

# (CV PA2) Structure From Motion

20225092 Huigyoeng Son

## Abstract

본 프로젝트에서는 다중 viewpoint에서 얻은 물체 이미지로부터 물체의 3차원 구조를 복원하는 **SFM(Structure From Motion)**을 구현하였다. 먼저, 처음 두 개의 viewpoint에 대해 5 point algorithm을 통해 **Two View SFM**을 수행한 후, 다른 viewpoint들에 대해서 차례대로 3 point algorithm과 bundle adjustment를 적용하여 **Multi View SFM**을 수행하였다. 각각의 SFM에서 point algorithm은 camera pose를 추출하기 위해 사용되는데, 이때 RANSAC 추정 방법을 사용해 신뢰도가 가장 높은 값을 사용하였다. camera pose를 추출하면 각 이미지의 matching point들에 대해 triangulation을 적용하여 3d cloudpoints를 구할 수 있다. 실험은 카메라 파라미터가 일정하다는 조건 하에 여러 viewpoint에서 촬영한 물체들에 대해 수행되었으며, 사용한 물체는 기본 데이터셋(moai, choonsik, nike, toothless)과 커스텀 데이터셋(eiffeltower, dinosour, bluedoll, dog)이다. Two View SFM에서 물체의 부분적인 3차원 구조를 복원할 수 있음을 확인하였고, Multi View SFM으로 물체의 더 완전한 3차원 구조를 복원할 수 있었다. 우리는 특히 물체 moai와 eiffeltower와 같이 특징점 정보가 많은 물체들에 대해 결과가 잘 나오는 것을 확인할 수 있었다. 해당 프로젝트에 사용한 코드는 다음의 github에서 확인할 수 있다: <https://github.com/gyoenge/cvpa2-structure-from-motion>.

## 1 Introduction

### 1.1 Two View SFM

Two View SFM은 두 개의 viewpoint에서 촬영한 물체 이미지로부터 물체의 부분적인 3차원 구조를 복원하는 과정이다. 이러한 과정은 (1) camera pose를 복원하는 5 point algorithm (2) 3d point를 복원하는 triangulation으로 이루어진다.

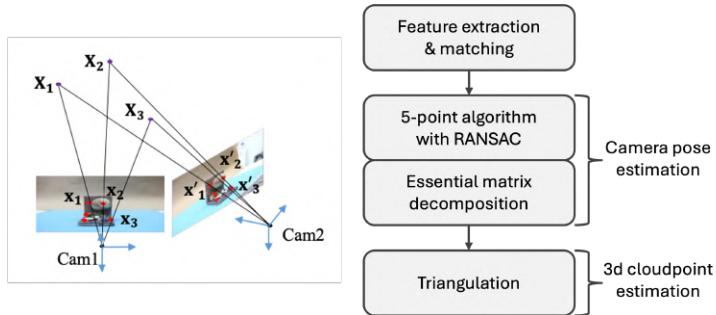


Figure 1: Overview framework of Two View SFM

두 개의 이미지에 대해 대응되는 물체의 2d point들을 찾기 위해 feature extraction & matching을 수행할 수 있다. 우리는 SIFT(Scale-Invariant Feature Transform) 알고리즘을 이용해 feature extraction을 수행하였고, BFMatcher(Brute-Force Matcher)를 이용해 추출된 feature에 대해 matching을 수행하였다. 추가적으로 보다 정확한 feature matching 결과를 얻기 위해 Lowe's ratio test를 이용할 수 있었다. 이렇게 구한 두 이미지의 2d point matches를 이용해 두 이미지 시스템을 촬영하고 있는 camera들의 기하학적인 관계를 추정할 수 있다. Epipolar geometry로부터  $\mathbf{x}_1^T \mathbf{E} \mathbf{x}_2 = 0$ 의 식으로 관계를 나타낼 수 있는데, 여기서 Essential matrix  $\mathbf{E}$ 는 두 camera pose간의 위치 관계를 포함하고 있는 행렬이다. 우리는 5개의 2d point 쌍을 사용하는 5-Point Algorithm을 사용해 식을 풀어 Essential matrix를 구할 수 있었다. 이때 RANSAC(RANdom Sample Consensus)기반의 추정 방식을 활용하여 가장 많은 inlier를 갖는 Essential matrix 해를 구하였다. 이후 Essential matrix를 분해하는 과정을 통해 각각의 camera pose 행렬을 도출할 수 있다. 최종적으로 inlier 2d point들에 대해 구한 camera pose와 함께 triangulation을 수행함으로써 3d cloudpoint를 복원한다.

### 1.2 Multi View SFM

더 많은 viewpoint 정보를 이용하면, 보다 완전한 물체의 3차원 구조를 복원할 수 있다. 이렇게 3개 이상의 viewpoint 정보를 이용하는 Multi View SFM은, (1) camera pose를 복원하는 3 point algorithm (2) 3d point를 복원하는 triangulation (3) 구한 camera pose와 3d point들을 최적화하는 bundle adjustment의 과정들을, 3개 viewpoint 쌍마다 반복하여 수행하는 것으로 이루어진다.

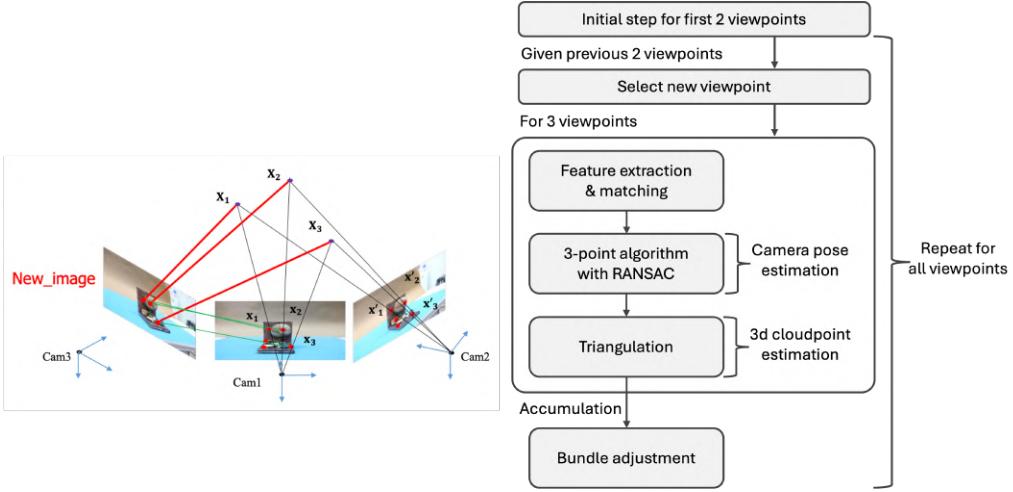


Figure 2: Overview framework of Multi View SfM

Multi View SfM은 처음 두 viewpoint들에 대해 Initial step을 수행하는 것으로 시작한다. 이 Initial step은 앞의 Two View SfM과 동일한 과정을 거쳐 camera pose와 3d points를 얻게 된다. 이후 다른 viewpoint를 하나씩 추가하는 것을 반복하여 이들의 camera pose와 3d points를 누적하여 결과로 도출하게 된다. 이러한 반복 과정은 3개 viewpoint 쌍에 대해 반복되는데, 예를 들어  $(\text{img}[1], \text{img}[2], \text{img}[3]) \rightarrow (\text{img}[2], \text{img}[3], \text{img}[4]) \rightarrow \dots \rightarrow (\text{img}[n-2], \text{img}[n-1], \text{img}[n])$  으로 loop가 수행되는 식이다. 각 반복 step i에서는 주어진 2 viewpoint  $(\text{img}[i], \text{img}[i+1])$  정보에 추가되는 viewpoint  $(\text{img}[i+2])$ 를 이용해 새로운 카메라 대한 camera pose와 3d point를 구하는 과정이 이루어진다. 먼저 Two View SfM과 동일한 방식을 새로운 이미지에 대해 feature extraction을 수행한 후, 이번에는 세 쌍의 이미지 모두에 공통되는 feature matching을 수행한다. 구한 feature 중에서 새로운 이미지에서의 2d point 3개와 이에 대응되는 주어진 3d point 3개를 이용하면, 3 point algorithm을 수행함으로써 새로운 카메라의 camera pose 행렬을 구할 수 있다. 여기서도 추가적으로 RANSAC기반 추정 방식을 활용함으로써, 새 camera pose를 이용한 projection error를 최소화하는 camera pose 해를 구하였다. 이후 새 이미지에 대해 triangulation을 수행함으로써 물체의 새로운 3d pointcloud들을 얻을 수 있다. 최종적으로 반복 과정을 수행할 때마다 새롭게 얻은 3d pointcloud 누적 결과에 대해 bundle adjustment를 수행함으로써 모든 이미지에서의 reprojection error를 최소화하는 개선된 결과를 얻을 수 있다. 이러한 Multi View SfM 결과는 Two View SfM에서보다 더 완성된 3d pointcloud structure 정보를 포함한다.

### 1.3 Dataset and Camera calibration

우리는 기본 물체 데이터셋(moai, choonsik, nike, toothless)에 더해, 새로운 커스텀 데이터셋(eiffeltower, dinosour, bluedoll, dog)도 함께 구성하여 실험하였다. 각 물체를 촬영할 때 camera parameter가 일정하게 유지되도록 fixed-focus 카메라를 사용한다. basic dataset은 각 물체에 대해 약 18~22개의 viewpoint 이미지로 구성되어 있고, custom dataset은 각 물체에 대해 약 4~8개의 viewpoint 이미지로 구성되어 있다.



Figure 3: Objects — Basic(moai, choonsik, nike, toothless) and Custom(eiffeltower, dinosour, bluedoll, dog)

여기서 각 basic dataset과 custom dataset은 다른 camera parameter를 가지게 되는데, 이를 반영하기 위해 camera calibration을 각각 수행하는 작업이 필요하다. camera calibration은 카메라에 대해 intrinsic matrix  $\mathbf{K}$ 를 구하는 과정이며, checker board를 다양한 각도에서 촬영한 사진을 이용하여 수행할 수 있다. 이러한 intrinsic matrix는 focal length, optical center (principal point), skew 등의 camera parameter 정보를 포함하며, 이를 이용해 픽셀 좌표계에서의 관측값을 카메라 좌표계의 기하학적 위치로 정확하게 변환할 수 있어, SfM에서의 essential matrix 계산, triangulation, bundle adjustment 등의 계산 과정에 활용할 수 있다.

## 2 Method

### 2.1 Feature extraction and matching

두 이미지에서 특징점을 추출하여 매칭하는 작업을 수행함으로써 SFM에 사용될 수 있는 2d point들을 추출할 수 있다. 먼저, 각각의 이미지에서 특징점을 추출하기 위해 **SIFT(Scale-Invariant Feature Transform)** 알고리즘을 사용한다 [2]. SIFT는 이미지 내에서 스케일 변화와 회전에 대해 불변한 특징점을 추출하는 데에 특화된 알고리즘으로, 우리의 실험 환경인 다양한 크기와 방향에 대한 이미지에서 적용하기에 적합하다. SIFT 알고리즘의 처리 과정은 다음과 같이 이루어진다. 먼저 이미지에서 다양한 스케일 정보를 얻기 위해 각 픽셀 포인트에 대해 Difference of Gaussian(DoG)를 계산하고, 이를 통해 극값(extrema)를 찾는다. 이 극값들이 잠재적인 keypoint 후보가 되며, 각 keypoint에 대해 주위의 dominant orientation을 계산하여 방향 정보를 할당한다. 이후 각 keypoint 주변의 국소 지역 이미지 patch들을 기반으로 SIFT descriptor(특징 벡터, 128차원)를 생성한다. 이 descriptor는 지역 이미지 패턴의 분포를 잘 요약함으로써 keypoint 간에 매칭을 수행할 수 있도록 해준다.

이후, 두 이미지의 SIFT 특징점을 매칭하기 위해 **BFMatcher(Brute-Force Matcher)**를 사용할 수 있다 [3]. BFMatcher는 keypoint간의 유사도를 비교하여 가장 가까운 대응점을 찾는 방식이다. 구체적으로 한 이미지의 각 SIFT descriptor 특징 벡터에 대해 다른 이미지의 모든 특징 벡터와의 L2 거리(유클리디안 거리)를 계산하여 가장 가까운 이웃을 찾는 방식으로 작동한다. 이때 매칭의 신뢰도를 높이기 위해, k-NN(k-Nearest Neighbors)기반 매칭을 수행하여 여러 후보(우리는  $k=2$  사용)들을 뽑은 후, **Lowe's ratio test**를 적용하여 가장 적합한 matching point를 찾는다. k-NN 방식은 한 특징점에 대해 가장 가까운  $k(k=2)$ 개의 후보 descriptor를 찾는 방식이다. Lowe's ratio test는 이들  $k(k=2)$ 개의 후보에 대해, 가장 가까운 두 후보 매칭 간의 거리 비율을 계산하여, 첫 번째 후보가 두 번째 후보보다 명확하게 가깝지 않으면 잘못된 매칭일 가능성성이 높다고 본다. 따라서 우리는 두 매칭의 거리 비율이 일정 threshold (우리는 대개 0.75 이하를 사용)보다 작은 경우에만 유효한 매칭으로 간주하고 필터링하여 사용한다.

### 2.2 Essential matrix estimation with RANSAC

Essential Matrix는 두 이미지 간의 기하학적 관계를 설명한다. 아래와 같이 두 이미지에 대해 대응되는 두 2d keypoint 와 해당 match가 표현하는 3d point, 그리고 각 카메라 원점을 모두 포함하는 평면을 만들 수 있다. 따라서 이 평면 조건을 이용하면 2d keypoints와 Essential Matrix  $E$ 으로 식  $\mathbf{x}_1^T E \mathbf{x}_2 = 0$ 를 구성할 수 있고, 이를 Epipolar constraint라고 한다. 2d keypoints 쌍을 충분히 알고 있는 상태에서, 이 식을 풀면 Essential Matrix  $E$ 를 구할 수 있다.

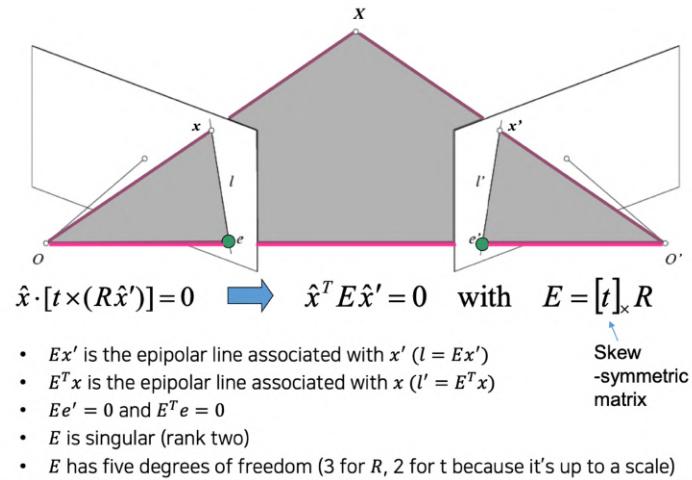


Figure 4: Epipolar Geometry

우리는 특히, 이 식에 대해 5개 점만으로도 Essential Matrix를 계산할 수 있는 Nistér의 **5-Point Algorithm**을 사용한다 [4]. 먼저, keypoint들을 camera intrinsic matrix  $K$ 의 역행렬을 곱하여 정규화하는 작업을 수행한다(이렇게 변환된 좌표는 카메라 보정된 좌표계 상에서의 위치를 의미한다). 이후 5쌍의 정규화된 2d points에 대해, Epipolar Constraint 식을 바탕으로 5개의 동차 방정식을 구한다. 이때 추가적인 제약 조건들도 만들어야 하는데,  $E$ 는  $\text{rank}(E) = 2$ 이고  $\det(E) = 0$ 이어야 하며,  $2EE^T E - \text{trace}(EE^T)E = 0$ 이라는 비선형 제약도 만족해야 한다. 이러한 제약 조건들을 포함하여 총 10 차 다항 방정식을 유도할 수 있고, 이를 통해 최대 10개의 실수 해를 계산할 수 있다. 이후 각 후보  $E$ 들에 대해 Epipolar constraint에서의  $\mathbf{x}_1^T E \mathbf{x}_2$ 를 오차로 사용하여, 오차를 최소화하는 가장 적합한 해를 구할 수 있다.

여기서 우리는 **RANSAC(RANdom Sample Consensus)** 기반의 추정 방식을 함께 도입하여 사용하였다. RANSAC은 입력 데이터 중 일부가 이상치(outlier)를 포함하고 있을 때, 이들의 영향을 배제하고 신뢰성 있는 모델 파라미터를 추정하기 위한 반복적 샘플링 기법이다. RANSAC 루프에서는 무작위로 5쌍의 매칭 키포인트를 샘플링하고, 5-Point Algorithm 결과인 Essential Matrix 후보 행렬들에 대해 모든 매칭 점 쌍의 오차를 계산하여, 오차를 최소화하는 Essential Matrix를 구한다. 이러한 루프를 max iter만큼 반복하면서, 루프들 중 threshold 이하의 오차를 가지는 inlier 쌍들을 최대한 많이 갖도록하는 Essential Matrix를 최적 결과로 도출하여, outlier에 강건하며 신뢰도 높은 해를 찾도록 할 수 있다.

## 2.3 Essential matrix decomposition

구한 Essential Matrix  $\mathbf{E}$ 를 분해(decomposition)하면, 각 camera pose 행렬을 구할 수 있다.

$$\mathbf{E} = [\mathbf{t}]_{\times} \mathbf{R} = \mathbf{R}[\mathbf{R}^T \mathbf{t}] \quad (\mathbf{R}: \text{rotation}, \mathbf{t}: \text{translation}) \quad (1)$$

위는 camera 1의 pose matrix를  $[\mathbf{I}|0]$ 로(원점으로) 두고, camera 2의 pose matrix를  $[\mathbf{R}|\mathbf{t}]$ 로 두었을 때의 식이다.  $[\mathbf{R}|\mathbf{t}]$ 를 구하는 과정은 다음과 같다: (1)  $\mathbf{E}$ 에 대해 SVD(Singular Value Decomposition) 수행 (2) decomposition 결과를 조합하여 4가지 pose matrix 후보 생성 (3) 두 카메라에 대한 depth가 모두 양수값을 가지는 best pose matrix를 선택. 여기서 (2)의 4가지 pose matrix 후보들은 다음과 같다:

$$\begin{cases} P_1 = [\mathbf{U}\mathbf{W}\mathbf{V}^T] + \mathbf{u}_3 \\ P_2 = [\mathbf{U}\mathbf{W}\mathbf{V}^T] - \mathbf{u}_3 \\ P_3 = [\mathbf{U}\mathbf{W}^T\mathbf{V}^T] + \mathbf{u}_3 \\ P_4 = [\mathbf{U}\mathbf{W}^T\mathbf{V}^T] - \mathbf{u}_3 \end{cases}, \quad \text{where} \quad \begin{cases} \text{SVD}(\mathbf{E}) = \mathbf{U} \cdot \text{diag}(1, 1, 0) \cdot \mathbf{V}^T \\ \mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \\ \mathbf{u}_3 = \mathbf{U} \cdot (0, 0, 1)^T \end{cases} \quad (2)$$

이들 후보들에 대해 (3)의 과정에서 depth를 구하기 위해 우리는 section 2.4.의 triangulation을 사용할 수 있다.

## 2.4 Triangulation

Triangulation을 통해 2d matched points와 camera pose matrices로부터 3d points를 생성할 수 있다. Extrinsic matrix를  $\mathbf{P}' = \mathbf{K} \cdot [\mathbf{R}|\mathbf{t}]$ 라 하면, camera 1의 extrinsic을  $\mathbf{P}'_1$ , camera 2의 extrinsic을  $\mathbf{P}'_2$ 로 표현할 수 있다. 이때 3d point를  $X$ 로 두고, 각 2d matched point를  $(x_1, y_1), (x_2, y_2)$ 라 하면, 다음과 같이 식을 구성할 수 있다:

$$\begin{cases} \mathbf{A}\mathbf{X} = 0 \\ \mathbf{A} = \begin{bmatrix} x_1 \mathbf{p}'_1{}^{3T} - \mathbf{p}'_1{}^{1T} \\ y_1 \mathbf{p}'_1{}^{3T} - \mathbf{p}'_1{}^{2T} \\ x_2 \mathbf{p}'_2{}^{3T} - \mathbf{p}'_2{}^{1T} \\ y_2 \mathbf{p}'_2{}^{3T} - \mathbf{p}'_2{}^{2T} \end{bmatrix} \end{cases} \quad (3)$$

위의 식을 풀면 homogeneous 한  $\mathbf{X}$ 를 구할 수 있고, 이를  $\mathbf{X}^4$ 로 나누어 정규화함으로써 최종 3d point를 구할 수 있다.

## 2.5 Growing step (Multi-view)

Multi View SFM은 위의 2.1 2.4 과정을 통해 initial step을 수행한다. 이후 Growing step으로써 새로운 카메라 view를 하나씩 추가하게 되는데, 각 Growing step에서는 새로운 카메라에 대한 camera pose matrix를 구하기 위해 **3-Point Algorithm**을 적용하게 된다 [5]. [5]에서 제시된 Three-Point Perspective Pose Estimation (3P-PPE) 알고리즘은 3d point 세 개와 그에 대응하는 이미지 상의 투영 좌표 (2d point) 세 개가 주어졌을 때, 카메라의 회전 행렬  $\mathbf{R}$ 과 이동 벡터  $\mathbf{t}$ 를 유일하게 복원하는 알고리즘이다. 우선 step 2.1과 같은 과정으로 새로운 이미지에 대해 feature extraction을 수행한 후, 앞의 두 이미지 쌍과 대응되는 feature를 필터링하여 2d points를 구한다. 주어진 2d point  $\mathbf{x}_i$ 는 camera intrinsic matrix  $\mathbf{K}$ 로 정규화한다. 이후, 각 3D point  $\mathbf{X}_i$ 는  $\mathbf{R}$ 과  $\mathbf{t}$ 에 의해 대응하는 방향에 위치해야 한다는 기하학적 제약 조건을 바탕으로, 위치 간 거리 및 각도 관계에 기반한 고차 방정식 시스템을 구성할 수 있다. 이 방정식은 일반적으로 최대 4개의 실수 해를 가지며, 각 해에 대해 재투영 오차  $\sum_i \|\mathbf{x}_i - \mathbf{K}\mathbf{P}\mathbf{X}_i\|^2$ 를 계산하여 가장 적합한 해를 구할 수 있다.

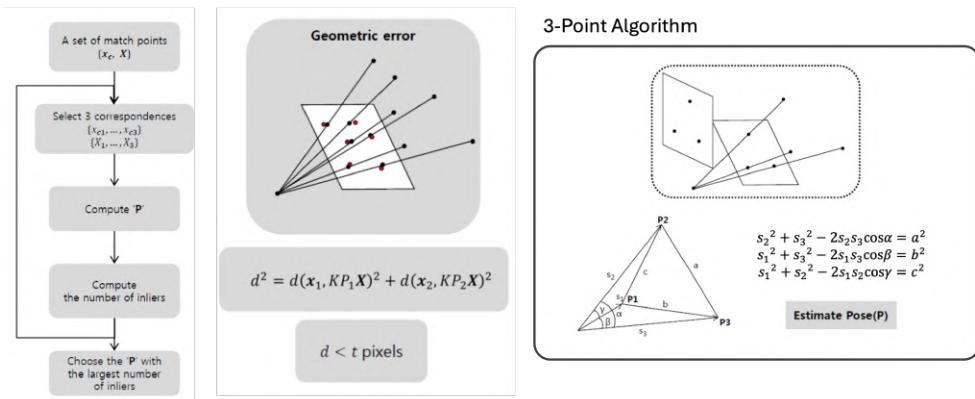


Figure 5: 3-Point Algorithm with RANSAC

우리는 3-Point Algorithm에 RANSAC 추정 방식을 결합하여 사용하였다. 위에서와 마찬가지로, RANSAC 루프마다 무작위로 3쌍의 3d-2d point 쌍을 샘플링하고, 3-Point Algorithm 결과인 Pose Matrix 후보 행렬들에 대해 재투영 오차를

계산하여, 오차를 최소화하는 Pose Matrix를 구한다. 이러한 루프를 max iter만큼 반복하면서, 루프들 중 threshold 이하의 오차를 가지는 inlier 쌍들을 최대한 많이 갖도록하는 Pose Matrix를 최적 결과로 도출하여, outlier에 강건하며 신뢰도 높은 해를 찾도록 할 수 있다.

그런 다음, 해당 Pose Matrix를 이용해 step 2.4와 같은 triangulation을 수행하여 새로운 viewpoint에 대한 3d points를 구한다. Growing step에서의 새로운 viewpoint에 대한 3d points와 pose matrix는 앞서 구한 viewpoints들의 것들과 합쳐져 누적되어 저장한다. 그러면 우리는 물체의 3d structure에 대해 서로 다른 부분들을 모두 합쳐 보다 완성도 있는 structure를 구성할 수 있다.

## 2.6 Optimization (Multi-view)

**Bundle Adjustment**은 다수의 이미지로부터 복원된 3d cloudpoints와 camera poses를 전역적으로 최적화하여 전체 재구성 정확도를 높이는 절차이다 [6, 7]. 이는 각 카메라에서 관측된 2D points  $\mathbf{u}_{ij}$ 와 그에 대응하는 3D points  $\mathbf{X}_j$ 를 주어진 camera poses를 통해 재투영한 결과  $(\mathbf{K}[\mathbf{R}_i|\mathbf{t}_i]\mathbf{X}_j)$  간의 거리인 재투영 오차를 최소화하는 비선형 최적화 문제로 구성된다. 수학적으로는  $\min_{\mathbf{P}, \mathbf{X}} \sum_{i=1}^n \sum_{j=1}^m w_{ij} \|\mathbf{u}_{ij} - \pi(\mathbf{P}_i \mathbf{X}_j)\|^2$  와 같은 형태의 목적함수를 가지며, 여기서  $w_{ij}$ 는 해당 점이 관측되었는지를 나타내는 indicator이다. 최적화는 일반적으로 Levenberg–Marquardt (LM) 알고리즘을 기반으로 한 비선형 최소자승 방식으로 수행되며, MATLAB의 lsqnonlin 함수 등의 도구를 통해 구현될 수 있다. 초기값의 품질이 전체 수렴 품질에 크게 영향을 미치므로, 보통 전처리 단계에서 PnP와 삼각측량을 통해 적절한 초기값을 확보한 후 수행된다. 시각적으로는 최적화 전에는 광선이 서로 수렴하지 않고 퍼져 있는 반면, 최적화 후에는 카메라 중심으로 광선이 집중되며 더 일관된 3D 구조와 정확한 카메라 파라미터를 얻을 수 있다.

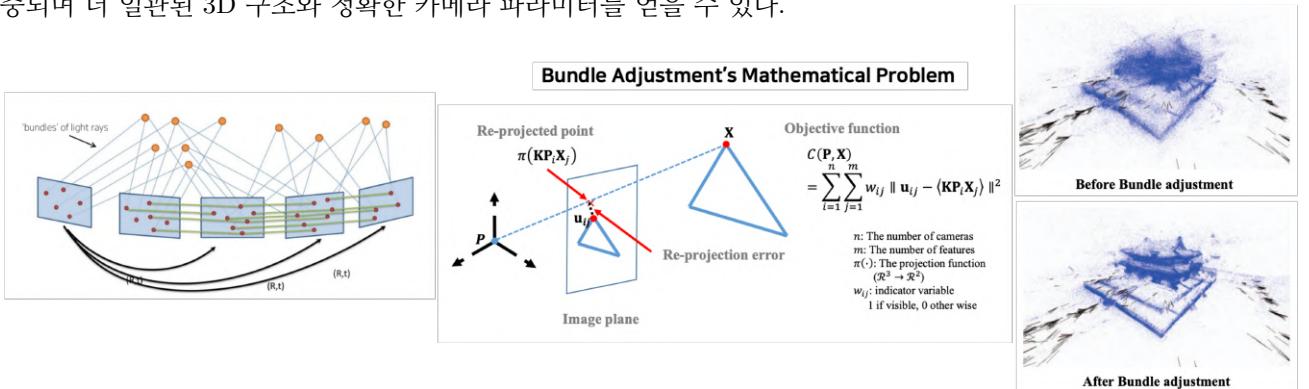


Figure 6: Bundle Adjustment

## 2.7 Camera calibration

**Camera Calibration**은 카메라에 대해 intrinsic matrix  $\mathbf{K}$ 를 구하는 과정이며, checker board를 다양한 각도에서 촬영한 사진을 이용하여 수행할 수 있다 [8, 9]. 이러한 intrinsic matrix는 focal length, optical center (principal point), skew 등의 camera parameter 정보를 포함하며, 이를 이용해 픽셀 좌표계에서의 관측값을 카메라 좌표계의 기하학적 위치로 정확하게 변환할 수 있어, SfM에서의 essential matrix 계산, triangulation, bundle adjustment 등의 계산 과정에 활용할 수 있다.

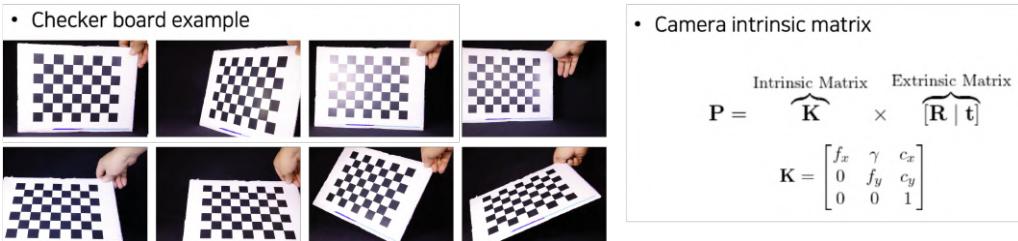


Figure 7: Camera calibration with Checker board

우리는 다음과 같은 절차로 Camera Calibration을 구현하였다. 먼저, OpenCV의 cv2.findChessboardCorners 함수를 사용하여 입력된 각 checkerboard 이미지에서 내부 코너를 탐지하고, cv2.cornerSubPix를 통해 서브픽셀 단위로 코너 위치를 정밀하게 보정한다. 이때 checkerboard는 (6, 8) 크기의 내부 코너를 가지며, 각 코너는 실제 월드 좌표계 상에서 Z=0 평면 위의 일정한 간격으로 배치된 3D 점  $\mathbf{X}_i$ 로 간주된다. 이후 모든 이미지에서 수집된 checkerboard corner 2D-3D 대응쌍을 바탕으로 cv2.calibrateCamera를 사용하여 카메라의 내부 파라미터  $\mathbf{K}$ 를 추정할 수 있다. 해당 함수는 비선형 최소자승 기반의 최적화 알고리즘을 통해 3D 점을 카메라 좌표계로 투영한 결과와 실제 이미지 상의 2D 코너 간의 재투영 오차(reprojection error)를 최소화하는  $\mathbf{K}$ 를 도출하는 방식을 사용한다.

### 3 Result and Discussion

#### 3.1 Feature Matching

Feature Matching 결과는 다음과 같다. 아래는 Base Dataset에 대한 Two View SfM을 시행할 때 Feature Matching 을 수행한 결과이다. 물체 moai에 대해, kp1 : 46498, kp2 : 46482, matches : 5366, inlier matches : 4236의 결과가 나왔다. 물체 choonsik에 대해, kp1 : 4306, kp2 : 3942, matches : 180, inlier matches : 99의 결과가 나왔다. 물체 nike에 대해, kp1 : 2952, kp2 : 2677, matches : 165, inlier matches : 71의 결과가 나왔다. 물체 toothless에 대해, kp1 : 13260, kp2 : 11584, matches : 325, inlier matches : 108의 결과가 나왔다.

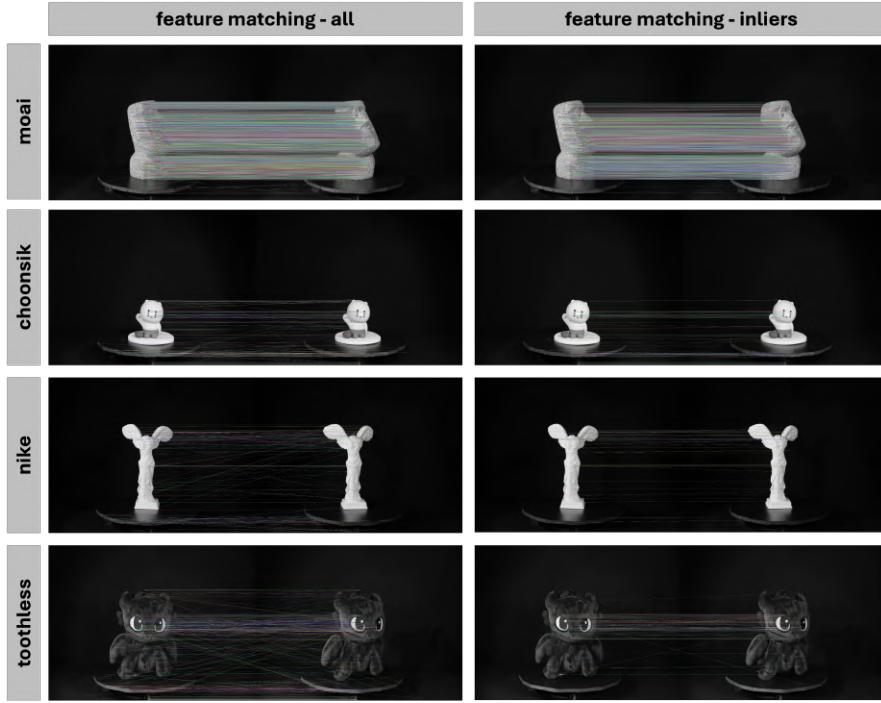


Figure 8: Feature Matching : Two View SFM (Base Dataset)

다음은 Custom Dataset에 대한 Two View SfM을 시행할 때 Feature Matching을 수행한 결과이다. 물체 eiffeltower에 대해, kp1 : 4493, kp2 : 4095, matches : 624, inlier matches : 259의 결과가 나왔다. 물체 dinosour에 대해, kp1 : 2388, kp2 : 1168, matches : 140, inlier matches : 15의 결과가 나왔다. 물체 bluedoll에 대해, kp1 : 3044, kp2 : 1029, matches : 98, inlier matches : 20의 결과가 나왔다. 물체 dog에 대해, kp1 : 799, kp2 : 575, matches : 120, inlier matches : 29의 결과가 나왔다.

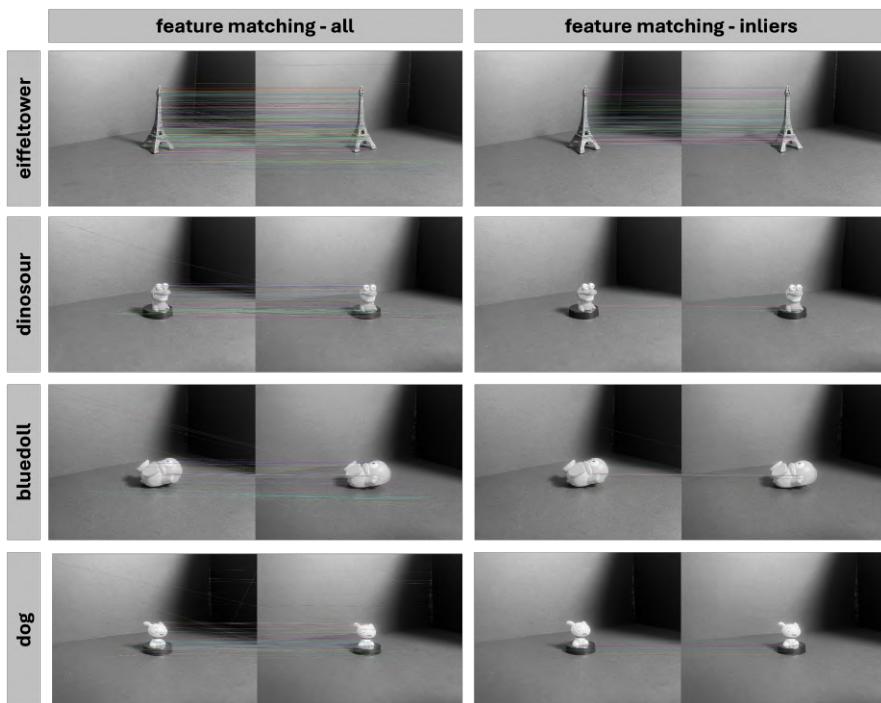


Figure 9: Feature Matching : Two View SFM (Custom Dataset)

두 Dataset에 대해, Basic Dataset에서 특히 물체 moai의 feature extraction & matching이 가장 잘 이루어지고 있는 것을 확인할 수 있으며, Custom Dataset에서 특히 물체 eiffeltower의 feature extraction & matching이 가장 잘 이루어지고 있는 것을 확인할 수 있다. 공통적으로 표면에 요철이 많은 물체에 대해 feature가 많이 생기고 matching 결과 또한 정확하게 나타나는 것으로 보인다. 또한 full feature matching과 SfM 수행 후의 inlier feature matching 결과를 비교하면, SfM에서의 RANSAC 수행으로 outlier들이 대체로 잘 필터링되고 있음을 확인할 수 있다.

다음은 물체 moai에 대한 Multi View SfM을 시행할 때 Feature Matching을 수행한 결과이다. 각 이미지별 keypoint 갯수 결과는 4.jpg: 44533, 3.jpg: 46498, 2.jpg: 46482, 1.jpg: 45787, 0.jpg: 44717, 19.jpg: 46184, 18.jpg: 47238, 17.jpg: 46471, 16.jpg: 46036이다. 두 이미지 쌍에서의 matches 갯수 결과는 4.jpg-3.jpg: 1736, 3.jpg-2.jpg: 2049, 2.jpg-1.jpg: 4758, 1.jpg-0.jpg: 926, 0.jpg-19.jpg: 4450, 19.jpg-18.jpg: 3887, 18.jpg-17.jpg: 1619, 17.jpg-16.jpg: 1939이다.

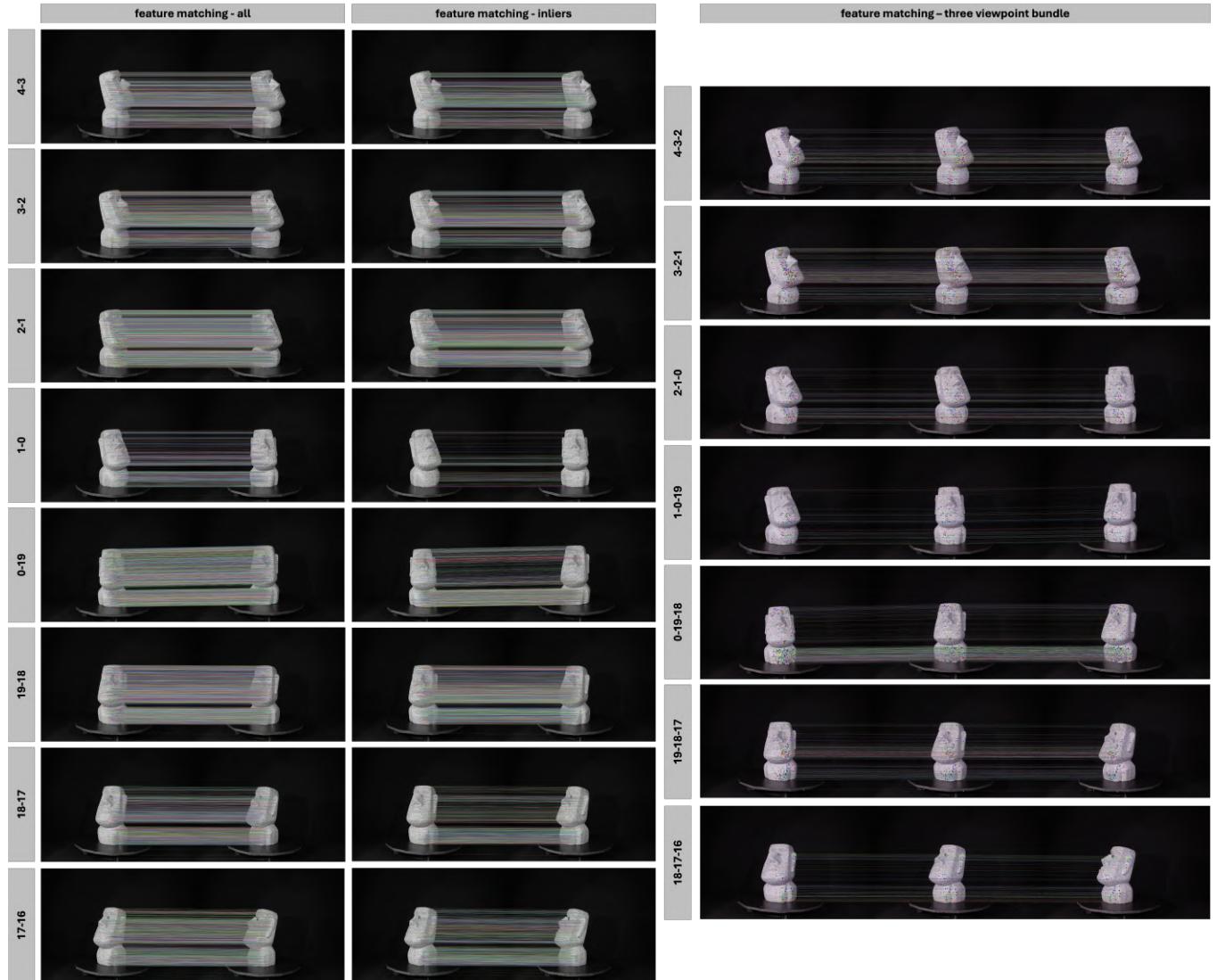


Figure 10: Feature Matching : Multi View SfM (Moai)

위의 Multi View 결과에 대해 두 이미지쌍에 대한 feature matching과 growing step에서의 세 이미지쌍에 대한 feature matching을 비교하였을 때, 세 이미지쌍에 대한 matching은 두 개의 두 이미지쌍 matching의 교집합을 찾는 것으로, 약 20% 정도의 비율로 찾아지는 것을 확인할 수 있었다. inlier feature matching은 첫 initial step에서의 matching 1쌍과 나머지 growing step에서의 matching 7쌍이 결과로 나와있다. outlier들이 필터링되어 full feature matching보다 적은 수의 matching을 가지는 것을 확인할 수 있으나, 여전히 많은 matching 정보를 보존하고 있는 것을 확인할 수 있다. inlier는 주로 각 view쌍에서의 물체 중심부에서 많이 관찰되는 것을 확인할 수 있었다. 결과적으로 multiview를 수행할 때 각 step에서 여러 view에 대한 특징점 정보를 충분히 포함하고 SfM이 수행되고 있는 것을 확인할 수 있다.

### 3.2 Two View SFM

다음은 Two View SFM을 수행한 결과이다. 먼저 base dataset(moai, choonsik, nike, toothless)에 대한 Two View SFM에 대한 결과는 다음과 같다. 각각의 물체에 대해 결과로 나온 inlier points, 3d cloudpoints, camera poses, 그리고 사용한 hyperparameter들을 표로 정리하였다.

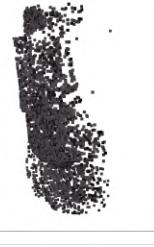
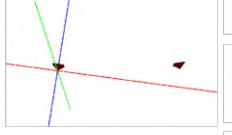
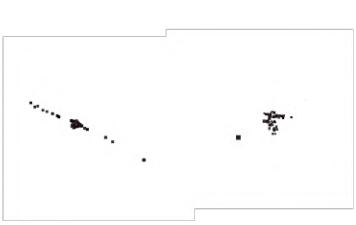
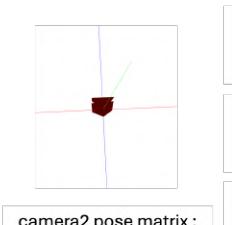
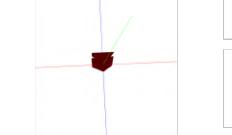
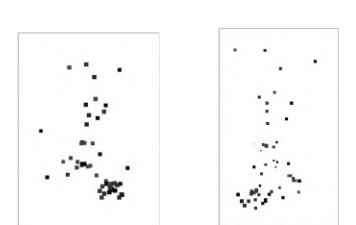
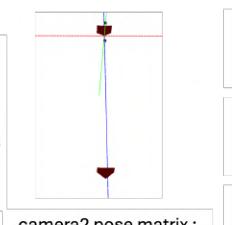
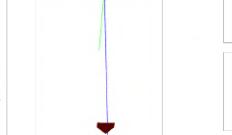
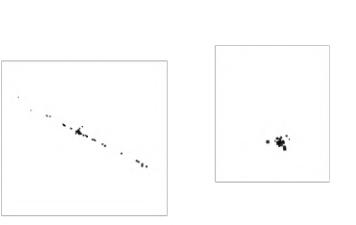
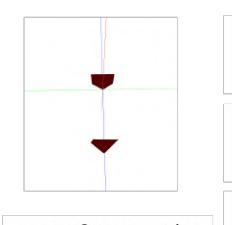
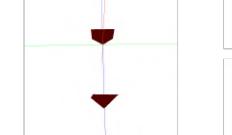
	Inliers	Result : 3d cloudpoints			Camera poses	hyperparameter
moai						matching threshold : 0.75 ransac iteration : 1000 inlier threshold : 1e-4  camera2 pose matrix : $\begin{bmatrix} 0.95973707 & -0.02610829 & 0.27968395 & -0.98750118 \\ 0.0247206 & 0.99965856 & 0.00846881 & -0.03171744 \\ -0.27981003 & -0.0012331 & 0.9600546 & 0.15449844 \end{bmatrix}$
choonsik						matching threshold : 0.8 ransac iteration : 10000 inlier threshold : 5e-4  camera2 pose matrix : $\begin{bmatrix} 0.99886111 & -0.23257393 & -0.041482 & 0.01066998 \\ 0.00864259 & -0.94442838 & 0.32860367 & -0.16909214 \\ -0.04692326 & 0.32787086 & 0.94356562 & -0.98541574 \end{bmatrix}$
nike						matching threshold : 0.8 ransac iteration : 10000 inlier threshold : 5e-4  camera2 pose matrix : $\begin{bmatrix} -0.99846123e-01 & -4.29991819e-04 & -2.66114508e-02 & 1.36938374e-02 \\ 1.97614757e-03 & -9.88717785e-01 & -5.05853819e-02 & 0.28359095e-02 \\ 1.097654607e-02 & -5.06200579e-02 & 0.98364606e-01 & -3.99072401e-01 \end{bmatrix}$
toothless						matching threshold : 0.8 ransac iteration : 10000 inlier threshold : 5e-4  camera2 pose matrix : $\begin{bmatrix} 0.99007852 & 0.12780307 & 0.04972221 & -0.04848635 \\ 0.12848491 & -0.99159847 & -0.01497026 & 0.00889691 \\ 0.04739424 & -0.02121777 & 0.99865089 & -0.99892334 \end{bmatrix}$

Figure 11: Result : Two View SFM (Base Dataset)

Moai가 가장 결과가 정확하고 잘 나온 것을 확인할 수 있는데, 그 이유는 추출된 특징점이 가장 많고 정확하기 때문이라고 말할 수 있다. inlier들에 대해 3d cloudpoint가 잘 형성된 것을 확인할 수 있으며, camera pose 또한 정확하게 추출되는 것을 확인할 수 있었다. camera2 pose matrix에 대해, 마지막 pose translation vector에서 y축 요소가 y에 가까운 것을 통해 baseline이 실제 3d 환경에서 거의 수평으로 촬영된 것이라고 유추할 수 있고, 이를 통해 pose가 잘 추출되었음을 확인할 수 있다. rotation에 대해서도 시각화 결과를 통해 camera1을 기준으로 실제 matching view와 일치하게 결과가 나오는 것을 확인할 수 있었다.

Choonsik, Nike, Toothless는 결과적으로 3d cloudpoint의 수가 희소하고 z축의 오차가 큰 것을 확인할 수 있었다. 이는 추출된 특징점 정보가 희소하기 때문에 추측할 수 있다. cloudpoint 수가 적은 것에 대해 matched feature 수가 적기 때문에 결과적으로 inlier 또한 적게 나와 최종적으로 3d point들도 희소하게 생성된 것이다. 오차에 가장 민감하게 반응하는 것은 cloudpoint의 z축이었다. cloudpoint의 z축 오차가 큰 것에 대해, essential matrix 및 pose matrix에서 오차가 발생하였기 때문이라고 말할 수 있다. 특히 camera pose에서의 pose translation에 대해, z축 및 x축의 오차가 큰 것을 확인할 수 있었고, rotation 또한 다소 부정확한 결과가 나왔다. 다만 결과 3d cloudpoint에서 xy평면 상의 위치는 물체 nike를 통해 상대적 위치를 근사적으로 잘 검출하였음을 확인할 수 있었다. 이는 nike의 inlier point들이 xy평면 상에 넓게 위치하여 xy평면 정보를 잘 전달하기 때문이라고 생각할 수 있다. 따라서 feature들이 좁게 분포할수록 오차(특히 z축 오차)가 크게 나타날 가능성이 높다고 유추할 수 있었다. 최종적으로 feature의 수가 많고 matching 결과가 정확할수록 SFM의 결과로 생성되는 3d cloudpoint가 많고 정확하다고 말할 수 있다.

다음으로, custom dataset(eiffeltower, dinosour, bluedoll, dog)에 대한 Two View SfM에 대한 결과는 다음과 같다.

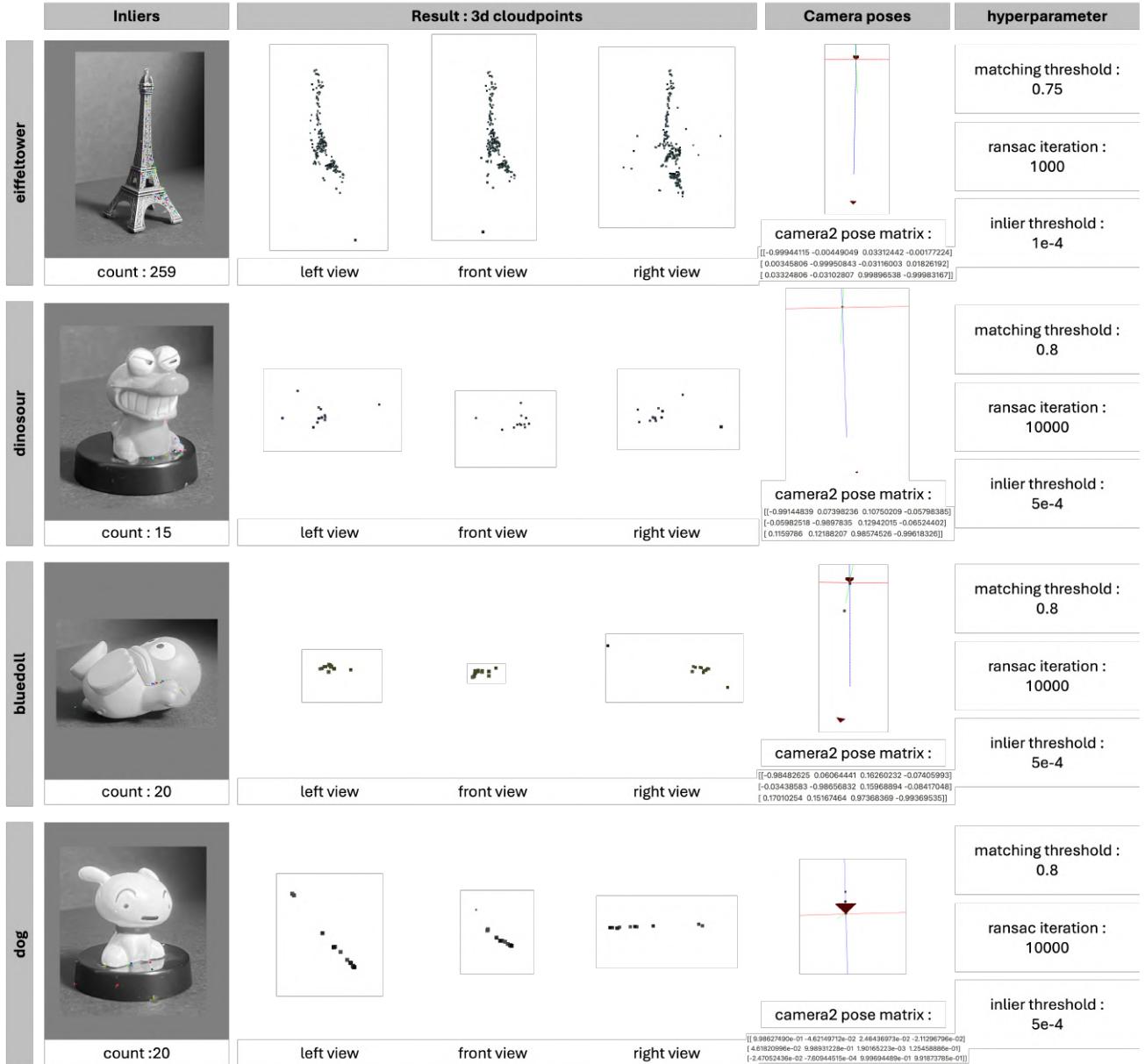


Figure 12: Result : Two View SfM (Custom Dataset)

Eiffeltower의 결과가 가장 잘 나온 것을 확인할 수 있으며, 마찬가지로 feature extraction & matching 단계에서 특징점은 정확하고 많게 추출하여 essential matrix 추정 과정에서 입력 정보의 품질이 좋았기 때문이라고 유추할 수 있다. 나머지 물체들인 Dinosour, Bluedoll, Dog에 대해서는 특징점이 희소하여 다소 부정확한 결과가 나왔다. 여기서도 마찬가지로, cloudpoint의 z축 오차가 가장 크게 나타나는 것을 확인할 수 있었고, pose matrix에서의 x축 z축 translation과 rotation에서의 오차가 나타나는 것을 확인할 수 있었다. 다만 여기서의 오차는 custom dataset을 구축할 때 자체에서 카메라 pose가 정확하게 조정되지 않은 것에서 발생한 오차도 있을 것이라고 생각된다.

### 3.3 Multi View SfM

Multi View SfM은 Two View SfM에서 가장 결과가 잘 나왔던 Basic dataset의 Moai와 Custom dataset의 Eiffel Tower에 대해 수행하였다. Moai에서 사용한 hyperparameter는 다음과 같다 : [matching] matching threshold knn 0.60, [initial] ransac iter 1000, em threshold 1e-4, [growing] three point ransac iter 2000, three point threshold 1e-4, three point inlier threshold 1e-1. Eiffel Tower에서 사용한 hyperparameter은 다음과 같다 : [matching] matching threshold knn 0.70, [initial] ransac iter 1000, em threshold 1e-4, [growing] three point ransac iter 2000, three point threshold 1e-3, three point inlier threshold 1e-1. Moai의 Multi View SfM 수행에서는 모든 growing step을 성공적으로 잘 마칠 수 있었고, 각 step 및 최종 cloudpoint 결과 또한 모두 잘 나타나는 것을 확인할 수 있었다. Eiffel Tower의 Multi View SfM 수행에서는 마지막 growing step을 제외하고 수행할 수 있었고, 첫 growing step에서는 after bundle cloudpoint가 정확하게 나왔으나, 두세번째 growing step에서 일렬로 정렬된 outlier들이 생겼고, 네번째 growing step에서 전체 after bundle cloudpoint에서 큰 오차가 생겼다.

따라서 성공적으로 결과가 나온 물체 Moai의 Multi View SFM을 다음과 같이 시각화하였다.

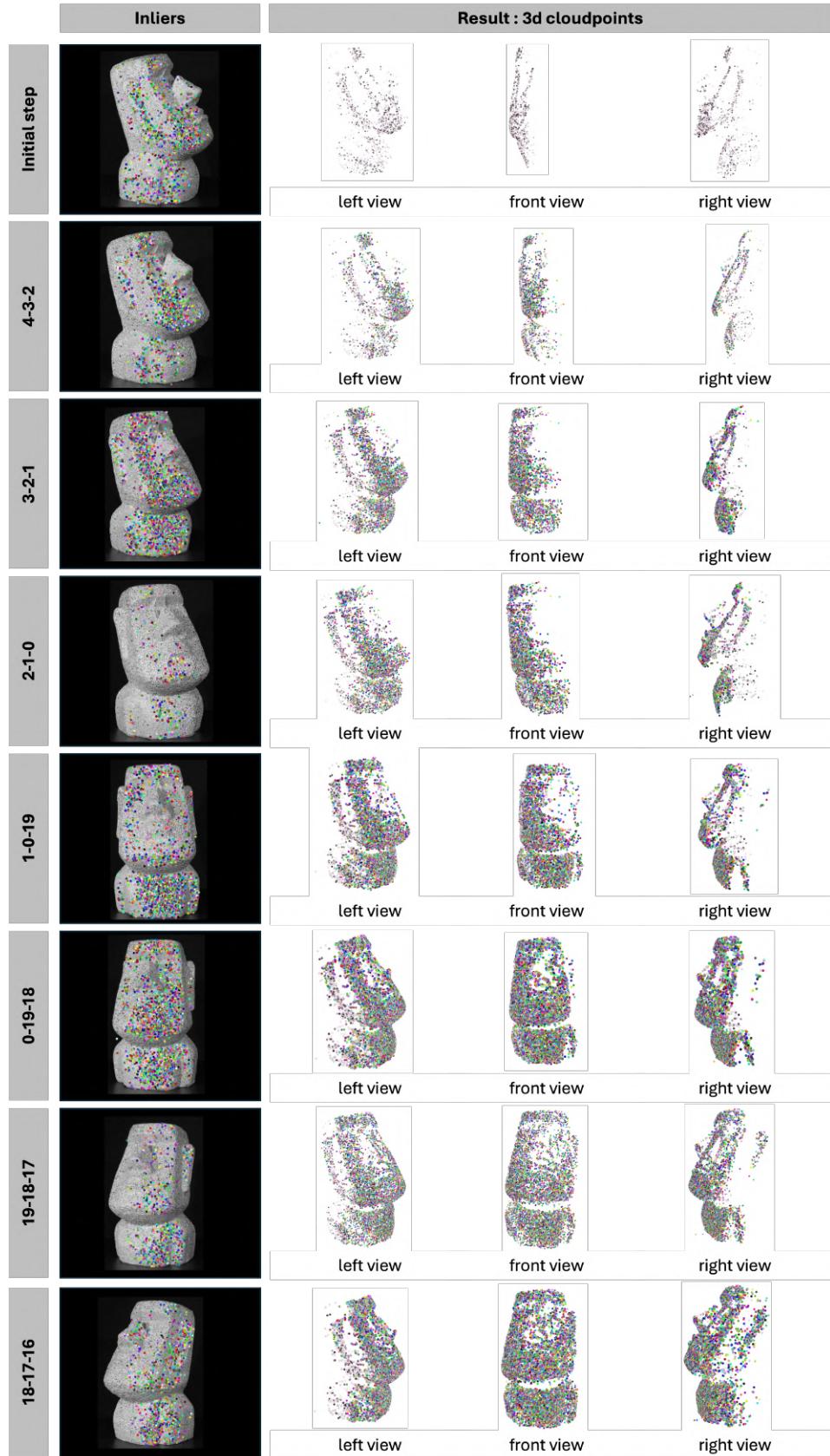


Figure 13: Result : Multi View SFM (moai)

initial step에 대응되는 점들은 물체의 원본 색상으로 표시하였고, 나머지 growing step들에 대해서는 initial step과 구별하기 위해 inlier들에 특징적인 색상들을 부여하여 시각화하였다. growing step마다 새로운 viewpoint에 대응되는 새로운 3d point 정보들을 추가하고 있는 모습을 시각적으로 확인할 수 있다. 결과적으로 multi view sfm에서 growing step을 수행함에 따라, 더욱 완성도 있는 3d structure 정보를 얻을 수 있음을 효과적으로 확인할 수 있었다.

### 3.4 Custom dataset with Camera calibration

구축한 Custom dataset은 다음과 같다.

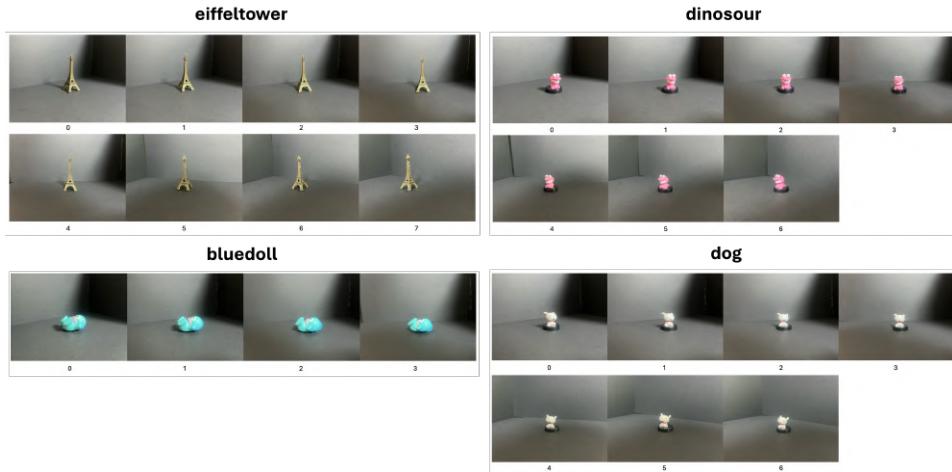


Figure 14: Custom Dataset (eiffeltower, dinosour, bluedoll, dog)

Custom dataset에 대해 Camera calibration을 수행하기 위해 다음과 같이 fixed-focus camera로 촬영한 30장의 checker board 사진을 사용하였으며, 결과로 다음과 같은 camera intrinsic matrix를 구할 수 있었다.

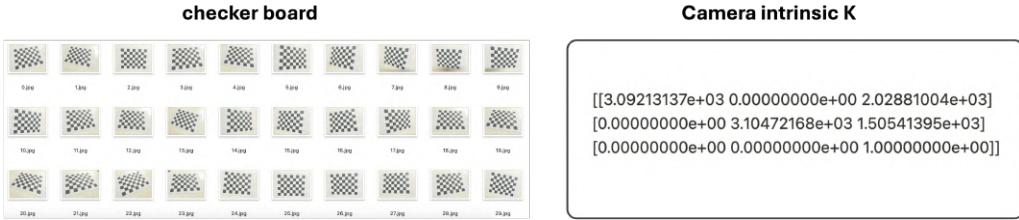


Figure 15: Custom Dataset Camera Calibration

## References

- [1] Richard Szeliski, *Computer Vision: Algorithms and Applications*, Draft version (2010), Available at: [http://szeliski.org/Book/drafts/SzeliskiBook\\_20100903\\_draft.pdf](http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf), Chapter 7: Structure from Motion, Accessed: 2025-05-09.
- [2] OpenCV, *SIFT Feature Detection*, Available at: [https://docs.opencv.org/4.x/da/df5/tutorial\\_py\\_sift\\_intro.html](https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html), Accessed: 2025-05-09.
- [3] OpenCV, *Feature Matching using Brute-Force Matcher*, Available at: [https://docs.opencv.org/4.x/dc/dc3/tutorial\\_py\\_matcher.html](https://docs.opencv.org/4.x/dc/dc3/tutorial_py_matcher.html), Accessed: 2025-05-09.
- [4] H. Stewénius, C. Engels, and D. Nistér, *Recent Developments on Direct Relative Orientation*, ISPRS Journal of Photogrammetry and Remote Sensing, Vol. 60, No. 4, pp. 284–294, June 2006.
- [5] Bert M. Haralick, Charles N. Lee, Karsten Ottenberg, and Michael Nölle, "Review and Analysis of Solutions of the Three Point Perspective Pose Estimation Problem," *International Journal of Computer Vision*, vol. 13, no. 3, pp. 331–356, 1994.
- [6] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon, "Bundle Adjustment—A Modern Synthesis," In *International Workshop on Vision Algorithms*, pp. 298–372, 1999.
- [7] Yekeun Jeong, Jinsun Park, Kyungdon Joo, and In So Kweon, "Pushing the Envelope of Modern Methods for Bundle Adjustment," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 34, no. 8, pp. 1605–1617, 2012.
- [8] foss4g.tistory.com, *OpenCV Camera Calibration*, Available at: <https://foss4g.tistory.com/1665>, Accessed: 2025-05-09.
- [9] OpenCV, *Camera Calibration*, Available at: [https://docs.opencv.org/4.x/dc/dbb/tutorial\\_py\\_calibration.html](https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html), Accessed: 2025-05-09.