Structure from Motion System in Modern Browsers

Pu Xu

Abstract

3D scene reconstruction is a crucial aspect of computer vision. In this paper, we will design a structure from motion system which take photos took by regular pinehole cameras as input and runs inside a modern bowser.

Photos took by regular digital camera i.e. pinhold camera model are the most common and aboundant material for scene reconstruction. On the other hand, the most pervasive computation platform is the web browser, which is compatible with virtually every platform regardless the difference of operating system or hardware. 3D scene reconstruction system on the most pervasive platform - web browser, using the most common source material - photos, minimized the requirements for this important task, has great potential.

Keywords: structure from motion, 3D scene analysis, javascript,

1 Introduction

The title “Structure from Motion System in Modern Browser” implies two key aspects, “Structure from Motion” and “Modern Browser”, and on both sides it has great potential.

On one hand, structure from motion has always been a classic problem in machine vision. It’s the problem of converting images or videos into 3D scene model, which is the exact reverse of CG(Computer Graphics). In resent years, structure from motion has became increasingly relavent because of the rising of technologies like VR (Virtual Reality), 3D printing and GIS (Geography Information System). Reconstruct 3D scene directly from regular photos i.e. follow pinehole camera model instead of specialized equipments can dramatically decreased the requirements for this important task, and open up the door to potentially unlimited resource for 3D reconstruction.

On the other hand, modern browser as a platform is gaining its popularity. It has became increasingly powerful over the years both in functionality and performance. Several novel features of the modern browser has made implemeting an SFM system possible. First is the WebWorkers. Javascript in browser is mostly a single threaded language. With WebWorker, parallel computing with clent-side javascript became possible. Another one is WebGL, which is the openGL interface for the web. It made the rendering of 3D model practical (SVG and canvas can also render 3D model, but not as convenient). Last but not the least, Indexed DB is also a very important element. It implements a light weight NoSQL database, enables object storage and file storage inside the browser. For projects dealing with a large dataset and need efficient data persistent such as WebSFM, they are no longer confined in the limited memory space, makes IndexedDB an extremely crucial feature.

In sum, the increasing need for 3D reconstruction in the industry and the maturity of web platform have made the SFM system in modern browser relavant and possible.

2 Overview

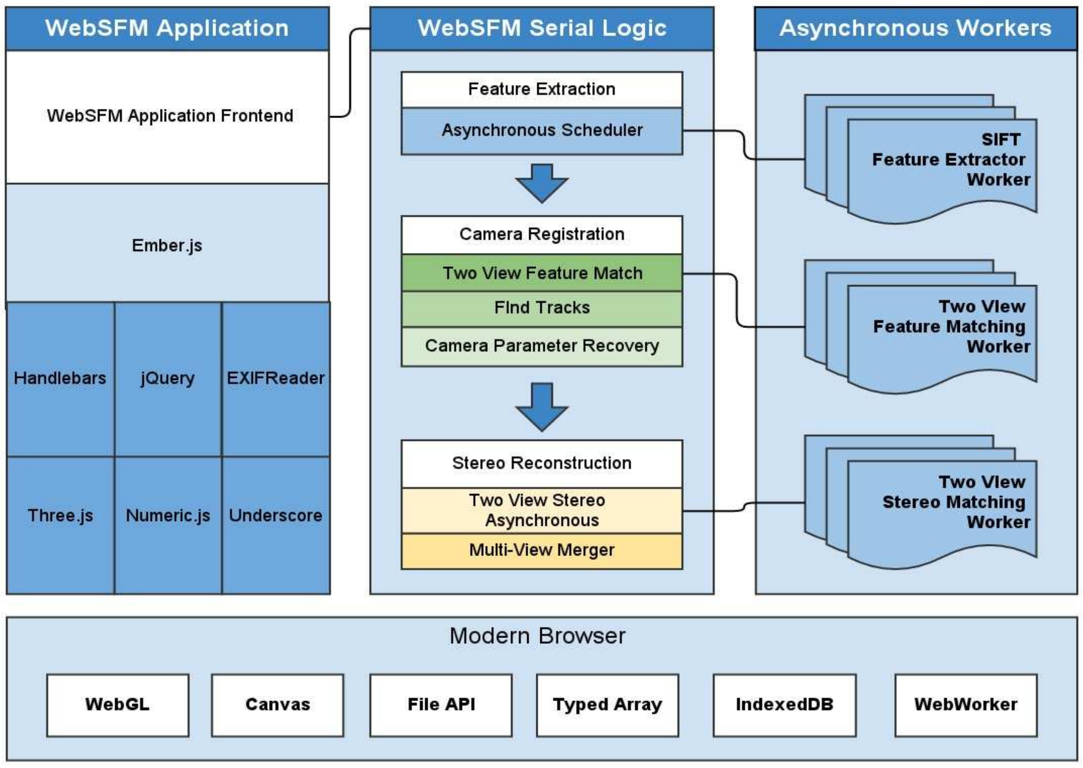

Figure 1. Framework of WebSFM

The main process of SFM system happens in the second thread dedicated for the serial logic of the SFM pipeline. The pipline is based on multiple exsiting implementations, such as [4][3]. And the sparse bundle adjustment part is influenced by SBA[2]. WebSFM has 3 phases:

First is feature extraction. Image features are extracted using Lowe's[1] SIFT. It uses Difference of Guassian of the grayscale image to approximate Laplassian of Guassian of the image. Feature points are first found in the Laplacian of Guassian then filtered, finally binded with a descriptor. Features are scale/orientation invariant, which mean they can be found and matched in arbitray scale and orientation.

Second is camera registration. We use the feature coorespondence between images to calibrate the cameras i.e. estimate the camera parameters. First we choose tracks i.e. globally unique and consistent feature matches. Then set up the initial pair of cameras. Afterwards, cameras are calibrated incrementally one after another, untill no reliable camera is left.

Finally is stereopsis. After cameras are calibrated, we can use stereo vision to obtain dense point cloud. For each pair of cameras, rectify their relative pose into standard stereo configuration, then search feature match along the scan line which coincides epipolar line because of rectification. Then merge the two view stereo together to form MVS (Multi-View Stereo).

Optionally, we can reconstruct surface or create mesh from the dense point cloud.

3 Mathematical Context

3.1 View Geometry

3.1.1 Coordinate System

Points in the SfM pipline have multiple cooridates according to different frames of reference.

- `Image Coordinate` – x
- `Camera Coordinate` – X_{cam}
- `World Coordinate` – X

Transition between these coordiate systems are accomplished by the use of homogenous coordinate and transition matrices.

- `Rotation Matrix(R) and Translation Vector(t)` – $X_{\text{cam}} = [R, t]X$
- `Calibration Matrix(K)` – $x = KX_{\text{cam}}$
- `Camera/Projection Matrix(P)` – $x = PX = K[R, t]X$

3.1.2 View Constrains

Two-view constrains:

- `Fundamental Matrix(F)` – $x_1^T F x_2 = 0$
- `Essential Matrix(E)` – $X_{\text{cam1}}^T E X_{\text{cam2}} = 0$

Fundamental matrix is used for un-calibrated cameras, essential matrix is used for calibrated cameras, their relation is $F = K_1^T E K_2$.

3.1.3 Eight Point Algorithm (Normalized)

Recover fundamental matrix of a two-view pair.

To be more specific, the algorithm used here is normalized 8-point-algorithm. it first normalize the image coordinates into $[-1, 1] \times [-1, 1]$.

$$x = Nx = \begin{pmatrix} 2/\text{width} & 0 & -1 \\ 0 & 2/\text{height} & -1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

the epipolar constrain of fundamental matrix and a pair of matching points can be represented as:

$$x^T F x' = \begin{pmatrix} x & y & 1 \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = 0$$

it can also be written as:

$$x' x f_{11} + x' y f_{12} + x' f_{13} + y' x f_{21} + y' y f_{22} + y' f_{23} + x f_{31} + y f_{32} + f_{33} = 0$$

or in matrix form:

$$\begin{pmatrix} x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \end{pmatrix} \begin{pmatrix} f_{11} & f_{12} & f_{13} & f_{21} & f_{22} & f_{23} & f_{31} & f_{32} & f_{33} \end{pmatrix}^T = 0$$

each row can also be written as the fatten version of $x'^T x$.

the solution shold be non-zero and up to scale, so 8 points are needed, solve can be obtained by SVD, use the sigular vector with the minimum sigular value as the solution.

By epipolar constrain, determinant of fundamental matrix is always 0 i.e. singular. So after the solution is found, we SVD the fundamental matrix

$$F' = U S V^T$$

then set its smallest singular value to 0

$$S' = \mathrm{dia}\begin{pmatrix} s_1 & s_2 & 0 \end{pmatrix}$$

then multiply them together again, so the epipolar constrain is enforced.

$$F'' = U S' V^T$$

at last, reverse the normalization process, obtain the final result.

$$F = N_1^T F'' N_2$$

3.1.4 Six Point Algorithm

six coorespondence yields the projection matrix(3*4).

get projection matrix

$$x = PX = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

each point can be written as two equations

$$x = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}, y = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}$$

the unknown matrix P can be represented as a vector:

$$p = \begin{pmatrix} p_{11} & p_{12} & p_{13} & p_{14} & p_{21} & p_{22} & p_{23} & p_{24} & p_{31} & p_{32} & p_{33} & p_{34} \end{pmatrix}^T$$

and the equation can turn it into:

$$\begin{pmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ & -x \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -yX & -yY & -yZ & -y \end{pmatrix} p = 0$$

each row can also be written as the flatten version of

$$\begin{pmatrix} 1 \\ 0 \\ -x \end{pmatrix} X^T \text{and} \begin{pmatrix} 0 \\ 1 \\ -y \end{pmatrix} X^T$$

with 6 pairs of coorespondence, 12×12 matrix A can be obtained:

$$\mathrm{Ap} = 0$$

Solution of p can be obtained by SVD the matrix A.

3.1.5 Two Point Algorithm (Triangulation)

Recover the coordinate of a point in the world coordiante system from two calibrated camera and th projection coordinates of the point on each image plane. The constrains can be written as:

$$\begin{cases} x_l = P_l X_l = (\; m_l^3 \;\; m_l^3 \;\; m_l^3 \;)^T X \\ x_r = P_r X_r = (\; m_l^3 \;\; m_l^3 \;\; m_l^3 \;)^T X \end{cases}$$

A linear equation can be obtained.

$$\begin{pmatrix} x_l m_l^{3T} - m_l^{1T} \\ y_l m_l^{3T} - m_l^{2T} \\ x_r m_r^{3T} - m_r^{1T} \\ y_r m_r^{3T} - m_r^{2T} \end{pmatrix} P = AP = 0$$

and the solution of P can be found by SVD the matrix $A^T A$.

3.1.6 Standard Stereo Rectification

Rectification is the process of changing the relative pose of a pair of cameras into standard stereo configuration. It is essentially a homography of the original images. The three steps are: First send the epipoles to infinity by rotating both cameras by R_{rect}.

$$R_{\text{rect}} = [e_1 \, e_2 \, e_3]^T$$

Each row of R_{rect} is obtained by the following algorithm.

$$\begin{cases} e_1 = \dfrac{T}{\|T\|} \\ e_2 = \dfrac{1}{\sqrt{T_x^2 + T_y^2}} [-T_y \, T_x \, 0]^T \\ e_3 = e_1 \times e_2 \end{cases}$$

Second, rotate the reference camera by the relative roation of the other camera R so the epipolar lines became parallel. Finally, adjust the focal length so two image plane coincides.

3.2 RANSAC

RANSAC(RANdom SAmple Consensus) is a data filter used cross the entire system. In a set of samples with an underlying constrain, but contains both inliers and outliers, RANSAC attempts to select a small inlier-only subset by random selection within a finite number of trials and estimate this underlying constrain to further filter the dataset. RANSAC contains three major components:

- `constrain` – the underlying constrian of the samples
- `constrain generator` – the algorithm to estimate the constrain from a subset of samples
- `dataset` – the "dirty" data set

And three parameters:

- `tolerance` – tolerated error
- `threshold` – estimated outlier percentage
- `N`– quota of trials

The process of RANSAC is a finite loop. Until no trials left, it randomly choose a subset of samples to estimate the underlying constrain e.g. fundamental matrix in two-view matching, then test the entire sample set, and calculate the percentage of rejected samples, if its greater than the threshold, this randomly selected subset may contain outlier which causes the estimated constrain to be inaccurate, and retry. If its less than the threshold, this estimated constrain is considered to be accuate, so we break out the loop and filter the dataset with this estimated constrain, and filter out the outliers using the estimated constrain.

3.3 Bundle Adjustment

Levenberg-Marqurdt Algorithm (LM) is the strategy we choose to perform non-linear optimization. Its the process to find the optimal solution so the difference between result and the measurement vector is minimized:

$$minimize\, S(p) = \|f(p) - y\| = \|\boldsymbol{\epsilon}_p\|$$

In our case, we always use bundle adjustment to minimize error, so the measurement vector is assumed to be zero.

In a small neighborhood, delta of the measurement is approximate to the prodect of Jacobian matrix and the step vector $\boldsymbol{\delta}_p$:

$$f(p+\boldsymbol{\delta}_p) \approx f(p) + J\boldsymbol{\delta}_p$$

from which we obtian the equation:

$$J^T J\boldsymbol{\delta}_p = J^T\boldsymbol{\epsilon}$$

where

$$N = J^T J \approx \text{Hessian}(\|\boldsymbol{\epsilon}_p\|)$$

What make the LMA special is it uses damping, the damped equation:

$$(N+\boldsymbol{\mu}I)\boldsymbol{\delta}_p = J^T\boldsymbol{\epsilon}$$

when far from optiamal, the damping value is large and the algorithm bahaves like steepest gradient descent, and when close to the optimal solution, damping value became smaller and behaves like Guass-Newton method. The damping value is updated during the iteration based on the current solution. The inital paramters are:

$$\boldsymbol{\mu}_0 = \max(N_{\text{ii}}), \boldsymbol{\lambda} = 2$$

For each iteration, if the solution with damp value $\boldsymbol{\mu}/\boldsymbol{\lambda}$ improved the solution, then the current optimal is updated and $\boldsymbol{\mu}/\boldsymbol{\lambda}$ became the default damp value. Otherwise, if the current default damp value yields better solution, then the current optimal is updated and the damp value is unchaged. When both $\boldsymbol{\mu}/\boldsymbol{\lambda}$ and $\boldsymbol{\mu}$ do not yield better solution, than multiply the current damp value by $\boldsymbol{\lambda}$ until the equation yields better solution than the current optimal.

4 Extract Image Features

There are several desirable feature extractors, we choose SIFT because the features it generates are scale/orientation invariant, and most existing SfM implementation uses SIFT as features extractor and yield disirable results.

4.1 Scale Invariant

Features obtained from SIFT is scale invariant, which means features can be found and matched regardless what scale it is in. To accomplish that, we need to find feature points in LoGs (Laplacian of Guassian) of the image, due to the complexity of computing the second derivative of an image, we use DoGs(Difference of Guassian) to approximate. This is acomplished by using a series of octaves and scales of the original image, they simulates the image in various scales. We use 4 octave times 5 scales as Lowe's [1] paper suggested.

To construct octave space, we directly shrink size of the image by half, each pixel is obtained by averaging the grayscale value of its window.In each octave, scalespace are constructed by convolution of the image with a series of a guassian kernels. The sigma of the guassian kernel is based on the relation between sigma of LoG and DoG.

$$L_{\boldsymbol{\sigma}} = \nabla^2 G_{\boldsymbol{\sigma}} = G_{\boldsymbol{\sigma}1} - G_{\boldsymbol{\sigma}2} \qquad \boldsymbol{\sigma}_1 = \boldsymbol{\sigma}/\sqrt{2}\,, \boldsymbol{\sigma}_2 = \sqrt{2}\boldsymbol{\sigma}$$

Sigma of guassian kernels in each octave is propotional by the factor of 2 to satisfy the euqation above. After scalespaces are obtained, we can approximate LoG using DoG (Difference of Guassian) between adjacent scales.

Afterwards, several filters are applied to choose high quality feature points. Contract filter will only accept maxium and minimum points in the DoG pyramid i.e. $3 \times 3 \times 3$ window around the point. Corner filter will only accept corner by comparing the derivative of the point on each direction. Finally, scale invariant feature points are selected.

4.2 Orientation Invariant

After finding scale inavriant feature points, we assign each feature point with an orientation from its neighborhood, then assign a discriptor to the point according to its orientation and neighborhood, so the descriptor is self-oriented, can be matched regardless the orientation of images. All following process are done in the DoG image where the point is found.

To calculate the orientation, we collect the gradients inside a window around the feature point and construct a histogram of gradient orientations, where orientations are divided into 36 discrete bins. All directions over 80% of the maxium is considered an valid orientation, which means one feaure point might coorespond to several oriented feature points.

Finally, to calculate the descriptor, we collect gradient informations from a 16×16 window around the point aligned with its orientation. Gradient dirction are guass-weighted and relative to the orientation of feaure point. The window is then divided into a 4×4 grid of 4×4 sub-windows. In each sub-window, collect the histogram of gradient orientations with 8 bins. Afterwards, it will yield $16 \times 8 = 128$ numbers as the descripter. Finally, achive illumination invariant by normalizing the descriptor.

4.3 Usages

SIFT will generate a huge amount of feature points, it will be used in both camera registration phase and stereopsis phase. But only a small subset of features will be used in the camera registration phase, it's because of camera registratration need features that are globally unique, so it can be used to esitimate camera pose and parameters.

5 Camera Registration

Camera registration i.e. camera calibration is the process of estimating camera parameters. Calibration matrix K, rotation matrix R and translation vector t are the three parameters we need to obtain in the calibration process.

Intrinsic calibration is the process of obtaining the calibraton matrix K. The primary variable of K is focal length in pixel. Ideally we can obtain focal length in pixel from EXIF information, but in practice, focal length are given by mm, and most EXIF tag does not have CCD size or alternative attributes which allow us to calculate focal length in pixel, so we have to esitimate it by track coorespondence.

Extrinsic calibration will recover the rotation matrix R and translation vector t. They can be extracted from both projection matrix and essential matrix, depend on the choice of algorithm.

Camera parameters are very sensitive in view geometry, it will affect the estimated cooridante of every point it observes, and the relative pose bwtween cameras, consequently affect the entire system. Although bundle adjustment can refine the parameters, but sparse bundle adjustment only promises local optimization, the inital solution is still critical.

On the other hand, each image yields up to 10,000 features, but camera calibration algorithms require only a very small amount of matches e.g. DLT, 5-point-algorithm, so outliers in the input may affetct the result dramatically.

So, from all points the feature extractor provides, choose matches with high percision, high contrast and globally unique is a crutial part of camera registration, and those points are called tracks, A track represents a distinct point in the scene, it have following constrains:

- Satisfy all geometric constrains
- Does not have multiply matches in a single view i.e distinct

5.1 Match Tracks

To select the desirable tracks, first match features for each pair of images. Instead using a threshold of the euclidean distance of two descriptor to indicate a match, we use a nearest neighbor strategy. When comparing two inages, for each feature in image 1, we find the 1st and 2nd nearest feature in image 2 and compare the ratio of the euclidean distance, if less then the ratio threshold, then accept as a match, which means it's a match and the only match in this image. then, use view constrains to filter the matches. for each image pair, use two view constrain, fundamental matrx. use RANSAC to estimate the fundamental matrix and filter outlier matches.

To satisfy the second requirment, we construct image connectivity graph from all feature matches, tracks that contains multiple matches on a single image will be discarded. In this process, not only inconsistencies are filtered, we also merged two-view matches into a global image connectivity graph and global tracks, which can be used in next phase.

5.2 Register the First Pair of Cameras

The first pair of camera is crutial, we need to calibrate the inital pair of cameras and triangulate the inital set of tracks, it will be used in the incremental recovery phase to calibrate new cameras.

We first guess the focal length in pixel. Then use 8-point-algorithm to estimate the fundamental matrix of the inital pair, and obtain its essential matrix using the guessed focal length, set one of the cmaera's local coordinate system as the world coordinate system i.e. set its rotation as I and translation as 0. Then carry out the other camera's extrinsic parameters from essential matrix. Then we select the tracks that are visible in this pair of cameras, and triangulate their world coordinates. Finally a global bundle adjustment is applied to refine the parameters, especially the focal length.

5.3 Incremetal Recovery

After the intial pair of cameras and tracks are registered, we can calibrate the rest of cameras one by one.

First find the next camera by selecting the one observes the most established tracks. Estimate its camera matrix from the coorespondence between each tracks's world coordinate and image coodinate using 6-point-algorithm. Then a bundle adjustment is applied, only the new camera and tracks it observes can change. With camera matrix, we can obtain the calibration matrix, rotation matrix and translation vector using QR decomposition. Afterwards, triangulate tracks that are visible in the new camera. After each new track is added, a global bundle adjustment is applied. Repeat the process above until all cameras are registered or no camera observes enough tracks left.

6 Stereopsis

Stereopsis phase genterates a dense point cloud. It first locally reconstruct each two-view stereo, then merge them together with global constrians. Such as depth, visibility and epipolar constrain.

6.1 Two-View Stereo

To perform two view stereo reconstruction, change the camera pose into the standard stereo configuration is recommanded, make epipolar line coincide the image's scan line, make disparty matching easier

we use feature based stereo for its simplicity, so only scan the feature points generated by SIFT, and compare them by their descriptor.

first step is image rectification, we need to convert the relative pose of cameras into standard stereo configuration. it is essentially a homography of the original image. Because of we use feature based stereo, instead of generate a new pair of images, we calculate the homography matrix and transform the feature points' coordinate into the new rectified plane, and arrange them each point's by scan line.

then we search for feature coorespondence in each scan line using nearest neighbor strategy. Because of the posible inaccuracy of feature coodinates, we search a set of near by scan lines. Instead of using the radio of two closest neighbor as threshold in the camera register phase to ensure its uniqueness, we use a simple distant threshold to filter out invalid matches because the search is already limited to a epipolar line i.e. scan line.

after the coorespodence are found, we convert them back to the original coorinates and triangulate them to obtain their coodinates in the scene.

6.2 Multi-View Stereo

merge the two view stereos together. In two-view stereo matching process, points are triangulated within the two-view domain. From the matching lists we find all matches for a specific point and choose the optimal baseline then triangulate it again.

7 Application Framework

7.1 Parallelization

There are two long-live threads:

- DOM Thread – The default thread created by the browser, have full access to the DOM, the application logic of WebSFM is inside it.
- WebSFM Thread – SfM thread is a singleton thread created by the DOM thread, represents the serial logic of the SFM system.

And three short-term threads

- SIFT feature generator thread
- Two-View feature matching thread
- Two-View stereo matching thread

Multiple short term threads are created dynamically by the WebSFM thread according to the number of cores assigned to the application. They are implemented to be asynchronous. Each represents a worker for a task with no dynamic data dependency in the SFM system. For simplicity and performance concern, we only extracted three tasks to be asynchronous and parallel, all of them are well comfined in a specific scope – a two-view pair or a single image.

7.2 Browser Related Details

There are some specific features that are crutial to the implementation of WebSFM, they provided us with a much more advanced programming environment in comparasion to the traditional client side javascript.

- Typed Arrays – Efficient data structure

- 2D Canvas – Image Processing
- IndexedDB – Efficient data persistent
- WebWorkers – Parallel computation
- WebGL – 3D Visualization

7.3 Samples

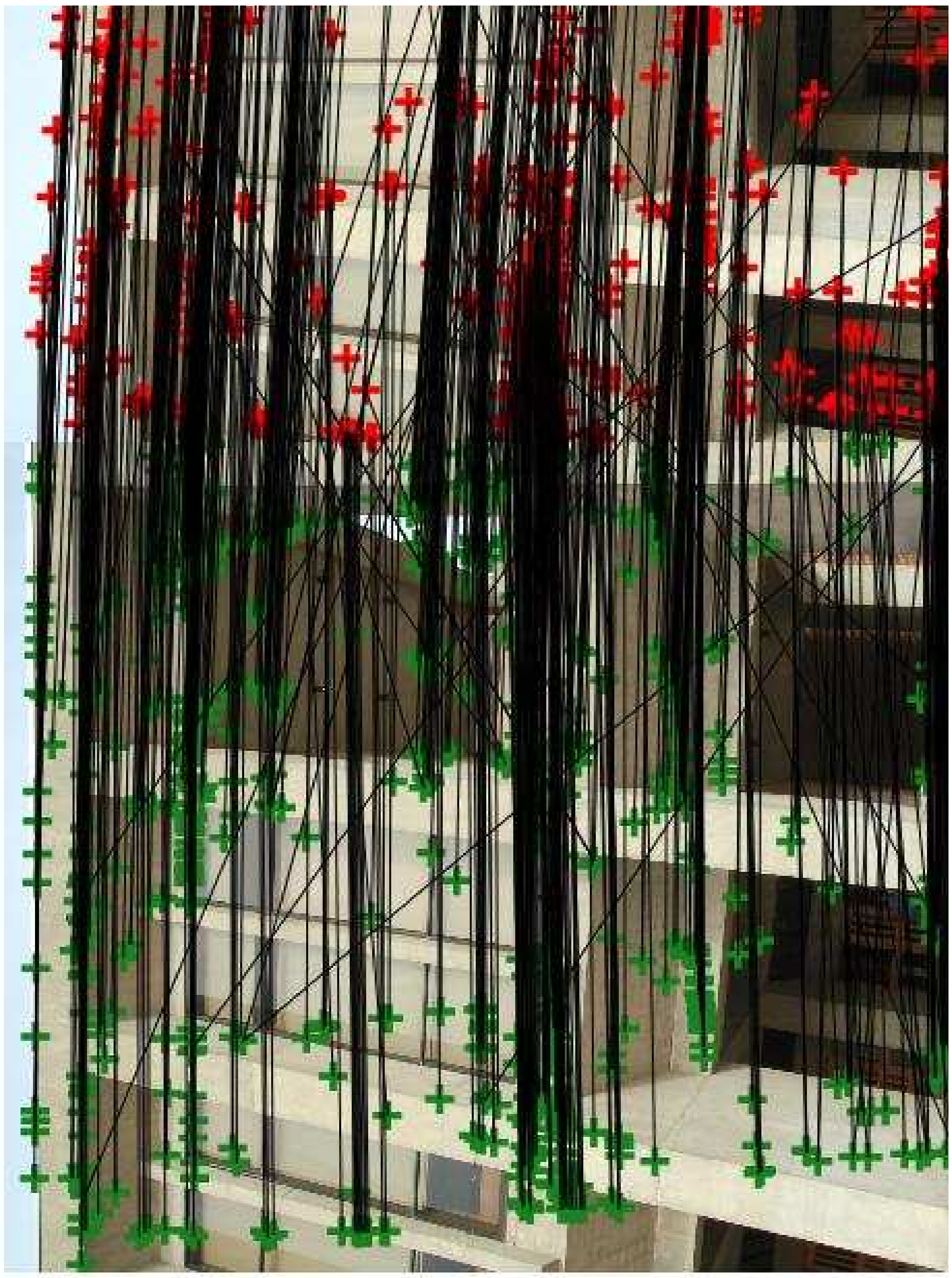

Figure 2. Two view matching using features generated by Lowe's SIFT

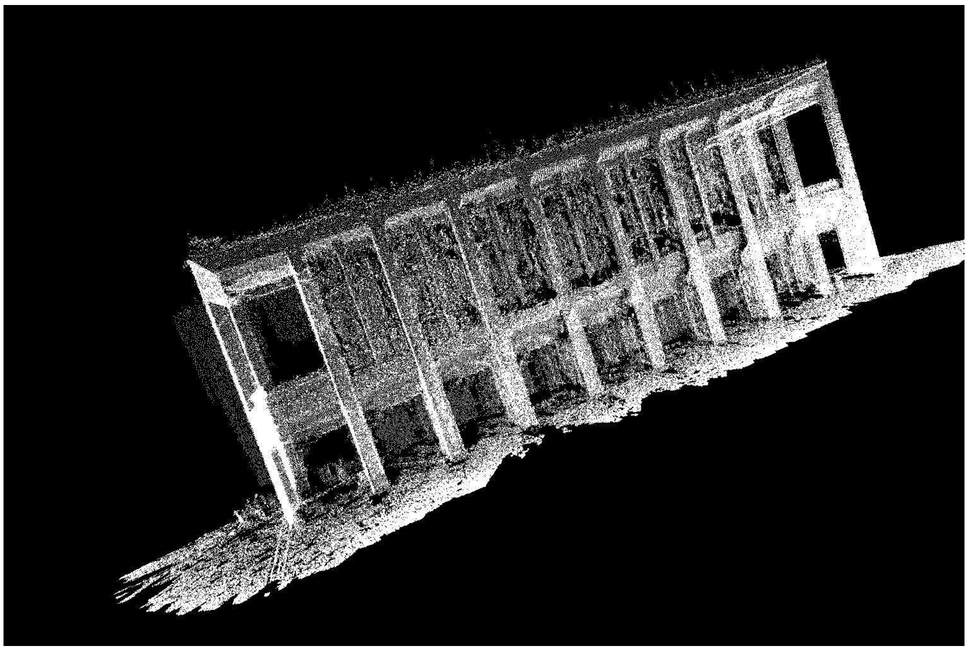

Figure 3. Dense point cloud generated by PMVS

Bibliography

[1] Distinctive image features from scale-invariant keypoints. *IJCV*, , 2004.

[2] The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. , , 2004.

[3] Modeling the world from internet photo collections. , , 2007.

[4] Visual modeling with a hand-held camera. .