中国科学技术大学
University of Science and Technology of China

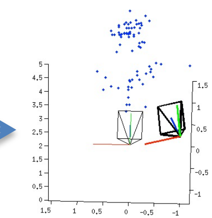
Bundle Adjustment

张举勇
中国科学技术大学

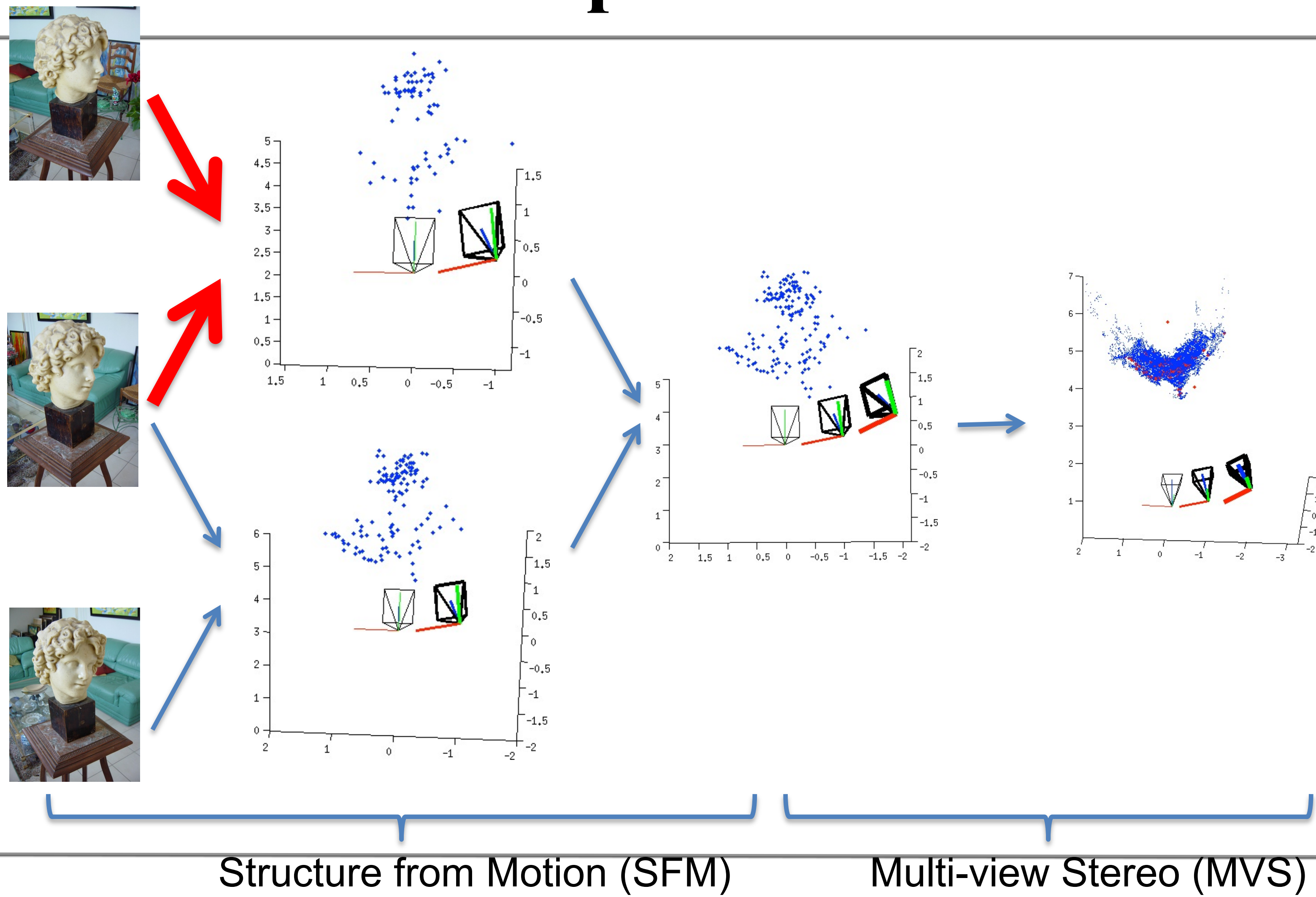
Two-view Reconstruction

keypoints → match
keypoints → match

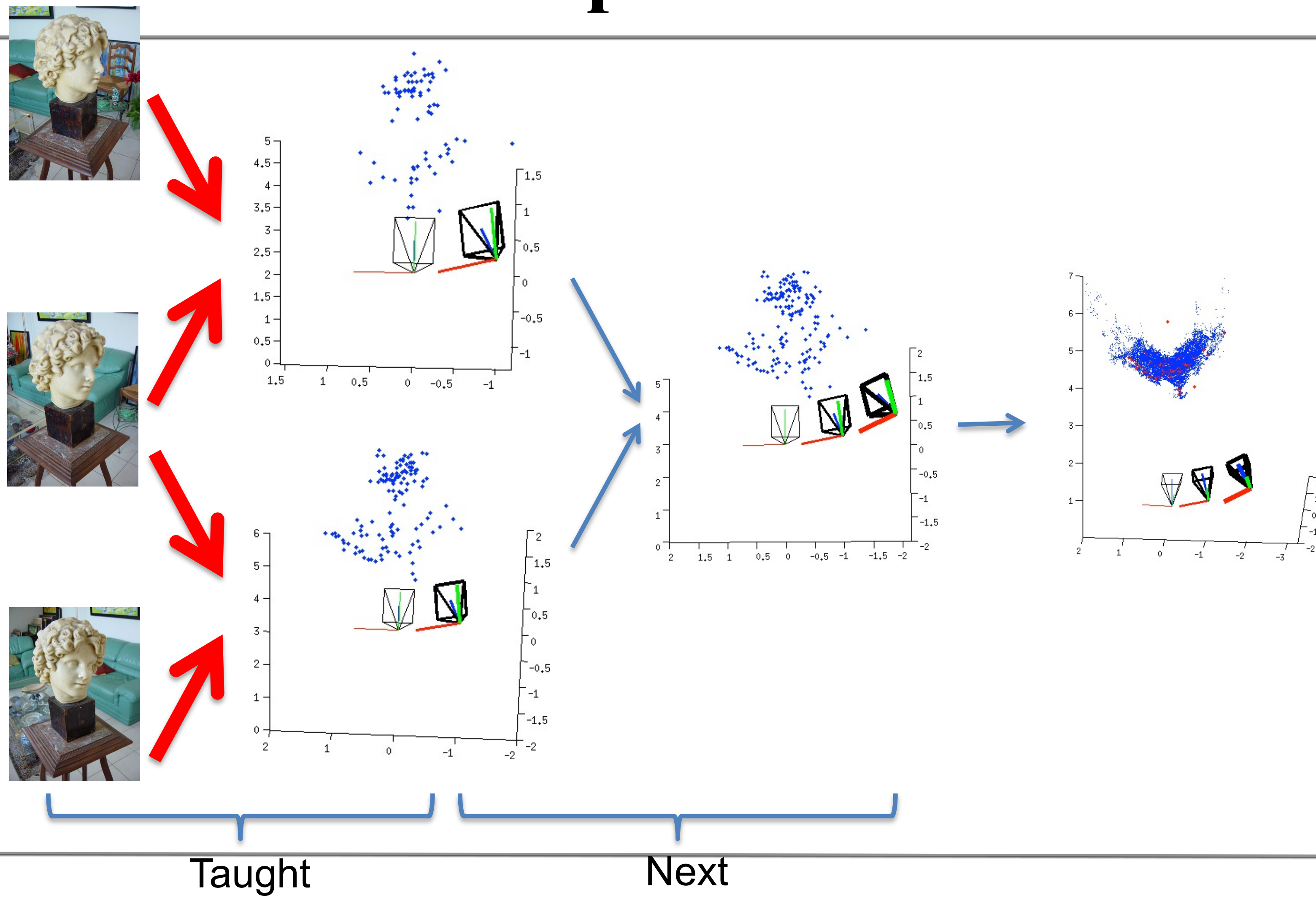
match → fundamental matrix → essential matrix → [R|t] → triangulation →



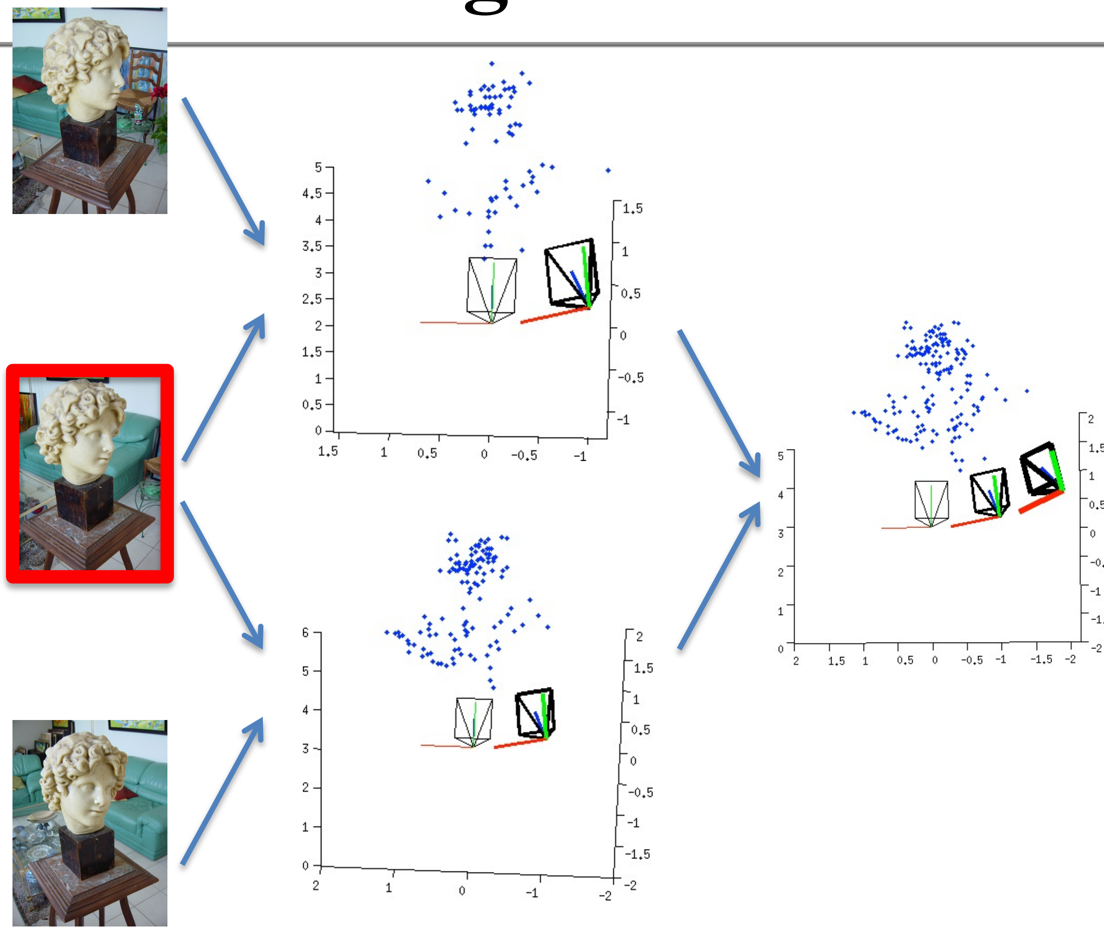
Pipeline



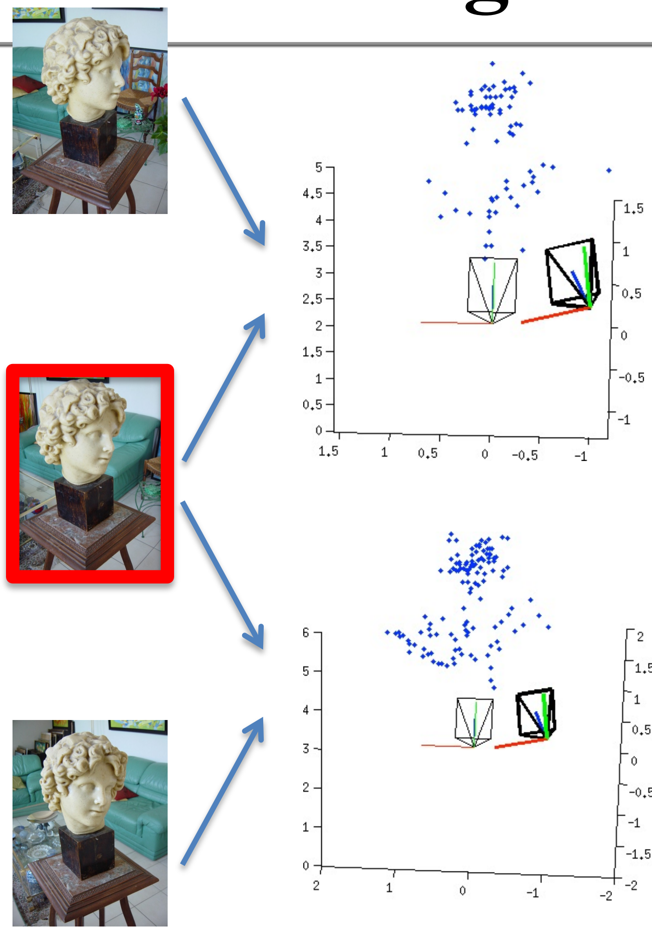
Pipeline

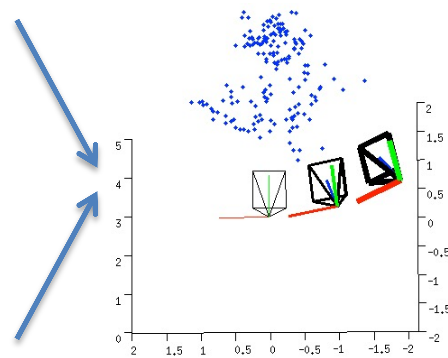


Merge Two Point Cloud



Merge Two Point Cloud





There can be only one $\left[\mathbf{R}_2 | \mathbf{t}_2\right]$

Merge Two Point Cloud

- From the 1st and 2nd images, we have

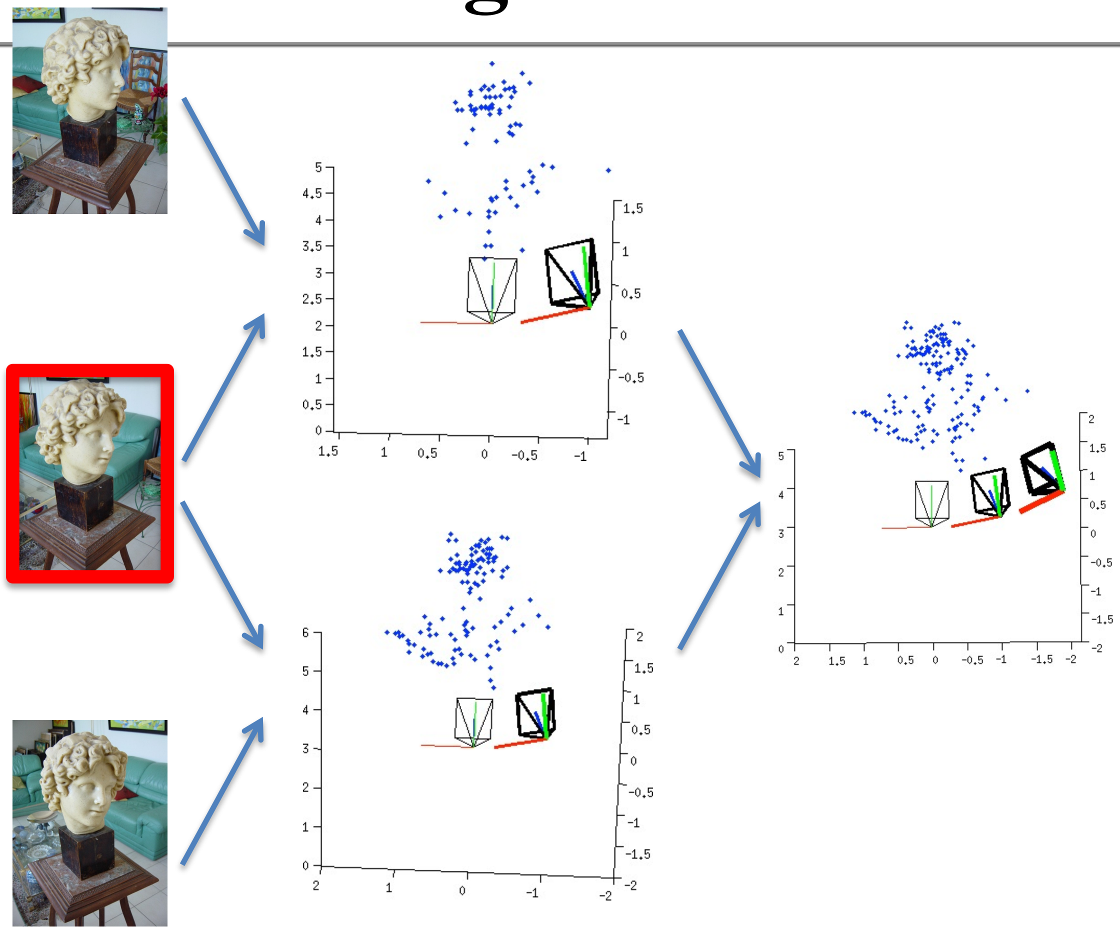
$$\left[\mathbf{R}_1|\mathbf{t}_1\right] \text{ and } \left[\mathbf{R}_2|\mathbf{t}_2\right]$$

- From the 2nd and 3rd images, we have

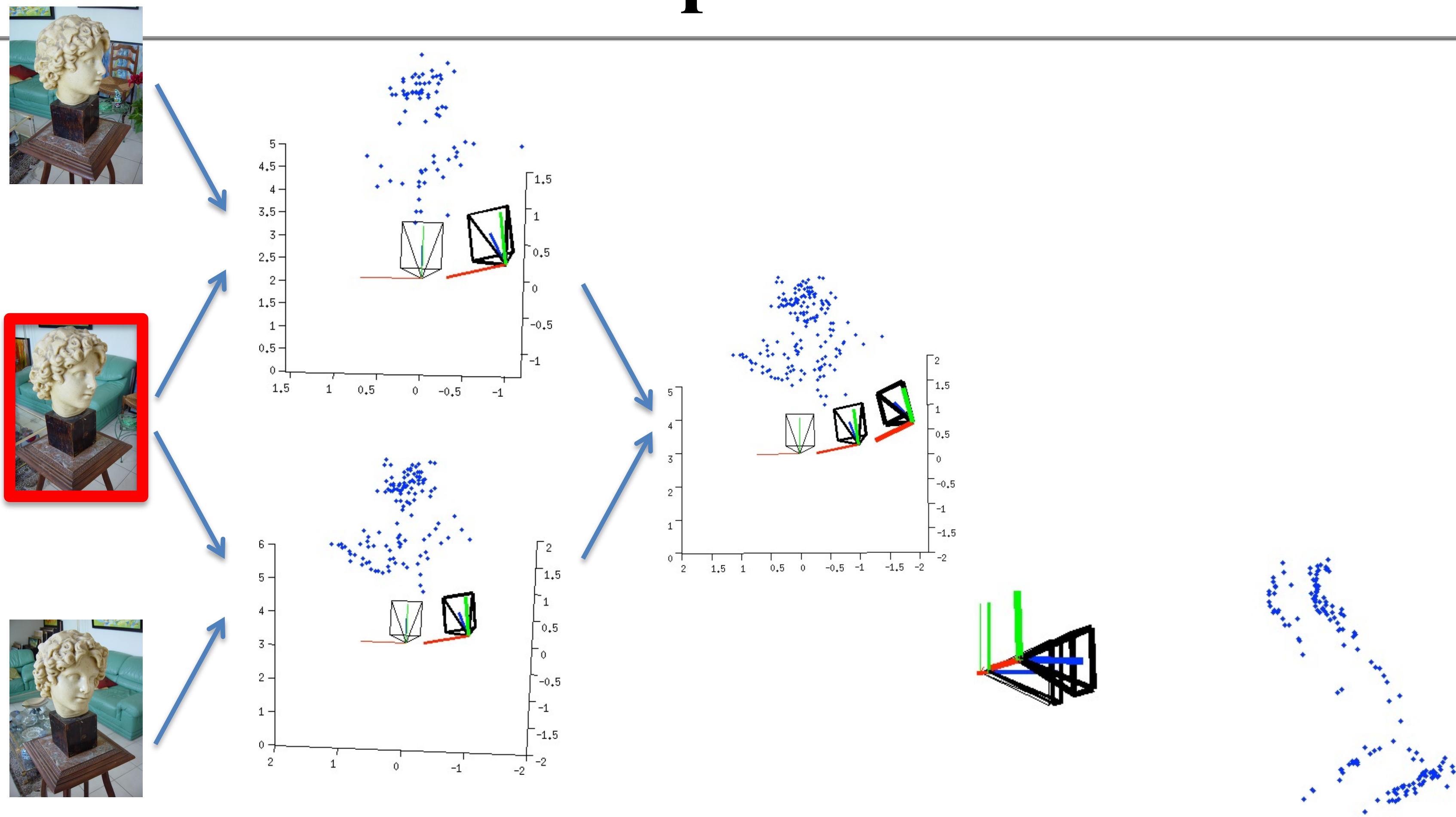
$$\left[\mathbf{R}_2|\mathbf{t}_2\right] \text{ and } \left[\mathbf{R}_3|\mathbf{t}_3\right]$$

- How to transform the coordinate system of the second point cloud to align with the first point cloud so that there is only one $\left[\mathbf{R}_2|\mathbf{t}_2\right]$?

Merge Two Point Cloud

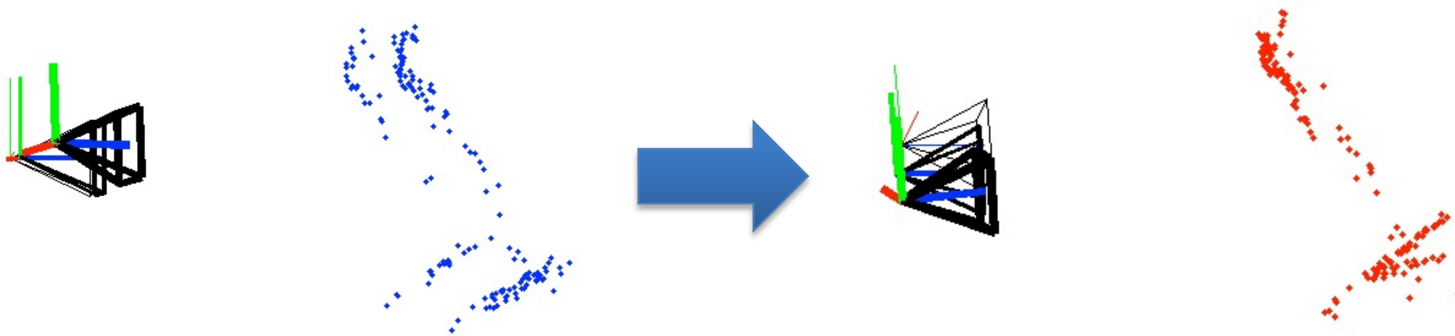


Oops



See From a Different Angle

Bundle Adjustment



Rethinking the SFM problem

- Input: Observed 2D image position

$$
\begin{matrix}
\tilde{\mathbf{x}}_1^1 & \tilde{\mathbf{x}}_1^2 & \\
\tilde{\mathbf{x}}_2^1 & \tilde{\mathbf{x}}_2^2 & \tilde{\mathbf{x}}_2^3 \\
\tilde{\mathbf{x}}_3^1 & & \tilde{\mathbf{x}}_3^3
\end{matrix}
$$

- Output:

Unknown Camera Parameters (with some guess)

$$\left[\mathbf{R}_1 \middle| \mathbf{t}_1\right], \left[\mathbf{R}_2 \middle| \mathbf{t}_2\right], \left[\mathbf{R}_3 \middle| \mathbf{t}_3\right]$$

Unknown Point 3D coordinate (with some guess)

$$\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \cdots$$

Bundle Adjustment

A valid solution $\left[\mathbf{R}_1|\mathbf{t}_1\right],\left[\mathbf{R}_2|\mathbf{t}_2\right],\left[\mathbf{R}_3|\mathbf{t}_3\right]$ and $\mathbf{X}^1,\mathbf{X}^2,\mathbf{X}^3,\cdots$ must let

$$\text{Re-projection}\left\{\begin{array}{lll} \mathbf{x}_1^1=\mathbf{K}\left[\mathbf{R}_1|\mathbf{t}_1\right]\mathbf{X}^1 & \mathbf{x}_1^2=\mathbf{K}\left[\mathbf{R}_1|\mathbf{t}_1\right]\mathbf{X}^2 & \\ \mathbf{x}_2^1=\mathbf{K}\left[\mathbf{R}_2|\mathbf{t}_2\right]\mathbf{X}^1 & \mathbf{x}_2^2=\mathbf{K}\left[\mathbf{R}_2|\mathbf{t}_2\right]\mathbf{X}^2 & \mathbf{x}_2^3=\mathbf{K}\left[\mathbf{R}_2|\mathbf{t}_2\right]\mathbf{X}^3 \\ \mathbf{x}_3^1=\mathbf{K}\left[\mathbf{R}_3|\mathbf{t}_3\right]\mathbf{X}^1 & & \mathbf{x}_3^3=\mathbf{K}\left[\mathbf{R}_3|\mathbf{t}_3\right]\mathbf{X}^3 \end{array}\right.$$

$$=$$

$$\text{Observation}\left\{\begin{array}{lll} \tilde{\mathbf{x}}_1^1 & \tilde{\mathbf{x}}_1^2 & \\ \tilde{\mathbf{x}}_2^1 & \tilde{\mathbf{x}}_2^2 & \tilde{\mathbf{x}}_2^3 \\ \tilde{\mathbf{x}}_3^1 & & \tilde{\mathbf{x}}_3^3 \end{array}\right.$$

Bundle Adjustment

A valid solution $[\mathbf{R}_1|\mathbf{t}_1],[\mathbf{R}_2|\mathbf{t}_2],[\mathbf{R}_3|\mathbf{t}_3]$ and $\mathbf{X}^1,\mathbf{X}^2,\mathbf{X}^3,\cdots$ must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

$$\min \sum_i \sum_j \left(\tilde{\mathbf{x}}_i^j - \mathbf{K}\left[\mathbf{R}_i|\mathbf{t}_i\right]\mathbf{X}^j \right)^2$$

Solving This Optimization Problem

- Theory:

  The Levenberg–Marquardt algorithm

  http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm

- Practice:

  The Ceres-Solver from Google

  http://code.google.com/p/ceres-solver/

Ceres-solver: A Nonlinear Least Squares Minimizer

Toy problem to solve $$\min(10 - x)^2$$

```
class SimpleCostFunction
  : public ceres::SizedCostFunction<1 /* number of residuals */,
                                    1 /* size of first parameter */> {
 public:
  virtual ~SimpleCostFunction() {}
  virtual bool Evaluate(double const* const* parameters,
                        double* residuals,
                        double** jacobians) const {
    const double x = parameters[0][0];
    residuals[0] = 10 - x;  // f(x) = 10 - x
    // Compute the Jacobian if asked for.
    if (jacobians != NULL && jacobians[0] != NULL) {
      jacobians[0][0] = -1;
    }
    return true;
  }
};
```

Ceres-solver: A Nonlinear Least Squares Minimizer

Toy problem to solve $\min(10-x)^2$

```
int main(int argc, char** argv) {
  double x = 5.0;
  ceres::Problem problem;

  // The problem object takes ownership of the newly allocated
  // SimpleCostFunction and uses it to optimize the value of x.
  problem.AddResidualBlock(new SimpleCostFunction, NULL, &x);

  // Run the solver!
  Solver::Options options;
  options.max_num_iterations = 10;
  options.linear_solver_type = ceres::DENSE_QR;
  options.minimizer_progress_to_stdout = true;
  Solver::Summary summary;
  Solve(options, &problem, &summary);
  std::cout << summary.BriefReport() << "\n";
  std::cout << "x : 5.0 -> " << x << "\n";
  return 0;
}
```

Ceres-solver: A Nonlinear Least Squares Minimizer

Toy problem to solve $\min(10 - x)^2$

```
0: f: 1.250000e+01 d: 0.00e+00 g: 5.00e+00 h: 0.00e+00 rho: 0.00e+00 mu: 1.00e-04 li:  0
1: f: 1.249750e-07 d: 1.25e+01 g: 5.00e-04 h: 5.00e+00 rho: 1.00e+00 mu: 3.33e-05 li:  1
2: f: 1.388518e-16 d: 1.25e-07 g: 1.67e-08 h: 5.00e-04 rho: 1.00e+00 mu: 1.11e-05 li:  1
Ceres Solver Report: Iterations: 2, Initial cost: 1.250000e+01,  \
Final cost: 1.388518e-16, Termination: PARAMETER_TOLERANCE.
x : 5 -> 10
```