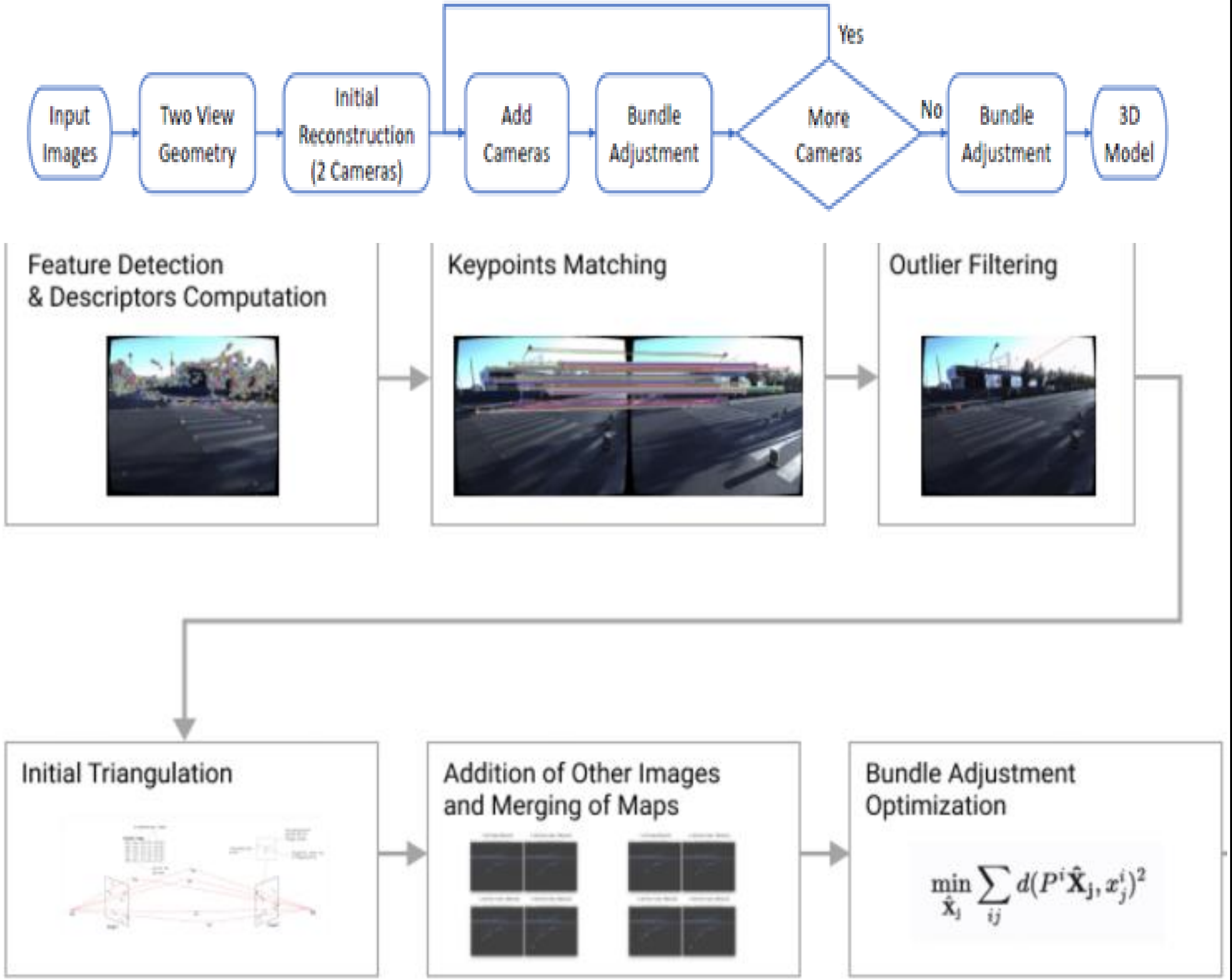


# SfM algorithm introduction

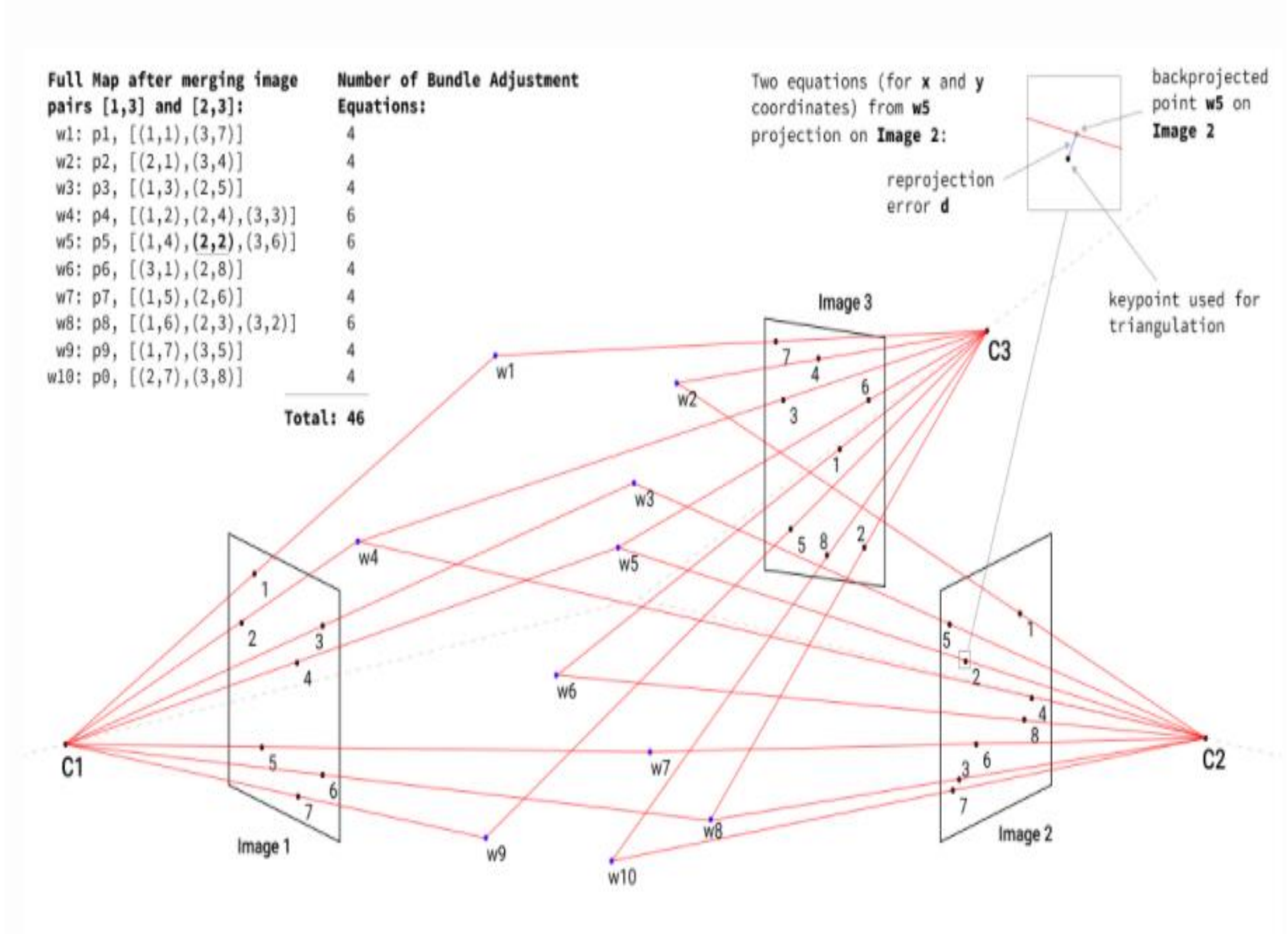
Incremental sfm

Workflow:



# SfM algorithm introduction



<table>
<tr>
<td>Incremental sfm</td>
<td>Bundle adjustment equations:


</td>
</tr>
</table>

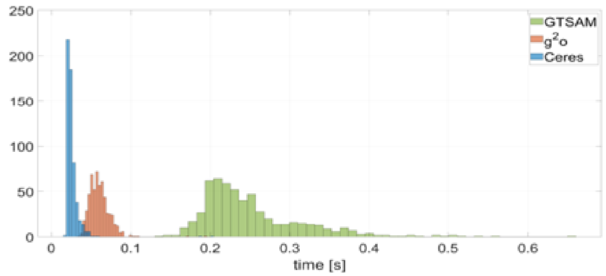
# SfM algorithm introduction



Incremental sfm problems

1)Large calculation time(mainly bundle adjustment will slow down as the number of images increases)

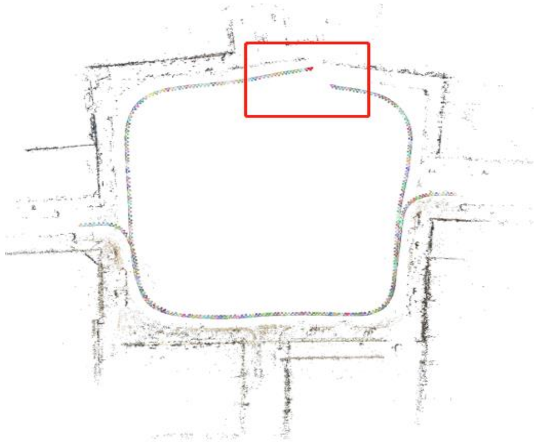
| | quantiles in [px] | | | |
|---|---|---|---|---|
| Frameworks | 25% | 50% | 75% | 99% |
| Ceres | 0.0105 | 0.0135 | 0.0172 | 0.0256 |
| GTSAM | 0.0104 | 0.0133 | 0.0173 | 0.0258 |
| g2o | 0.0107 | 0.0139 | 0.0176 | 0.0265 |




*Backend optimization frameworks*

*(**reference: Dense 3D-Reconstruction from Monocular Image Sequencesfor Computationally Constrained UAS**)*

2) Accumulation of errors caused drift

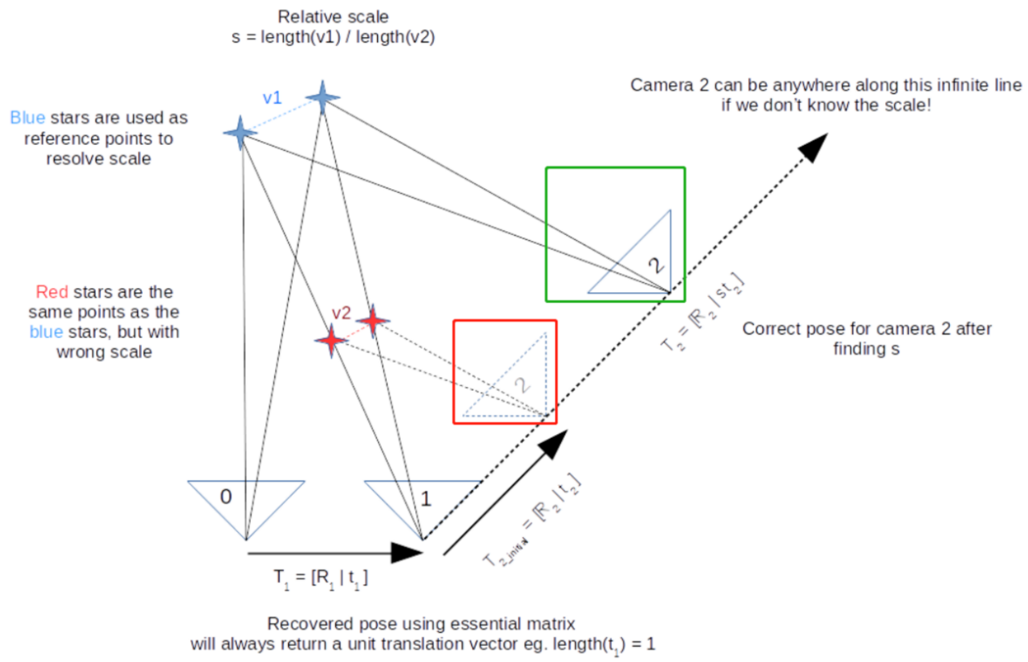


pipelines:

. Extract features

. Match features between all images

. Recover motion between previous to current image and triangulate points(*an ambiguity in the scale recovered from the essential matrix (hence only 5 degrees of freedom), don't know the length of the translation between each image, just the direction*)




. Run GTSAM bundle adjustment

# SfM C++ demo(opencv +gtsam)

*Coding demo is as follows:*

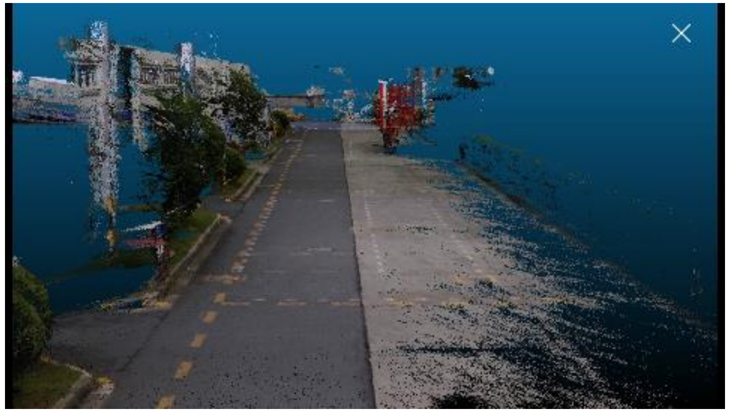
[https://github.com/yuancaimaiyi/sfm-opencv-gtsam-cmake-](https://github.com/yuancaimaiyi/sfm-opencv-gtsam-cmake-)

```
chengli@chengli-Legion-Y7000-2019:~/下载/SFM_example-master/src/build$ ./gtsam_sfm
Feature matching 0 1 ==> 1167/4505
Feature matching 0 2 ==> 401/4505
Feature matching 0 3 ==> 213/4505
Feature matching 0 4 ==> 81/4505
Feature matching 1 2 ==> 1248/4434
Feature matching 1 3 ==> 645/4434
Feature matching 1 4 ==> 275/4434
Feature matching 2 3 ==> 1076/4205
Feature matching 2 4 ==> 478/4205
Feature matching 3 4 ==> 1392/4468

initial camera matrix K
[1077, 0, 684;
 0, 1077, 456;
 0, 0, 1]

image 2 ==> 1 scale=1.23606 count=144453
image 3 ==> 2 scale=1.31063 count=129795
image 4 ==> 3 scale=1.09828 count=120786

initial graph error = 7604.91
final graph error = 124.631

final camera matrix K
    1067.89 6.57061e-06     693.519
          0      1062.5     447.738
          0           0           1
DSC02638.JPG roll:0.003299 pitch:0.00481253 yaw:0.000913918
DSC02639.JPG roll:-3.12814 pitch:-3.10734 yaw:3.05691
DSC02640.JPG roll:-3.11851 pitch:-3.00066 yaw:2.97453
DSC02641.JPG roll:-3.10082 pitch:-2.84757 yaw:2.81902
DSC02642.JPG roll:-2.98549 pitch:-2.63608 yaw:2.57972
```

**SfM application** :

. 3D reconstruction

1) Auto-drive(HD Map)



2) Ancient buildings reconstruction(Rome was built in a day)

# SfM application future and improvements

**SfM application** :

. Image mosaic(Emergency response)

Bilibili: https://www.bilibili.com/video/BV1uv411t7aX

# SfM application future and improvements



**SfM future and improvements :**

. Frontend algorithm(feature extracting and matching)

1) replace custom features with deep-learning features(SuperPoint 、d2-net,etc)- [https://github.com/ethz-asl/hfnet](https://github.com/ethz-asl/hfnet)

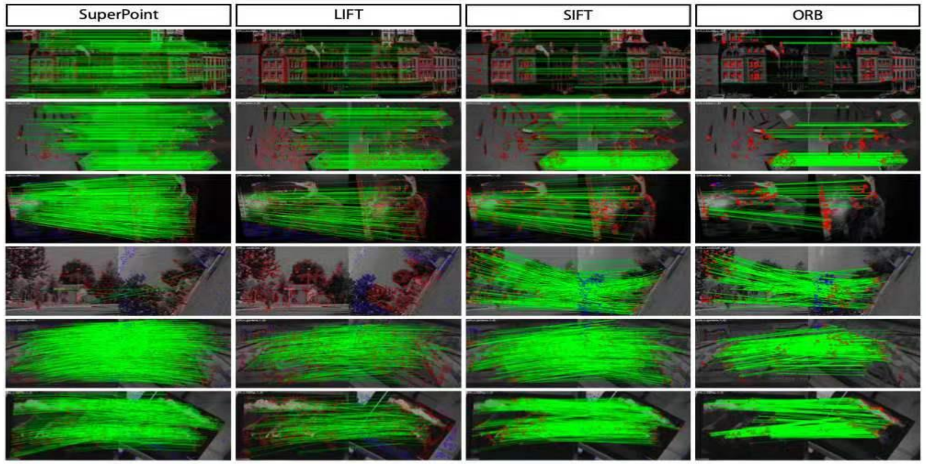



Figure 8. **Qualitative Results on HPatches**. The green lines show correct correspondences. SuperPoint tends to produce more dense and correct matches compared to LIFT, SIFT and ORB. While ORB has the highest average repeatability, the detections cluster together and generally do not result in more matches or more accurate homography estimates (see 4). Row 4: Failure case of SuperPoint and LIFT due to extreme in-plane rotation not seen in the training examples. See Appendix D for additional homography estimation example pairs.

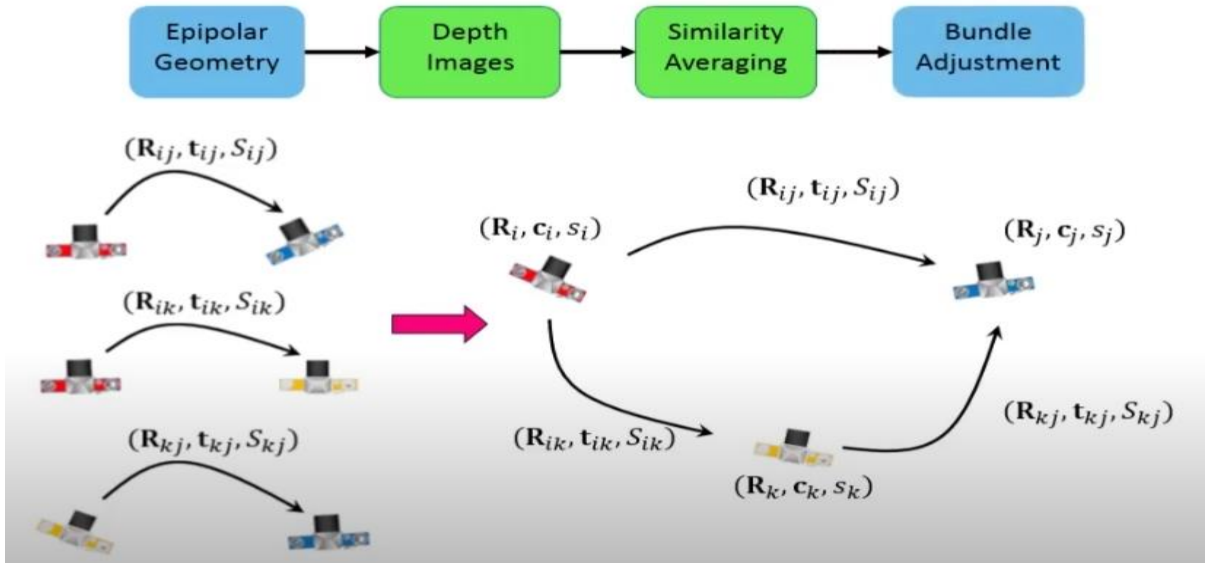
2) colmap (SiftGPU(2013) is rather old and not optimized with the latest GPU architecture ,replace with popsift- ***reference：*** ***Popsift: a faithful SIFT implementation for real-time applications*** )

**SfM future and improvements :**

. Global SfM

1) Due to the challenges faced by global, depth image of each camera help to upgrade an essential matrix to a similarity transformation(***reference:Global Structure-from-Motion by Similarity Averaging***)

https://youtu.be/UzXX9kbSrqg (*Zhaopeng Cui*)



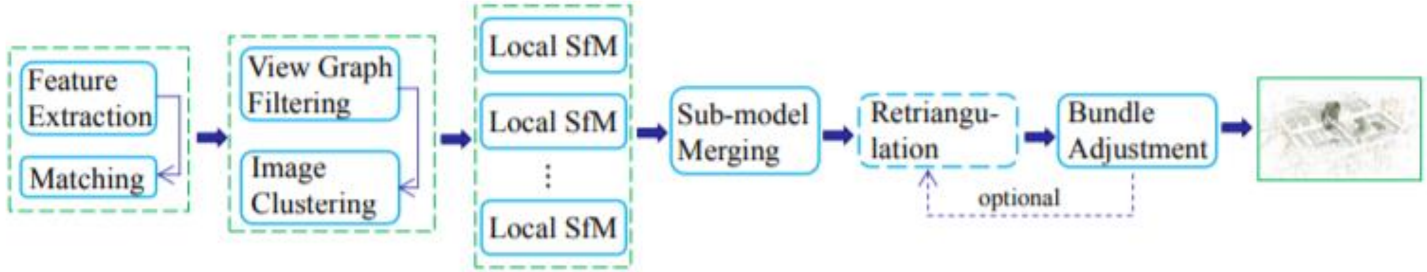
**SfM future and improvements :**

. Incremental SfM

1) Incremental SfM is more robust than global SfM, but the SfM's future trend is large-scale scene (***reference:DAGSfM: Distributed and Graph-Based Structure-from-Motion Library***)

Reference project:

https://github.com/AIBluefisher/DAGSfM (*Chenyu*)




2) The backend optimization framework-ceres solver is still slow(when the images are increasing) ,so manual differentiation by yourself instead of ceres solver automatic differentiation

## SfM future and improvements :

. Incremental SfM

3) Fusion other sensors,like GPS、imu,etc.(port openmvg Non-Rigid Registration module to colmap)

```
// Ceres CostFunctor used for SfM pose center to GPS pose center minimization
struct PoseCenterConstraintCostFunction
{
  Vec3 weight_;
  Vec3 pose_center_constraint_;

  PoseCenterConstraintCostFunction
  (
    const Vec3 & center,
    const Vec3 & weight
  ): weight_(weight), pose_center_constraint_(center)
  {
  }

  template <typename T> bool
  operator()
  (
    const T* const cam_extrinsics, // R_t
    T* residuals
  )
  const
  {
    using Vec3T = Eigen::Matrix<T,3,1>;
    Eigen::Map<const Vec3T> cam_R(&cam_extrinsics[0]);
    Eigen::Map<const Vec3T> cam_t(&cam_extrinsics[3]);
    const Vec3T cam_R_transpose(-cam_R);

    Vec3T pose_center;
    // Rotate the point according the camera rotation
    ceres::AngleAxisRotatePoint(cam_R_transpose.data(), cam_t.data(), pose_center.data());
    pose_center = pose_center * T(-1);

    Eigen::Map<Vec3T> residuals_eigen(residuals);
    residuals_eigen = weight_.cast<T>().cwiseProduct(pose_center - pose_center_constraint_.cast<T>());

    return true;
  }
};
```

## SfM future and improvements :

. Incremental SfM

3) Fusion other sensors,like GPS、imu,etc.(mavmap rotation constraint)

```cpp
// 增加角度约束
void _bundle_adjustment_add_pose_constraints(
        ceres::Problem& problem,
        FeatureManager& feature_manager,
        const std::vector<size_t>& free_image_ids,
        const std::vector<size_t>& fixed_image_ids,
        const double constrain_rotation_weight,
        const std::unordered_map<size_t, Eigen::Vector3d>& rotation_constraints,
        std::unordered_map<size_t, Eigen::Vector3d*>& rvecs) {

  // Determine rotation between constraints and SfM rotations from first
  // fixed image
  const size_t image_id = fixed_image_ids[0];
  // const Eigen::Vector3d rvec_FM = ;
  // const Eigen::AngleAxisd rot_FM(rvec_FM);
  const Eigen::Matrix3d R_FM
    = angle_axis_from_rvec(*rvecs[image_id]).toRotationMatrix();
  const Eigen::Matrix3d R_C
    = angle_axis_from_rvec(
       rotation_constraints.at(image_id)).toRotationMatrix();

  // "Difference" between rotations, i.e. rotation transformation from SfM to
  // constrained coordinate system
  Eigen::Matrix<double, 3, 4> matrix_FM_C
    = Eigen::Matrix<double, 3, 4>::Zero();
  matrix_FM_C.block<3, 3>(0, 0) = R_FM.transpose() * R_C;
  SimilarityTransform3D rot_FM_C(matrix_FM_C);

  // Rotate all camera poses and 3D points in feature manager
  for (auto it=feature_manager.rvecs.begin();
        it!=feature_manager.rvecs.end(); ++it) {
    rot_FM_C.transform_pose(it->second, feature_manager.tvecs[it->first]);
  }
  for (auto it=feature_manager.points3D.begin();
        it!=feature_manager.points3D.end(); ++it) {
    rot_FM_C.transform_point(it->second);
  }

  // Finally, add rotation constraints
  for (auto it=free_image_ids.begin(); it!=free_image_ids.end(); ++it) {

    const size_t image_id = *it;

    // rotation to be estimated
    Eigen::Vector3d* rvec = rvecs[image_id];

    // rotation constraint, e.g. from IMU sensor
    const Eigen::Vector3d& rvec0 = rotation_constraints.at(image_id);

    ceres::CostFunction* cost_function
      = BARotationConstraintCostFunction::create(constrain_rotation_weight,
                                                  rvec0);
    problem.AddResidualBlock(cost_function, NULL, rvec->data());

  }
```

```cpp
BARotationConstraintCostFunction::create(const double weight,
                                         const Eigen::Vector3d& rvec0) {
  return (new ceres::AutoDiffCostFunction
    <BARotationConstraintCostFunction, 1, 3>
      (new BARotationConstraintCostFunction(weight, rvec0)));
}


template <typename T> bool
BARotationConstraintCostFunction::operator()(const T* const rvec,
                                             T* residuals) const {

  T rotmat[9];

  ceres::AngleAxisToRotationMatrix(rvec, rotmat);

  // norm(rotmat.T - rotmat0, 'fro')

  // Using the inverse (transpose) as rotmat0 is the extrinsic rotation of
  // the camera

  residuals[0] = T(0);

  T diff;

  diff = rotmat[0] - T(rotmat0_[0]);
  residuals[0] += diff * diff;
  diff = rotmat[3] - T(rotmat0_[1]);
  residuals[0] += diff * diff;
  diff = rotmat[6] - T(rotmat0_[2]);
  residuals[0] += diff * diff;
  diff = rotmat[1] - T(rotmat0_[3]);
  residuals[0] += diff * diff;
  diff = rotmat[4] - T(rotmat0_[4]);
  residuals[0] += diff * diff;
  diff = rotmat[7] - T(rotmat0_[5]);
  residuals[0] += diff * diff;
  diff = rotmat[2] - T(rotmat0_[6]);
  residuals[0] += diff * diff;
  diff = rotmat[6] - T(rotmat0_[7]);
  residuals[0] += diff * diff;
  diff = rotmat[8] - T(rotmat0_[8]);
  residuals[0] += diff * diff;

  residuals[0] = T(weight_) * sqrt(residuals[0]);

  return true;
}
```

# 感谢聆听

Thanks for Listening


**李城**

**工作经历：**

**成都新橙北斗- 图形图像算法工程师**

**驭势科技-高精度地图算法工程师**

**(SfM、3D reconstruction、image mosaic)**